Quantifier Elimination (QE)

12/04/22 Recitation
Katherine Cordwell

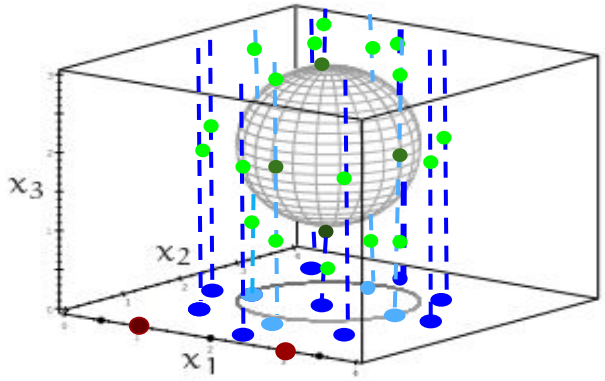


Image credit: Jirstrand (modified)

Background

- Quantifier elimination (QE) is the process that takes a quantified statement and finds an equivalent quantifier-free statement
- For our purposes, all variables are real-valued

Background

- An example with two variables

$$\forall x_1 \exists x_2 \; x_1 + x_2 = 3$$

After quantifier elimination: ?

Background

- An example with two variables

$$\forall x_1 \exists x_2 \; x_1 + x_2 = 3$$

After quantifier elimination: True

Background

- An example with two variables

$$\forall x_1 \exists x_2 \; x_1 + x_2 = 3$$

In this example, all variables are quantified.

Background

- An example with a free variable:

$$\forall x \forall y \, (x^2 > -0.25 \wedge ay^2 \geq 0)$$

After quantifier elimination: ?

Background

- An example with a free variable:

$$\forall x \forall y \, (x^2 > -0.25 \wedge ay^2 \geq 0)$$

After quantifier elimination: $a \geq 0$

Background

- A more complicated example with a free variable (from Pablo Parrilo's lecture notes):

$$\forall x \forall y (x^2 + ay^2 \leq 1) \implies (ax^2 - a^2xy + 2 \geq 0))$$

Background

- A more complicated example with a free variable (from Pablo Parrilo's lecture notes):

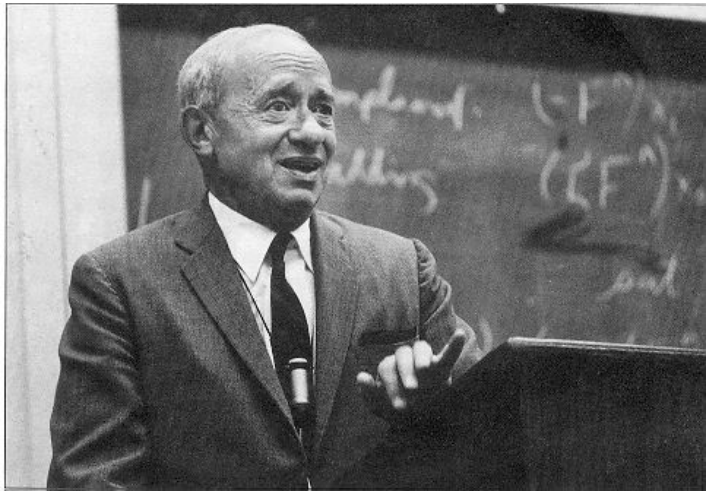
$$\forall x \forall y (x^2 + ay^2 \leq 1) \implies (ax^2 - a^2xy + 2 \geq 0))$$

After quantifier elimination:

$$(a \geq 0) \text{ and } (a^3 - 8a - 16 \leq 0)$$

Real-valued QE is decidable

- Miraculously.
- But the algorithms are complicated.



Alfred Tarski

Image credit: http://www2.cs.unibw.de/Proj/relmics/html/pictures/history/Tarski.html

QE: An abbrev history

- Tarski (1951): Quantifier elimination is decidable (so you can solve it with an algorithm)
- Seidenberg (1954): Another method for quantifier elimination (poor complexity)

QE: An abbrev history

- Tarski (1951), Seidenberg (1954)
- Collins (1975): CAD algorithm (implementable!)
  - McCallum (1984, 1988, 1998): Improved projection phase
  - Brown (2001): Improved projection phase
  - Many other researchers have made contributions!

QE: An abbrev history

- Tarski (1951), Seidenberg (1954) (Early progress)
- Collins (1975), Brown, McCallum, & more (CAD)
- Ben-Or, Kozen, Reif (1986) and others (Use parallelism)

CAD is still the most efficient QE algorithm, though it's doubly exponential in the number of variables

Let’s talk about CAD

The key to CAD

Turn a continuous problem into a discrete one.

The key to CAD

- Say we have some QE query

  $\forall x \forall y (x^2 > -0.25 \wedge ay^2 \geq 0)$

- That means we care about the signs of the polynomials

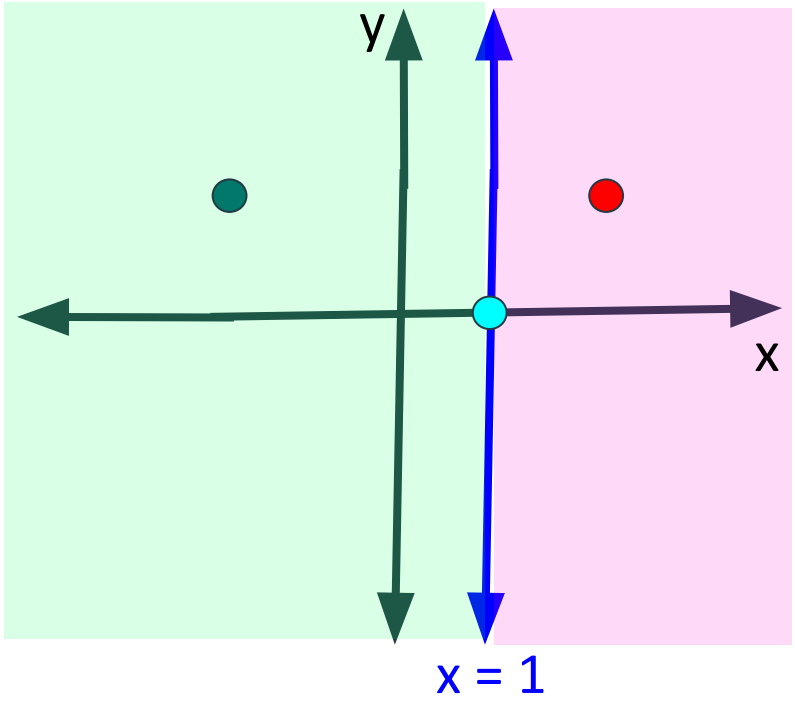
  $x^2 + 0.25$ and ay^2

- We want to turn this problem of finding information about the signs of (continuous) polynomials into a discrete problem

