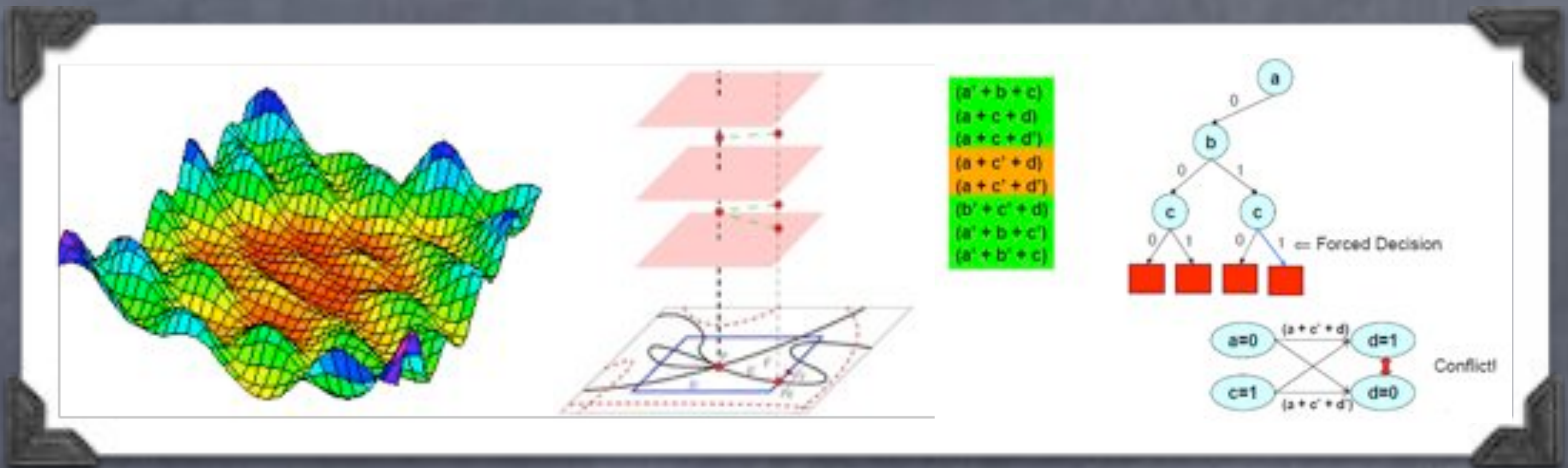
(Exact Global Nonlinear) Optimization on Demand



Leonardo de Moura & Grant Olney Passmore
MSR, Redmond, USA Edinburgh and Cambridge, UK

ADDCT-2013 ~ CADE-24
(presentation only)

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers
- Three main conceptual ingredients:

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers
- Three main conceptual ingredients:

CAD-based approach to optimization

eager method for nonlinear optimization, in Mathematica v9.x

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers
- Three main conceptual ingredients:

CAD-based approach to optimization

eager method for nonlinear optimization, in Mathematica v9.x

nlsat / existential CAD `on demand'

lazy CDCL-like approach to Exists RCF, in Z3 (Jovanović - de Moura, 2012)

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers
- Three main conceptual ingredients:

CAD-based approach to optimization

eager method for nonlinear optimization, in Mathematica v9.x

nlsat / existential CAD `on demand'

lazy CDCL-like approach to Exists RCF, in Z3 (Jovanović - de Moura, 2012)

computable nonstandard RCFs

computable RCFs containing infinitesimals (de Moura - Passmore, 2013)

Big Picture

- A CDCL-like approach to exact nonlinear global optimization over the real numbers
- Three main conceptual ingredients:

CAD-based approach to optimization

eager method for nonlinear optimization, in Mathematica v9.x

nlsat / existential CAD `on demand'

lazy CDCL-like approach to Exists RCF, in Z3 (Jovanović - de Moura, 2012)

computable nonstandard RCFs

computable RCFs containing infinitesimals (de Moura - Passmore, 2013)

- A *practical application* of nonstandard models!

