Recent Developments in Real Quantifier Elimination Technology

Matthew England
Coventry University
Centre for Computational Science and Mathematical Modelling

Warwick Mathematics Colloquium
8th March 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 will cover recent developments the author has been involved in that improve CAD with ideas from (a) satisfiability checking; (b) proof systems; and (c) machine learning.

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

Joint work with: Chris Brown, Erika Abraham, Russell Bradford, Kelly Cohen, James Davenport, Tereso del Río, Gereon Kremer, Lynn Pickering, Jasper Nalbach, AmirHosein Sadeghi Manesh 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 $b = 3$:

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

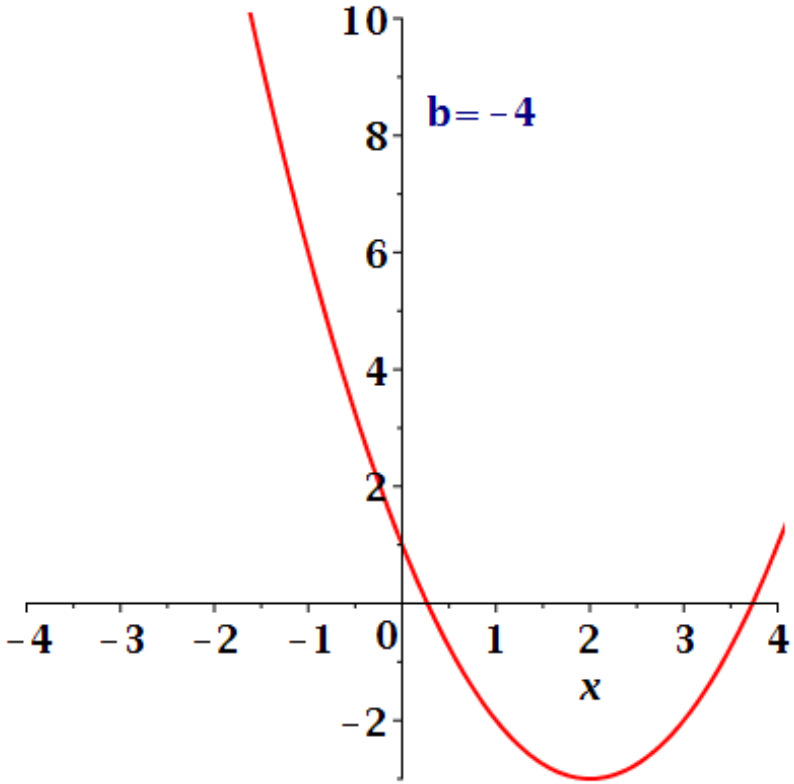
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example

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

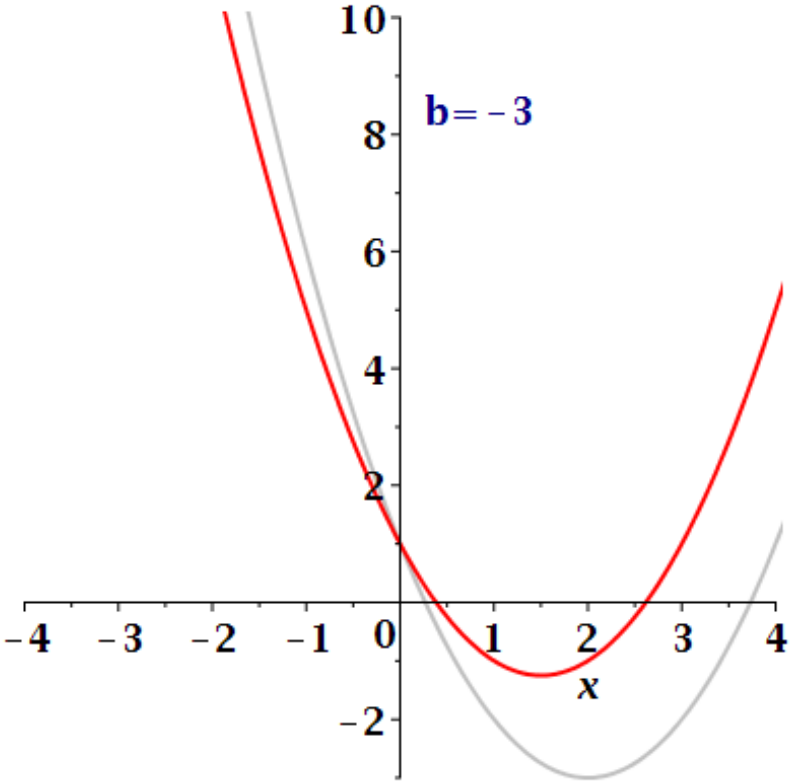
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True} \ ,$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False} \ .$$

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 automatically.



Partially quantified Tarski formulae example

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

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

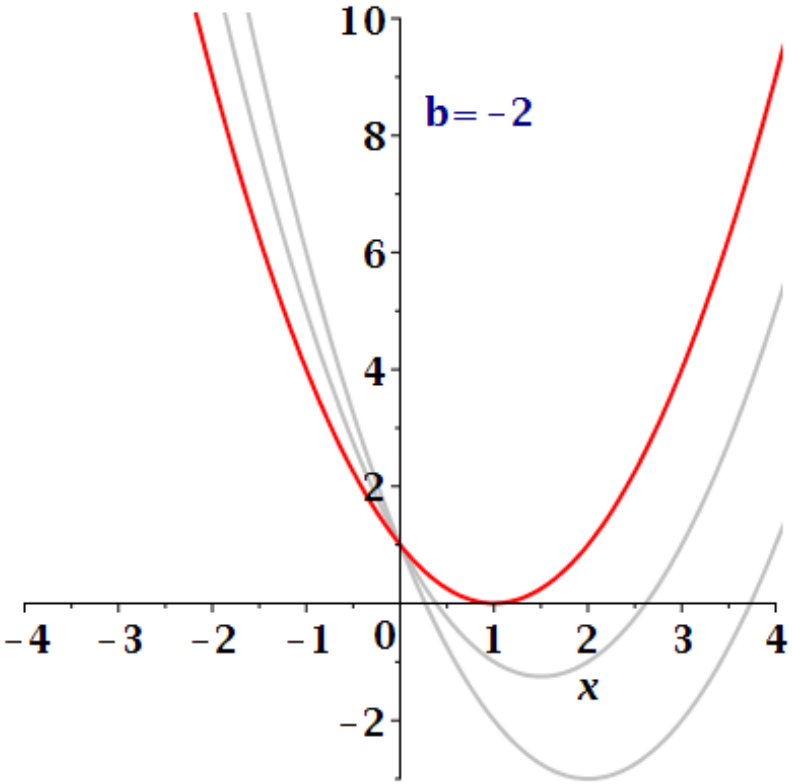
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example

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

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

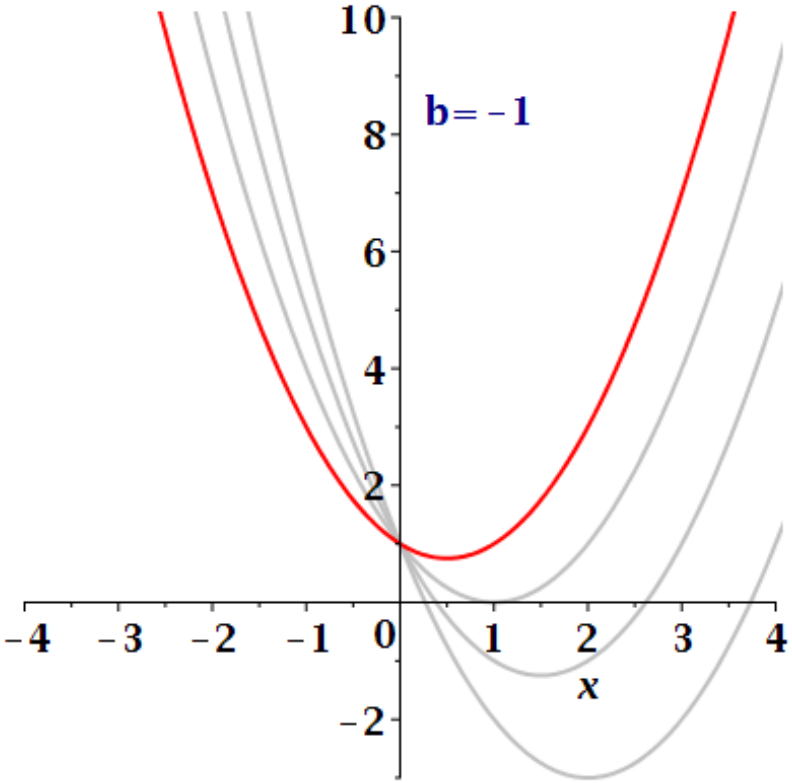
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example (Slide 3/43)

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

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

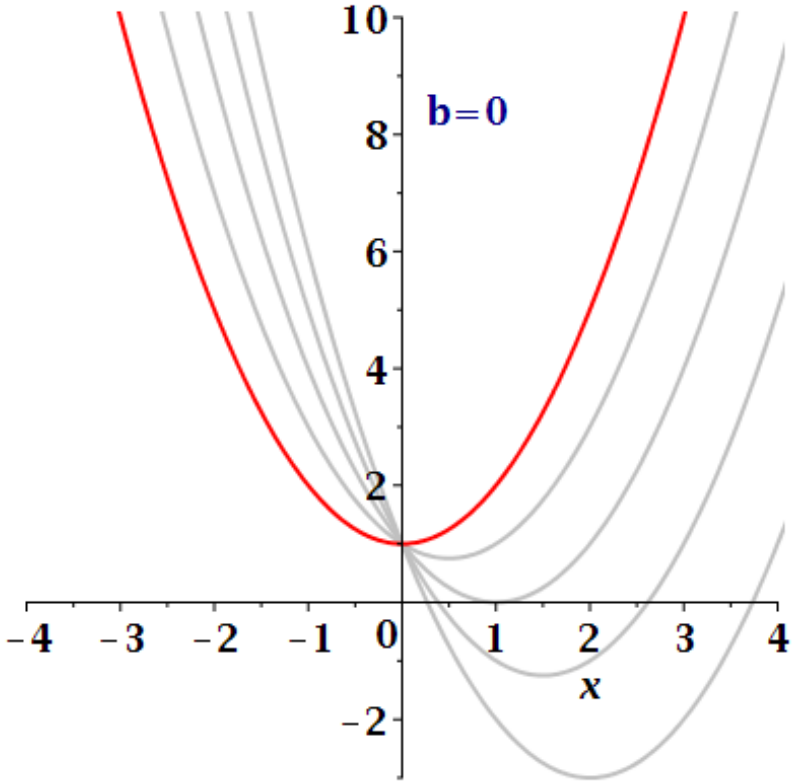
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example (Slide 3/43)

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

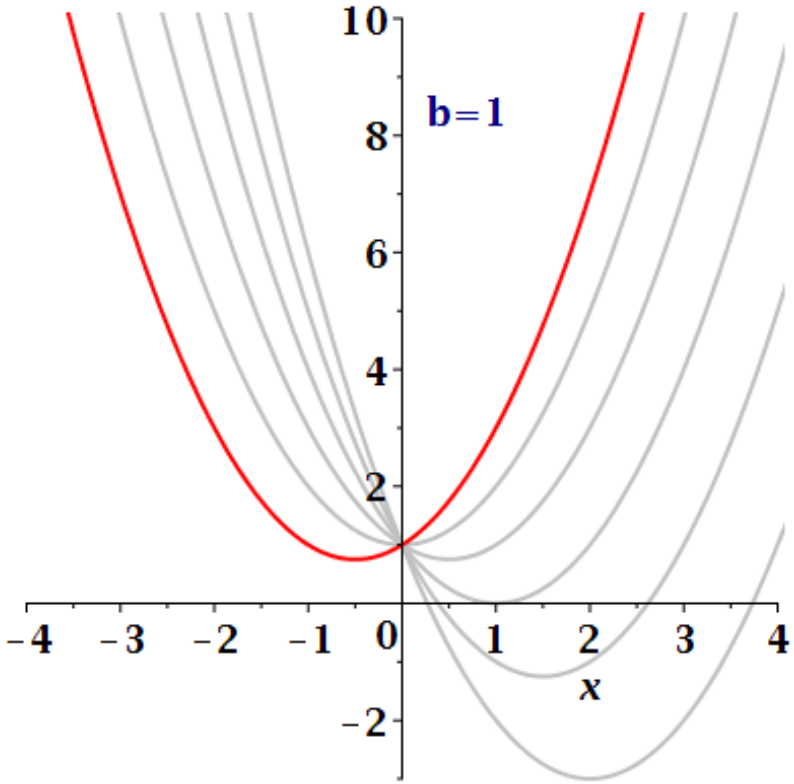
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example

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

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

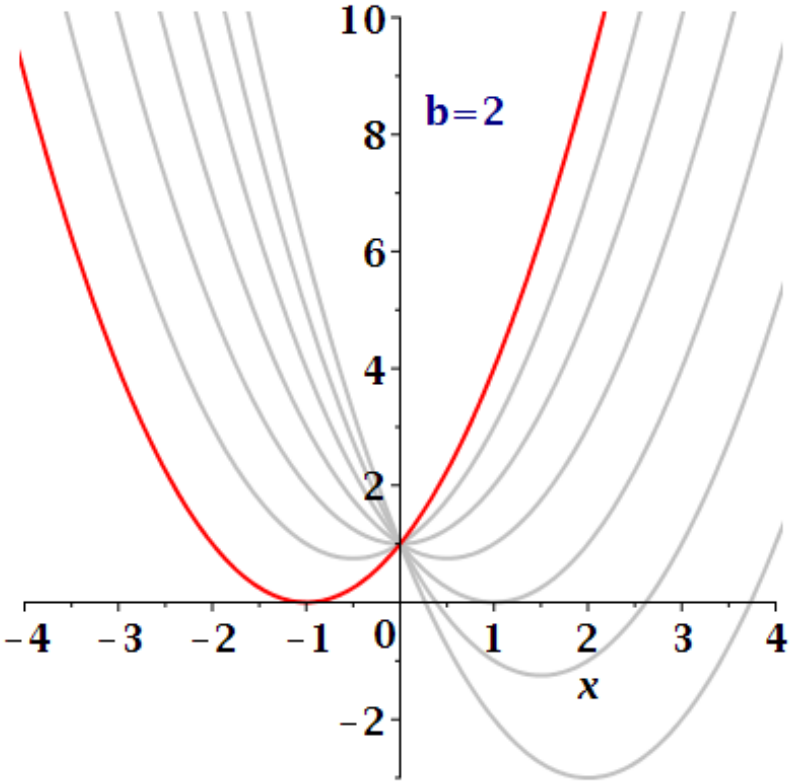
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example (Slide 3/43)

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

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

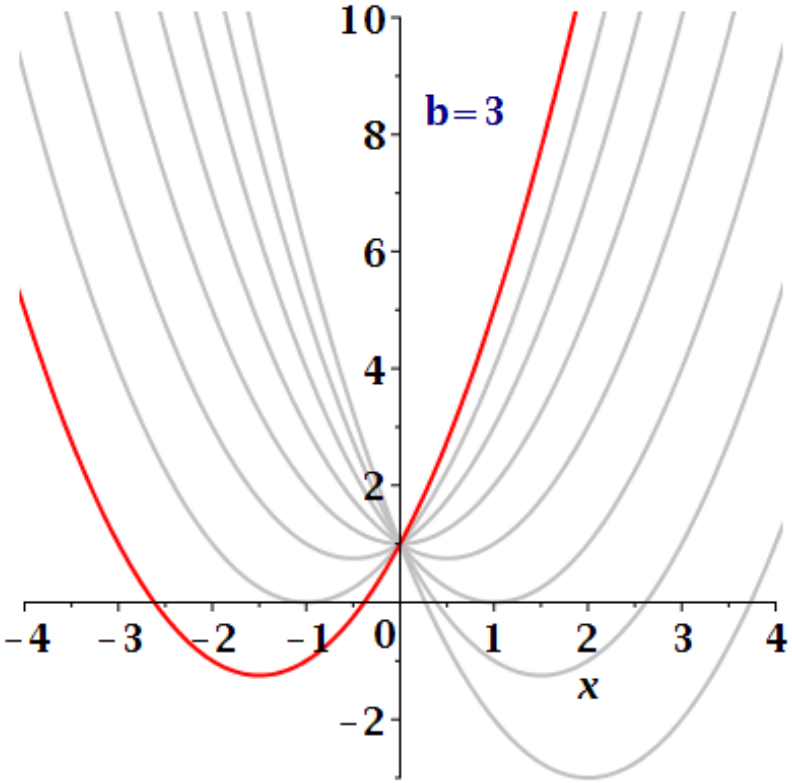
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

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 automatically.



Partially quantified Tarski formulae example (Slide 3/43)

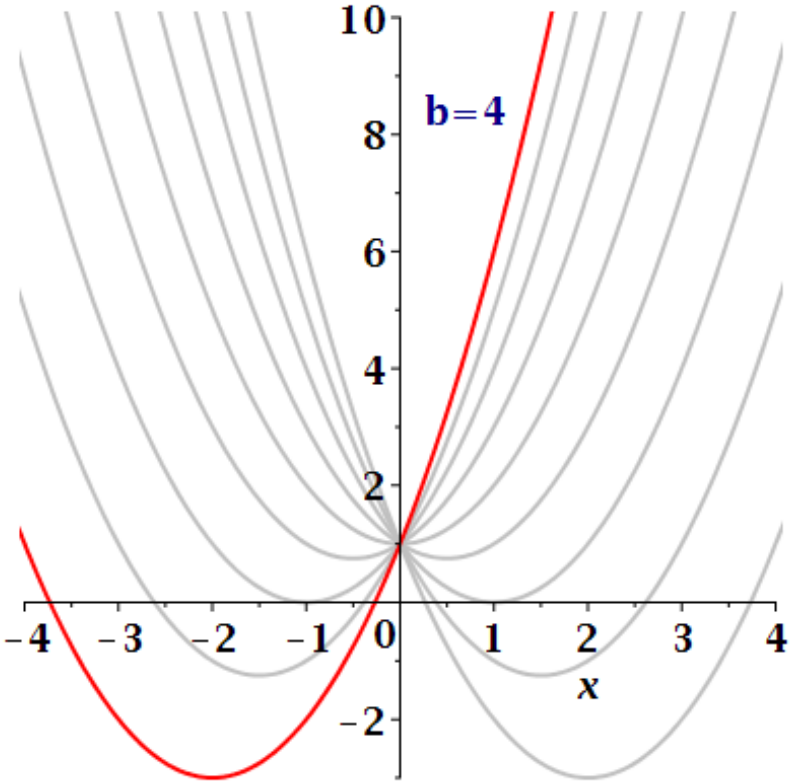
Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

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

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 automatically.



Numeric vs. Symbolic solution? (Slide 4/43)

The solution in the previous slide was essentially numeric: try what happens for many different values and interpolate a solution.

The alternative is to use **Symbolic Computation:** algorithms and data-structures for manipulating exact mathematical expressions and objects. Traditionally implemented in *Computer Algebra Systems* (e.g. Macaulay2, Maple, Mathematica, Sage).

Advantages of Symbolic Computation:

- Can solve numerically ill-conditioned problems.
- Provide guarantees for safety critical systems.
- Provide fundamental insight into the system.

Disadvantage of Symbolic Computation: much more expensive!

Outline

Cylindrical Algebraic Decomposition (Slide 5/43)

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 **cylindrical** meaning the projection of any two onto 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 $\mathbb{R}^{m-1}$).

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 finite set of sample points.

Most applications (e.g. QE) actually provide as input a logical formulae built from polynomial constraints and require as output a **truth-invariant CAD**: one such that each formula has constant truth value in each cell.
Such a CAD allows us to find solution sets from the descriptions of true cells: semi-algebraic; easy to visualise and check membership.

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 finite set of sample points.

Most applications (e.g. QE) actually provide as input a logical formulae built from polynomial constraints and require as output a **truth-invariant CAD**: one such that each formula has constant truth value in each cell.
Such a CAD allows us to find solution sets from the descriptions of true cells: semi-algebraic; easy to visualise and check membership.