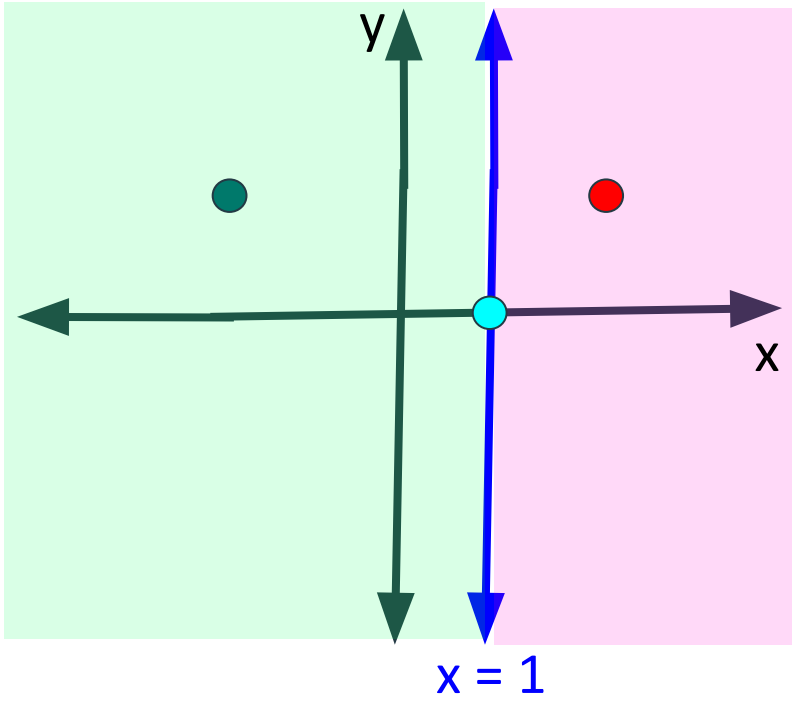
The CAD within CAD

- Start with some set of polynomials (say, in n variables)
- Divide n-space into *sign-invariant* regions: within a region, each polynomial takes only one sign (>, =, or <)
- A CAD for a set of polynomials is a set of sign-invariant regions and a sample point for each region

CAD for f(x, y) = x - 1, g(x, y) = y^2 + 1



CAD for f(x, y) = x - 1, g(x, y) = y^2 + 1



The line x = 1
The region x > 1
The region x < 1

A sample point from each region

What is a CAD?

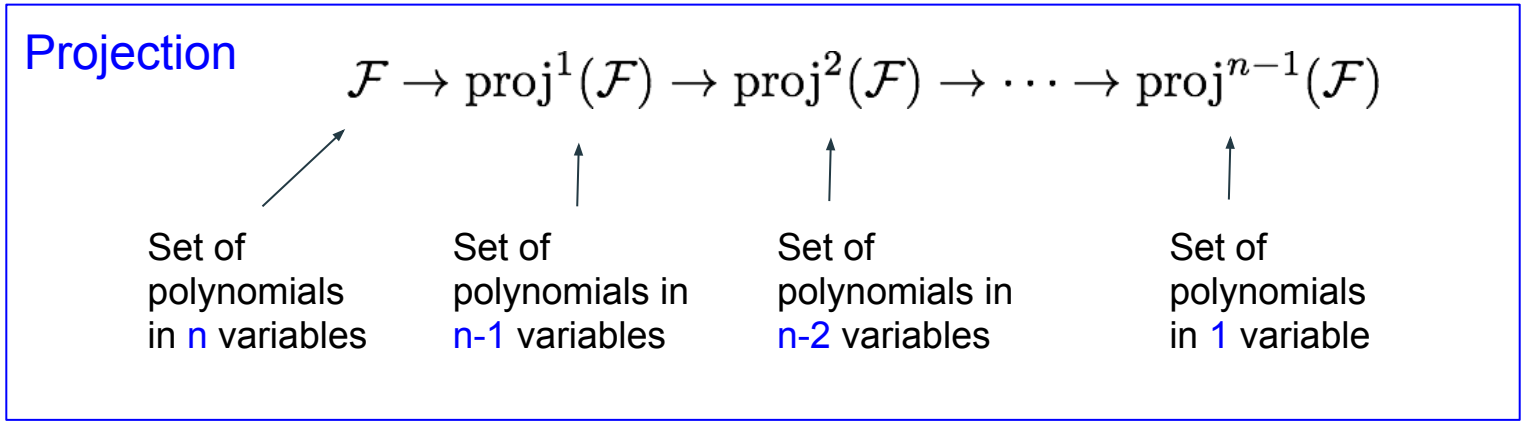
- A CAD for a set of polynomials is a set of sign-invariant regions and a sample point for each region
- How does having these points help us with QE?

Overview of CAD

- Construct a CAD in 2 phases: projection and lifting

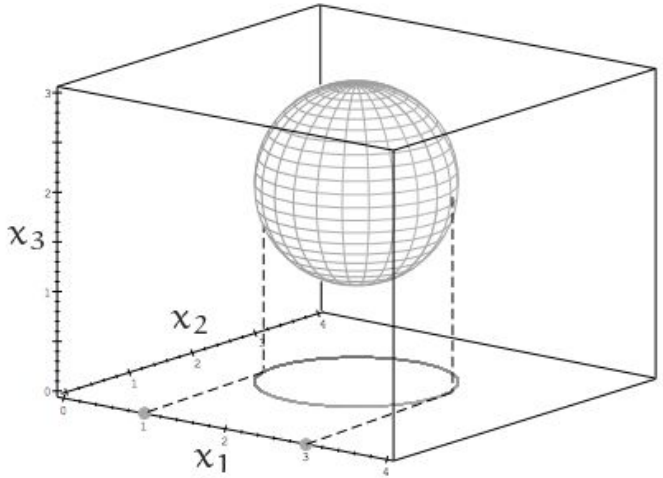
Overview of CAD

- Construct a CAD in 2 phases: projection and lifting



Intuition behind proj

"The zero sets of the resulting polynomials of each step is the projection of 'significant' points of the zero set of the preceding polynomials, i.e., self-crossings, isolated points, vertical tangent points etc." --Jirstrand