Exact Global Optimization

$$\begin{aligned} &\underset{\vec{x}}{\text{minimize}} \quad && f(\vec{x}) \\ &\text{subject to} \quad && \bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \end{aligned}$$

Many classes of optimization problems, based on restrictions of f's and b's

Exact Global Optimization

$$\begin{array}{ll} \underset{\vec{x}}{\text{minimize}} & f(\vec{x}) \\ \text{subject to} & \displaystyle\bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \end{array}$$

Many classes of optimization problems, based on restrictions of f's and b's

nonlinear, computable

What is a Real Closed Field?

$$\mathrm{RCF} = Th(\langle \mathbb{R}, +, *, <, 0, 1 \rangle)$$

What is a Real Closed Field?

$$\mathrm{RCF} = Th(\langle \mathbb{R}, +, *, <, 0, 1 \rangle)$$

Examples:

- The reals: $\langle \mathbb{R}, +, *, <, 0, 1 \rangle$
- The algebraic reals: $\langle \mathbb{R}_{alg}, +, *, <, 0, 1 \rangle$
- The (a!) Hyperreals: $\left(\prod_{\mathbb{N}} \langle \mathbb{R}, +, *, <, 0, 1 \rangle \right) / \, \mathcal{U}$
- Real closures: $\widetilde{\mathbb{K}}$ s.t. $\mathbb{K} = \mathbb{Q}(t_1, \ldots, t_n, \epsilon_1, \ldots, \epsilon_m)$

Optimization using RCF QE - I

RCF admits quantifier elimination (QE)

Optimization using RCF QE - I

RCF admits quantifier elimination (QE)

In theory, one can exploit RCF QE to solve nonlinear optimization problems over the reals:
Let's see how! In the next slide...

Optimization using RCF QE - I

RCF admits quantifier elimination (QE)

In theory, one can exploit RCF QE to solve nonlinear optimization problems over the reals:
Let's see how! In the next slide...

In practice, this is not a viable solution:
RCF QE is infeasible: O(2^2^(Omega(n)))

Optimization using RCF QE - II

$$\underset{\vec{x}}{\text{minimize}} \quad f(\vec{x})$$

$$\text{subject to} \quad \bigwedge_{i=1}^{m} f_i(\vec{x}) \le b_i$$

Optimization using RCF QE - II

$$\begin{aligned} &\underset{\vec{x}}{\text{minimize}} && f(\vec{x}) \\ &\text{subject to} && \bigwedge_{i=1}^{m} f_i(\vec{x}) \le b_i \end{aligned}$$

Step 1: New coordinate function y

$$F(\vec{x}, y) \triangleq \left(y = f(\vec{x}) \wedge \bigwedge_{i=1}^{m} f_i(\vec{x}) \le b_i \right)$$

Optimization using RCF QE - II

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & \bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \end{aligned}$$

Step 1: New coordinate function y

$$F(\vec{x}, y) \triangleq \left(y = f(\vec{x}) \wedge \bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \right)$$

Step 2: QE (project onto y)

Use RCF QE to eliminate $\exists \vec{x}$ from $\exists \vec{x} F(\vec{x}, y)$, obtaining $\varphi(y)$ s.t.

$$\varphi(y) \triangleq \bigvee_i \bigwedge_j (p_{i,j}(y) \bowtie_{i,j} 0), \qquad \bowtie_{i,j} \in \{<, \leq, =, \geq, >\}, \quad p_{i,j} \in \mathbb{Z}[y].$$

Step 2: QE (project onto y)

Use RCF QE to eliminate $\exists\vec{x}$ from $\exists\vec{x}F(\vec{x},y)$, obtaining $\varphi(y)$ s.t.

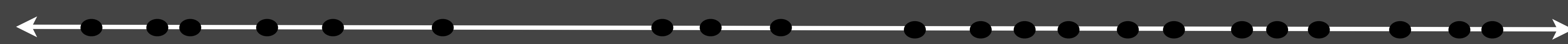
$$\varphi(y) \triangleq \bigvee_i \bigwedge_j (p_{i,j}(y) \bowtie_{i,j} 0), \qquad \bowtie_{i,j} \in \{<, \leq, =, \geq, >\}, \;\; p_{i,j} \in \mathbb{Z}[y].$$

Step 3: Real Root Isolation
(note sign invariance: IVT!)