Example: Circle — visualisation

Cell 1: $x < -1$, y free

Cell 2: $x = -1, y < 0$

Cell 3: $x = -1, y = 0$

Cell 4: $x = -1, y > 0$

Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$

Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$

Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$

Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$

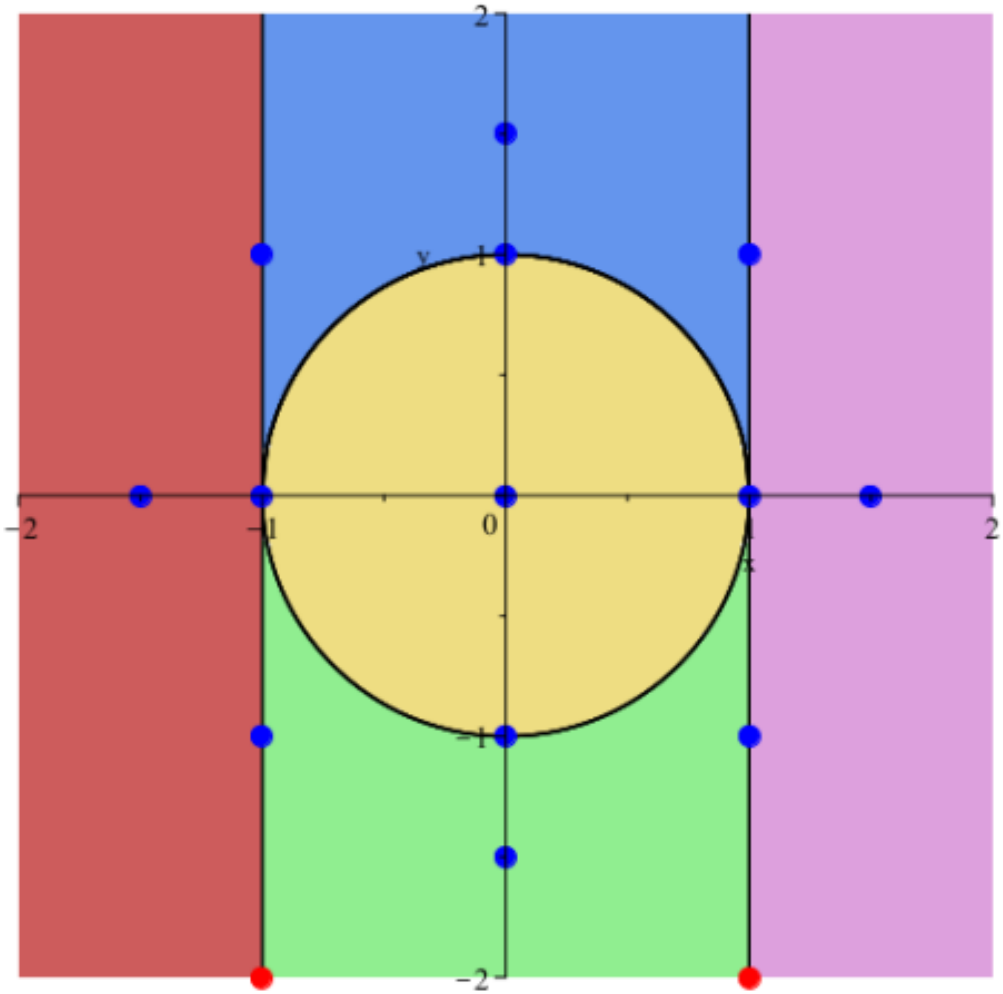
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

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

Cell 13: $x > 1$, y free



Example: Circle — visualisation

Cell 1: $x < -1$, y free

Cell 2: $x = -1, y < 0$

Cell 3: $x = -1, y = 0$

Cell 4: $x = -1, y > 0$

Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$

Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$

Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$

Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$

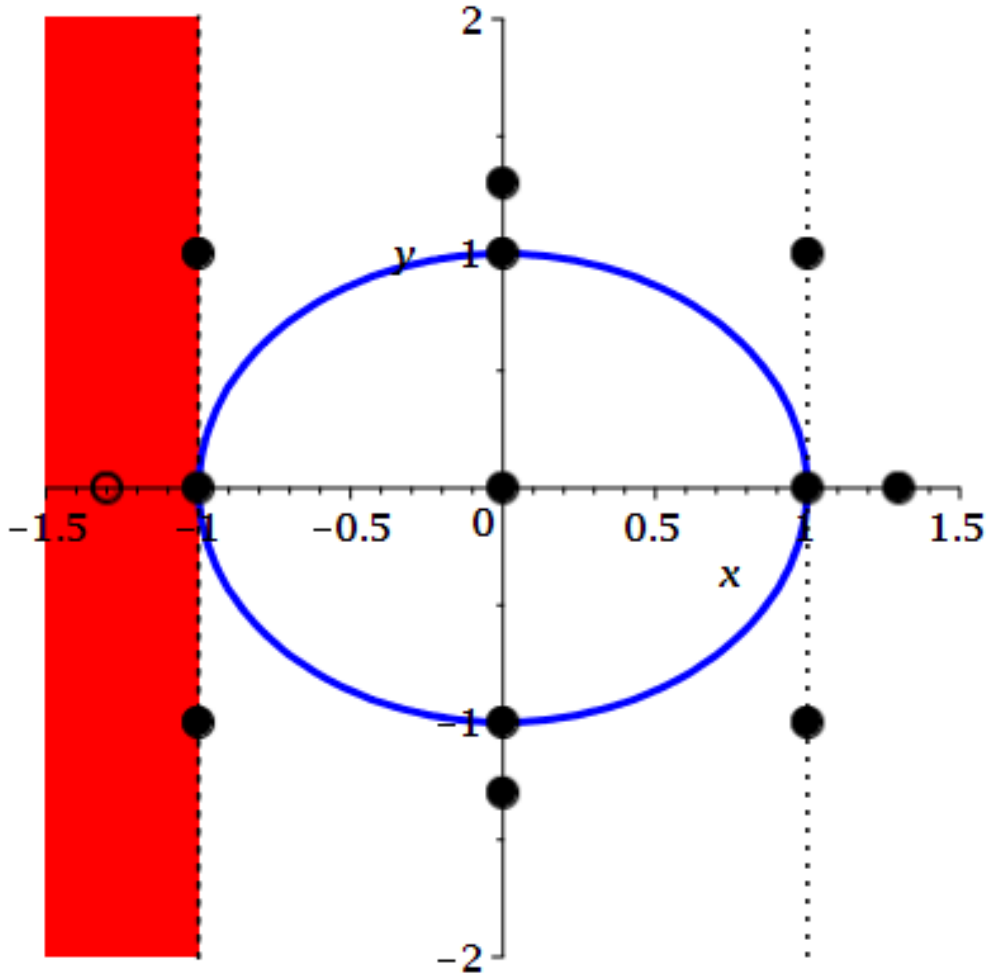
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

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

Cell 13: $x > 1$, y free



Example: Circle — visualisation

Cell 1: $x < -1$, y free

Cell 2: $x = -1, y < 0$

Cell 3: $x = -1, y = 0$

Cell 4: $x = -1, y > 0$

Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$

Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$

Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$

Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$

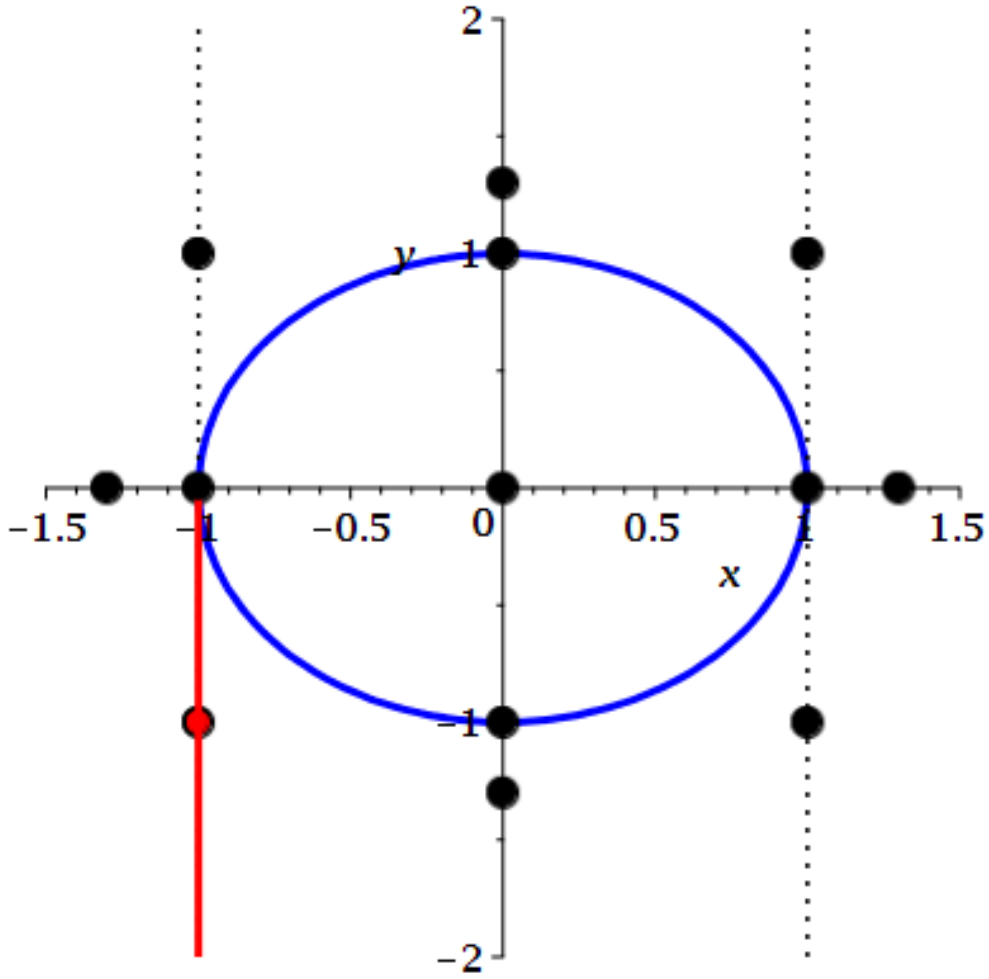
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

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

Cell 13: $x > 1$, y free



Example: Circle — visualisation

Cell 1: $x < -1$, y free

Cell 2: $x = -1, y < 0$

Cell 3: $x = -1, y = 0$

Cell 4: $x = -1, y > 0$

Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$

Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$

Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$

Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$

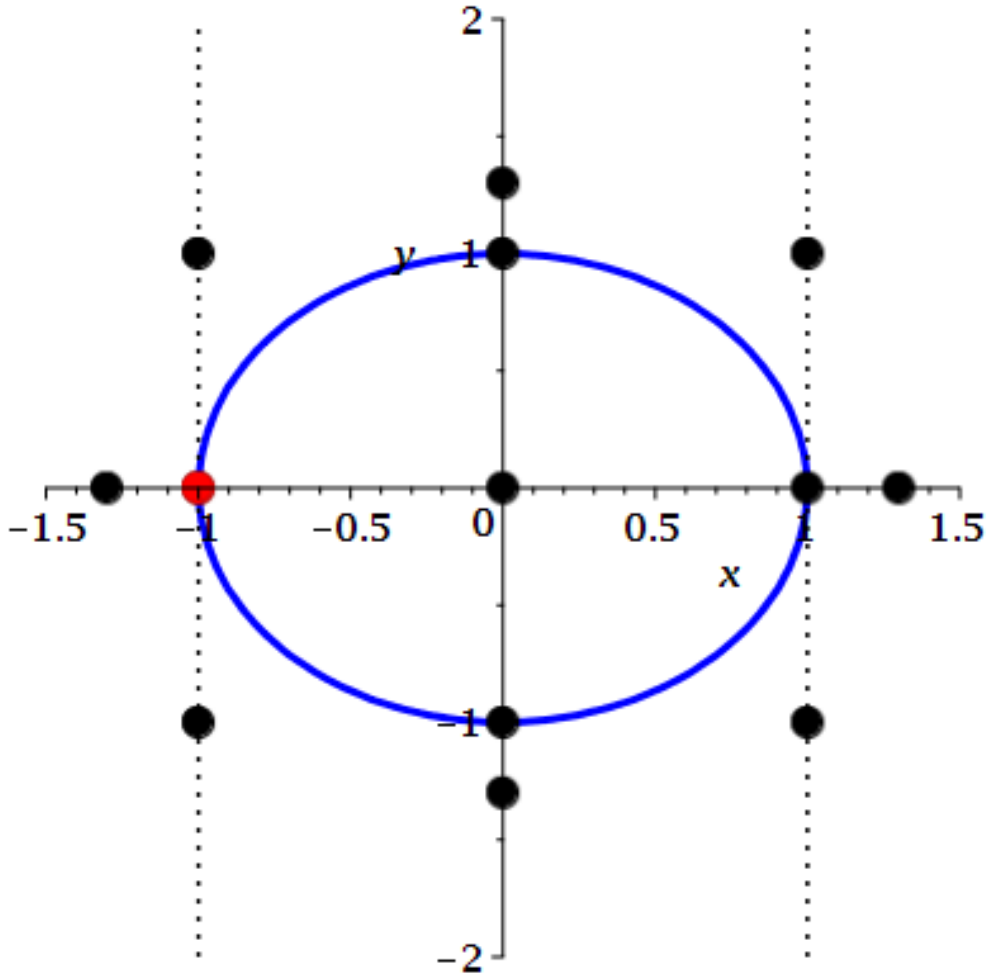
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

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

Cell 13: $x > 1$, y free



Example: Circle — visualisation

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

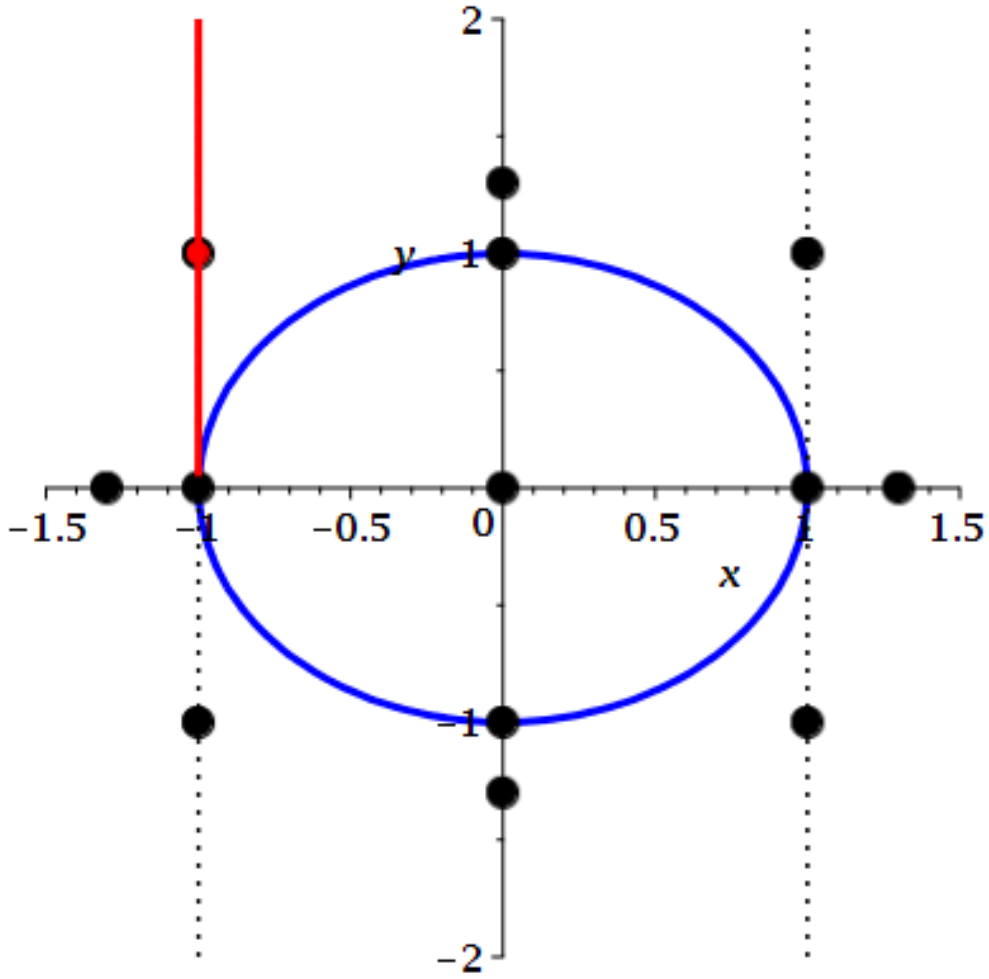
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

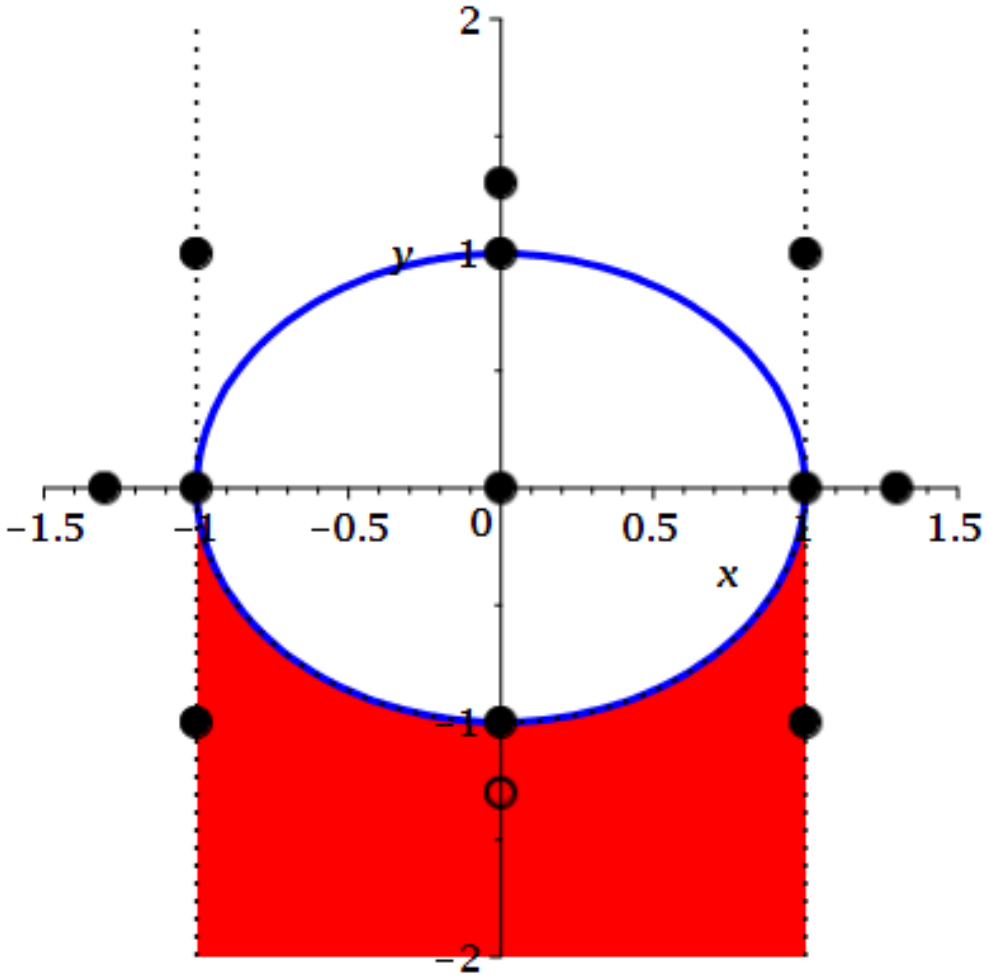
Cell 9: $-1 < x < 1$, $y^2 + x^2 - 1 > 0, y > 0$