Shows the zero sets of the polynomials in each phase of projection

Image credit: Jirstrand

Overview of CAD

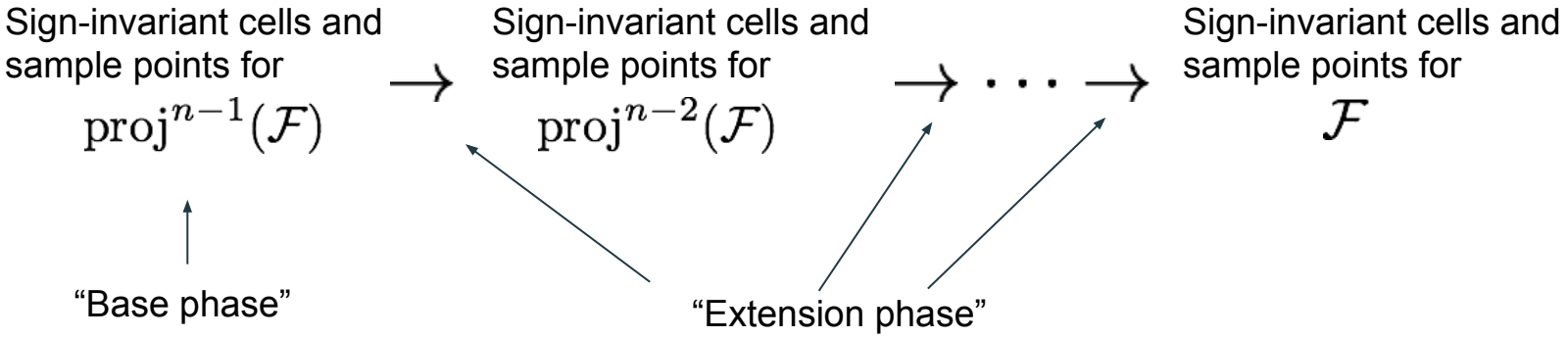
- Construct a CAD in 2 phases: projection and lifting

Lifting

Sign-invariant cells and sample points for $\text{proj}^{n-1}(\mathcal{F})$ $\rightarrow$ Sign-invariant cells and sample points for $\text{proj}^{n-2}(\mathcal{F})$ $\rightarrow \cdots \rightarrow$ Sign-invariant cells and sample points for $\mathcal{F}$ (the CAD)

Lifting: Two Phases

- "Base phase": construct a CAD for $\mathrm{proj}^{n-1}(\mathcal{F})$
- "Extension phase": Given a CAD for proj^{i-1}, construct a CAD for proj^{i}



Lifting: Base Phase

- $\text{proj}^{n-1}(\mathcal{F})$ is a set of univariate polynomials
- Find all roots, up to desired precision
- These roots and the intervals around the roots are the cells of the CAD
- NOTE: for each interval, we can choose a rational sample point
  - Why does this matter?

Lifting: Base Phase

- $\text{proj}^{n-1}(\mathcal{F})$ is a set of univariate polynomials
- Find all roots, up to desired precision
- These roots and the intervals around the roots are the cells of the CAD
- NOTE: for each interval, we can choose a rational sample point
  - Why does this matter?

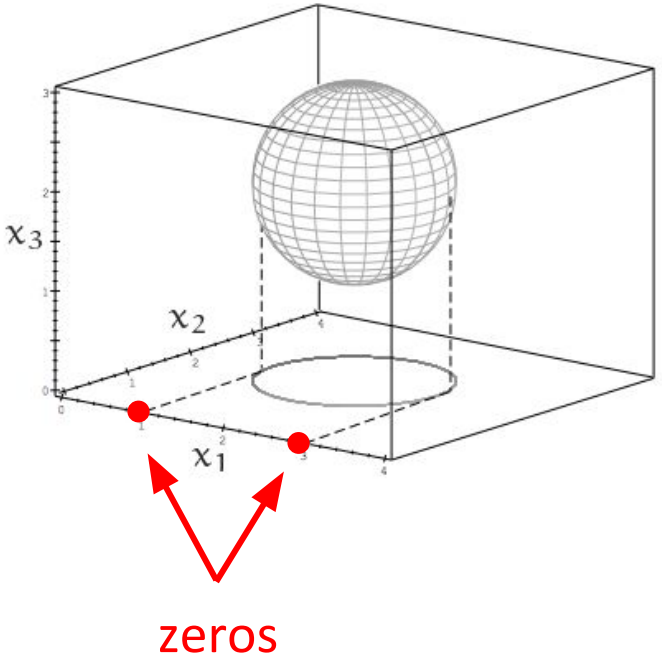
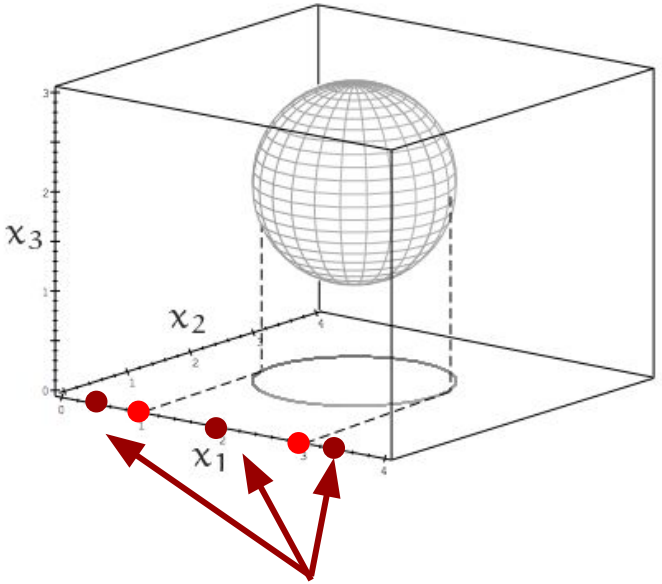


Image credit: Jirstrand (color added)

Lifting: Base Phase

- $\text{proj}^{n-1}(\mathcal{F})$ is a set of univariate polynomials
- Find all roots, up to desired precision
- These roots and the intervals around the roots are the cells of the CAD
- NOTE: for each interval, we can choose a rational sample point
  - Why does this matter?