Use univariate real root isolation (e.g., via Sturm sequences) to isolate all roots of $p_{i,j}(y) \in \mathbb{Z}[y]$. This partitions $\mathbb{R}$ into $2k+1$ connected components.

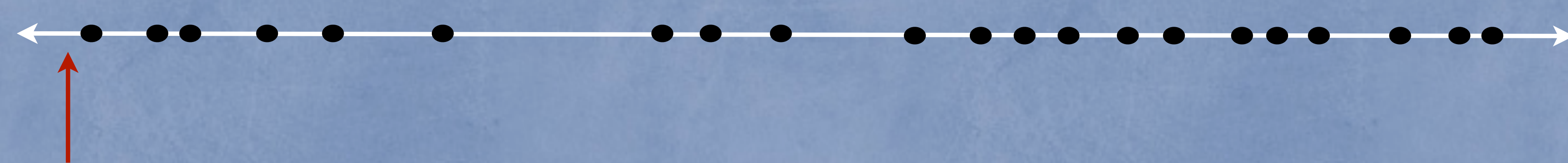
Step 3: Real Root Isolation
(note sign invariance: IVT!)

Use univariate real root isolation (e.g., via Sturm sequences) to isolate all roots of $p_{i,j}(y) \in \mathbb{Z}[y]$. This partitions $\mathbb{R}$ into $2k+1$ connected components.



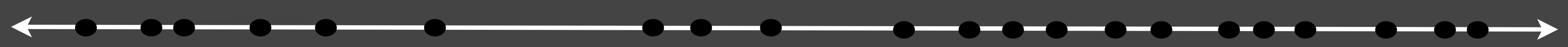
Step 4: Search!

Sweep from L to R, looking for first connected component satisfying $\varphi(y)$



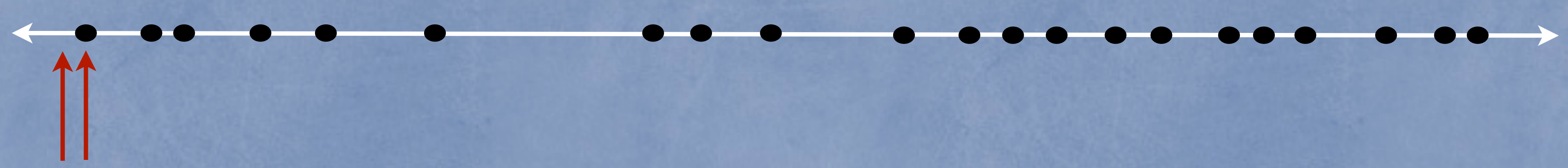
Step 3: Real Root Isolation
(note sign invariance: IVT!)

Use univariate real root isolation (e.g., via Sturm sequences) to isolate all roots of $p_{i,j}(y) \in \mathbb{Z}[y]$. This partitions $\mathbb{R}$ into $2k + 1$ connected components.



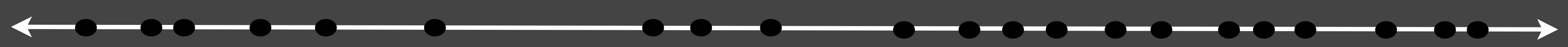
Step 4: Search!

Sweep from L to R, looking for first connected component satisfying $\varphi(y)$



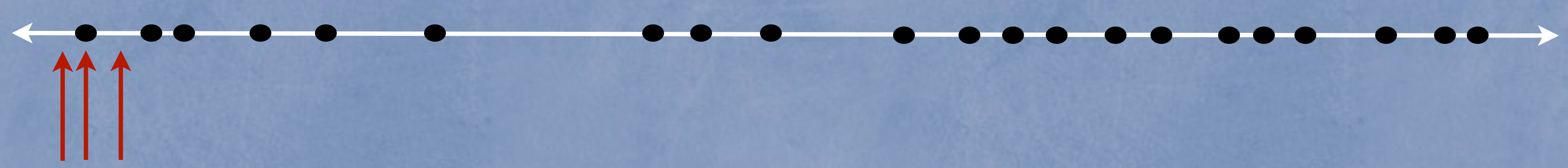
Step 3: Real Root Isolation
(note sign invariance: IVT!)