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

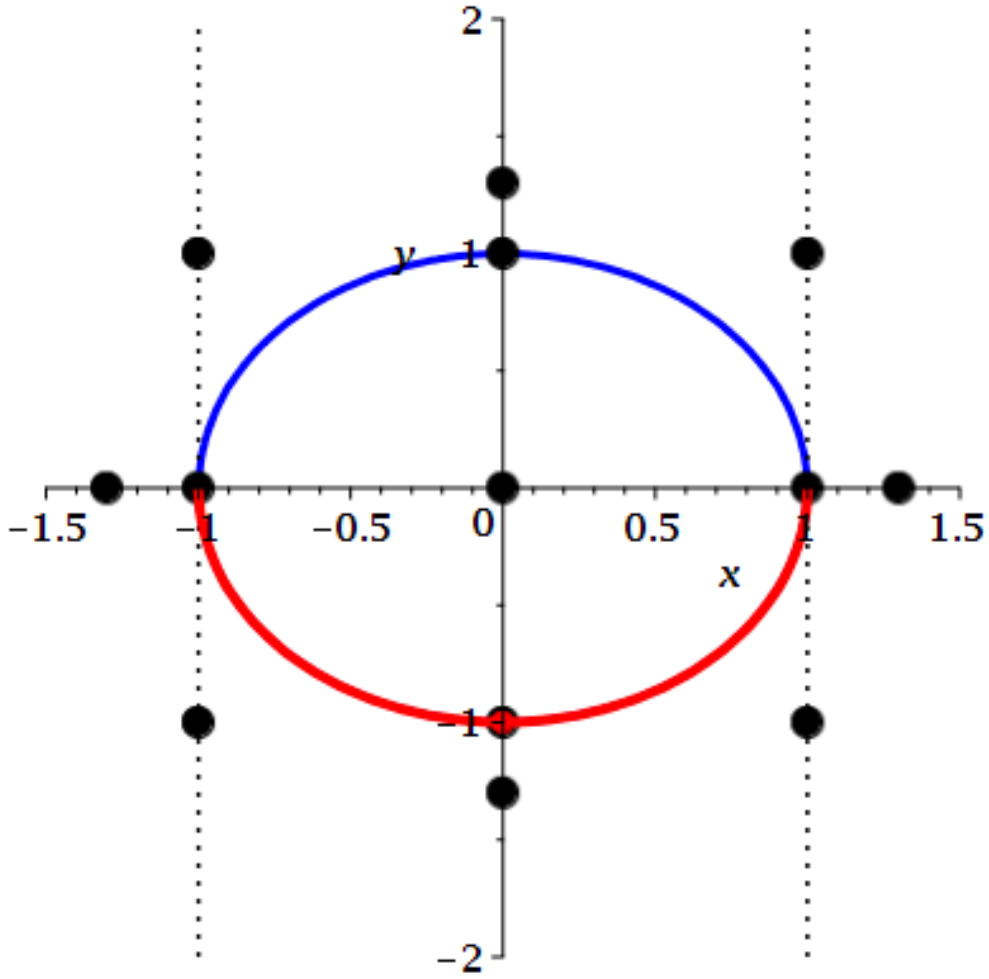
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

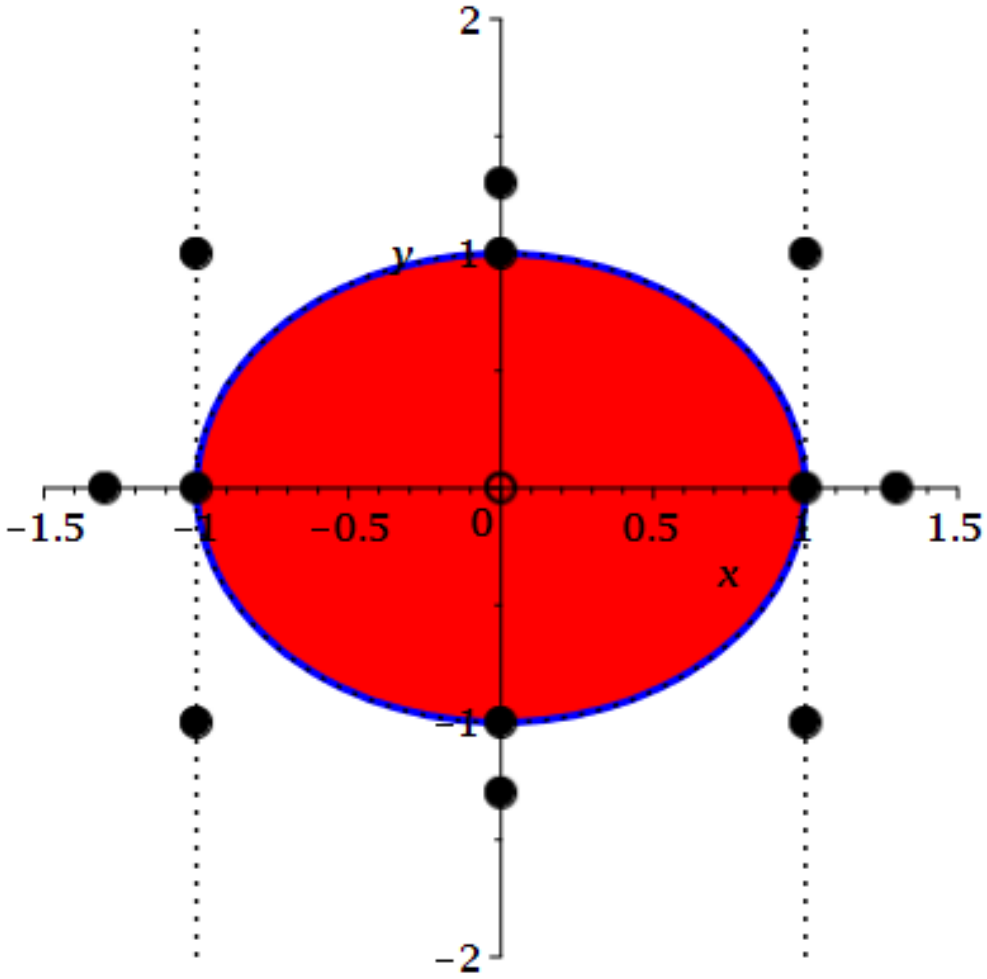
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

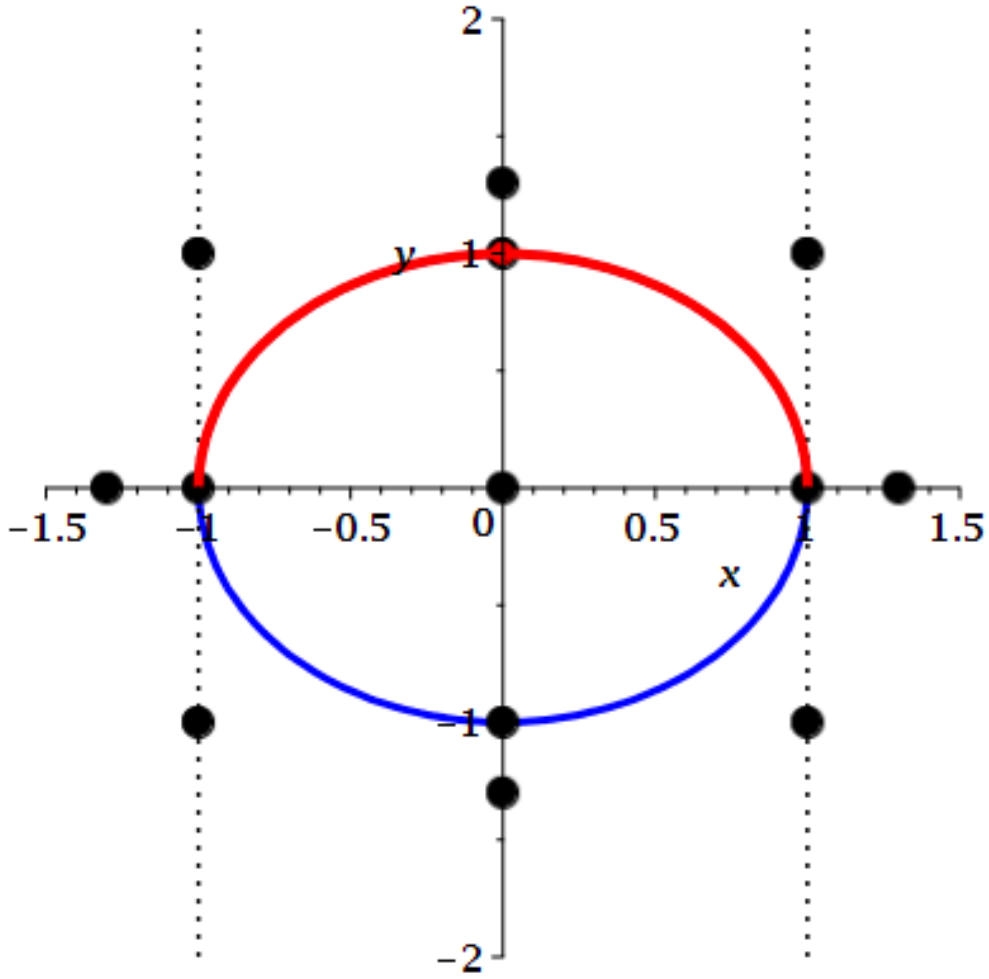
Cell 12: $x = 1, y > 0$

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

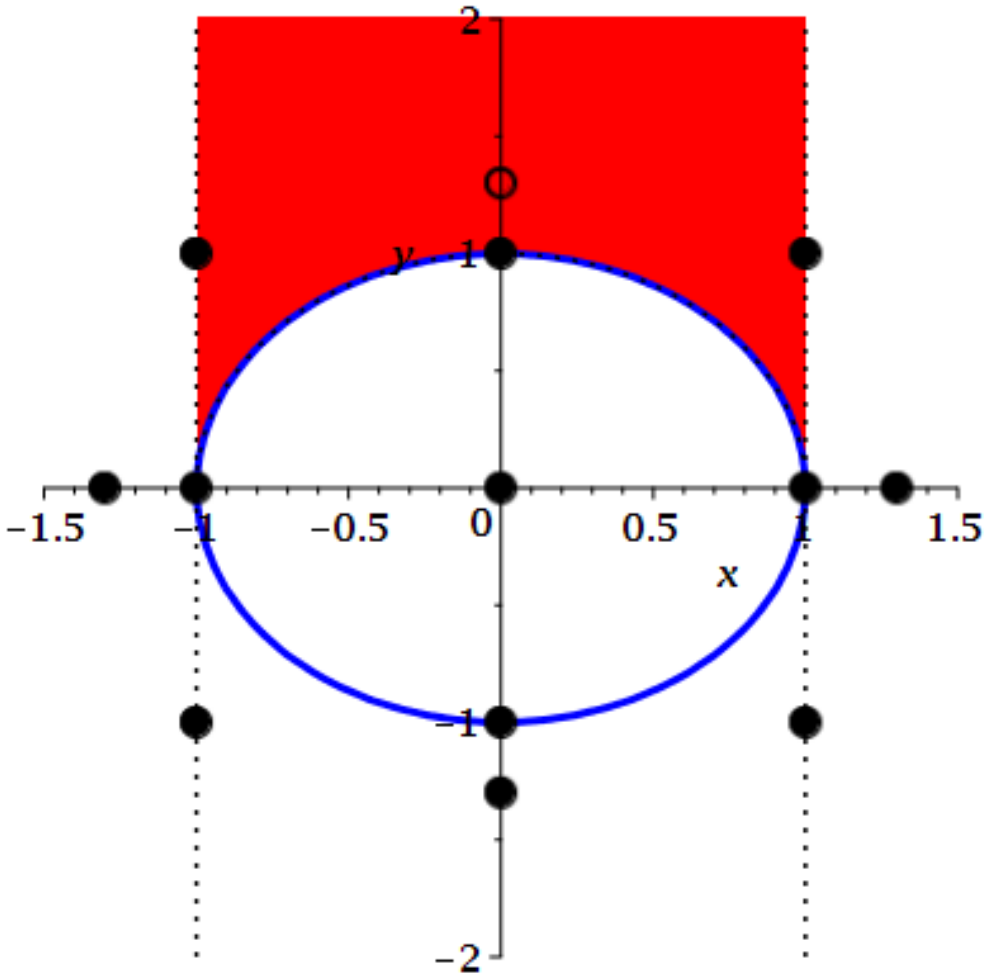
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

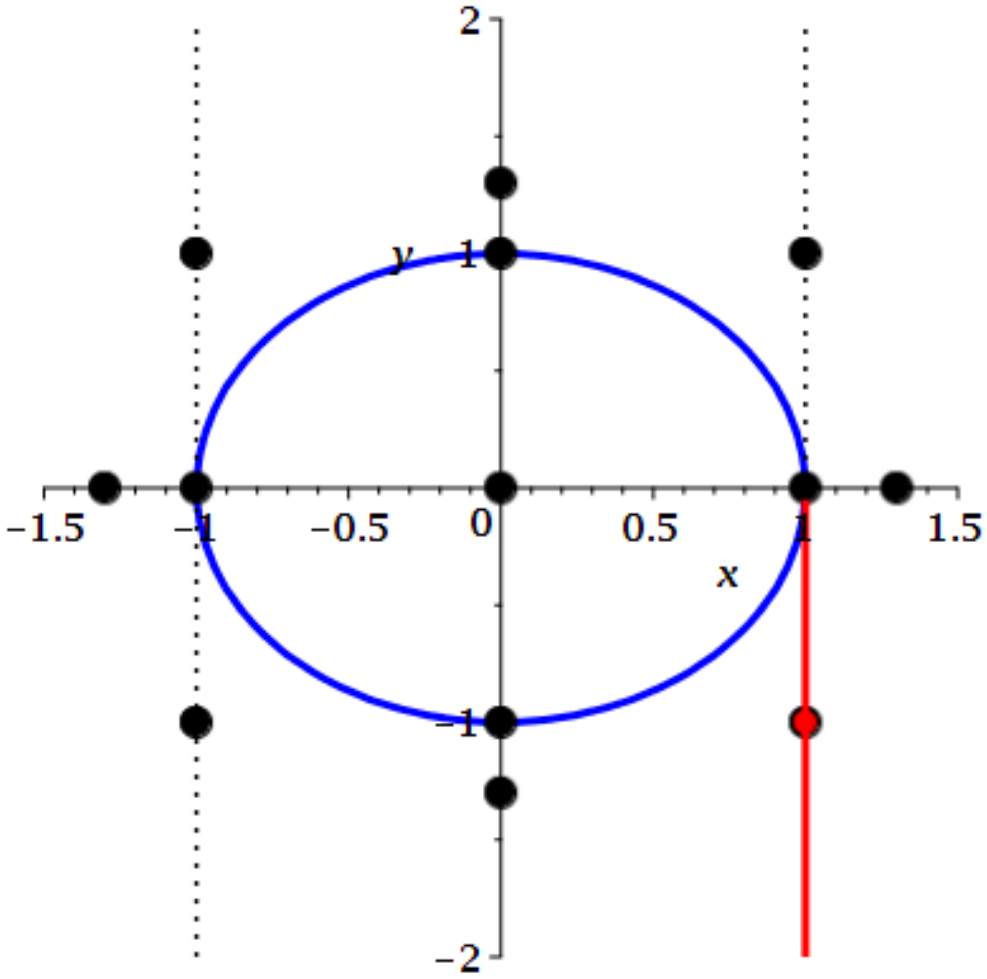
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

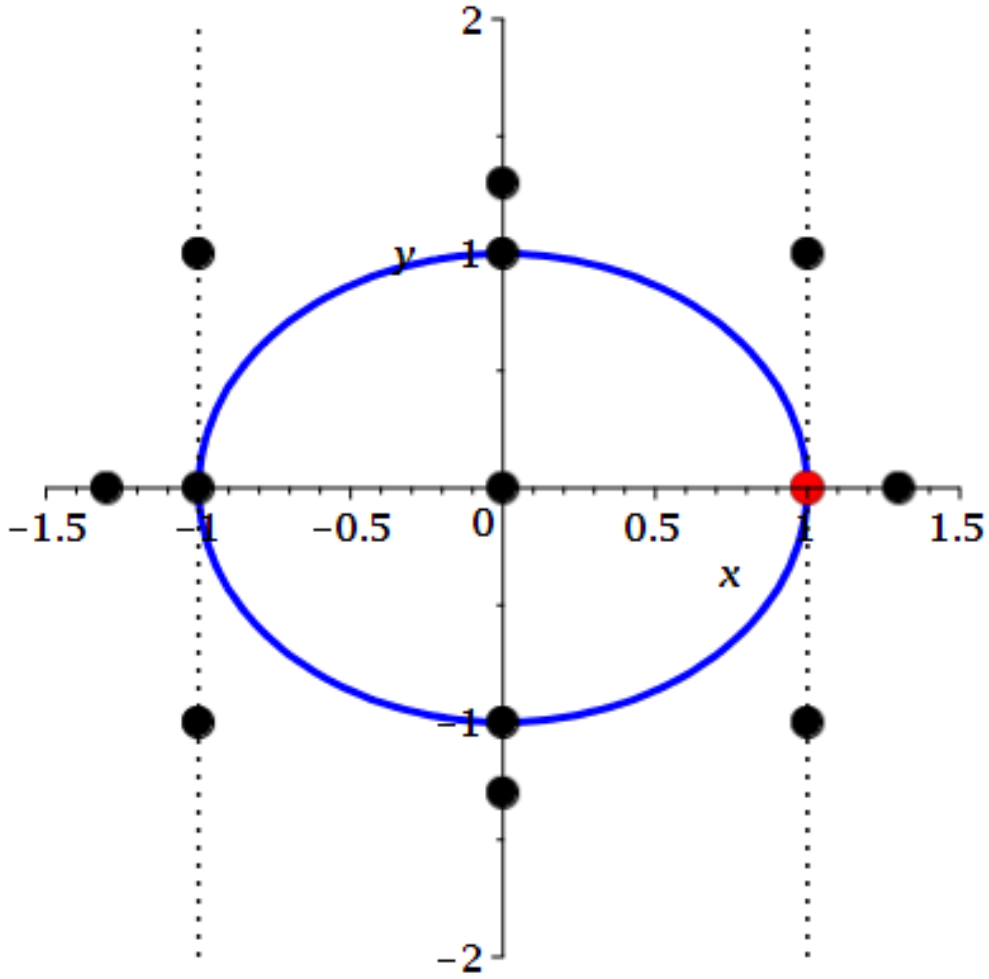
Cell 12: $x = 1, y > 0$

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

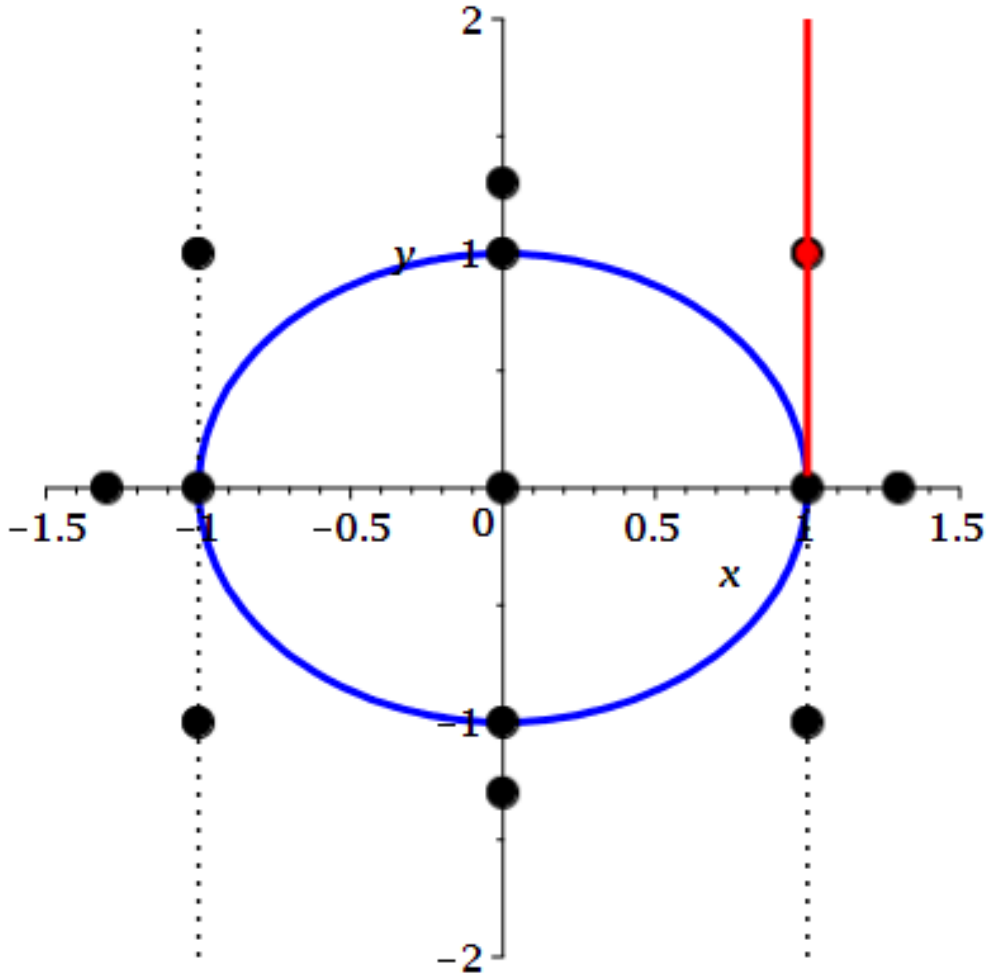
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