Rational points

Image credit: Jirstrand (color added)

Lifting: Extension Phase

- Take a cell in our CAD for $proj^{i-1}$
- Evaluate all of the polynomials in $proj^{i}$ at the sample point of that cell
- This gives a set of polynomials in one variable
- Use the technique of the base phase!

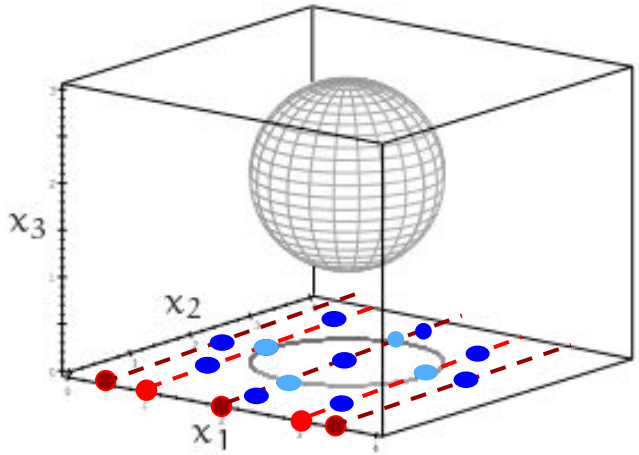


Image credit: Jirstrand (color added)

Lifting: Extension Phase

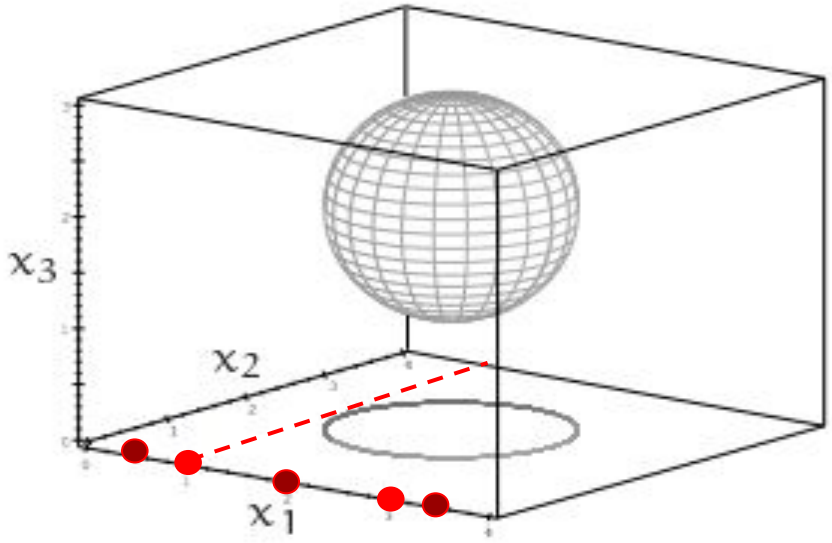


Image credit: Jirstrand (color added)

Lifting: Extension Phase

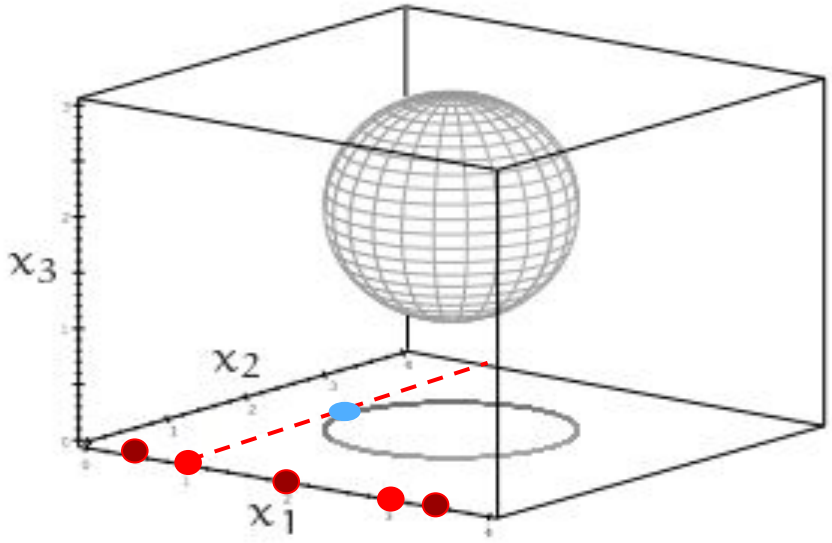


Image credit: Jirstrand (color added)

Lifting: Extension Phase

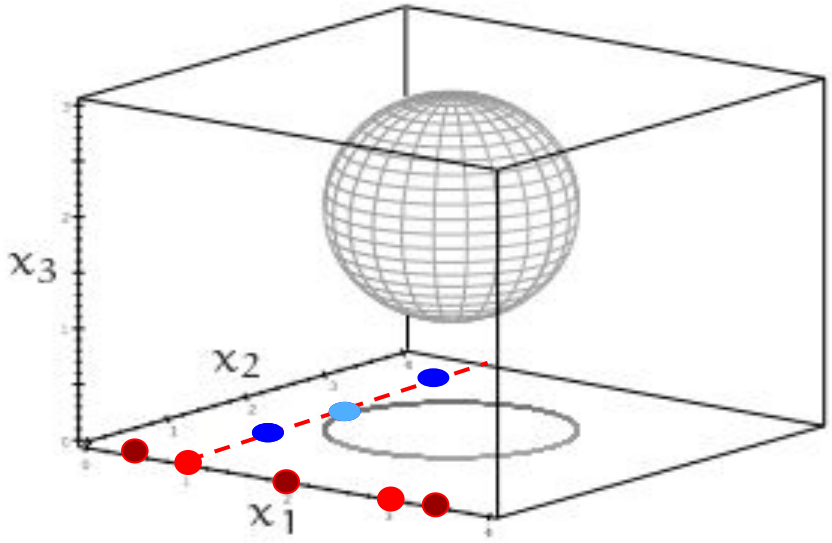


Image credit: Jirstrand (color added)

Lifting: Extension Phase

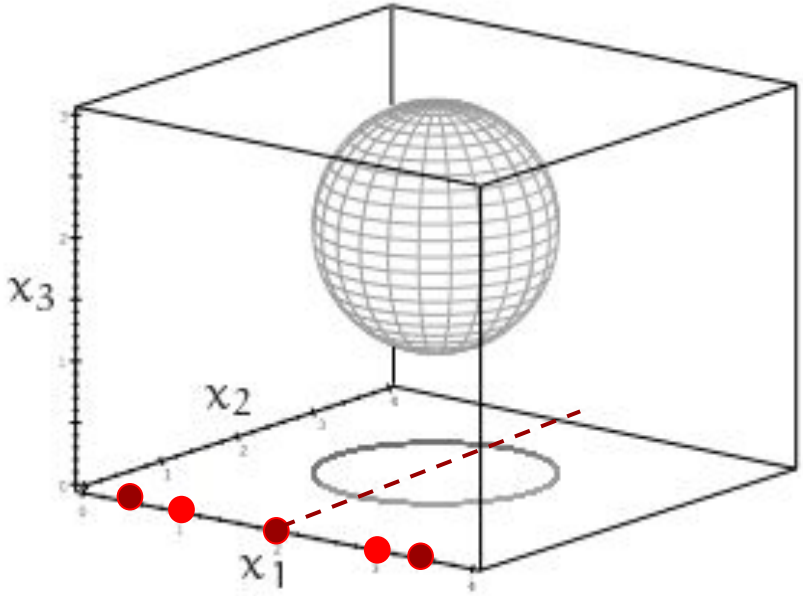


Image credit: Jirstrand (color added)

Lifting: Extension Phase

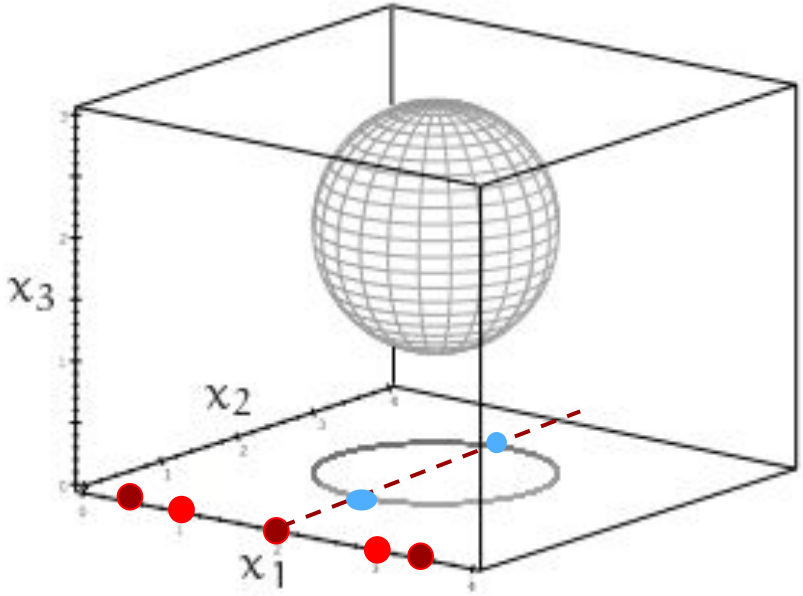


Image credit: Jirstrand (color added)

Lifting: Extension Phase

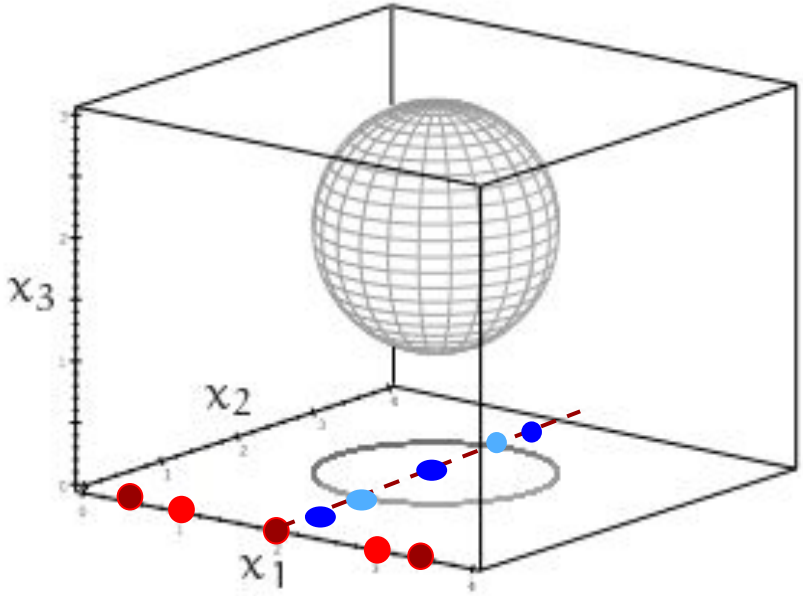


Image credit: Jirstrand (color added)

Lifting: Extension Phase

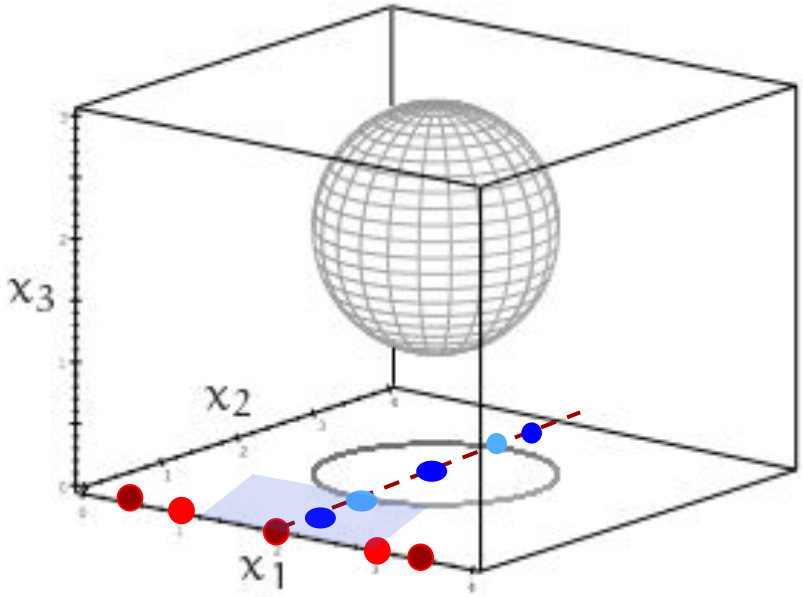


Image credit: Jirstrand (color added)