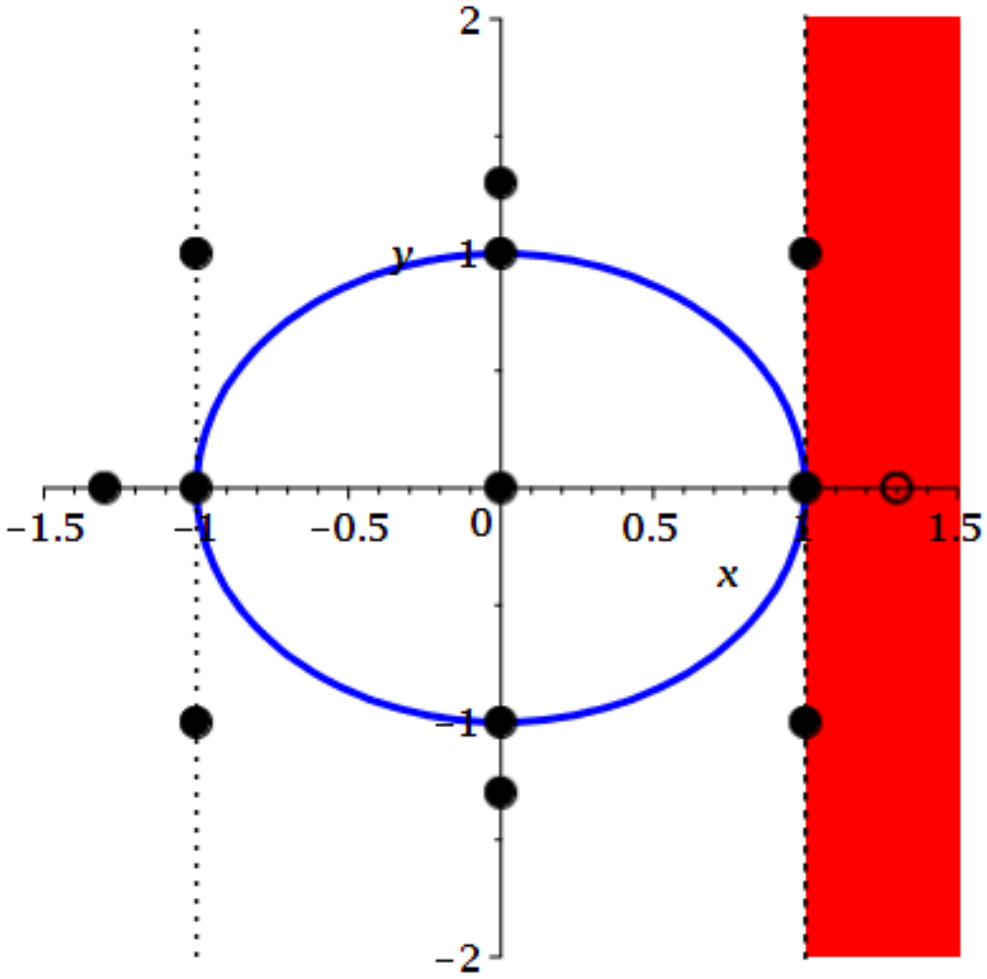
Cell 12: $x = 1, y > 0$

Cell 13: $x > 1$, y free



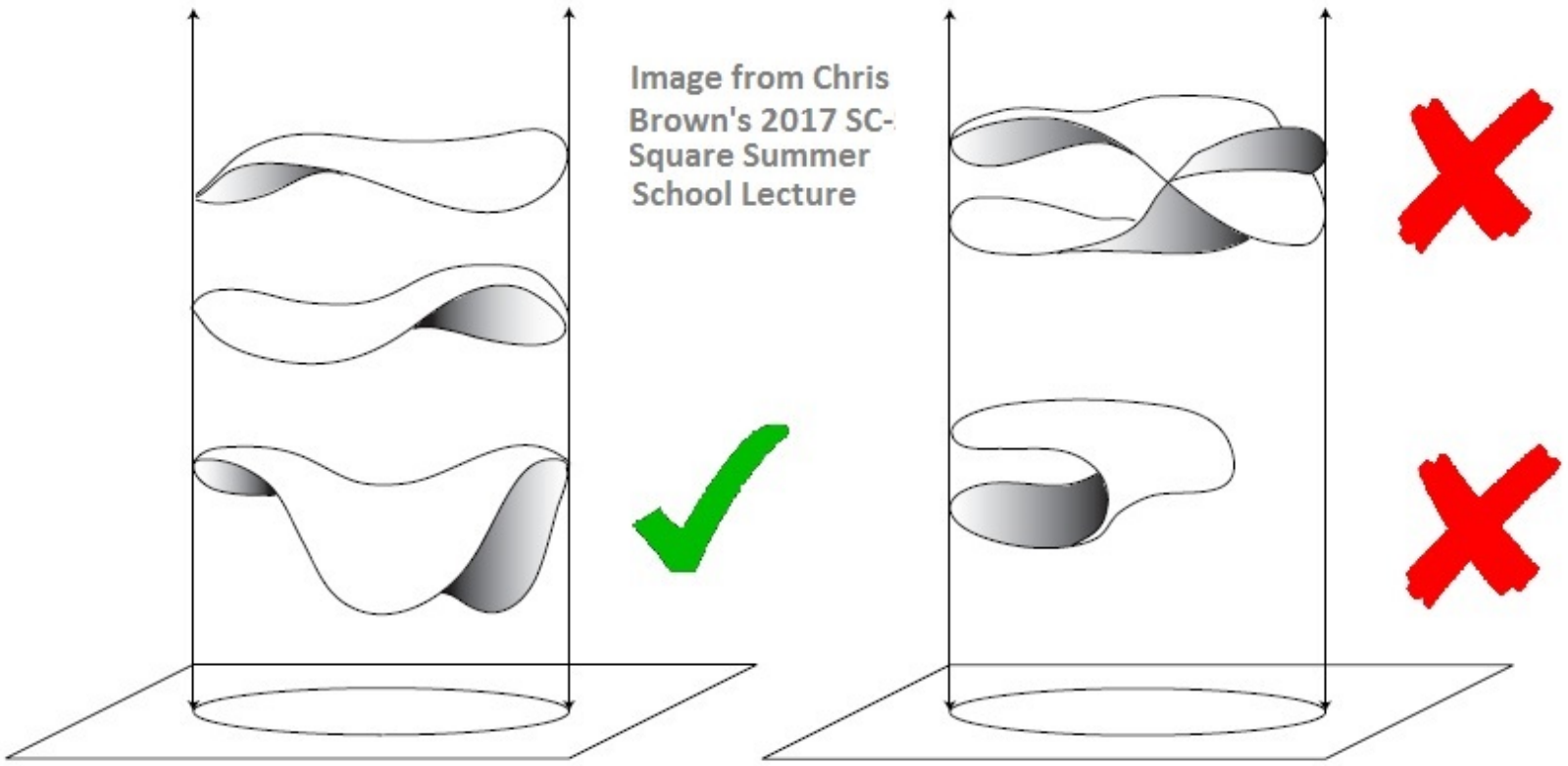
Example: Circle — visualisation

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?

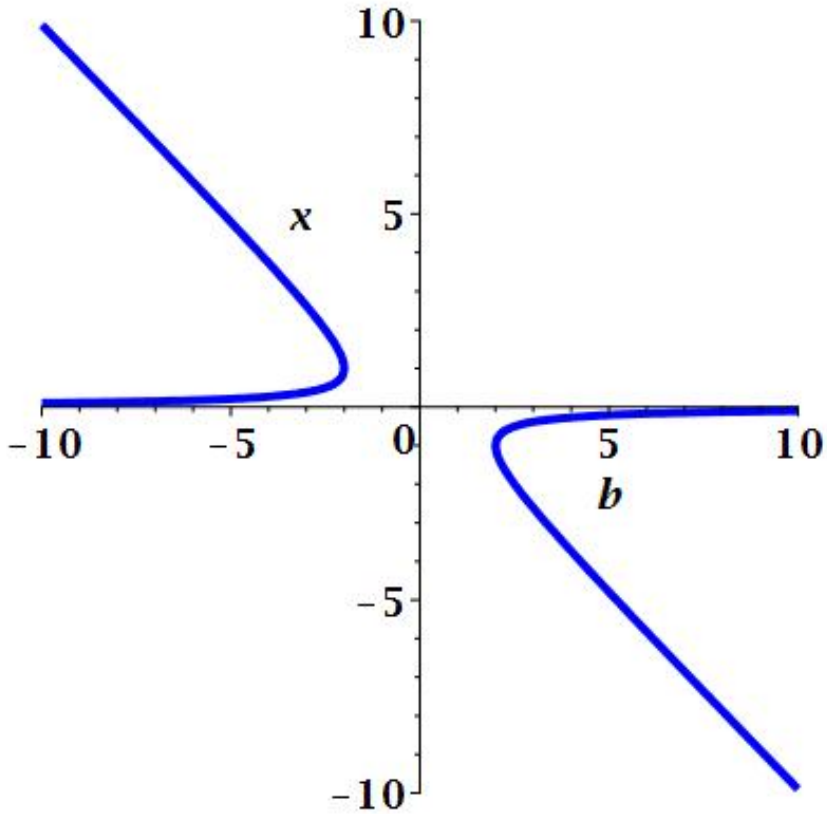
The usual approach is to calculate projection polynomials whose roots indicate changes in the behaviour of the input set; decompose with respect to these and then lift back working at a sample point. Sound if the projection provides **delineability**.



QE via CAD Example

How to determine with CAD?

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



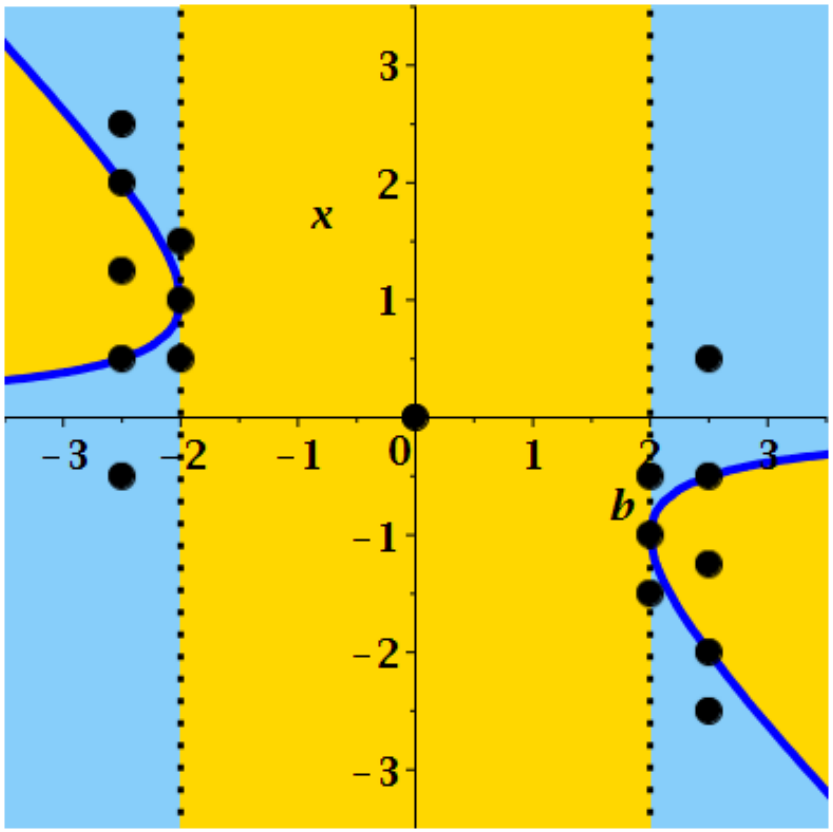
QE via CAD Example

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

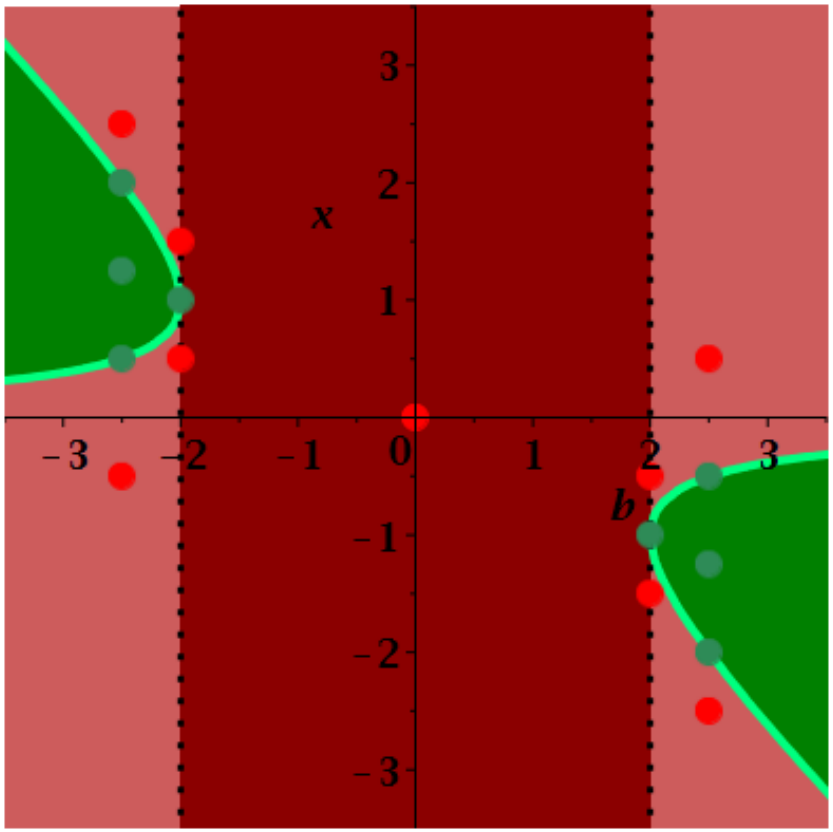
How to determine with CAD?

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

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?

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

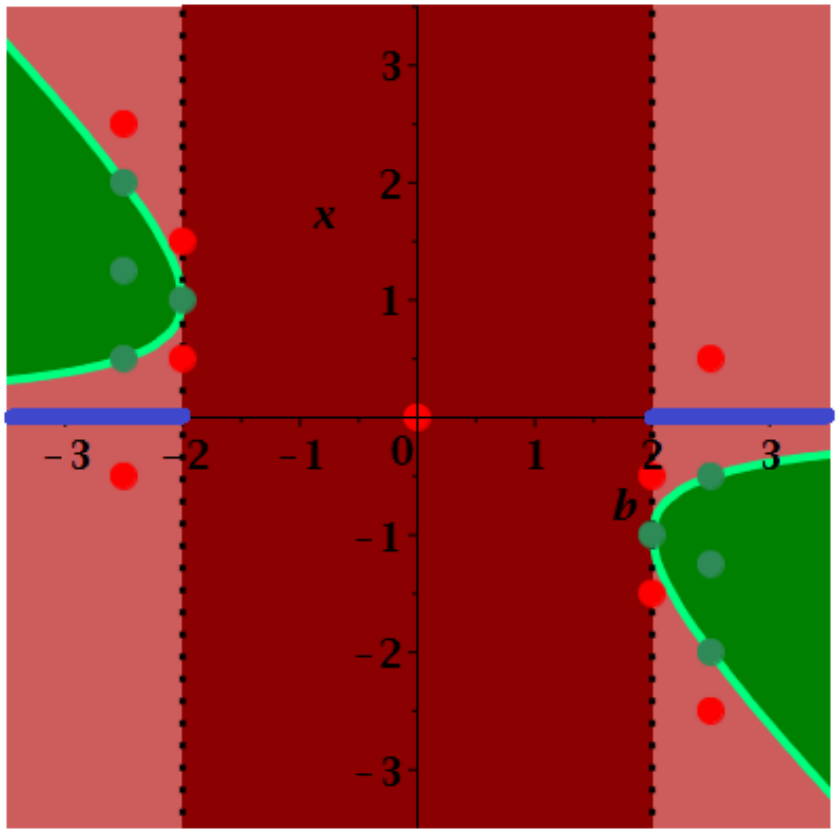
To solve we:

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?

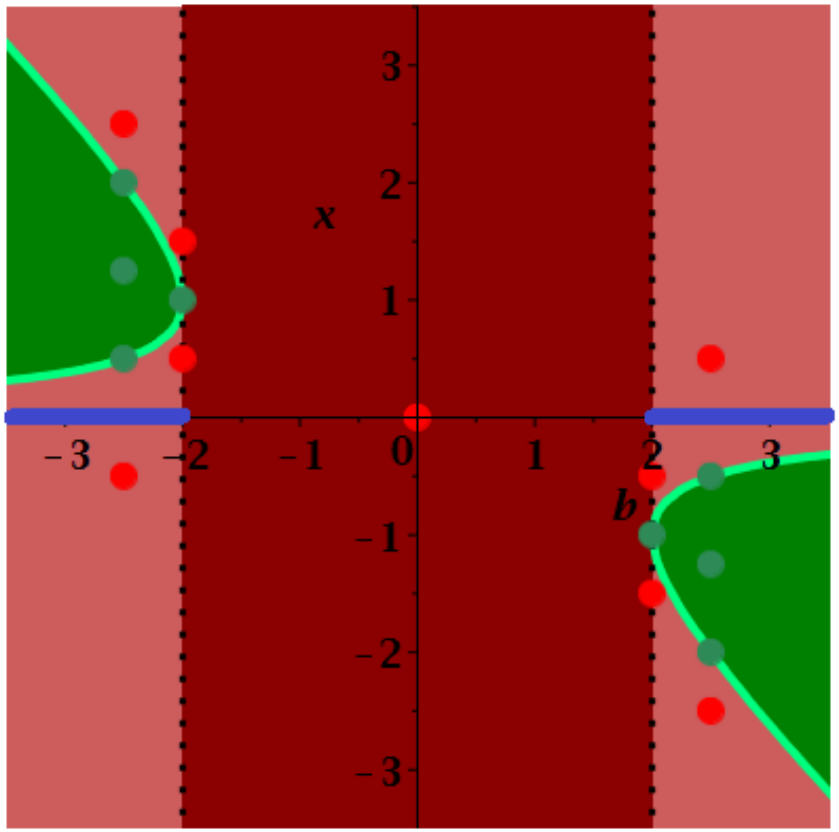
$$\exists x, x^2 + bx + 1 \leq 0$$

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 on a decomposition as follows.

- Eliminate existential quantifiers by projecting the true cells, as in the previous example.
- Eliminate universal quantifiers by using the relation

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

and then proceeding with existential QE.

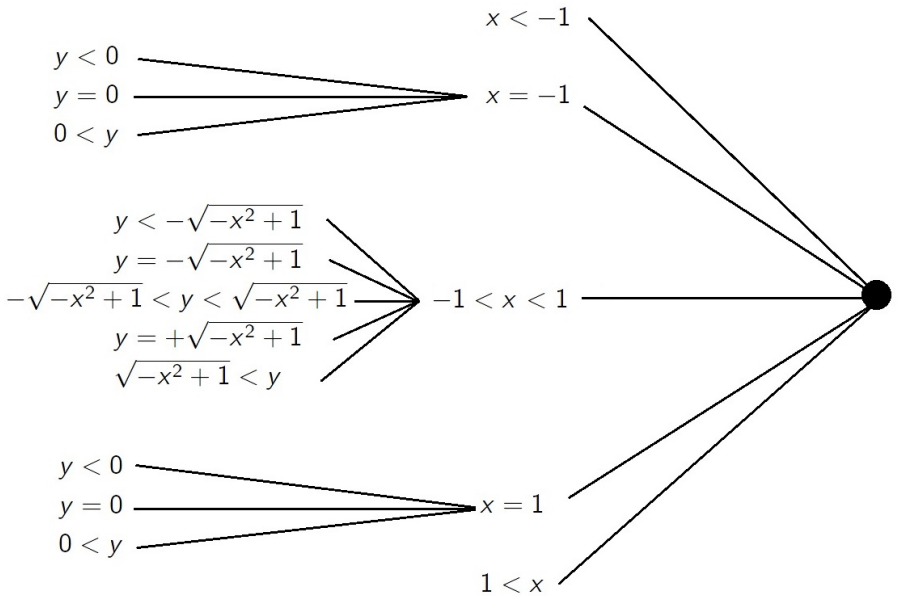
Recall our original example was $\forall x, x^2 + bx + 1 > 0$. The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$ which from the previous slide we know has solution $b \leq -2 \vee b \geq +2$. The solution to our universally quantified problem is then the negation of this: $-2 < b \wedge b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

Cylindricity Property

Decompositions allow us to understand infinite space with a finite number of samples and (semi-) algebraic cells allows us to construct solution formulae easily. But why cylindricity?