Lifting: Extension Phase

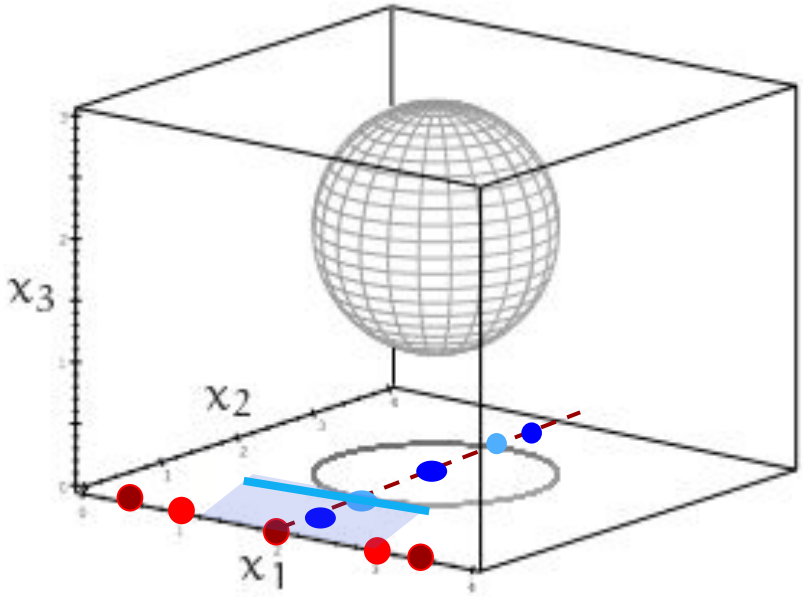


Image credit: Jirstrand (color added)

Lifting: Extension Phase

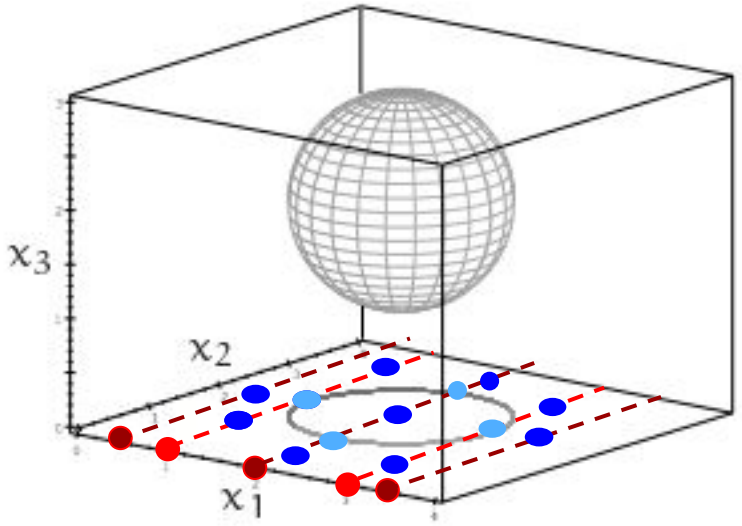


Image credit: Jirstrand (color added)

Lifting: Extension Phase

- The projection operator, proj, satisfies a key structural property in order to make this method of lifting sound.
- CAD gets its name because a lifted CAD is “cylindrically arranged” over its predecessor.

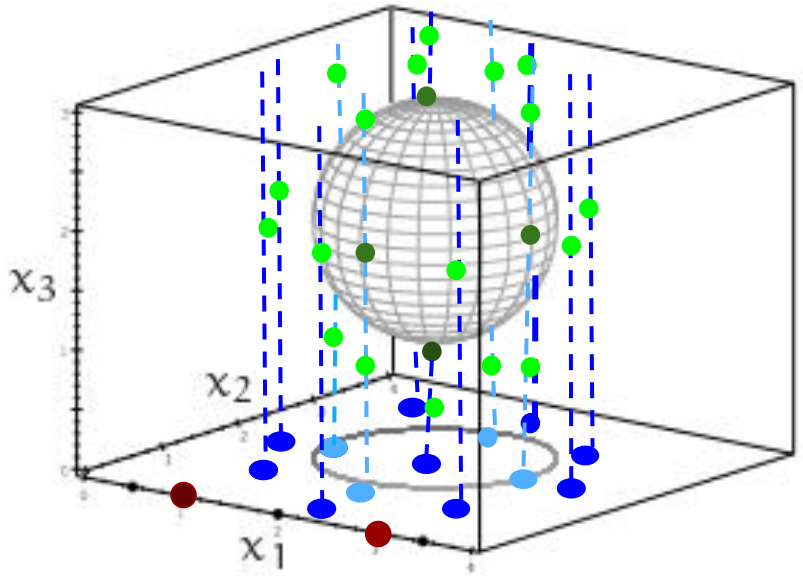


Image credit: Jirstrand (color added)

One optimization for CAD

- What if you know your QE problem only has strict inequalities (so no =, ≥, or ≤)?
- McCallum's CADMD: Only consider full-dimensional cells
  - Why does this help?

Connections between CAD and Virtual Substitution

CAD and VSubst

- This notion of “sample points” that capture all relevant information is also relevant in virtual substitution
- Can you see how?

VSubst and sample points

Say we have a cubic polynomial f(x) with roots f1, f2, f3, as pictured below:

What points do we need to virtually substitute if we have a formula of the form Exists x. f(x)=0 & (Other Formula) ?

VSubst and sample points

If we have a formula of the form Exists x. f(x)=0 & (Other Formula), we should virtually substitute the roots of f into “Other Formula”

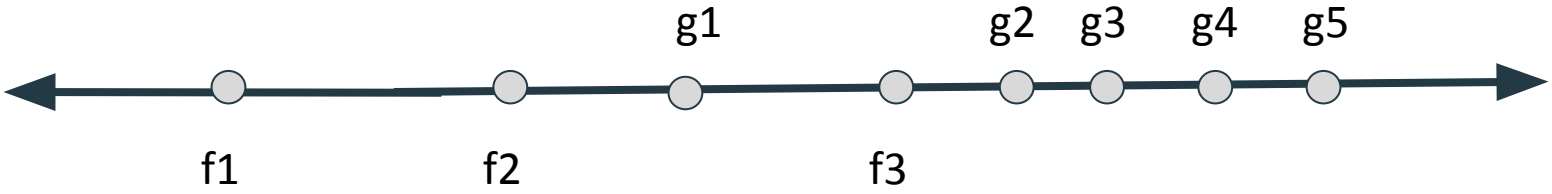
VSubst and sample points

What do we need to virtually substitute if we have a formula of the form Exists x. f(x)>0 & (Other Formula) ?

TRICK QUESTION! We can’t use virtual substitution with an arbitrary Other Formula in this case.

VSubst and sample points

For example: Exists x. f(x)>0 & g(x)= 0,where g(x) has 5 roots, and where f(g1) > 0.



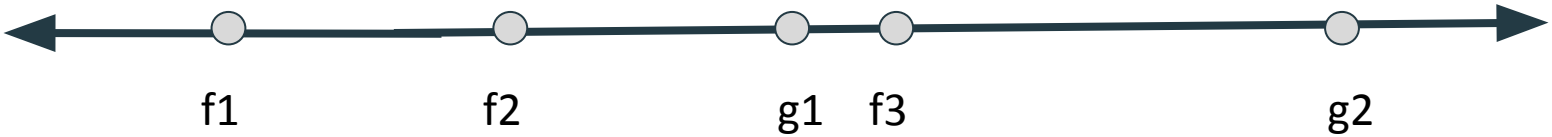
Can we use virtual substitution here?

No! We can't solve for the roots of g (quintic insolubility). And virtual substitution won't find these roots.

We should use CAD instead.

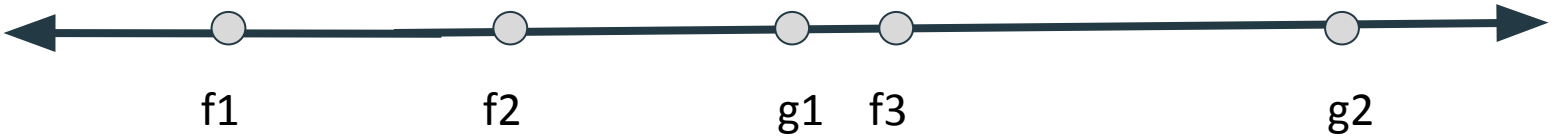
VSubst and sample points

Now, say we have a cubic polynomial f with roots f1, f2, f3 and a quadratic polynomial g with roots g1, g2, as pictured below:



VSubst and sample points

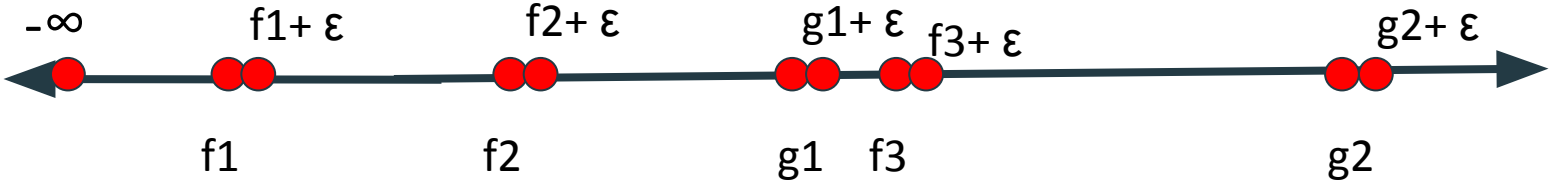
Roots of f: f1, f2, f3
Roots of g: g1, g2



What points do we need to virtually substitute if we have a formula of the form Exists x. f>=0 & g>=0 ?

VSubst and sample points

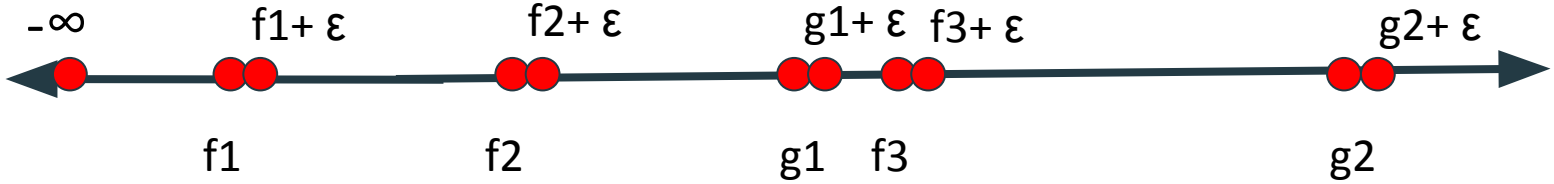
Roots of f: f1, f2, f3
Roots of g: g1, g2



If we have a formula of the form Exists x. f>=0 & g>=0, we should substitute -∞, the roots, and the off-roots

VSubst and sample points

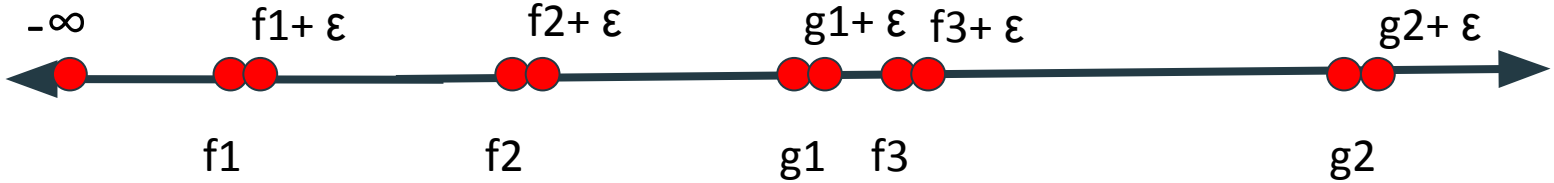
Roots of f: f1, f2, f3
Roots of g: g1, g2



What information does g1+ ε capture?
Information for the entire interval (g1, f3)

VSubst and sample points

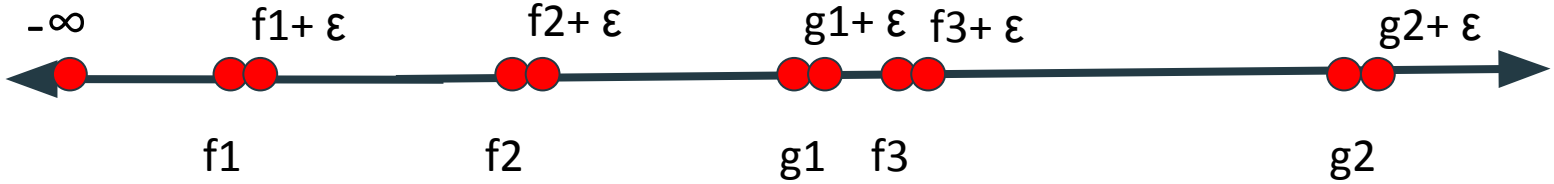
Roots of f: f1, f2, f3
Roots of g: g1, g2