The Real QE solution process requires us to project cells (and combine those projections) and to calculate the complement of cells: both of these things are trivial for cells arranged cylindrically.

Cylindricity means we can think of CAD as a tree branching by variable.



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!

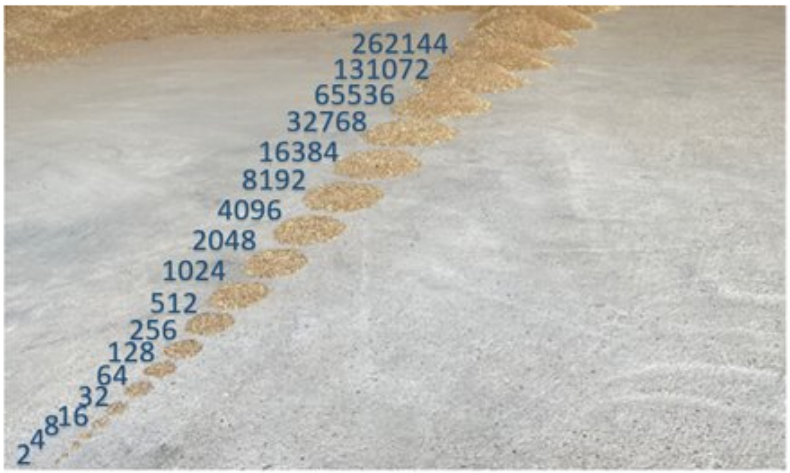
C. Brown and J.H. Davenport.
The complexity of quantifier elimination and cylindrical algebraic decomposition.
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

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

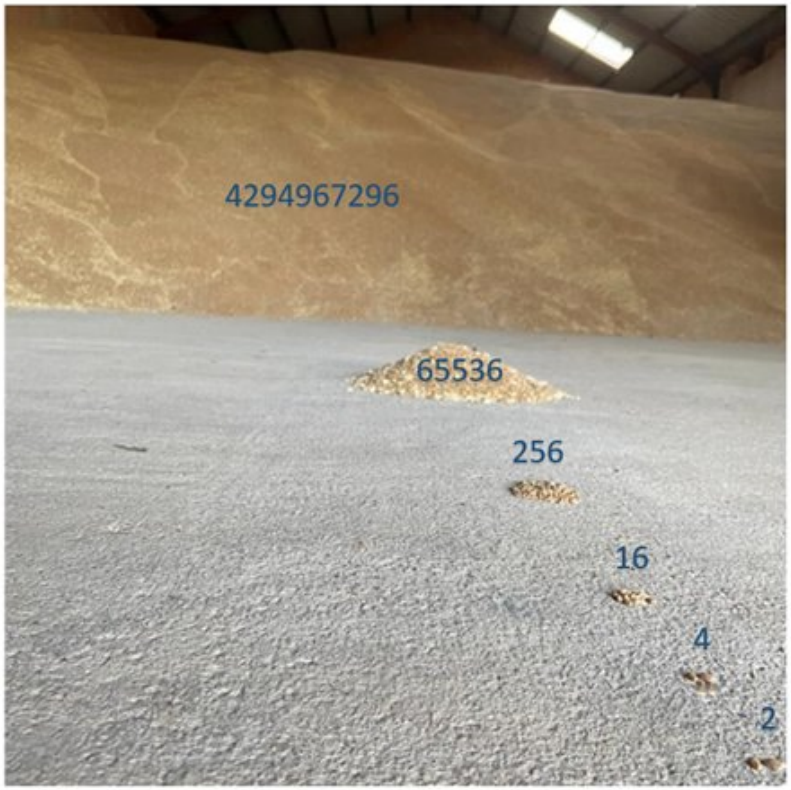
$$2^{2^3} = 256 \qquad 2^{2^4} = 65,536 \qquad 2^{2^5} = 4,294,967,296$$

The Doubly Exponential Wall

Images by Tereso del Rio



Exponential Growth



Doubly Exponential Growth

QE via CAD Implementations (Slide 14/43)

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

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 SAGE, and can now even run in your browser!

`http://tarski.tk/`

CAD implementations also exist in the SMT-solvers, SMT-RAT, YICES2, Z3 and CVC5. All are available for free – but note these can only answer fully quantified problems.

Alternatives to CAD for QE

There exists algorithms to achieve Real QE other than via CAD:

- Virtual Term Substitution: more efficient but degree limitations. Implementations in MATHEMATICA and REDLOG.
- QE via Comprehensive Gröbner Bases: more efficient, especially if there are many equations. Implementation in SYNNRAC package for MAPLE.
- Family of algorithms for Real QE with complexity doubly exponential in the number of quantifier eliminations. No implementations and analysis suggests that the crossover point where the asymptomatic growth becomes relevant is to far away for practical use at the moment.
- Variety of specialist Real QE algorithms for input of certain shape.

Alternatives to CAD for QE

There exists algorithms to achieve Real QE other than via CAD:

- Virtual Term Substitution: more efficient but degree limitations. Implementations in MATHEMATICA and REDLOG.
- QE via Comprehensive Gröbner Bases: more efficient, especially if there are many equations. Implementation in SYNNRAC package for MAPLE.
- Family of algorithms for Real QE with complexity doubly exponential in the number of quantifier eliminations. No implementations and analysis suggests that the crossover point where the asymptomatic growth becomes relevant is to far away for practical use at the moment.
- Variety of specialist Real QE algorithms for input of certain shape.

Alternatives to CAD for QE References

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, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 220–241. Springer-Verlag, 1998.

H.P. Le and M. Safey El Din.

Faster one block quantifier elimination for regular polynomial systems of equations.

In Proc. ISSAC '21, pages 265–272. ACM, 2021.

Outline

Real QE Applications

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

- derivation of optimal numerical schemes.
- artificial intelligence to pass university entrance exam.
- automated theorem proving.
- automated loop parellisation.
- structural design (minimising the weight of trusses).
- ⋮

There is no lack of potential applications: the problem is scaling up in the face of high complexity!

QE Applications References


M. Erascu and H. Hong.
Real quantifier elimination for the synthesis of optimal numerical algorithms (Case study: Square root computation).
Journal of Symbolic Computation, 75:110–126, 2016.


N.H. Arai, T. Matsuzaki, H. Iwane, and H. Anai.
Mathematics by machine.
In Proc. ISSAC '14, pages 1–8. ACM, 2014.


L.C. Paulson.
Metitarski: Past and future.
In Proc ITP '12, pages 1–10. Springer, 2012.


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


A.E. Charalampakis and I. Chatzigiannelis.
Analytical solutions for the minimum weight design of trusses by cylindrical algebraic decomposition.
Archive of Applied Mechanics, 88(1):39–49, 2018.

Chemical Reaction Networks

A **Chemical Reaction Network** (CRN) models the behaviour of a chemical system. Toy Example:

$$2X_1 + X_2 \xrightarrow{k_1} 3X_1 \xrightarrow{k_2} X_1 + 2X_2 \xrightarrow{k_3} 3X_2 \ .$$
$$\xleftarrow[\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad]{k_4}$$

From this we define a dynamical system:

$$\begin{cases} \dot{x_1} = (3-2)k_1 x_1 x_1 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3, \\ \dot{x_2} = -k_1 x_1^2 x_2 + 2k_2 x_1^3 + k_3 x_1 x_2^2 - 2k_4 x_2^3. \end{cases}$$

Steady states when all the derivatives are zero:

$$\begin{cases} k_1 x_1^2 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3 = 0, \\ x_1 + x_2 - k_5 = 0. \end{cases}$$

Parametric real semi-algebraic geometry problem.

Chemical Reaction Networks

A **Chemical Reaction Network** (CRN) models the behaviour of a chemical system. Toy Example:

$$2X_1 + X_2 \xrightarrow{k_1} 3X_1 \xrightarrow{k_2} X_1 + 2X_2 \xrightarrow{k_3} 3X_2 \ .$$
$$\xleftarrow[k_4]{}$$

From this we define a dynamical system:

$$\begin{cases} \dot{x_1} = (3-2)k_1 x_1 x_1 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3, \\ \dot{x_2} = -k_1 x_1^2 x_2 + 2k_2 x_1^3 + k_3 x_1 x_2^2 - 2k_4 x_2^3. \end{cases}$$

Steady states when all the derivatives are zero:

$$\begin{cases} k_1 x_1^2 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3 = 0, \\ x_1 + x_2 - k_5 = 0. \end{cases}$$

Parametric real semi-algebraic geometry problem.

Chemical Reaction Networks

A **Chemical Reaction Network** (CRN) models the behaviour of a chemical system. Toy Example:

$$2X_1 + X_2 \xrightarrow{k_1} 3X_1 \xrightarrow{k_2} X_1 + 2X_2 \xrightarrow{k_3} 3X_2 \ .$$
$$\xleftarrow[k_4]{}$$

From this we define a dynamical system:

$$\begin{cases} \dot{x_1} = (3-2)k_1 x_1 x_1 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3, \\ \dot{x_2} = -k_1 x_1^2 x_2 + 2k_2 x_1^3 + k_3 x_1 x_2^2 - 2k_4 x_2^3. \end{cases}$$

Steady states when all the derivatives are zero:

$$\begin{cases} k_1 x_1^2 x_2 - 2k_2 x_1^3 - k_3 x_1 x_2^2 + 2k_4 x_2^3 = 0, \\ x_1 + x_2 - k_5 = 0. \end{cases}$$

Parametric real semi-algebraic geometry problem.

Multistationarity

A CRN exhibits **multistationarity** if there exists a choice of parameter values for which the system has more than one (positive real) solution.

Why care about multistationarity?

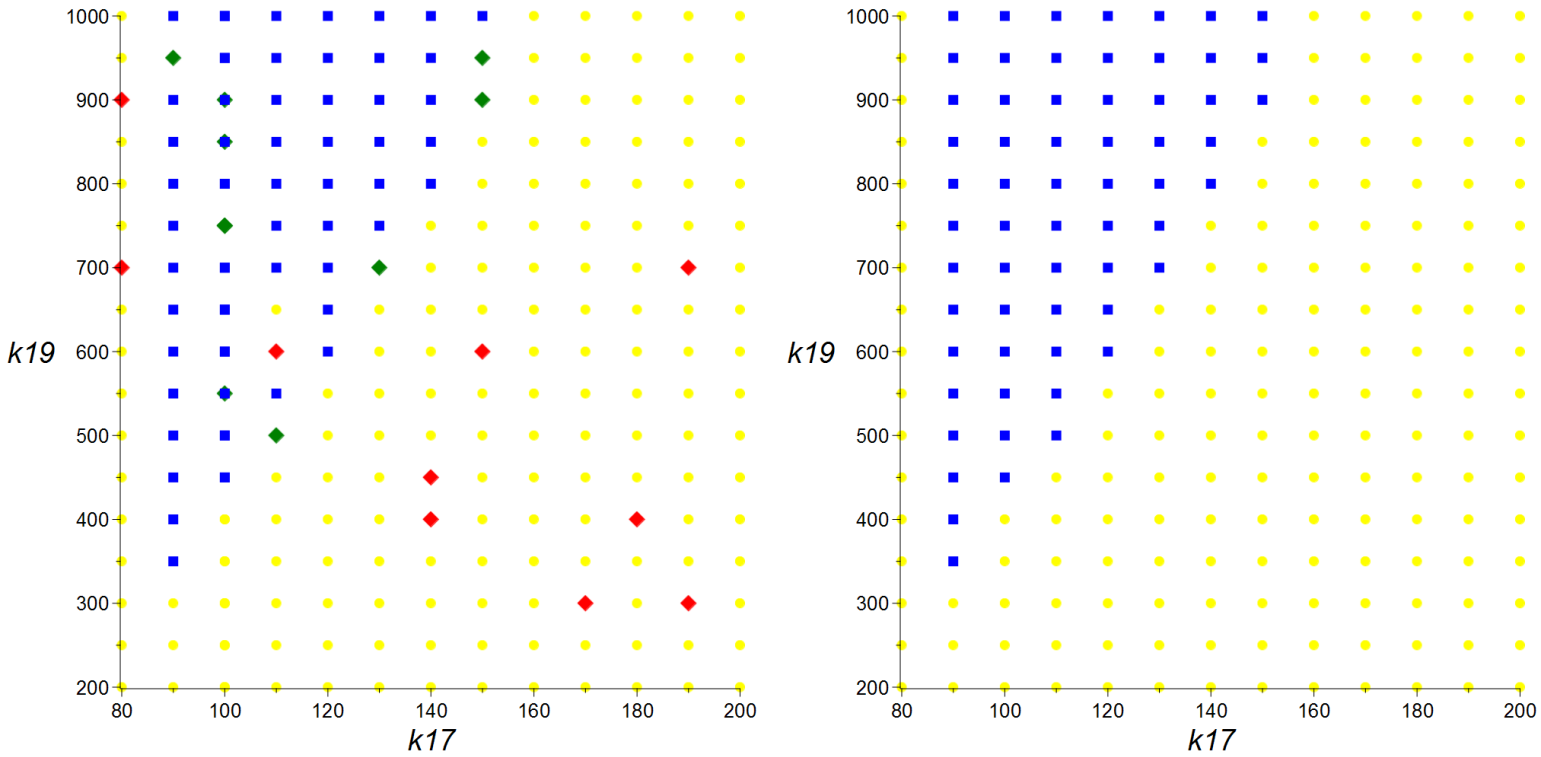
- Used for switch behaviour.
- Instrumental to cellular memory and cell differentiation during development.
- Used by micro organisms in survival strategies.
- Used in decision making processes in the cell division cycle.

There are a variety of efficient methods for answering the Boolean question as to whether or not a system can exhibit multistationarity. Less studied is the question of determining the actual parameter values where multistationarity occurs.

Numeric Methods for Multistationarity 1 (Slide 19/43)

Numeric sampling of the parameter space is used. However:

- Incorrect results can be obtained at ill-conditioned points.
- No guarantee all areas of interest will be sampled.



Numeric Methods for Multistationarity 2 (Slide 20/43)

Numeric sampling of the parameter space is used. However:

- Incorrect results can be obtained at ill-conditioned points.
- No guarantee all areas of interest will be sampled.

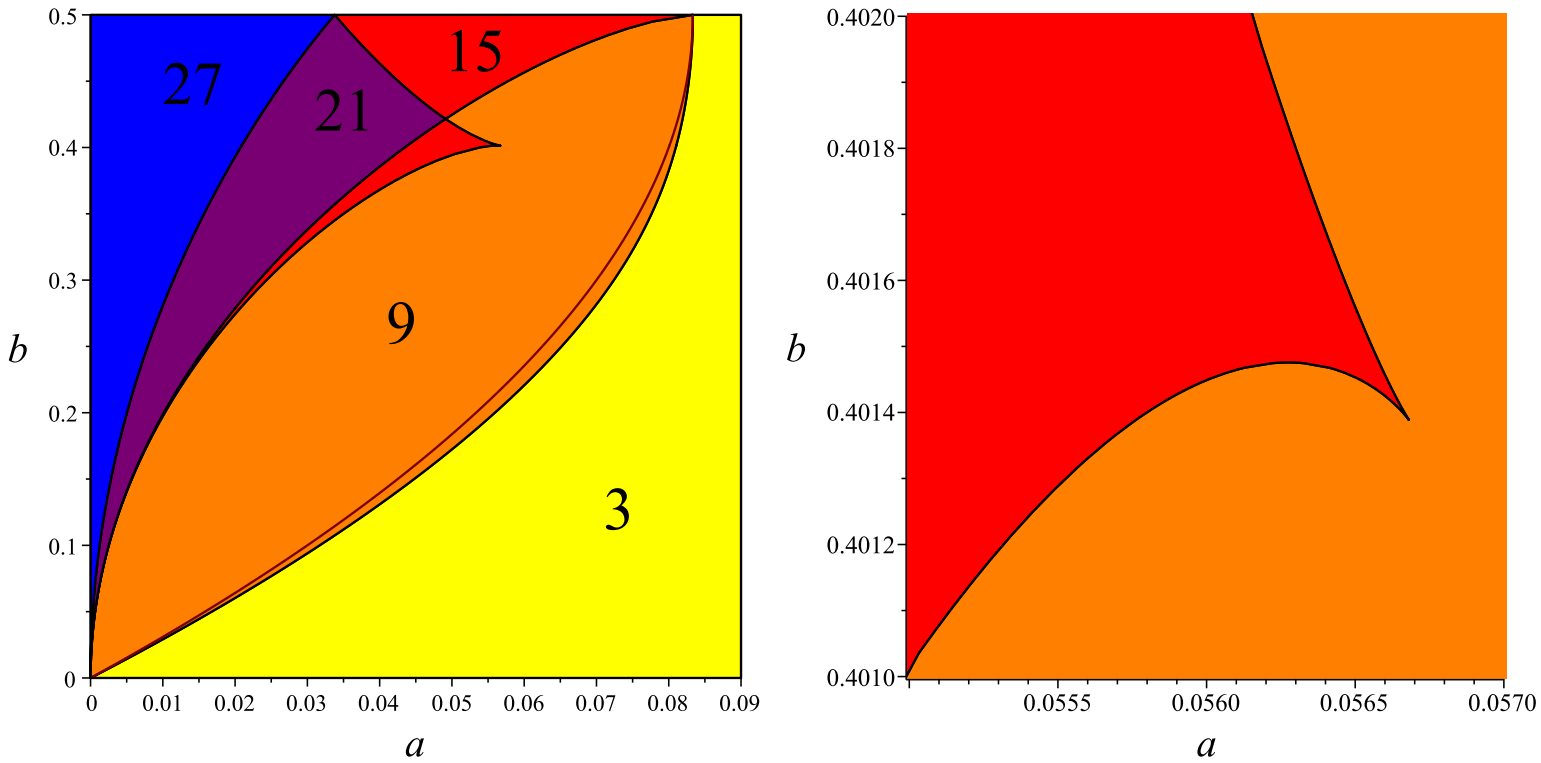


Image courtesy of AmirHosein SadeghiManesh.

Biology Example References

R. Bradford, J.H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Košta, O. Radulescu, T. Sturm, and A. Weber.

Identifying the parametric occurrence of multiple steady states for some biological networks.

Journal of Symbolic Computation, 98:84–119, 2020.

D. Lazard and F. Rouillier.

Solving parametric polynomial systems.

Journal of Symbolic Computation, 42(6):636–667, 2007.

G. Röst and A. Sadeghimanesh.

Exotic bifurcations in three connected populations with Allee effects.

International Journal of Bifurcation and Chaos, 31(13):2150202, 2021.

A. Sadeghimanesh and M. England.

Polynomial Superlevel Set Representation of the Multistationarity Region of Chemical Reaction Networks.

BMC Bioinformatics, 23 Article number 391, 26 pages, 2022.

Framework for Reasoning in Economics (Slide 21/43)

Determine whether, with variables $\Lambda = (v_1, \ldots, v_n)$, the hypotheses $H(\Lambda)$ follow from the assumptions $A(\Lambda)$. I.e. answer

$$\forall \Lambda \,.\, A(\Lambda) \Rightarrow H(\Lambda)?$$

Logically the answer must be `True` or `False`. But economists are interested also in the following:

- Are the assumptions themselves contradictory?
- If `False`, can additional assumptions be made to give `True`?
- If `True`, can any assumptions be removed?

Such questions can be answered by Real QE in many cases.

TheoryGuru Framework

TheoryGuru package for Mathematica led by Casey Milligan (Chicago). For a proposed economics theorem, we check both:

- the existence of an example
  $\exists \Lambda, A \wedge H$,
- and the existence of a counterexample
  $\exists \Lambda, A \wedge \neg H$.

Then categorize the proposed theorem as:

	$\neg\exists \Lambda, A \wedge \neg H$	$\exists \Lambda, A \wedge \neg H$
$\exists \Lambda, A \wedge H$	`True`	`Mixed`
$\neg\exists \Lambda, A \wedge H$	Contradictory Assumptions	`False`

An economist can explore: e.g. strengthen assumptions of `Mixed` result, or weaken assumptions of `True` result.