What information does -∞ capture?
The entire range (-∞, f1)

VSubst and sample points

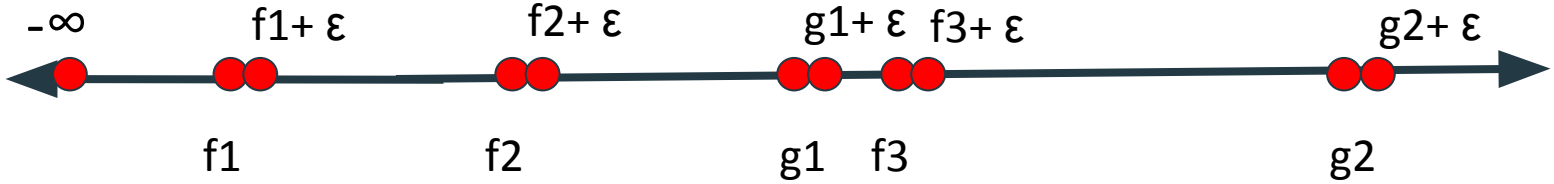
Roots of f: f1, f2, f3
Roots of g: g1, g2



Why don't we also need ∞?
That interval is already captured by g2 + ε

VSubst and sample points

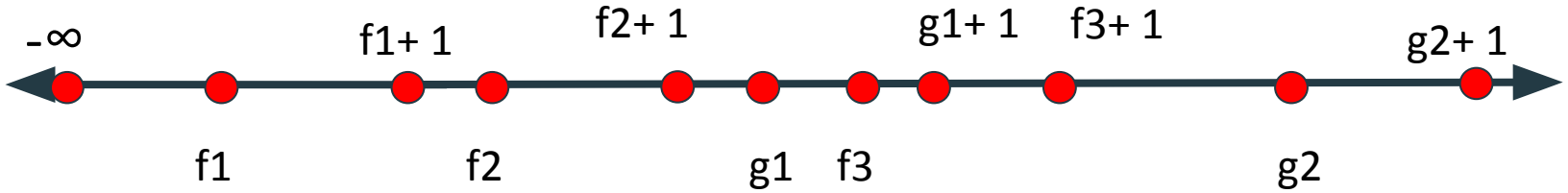
Roots of f: f1, f2, f3
Roots of g: g1, g2



Why do we need the ε to be arbitrary? Why not use some fixed ε, like ε = 1?

VSubst and sample points

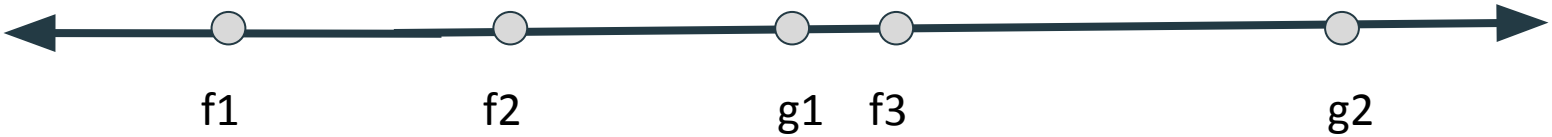
Roots of f: f1, f2, f3
Roots of g: g1, g2



What is wrong here?
We have lost information about the interval (g1, f3). 1 was too big.
Using arbitrary ε prevents this problem.

VSubst and sample points

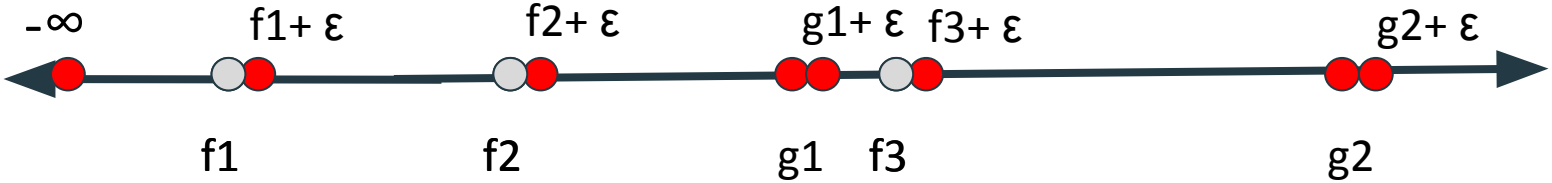
Roots of f: f1, f2, f3
Roots of g: g1, g2



What points do we need to virtually substitute if we have a formula of the form Exists x. f>0 & g>=0 ?

VSubst and sample points

Roots of f: f1, f2, f3
Roots of g: g1, g2



If our formula is Exists x. f>0 & g>=0, we no longer need to substitute the roots of f.

The key to CAD and Vsubst

Turn a continuous problem into a discrete one.

Quantifier elimination & KeYmaera X

Outline

- ❏ Why we can’t trust KeYmaera X
- ❏ Why KeYmaera X is still state-of-the-art
- ❏ How we can improve the situation

Why we can’t trust KeYmaera X

- Recall how KeYmaera X handles QE…

Image credit: Wikipedia

Why we can't trust KeYmaera X

- Recall how KeYmaera X handles QE…
- Should we trust this?

Image credit: Wikipedia

Why does it do this?

- Ideally KeYmaera X would outsource to a tool with support for formally verified QE
- What does formally verified support for QE look like?

Formally Verified Support for QE

- Very few general-purpose FV'd algorithms.
  - Mahboubi and Cohen (formalized Tarski's algorithm in Coq)
  - John Harrison (proof-producing procedure in HOL Light)
- A variety of univariate FV'd algorithms (in Isabelle, Coq, PVS, etc.)

Can we do better?

- KeYmaera X needs to use QE to close proofs.
- For complex real-world case studies, outsourcing to unverified software may be the only way to get an answer.

Can we do better?

- My current research focuses on improving formally verified support for QE.
- I work in the Isabelle theorem prover.

Isabelle Tutorial!

References

- Jirstrand, Mats. *Algebraic methods for inequality constraints in control*. Diss. PhD thesis, Linköping University, Department of Electrical Engineering, Division of Automatic Control, 1998
- Pablo Parrilo's lecture notes may be found here: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-972-algebraic-techniques-and-semidefinite-optimization-spring-2006/lecture-notes/lecture_18.pdf

For more information on CAD, please check out the CAD papers!