Example #0013 Context

Context: Disagreement between Mulligan and Krugman on causes of recession. In particular, latter asserted that whenever taxes on labour supply are primarily responsible for a recession then wages increase. Benchmark Example #0013 considers this claim.

Two scenarios to track: what actually happens (act) when taxes (t) and demand forces (a) together create a recession, and what would have happened (hyp) if taxes on labour supply had been the only factor affecting the market.

The labour demand and supply functions $D(w, a)$ and $S(w, t)$ meet at the labour market equilibrium to supply quantity of labour q.

Example #0013 Context

Context: Disagreement between Mulligan and Krugman on causes of recession. In particular, latter asserted that whenever taxes on labour supply are primarily responsible for a recession then wages increase. Benchmark Example #0013 considers this claim.

Two scenarios to track: what actually happens (act) when taxes (t) and demand forces (a) together create a recession, and what would have happened (hyp) if taxes on labour supply had been the only factor affecting the market.

The labour demand and supply functions $D(w, a)$ and $S(w, t)$ meet at the labour market equilibrium to supply quantity of labour q.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0$$
$$\wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1$$
$$\wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big)$$
$$\wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big)$$
$$\wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0$$
$$\wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0$$
$$\wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1$$
$$\wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big)$$
$$\wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big)$$
$$\wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0$$
$$\wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\left(D(w,a) = q = S(w,t)\right) \wedge \frac{d}{dhyp}\left(D(w,a) = q = S(w,t)\right) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Logic

$$A \equiv \frac{\partial D(w,a)}{\partial w} < 0 \wedge \frac{\partial S(w,t)}{\partial w} > 0 \wedge \frac{\partial D(w,a)}{\partial a} = 1 \wedge \frac{\partial S(w,t)}{\partial t} = 1 \wedge \frac{d}{dact}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{d}{dhyp}\big(D(w,a) = q = S(w,t)\big) \wedge \frac{dt}{dact} = \frac{dt}{dhyp} \wedge \frac{da}{dhyp} = 0 \wedge \frac{dq}{dhyp} < \frac{1}{2}\frac{dq}{dact} < 0$$

$$H \equiv \frac{dw}{dact} > 0.$$

Assumptions:

- Usual slope restrictions.
- Standard normalizations.
- Scenarios both move labour market equilibrium over course of recession.
- Scenarios have the same tax change but only the *act* scenario has a demand shift.
- Majority of the reduction in labour was due to supply.

Hypothesis: wages are higher at the end of the recession.

Example #0013 Result

We may view this as a Tarski formula in 12 variables,

$$\Lambda = \left\{ \frac{da}{dact}, \frac{da}{dhyp}, \frac{dt}{dact}, \frac{dt}{dhyp}, \right.$$
$$\frac{dq}{dact}, \frac{dq}{dhyp}, \frac{dw}{dact}, \frac{dw}{dhyp},$$
$$\left. \frac{\partial D(w,a)}{\partial a}, \frac{\partial S(w,t)}{\partial t}, \frac{\partial D(w,a)}{\partial w}, \frac{\partial S(w,t)}{\partial w} \right\},$$

Each is representing a partial derivative describing the supply and demand function or a total derivative indicating a change over time within a scenario.
Evaluating the two existential problems shows that both examples and counterexamples exist: the theorem is not universally true.

Example #0013 Exploration

If we leave $\frac{\partial D(w,a)}{\partial w}$ and $\frac{\partial S(w,t)}{\partial w}$ as free variables then QE recovers a disjunction of three quantifier-free formulae. Two of them contradict the assumptions, but the third, below, could be added to A to guarantee the truth of H.

$$\frac{\partial S(w,t)}{\partial w} \geq -\frac{\partial D(w,a)}{\partial w} > 0.$$

The added assumption states that labour supply is at least as sensitive to wages as labour demand.

The algebra and logical reasoning above is not political! Any politics comes with whether the assumptions hold. We cannot remove the politics, but at least gain clarity over which arguments are political and which not.

Economics Example References

C. Mulligan, R. Bradford, J.H. Davenport, M. England, and Z. Tonks.
Non-linear real arithmetic benchmarks derived from automated reasoning in economics.
In Proc. SC^2 '1), CEUR-WS 2189, pages 48–60, 2018.
URL: http://ceur-ws.org/Vol-2189/.

C.B. Mulligan, J.H. Davenport, and M. England.
TheoryGuru: A Mathematica package to apply quantifier elimination technology to economics.
In Proc. ICMS 2018, LNCS 10931, pages 369–378. Springer, 2018.
URL: https://doi.org/10.1007/978-3-319-96418-8_44.

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

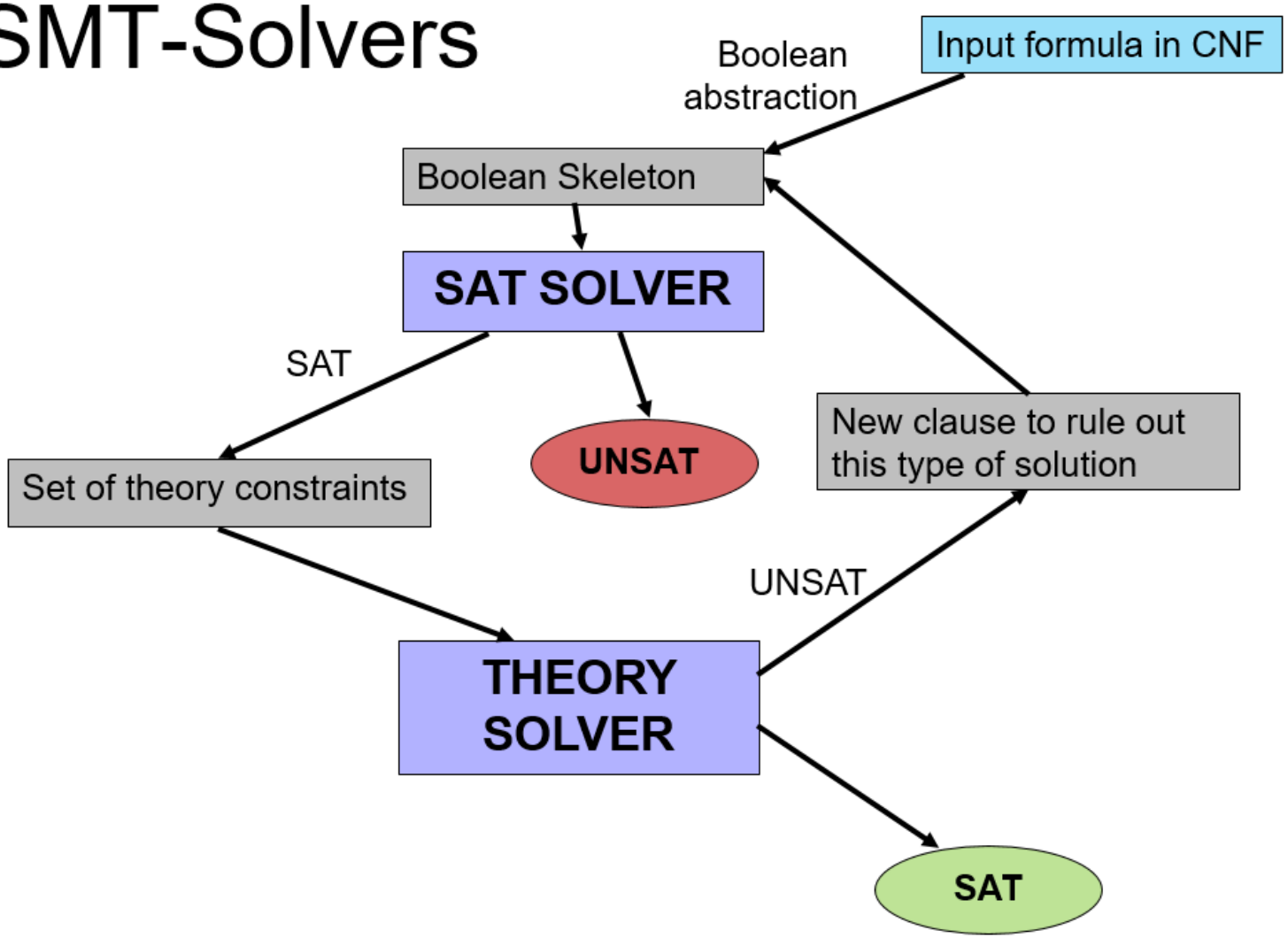
One approach to such satisfiability problems is to separate out the logic from the arithmetic theory.

- 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 involved).

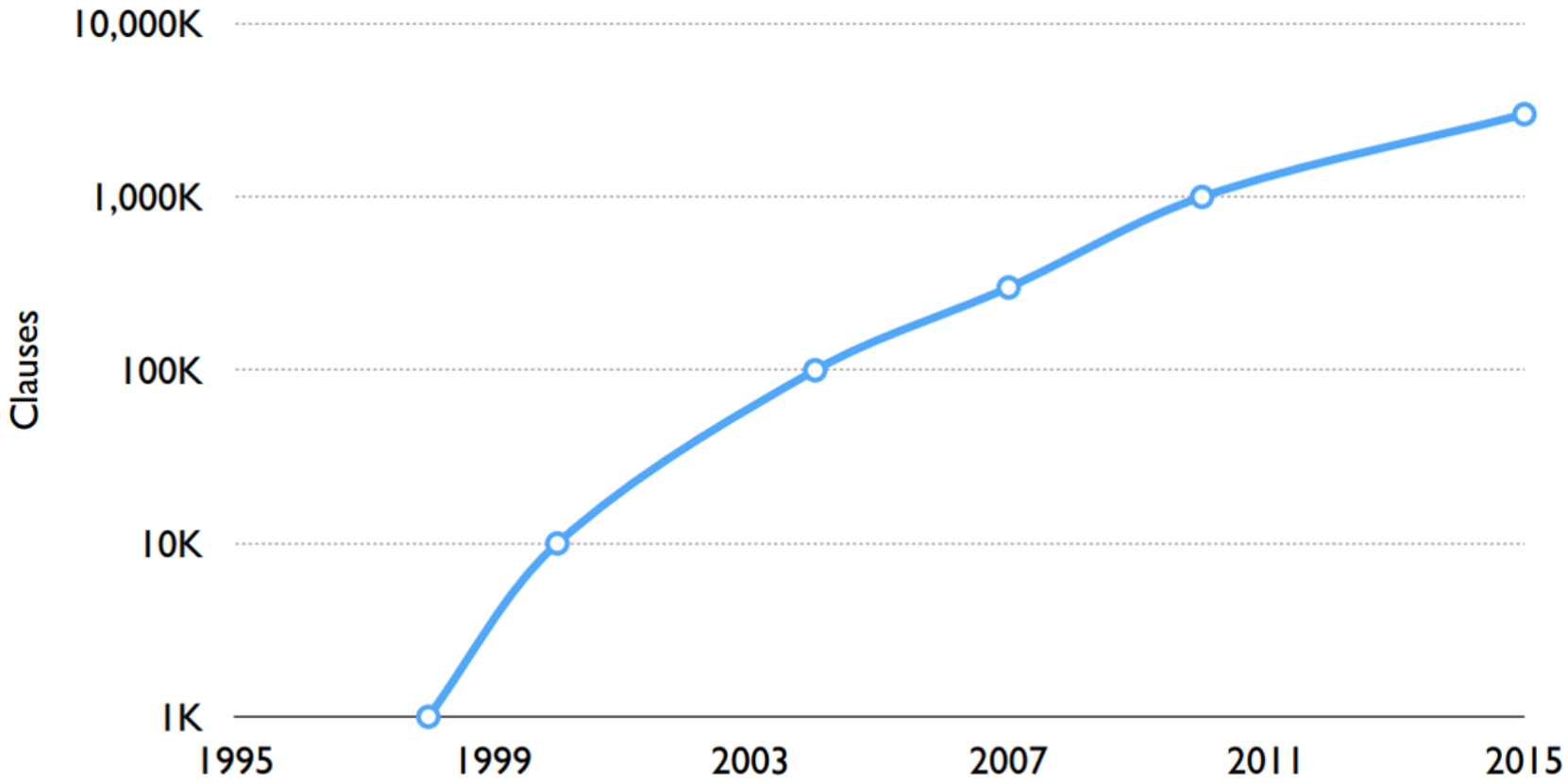
This approach is called **Satisfiability Modulo Theories** (SMT) or sometimes CDCL(T).

SMT-Solvers


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

Why the SMT Approach?

To take advantage of the incredible progress in SAT solvers!



Based on a slide from Vijay Ganesh

Why are SAT solvers so good? (Slide 31/43)

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; heuristics for non-critical choices; ML meta-solvers; ML for interior heuristics, 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 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.

(Q) How to adapt CAD further to avoid this?
(A) Try to follow the 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

Conflict Driven Cylindrical Algebraic Covering (Slide 34/43)

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

Features:

- Search based: choose sample point and if not satisfying build cell around it using CAD technology.
- 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 away from past conflicts.

CDCAC: Basic Idea

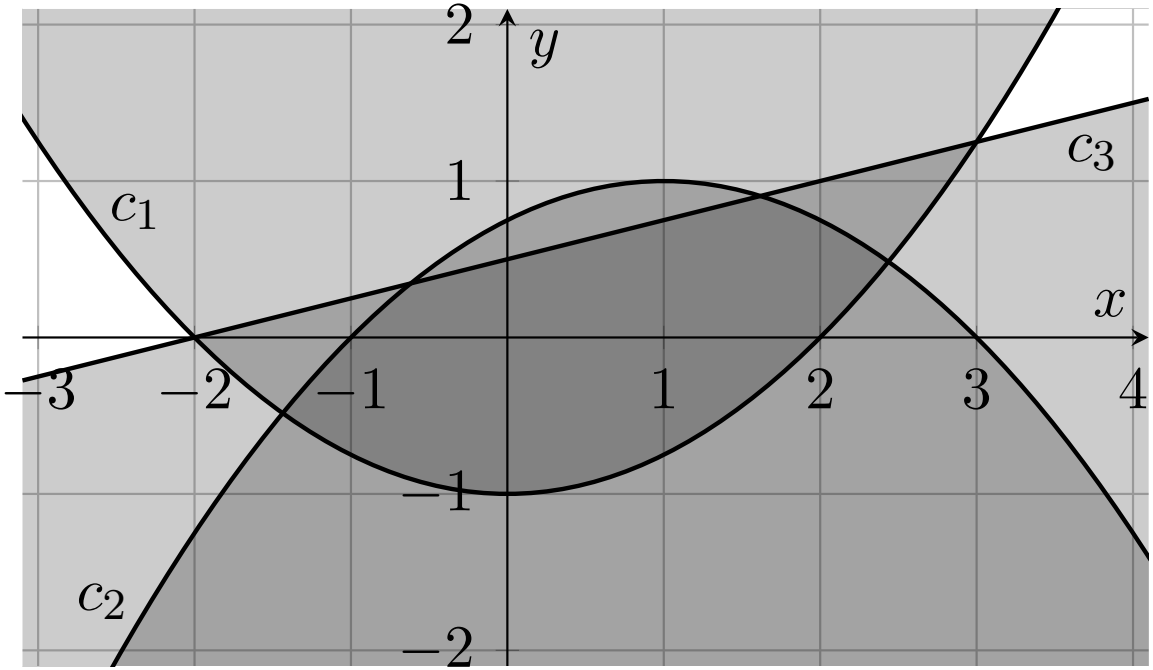
- Pick sample for lowest variable in ordering.
- Extend to increasingly higher dimensions in reference to those constraints made univariate.
- If all constraints satisfied then conclude SAT.
- If a constraint cannot be satisfied generalise to CAD cell in current dimension.
- Search outside the cell in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

Following slides by Gereon Kremer show simple example.

RWTH AACHEN UNIVERSITY

An example

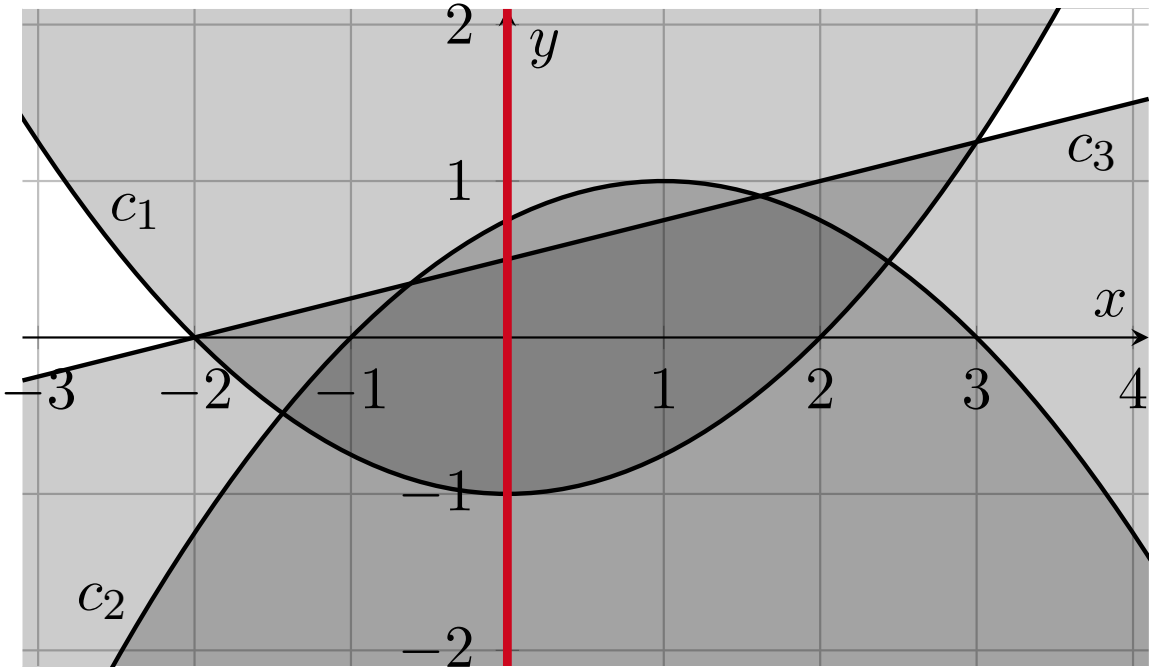
$c_1 : 4 \cdot y < x^2 - 4$ $\quad$ $c_2 : 4 \cdot y > 4 - (x-1)^2$ $\quad$ $c_3 : 4 \cdot y > x + 2$



RWTH AACHEN UNIVERSITY

An example

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



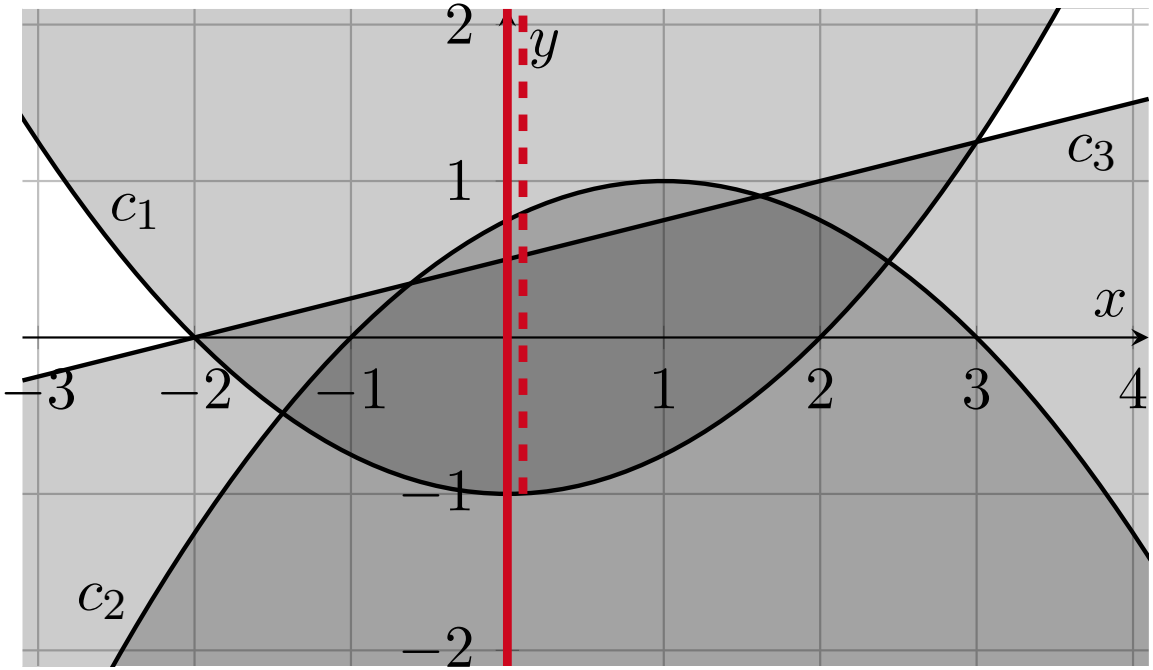
No constraint for x

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

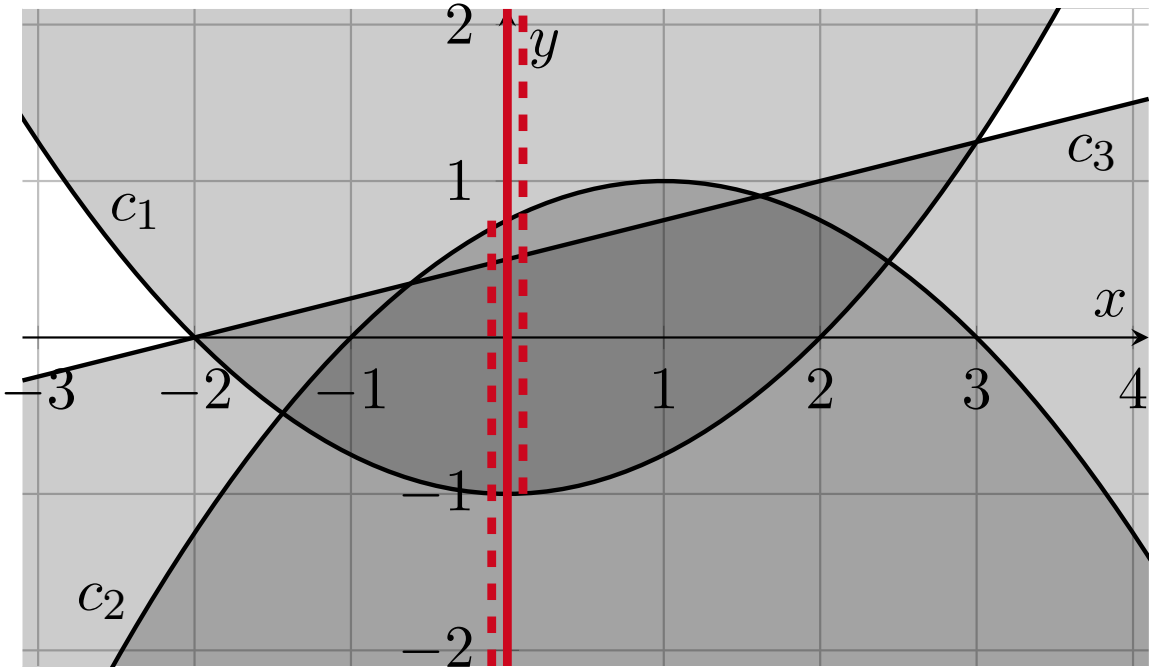
Guess $x \mapsto 0$

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

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 $x \mapsto 0$

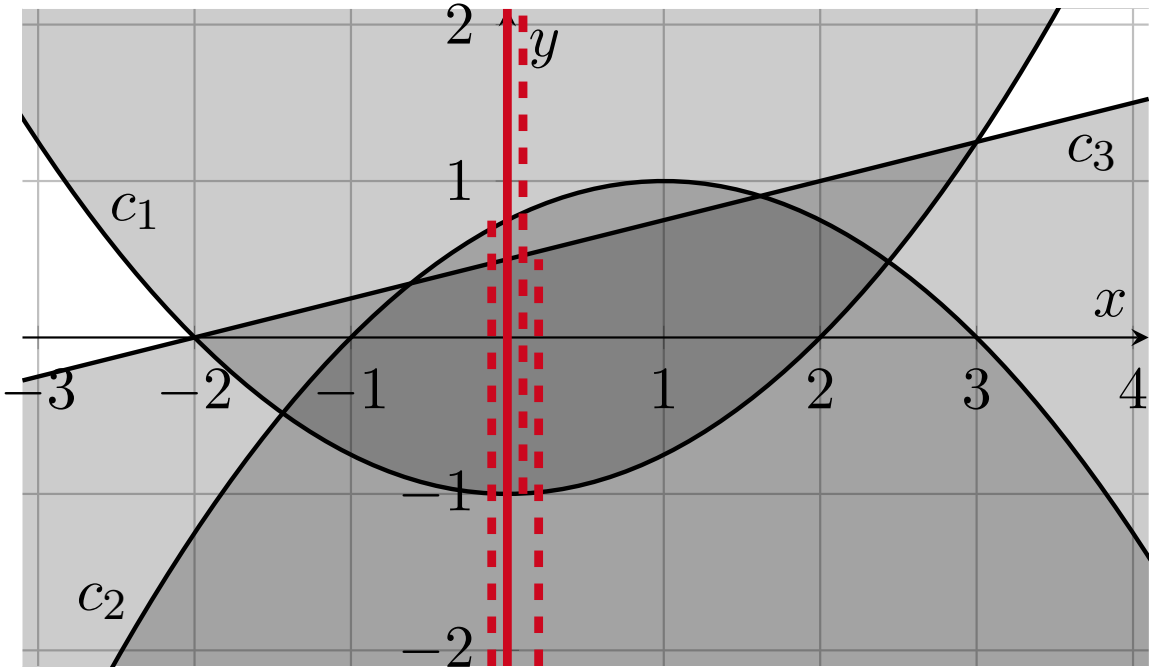
$c_1 \to y \notin (-1, \infty)$

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

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 $x \mapsto 0$

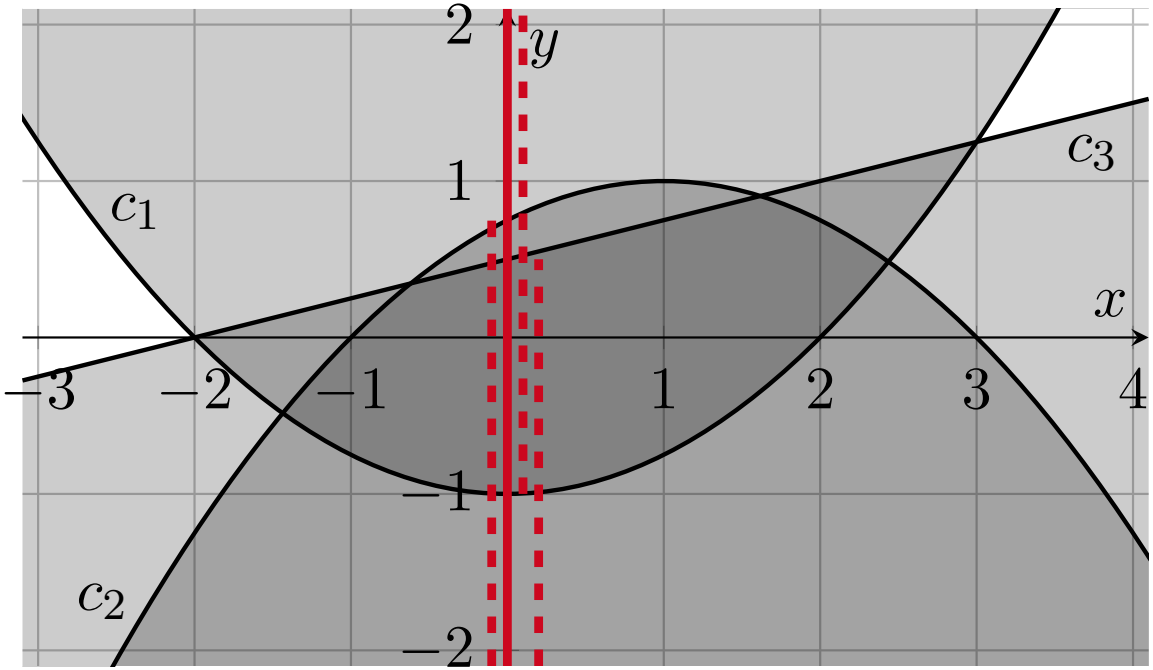
$c_1 \to y \notin (-1, \infty)$

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

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

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 $x \mapsto 0$

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

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

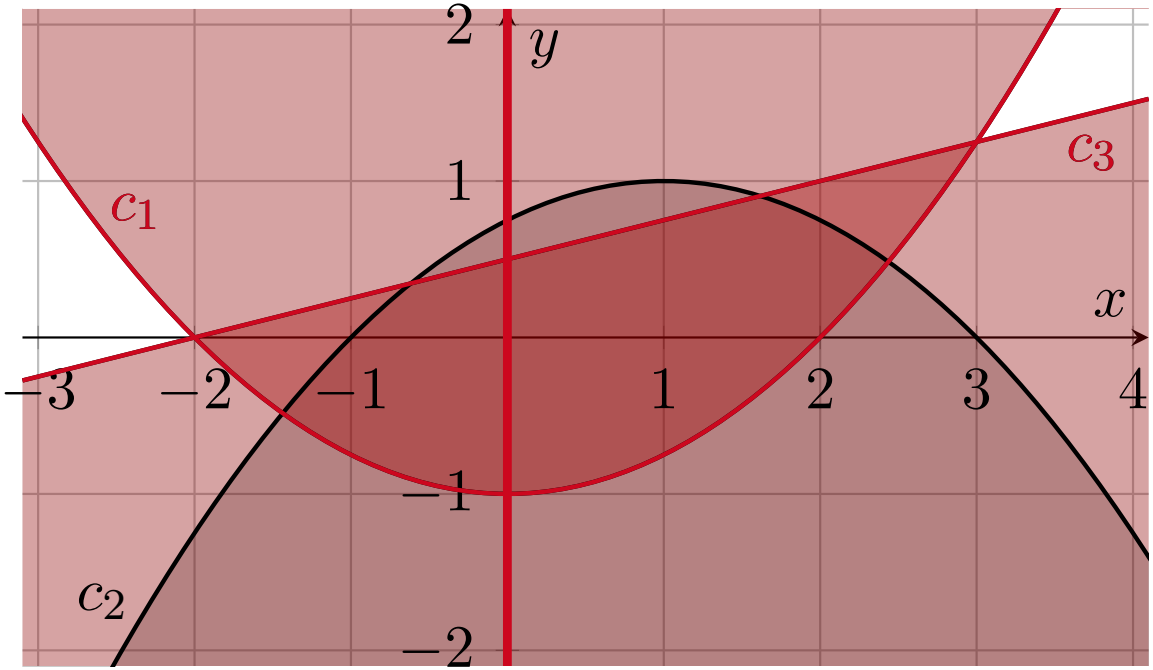
$c_3 \to y \notin (-\infty, 0.5)$

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 $x \mapsto 0$

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

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

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

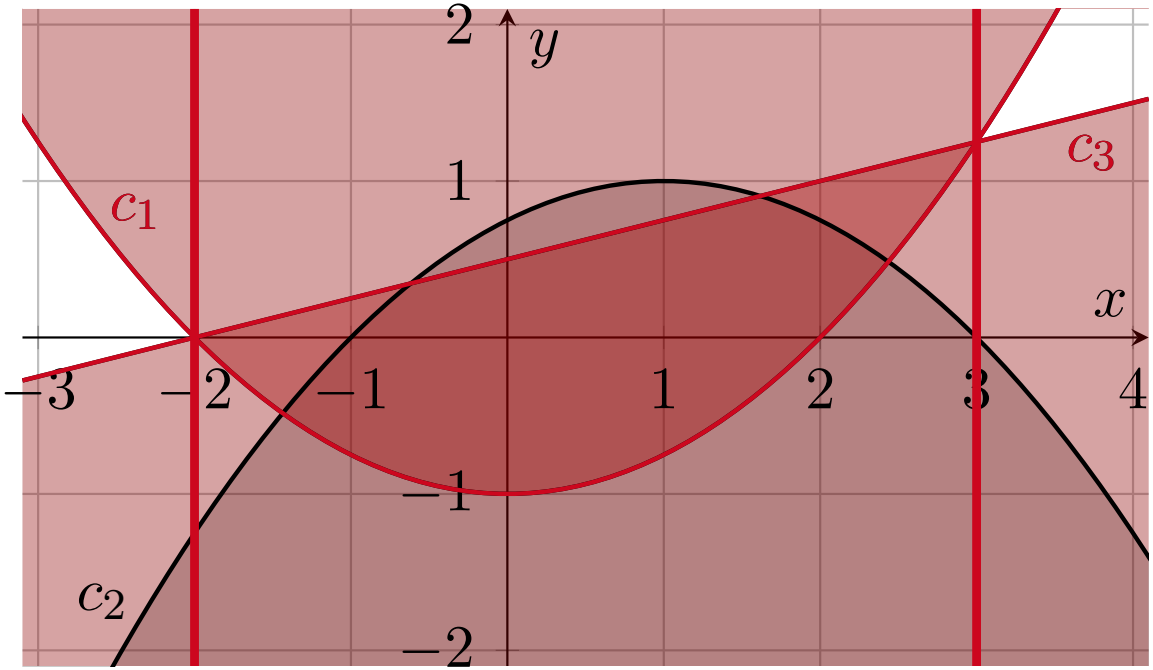
Construct covering

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

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

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

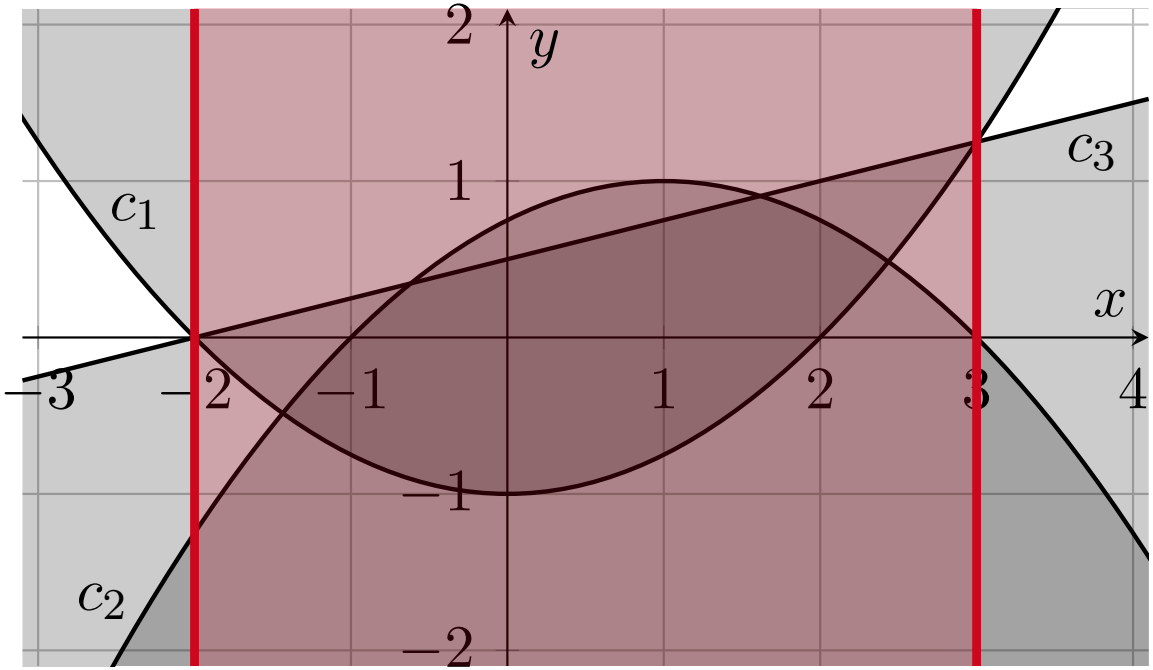
Construct interval for x

$x \notin (-2, 3)$

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 in cvc5

cvc5 is a popular open-source SMT-solver used by prominently in academia and industry (e.g. Amazon Web Services). It won the 2022 SMT Competition overall, and the QFNRA track. cvc5 used CDCAC as the core algorithm for non-linear real arithmetic.

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), pages 95-105. Springer International Publishing, 2022.

Outline

Certificates and Verification

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

But in the case of unsatisfiability, one would need to verify that:
(a) the cells form a decomposition / covering (not too hard given the cylindrical structure); and
(b) each cells satisfies the stated invariance property (difficult).

An attempt was made to formalise CAD in Coq by Cohen and Maboubi but this was never completed.

C. Cohen and A. Mahboubi.

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

Logical Methods in Computer Science, 8(1):1–40, 2012.
https://doi.org/10.2168/LMCS-8(1:2)2012

Proofs for SMT Verification

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

H. Barbosa et al.

Flexible Proof Production in an Industrial-Strength SMT Solver.

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

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

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

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

New Opportunities for the Formal Proof of Computational Real Geometry?

Proc. SC^2 2020, CEUR-WS 2752, pages 178–188, 2020.

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

Proof System for cylindrical cell

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

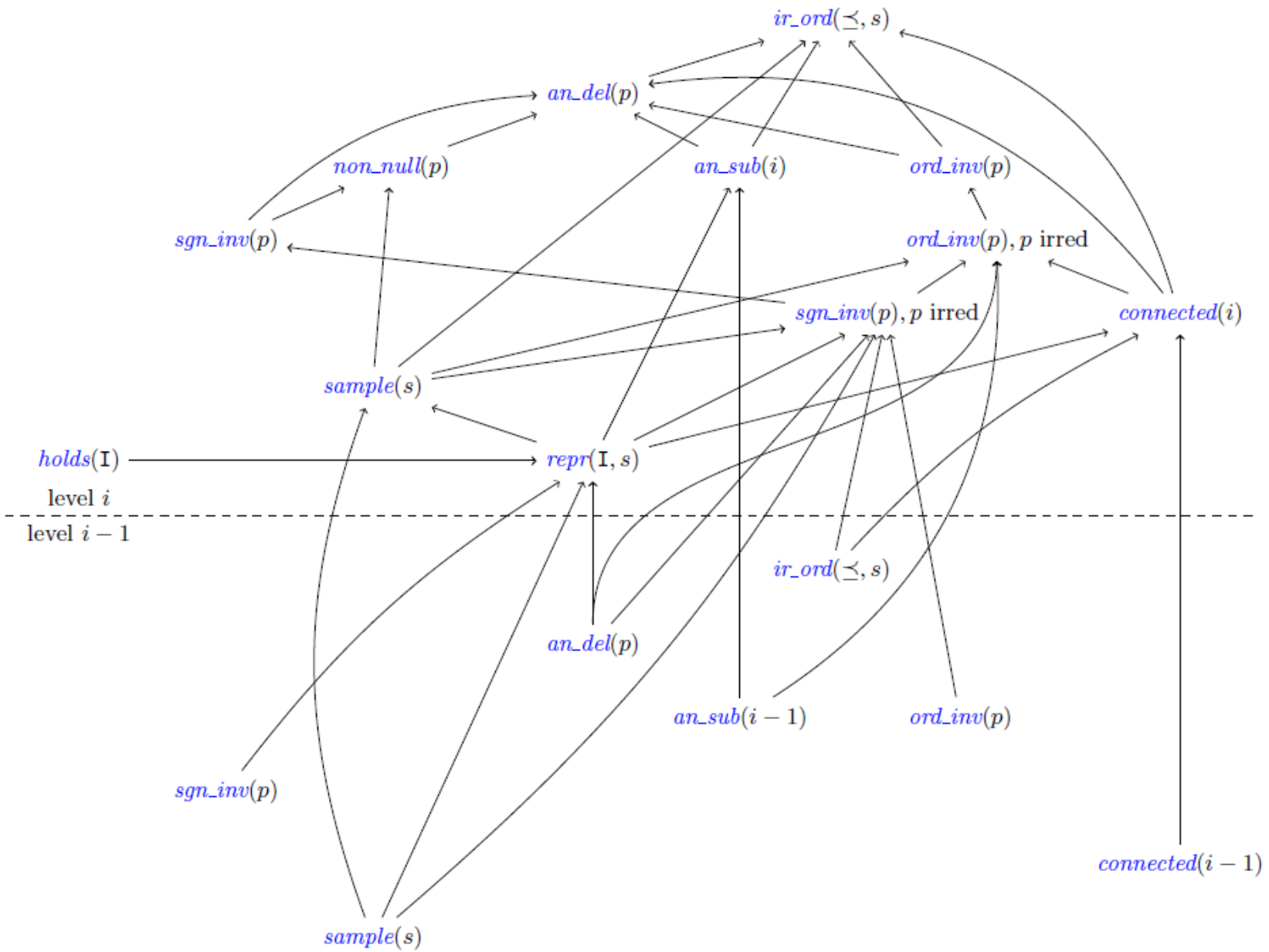
The algorithm then simply searches for a chain of proof rules to prove a desired property (employing heuristics to take choices where there is freedom in how the chain can to be built).

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

Levelwise construction of a single cylindrical algebraic cell.

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 Presentations

Unlike traditional pseudo-code, a proof system presentation clearly separates out the parts of the algorithm upon which correctness rests, from those parts where heuristic decisions may be taken. It :

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

Such presentation is prevalent 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

Geometric CAD

Rizeng Chen.

Geometric Fiber Classification of Morphisms and a Geometric Approach to Cylindrical Algebraic Decomposition.

Preprint, arXiv:2311.10515 (Dec 2023).
https://doi.org/10.48550/arXiv.2311.10515

Gives an algebraic geometry study of CAD. Leads to an alternative algorithm relying on GB calculations instead of the resultant calculations of traditional CAD. Iterated resultants known to capture many spurious solutions compared to GB; so this is potentially much faster!

Challenges: check validity; extend 50 years of CAD optimisations to this new framework?

Block CAD and Block QE

CAD (both traditional and geometric) works incrementally one variable at a time. Can we instead work in blocks: projecting and lifting multiple dimensions at a time? E.g. when we have multiple equational constraints?.

S. McCallum and C.W. Brown.

On delineability of varieties in CAD-based quantifier elimination with two equational constraints.

In Proceedings ISSAC 2009, pages 71-78, 2009.
https://doi.org/10.1145/1576702.1576715

J.H. Davenport, M. England, S. McCallum, and A.K. Uncu.

Iterated Resultants and Rational Functions in Real Quantifier Elimination.

Submitted, 2024. Preprint: arXiv:2312.16210.
https://doi.org/10.48550/arXiv.2312.16210

Can the geometric lens help with this?

Outline

Summary In One Slide

- The variable ordering of CAD has a huge effect on its practical efficiency.
- Machine learning models may be trained to choose the ordering better than any human-designed heuristic.
- Although naturally a classification problem (choosing one ordering from a discrete set of possibilities) performance can be improved by recasting as a regression problem (predicting the time of an ordering and going with the lowest prediction).
- Explainable AI techniques can allow us to analyse the output of such ML models and form new *human-level* heuristics that run independent of AI.

If you would like to know more...

Advert

I will give another talk at Warwick in the **Computational Algebraic Geometry Research Network** meeting on 21 – 22 March 2024.

I will talk more on the ML details of the work there.

`https://sites.google.com/view/computationalalgebraicgeometry/warwickmeeting2024`

Thanks for Listening

Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/

Symbolic Computation vs Machine Learning (Slide 46/43)

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. E.g. for the solution of differential equations.

ODE solving is well suited because it is cheap to symbolically check the correctness of the answer. This is not the case for most symbolic computation.

ML can only offer probabilistic guidance, but symbolic computation prizes exact results. 99% accuracy is great for image recognition but would not be acceptable for a mathematical proof.

Symbolic Computation vs Machine Learning (Slide 46/43)

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. E.g. for the solution of differential equations.

ODE solving is well suited because it is cheap to symbolically check the correctness of the answer. This is not the case for most symbolic computation.

ML can only offer probabilistic guidance, but symbolic computation prizes exact results. 99% accuracy is great for image recognition but would not be acceptable for a mathematical proof.

Symbolic Computation WITH Machine Learning (Slide 47/43)

ML can be applied to symbolic computation and still ensure exact results; by having it guide existing algorithms rather than replace them entirely.

Symbolic Computation algorithms often come with choices that need to be made, 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).

Such choices are often either left to the user, hard coded by the developer, or made based on a simple heuristic. ML may be able to offer a superior choice.

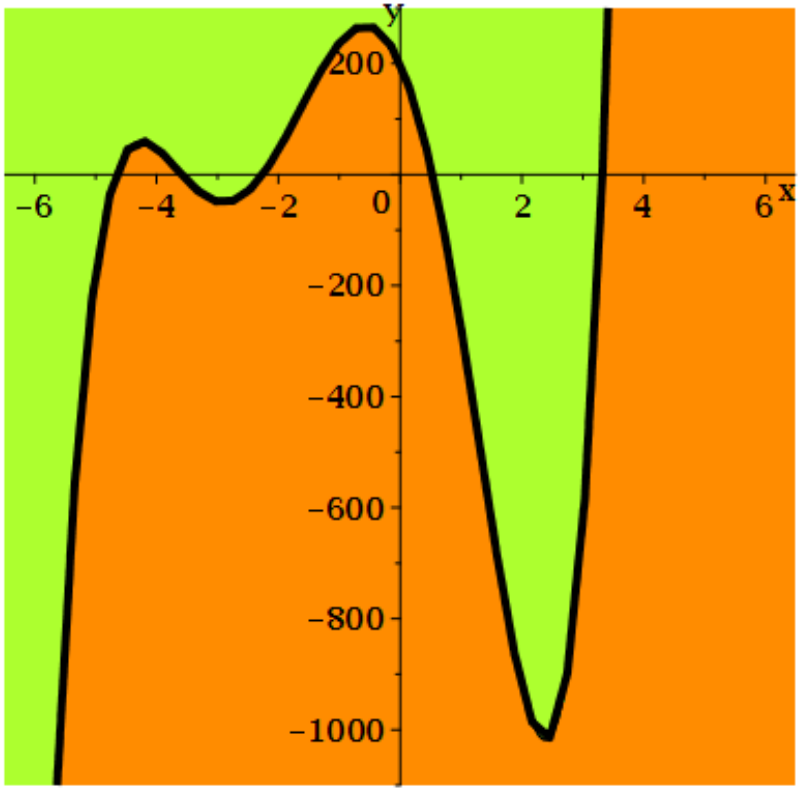
This is INTERESTING Machine Learning (Slide 48/43)

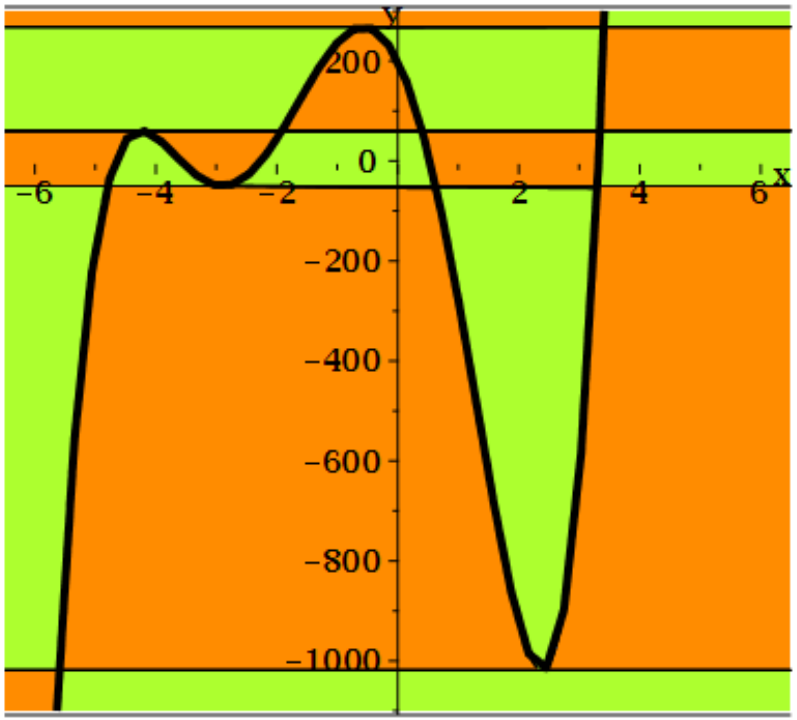
So ML has great potential for Symbolic Computation. But note this is also a particularly challenging / interesting ML domain:

- No a priori limit on the input space.
- Supervised learning hard: because labelling dataset needs lots of expensive symbolic computation.
- Unsupervised learning is hard: because it is 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 to allow for good generalisability on problems of interest?

Example: Variable Ordering for CAD

CAD requires a variable ordering: there can be multiple valid orderings which 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. There are various human-designed heuristics to make the choice:

- Some, e.g. Brown's heuristic, use only simple measures of the polynomials (degrees, sparsity etc.)
- Others, have uses more involved algebraic computations: significantly more expensive but not showing results that much better.

We prefer to focus on the former.

ML for CAD variable ordering literature (Slide 51/43)

Actually the first application of ML for CAS optimisation:

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

EPSRC ML4QE project led to new techniques in feature representation and hyperparameter selection:

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

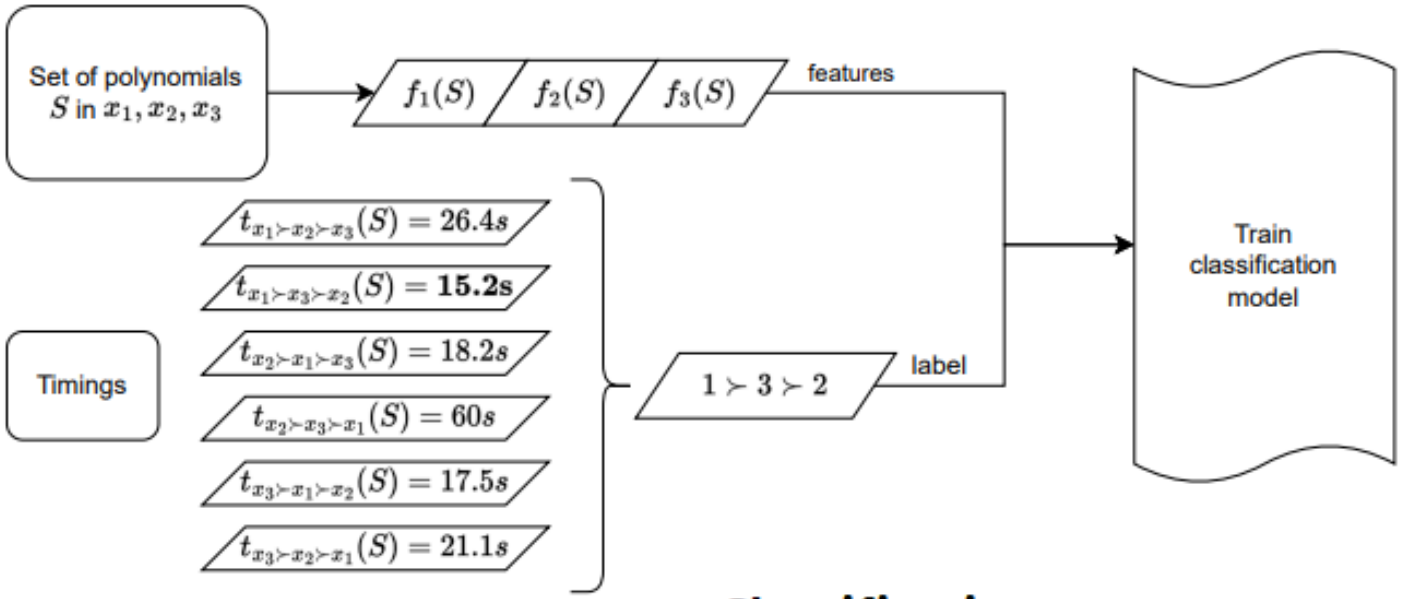
From Classification to Regression

The previous work framed this naturally as a Machine Learning *classification problem* (to choose from the discrete set of variable orderings). Recent work reframed this as a *regression problem* to predict the time taken with a particular ordering (with multiple predictions then compared to choose the final ordering).

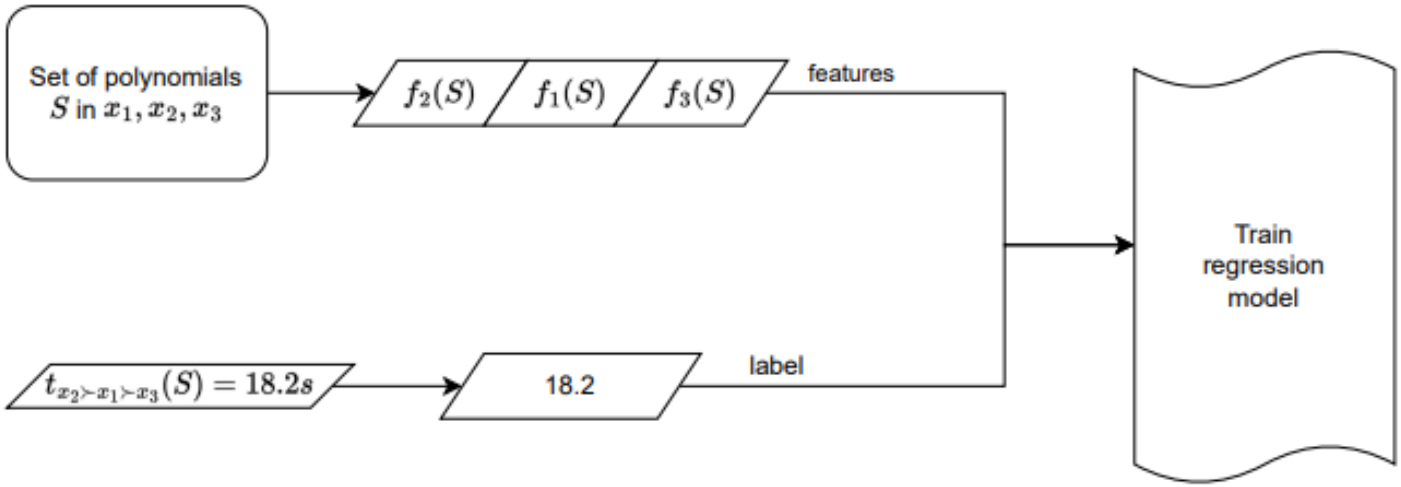
- A model trained with regression has access to more information: not just which ordering did best but also which came second, which third etc.
- Regression is a more difficult task, but we only need to be good enough at it to make a ranking.

So we hypothesised that regression would outperform classification.

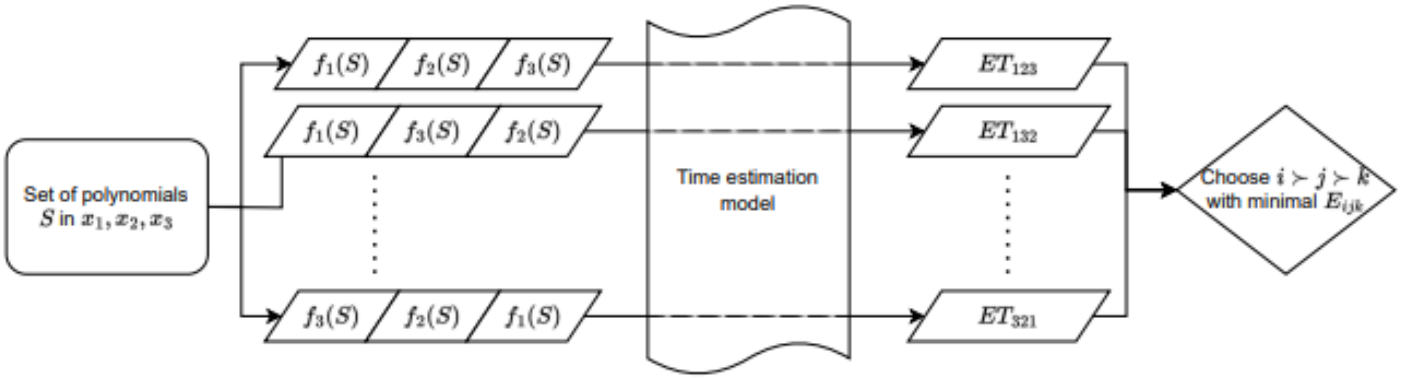
Classification vs. Regression Flowcharts



Classification vs. Regression Flowcharts

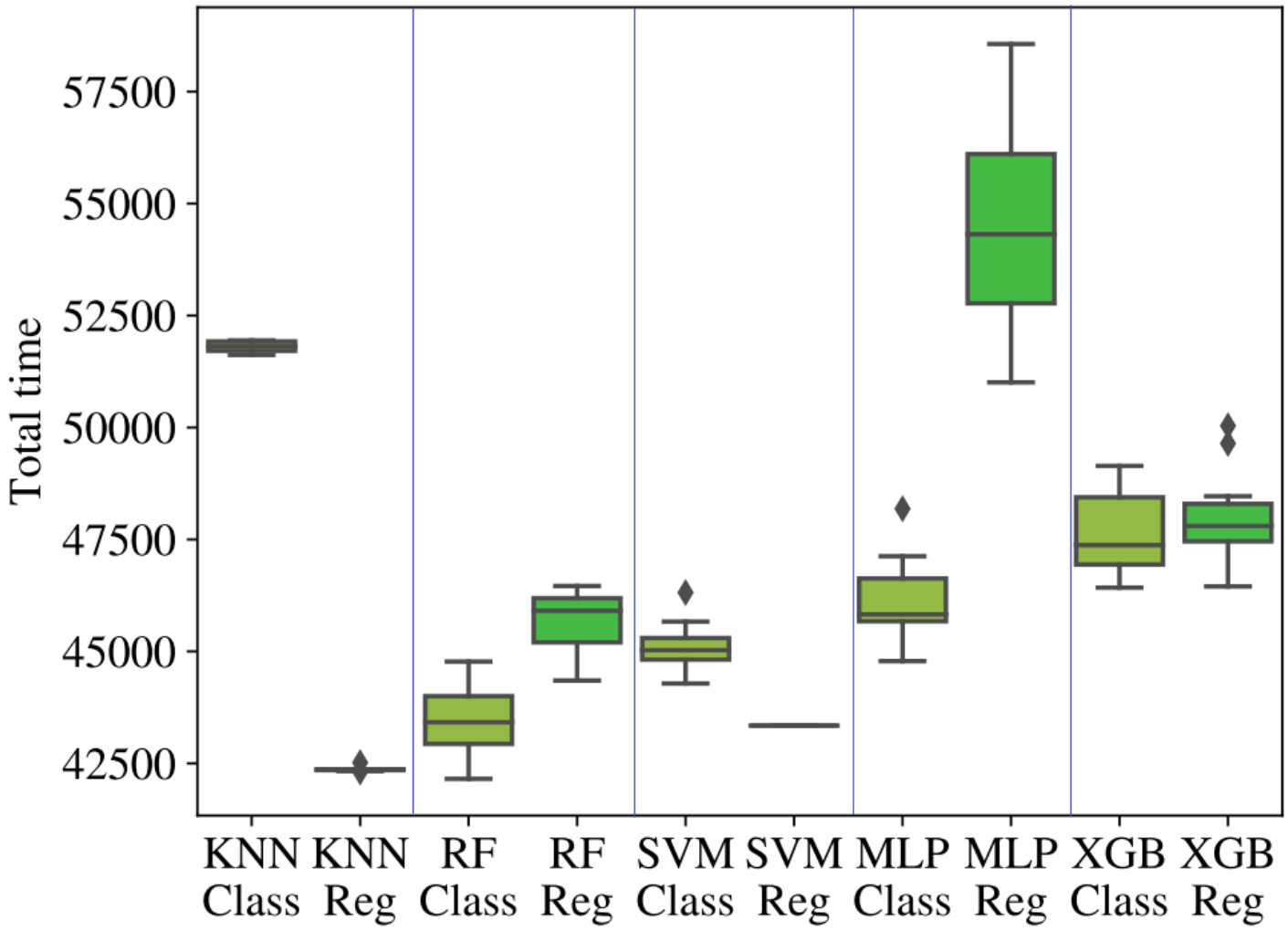


Regression Training
Regression Testing



Classification vs. Regression Results

Not universally beneficial, but new state-of-the-art results.



Recent work on XAI for QE variable ordering (Slide 55/43)

L. Pickering, T. Del Rio 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.

We applied the SHAP XAI tool to analyse the features used in a CAD ML-based variable ordering classifier. Some identified as impactful had been known before but others were new.

Next constructed non-ML heuristics from trio's of these features, in a similar design to prior human-designed heuristics. The best of these outperform's the state-of-the-art on standard benchmarks.

Demonstrates the potential for XAI as an exploratory tool.

Thanks for Listening

Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/