Use univariate real root isolation (e.g., via Sturm sequences) to isolate all roots of $p_{i,j}(y) \in \mathbb{Z}[y]$. This partitions $\mathbb{R}$ into $2k+1$ connected components.



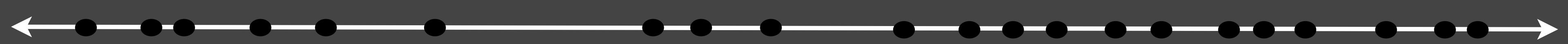
Step 4: Search!

Sweep from L to R, looking for first connected component satisfying $\varphi(y)$



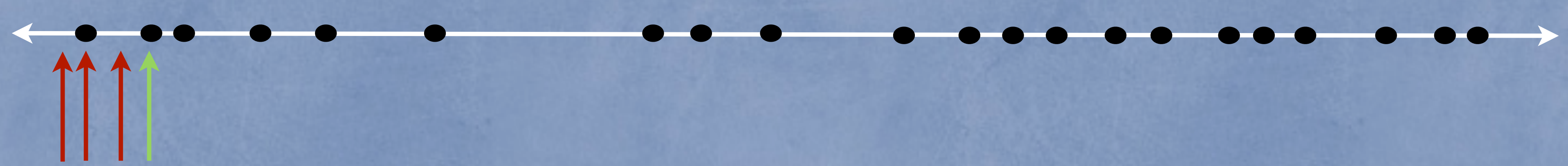
Step 3: Real Root Isolation
(note sign invariance: IVT!)

Use univariate real root isolation (e.g., via Sturm sequences) to isolate all roots of $p_{i,j}(y) \in \mathbb{Z}[y]$. This partitions $\mathbb{R}$ into $2k+1$ connected components.



Step 4: Search!

Sweep from L to R, looking for first connected component satisfying $\varphi(y)$



exact minimum found!

Four Possible Outcomes

- No satisfying region: Infeasible
- (-inf, r) : Unbounded
- [r] : Exact minimum
- (r, _) : No minimum, but exact infimum

Five ~~Four~~ Possible Outcomes

- No satisfying region: Infeasible
- (-inf, r) : Unbounded
- [r] : Exact minimum
- (r, _) : No minimum, but exact infimum
- RUN OUT OF MEMORY AND/OR TIME!!!

Computing $\varphi(y)$ explicitly is a bad idea!

A CAD-based Approach

- Used by Mathematica
- Doesn't require explicit computation of Phi(y)
- But, it is *eager* and *pessimistic*
- Our new approach is *lazy* and *optimistic*
- First, let's understand the CAD-based approach...

Cylindrical Algebraic Decomposition

CAD: A partitioning of $\mathbb{R}^n$

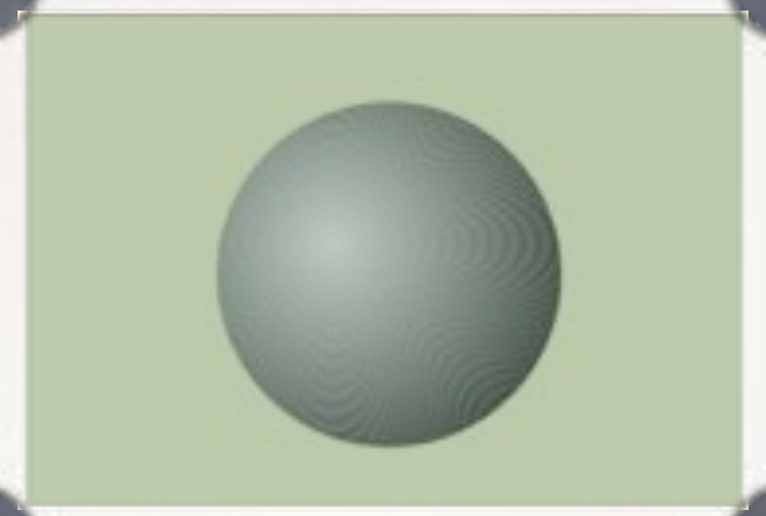
into finitely many RCF-definable connected components which "behaves nicely" w.r.t. projections onto lower dimensions.

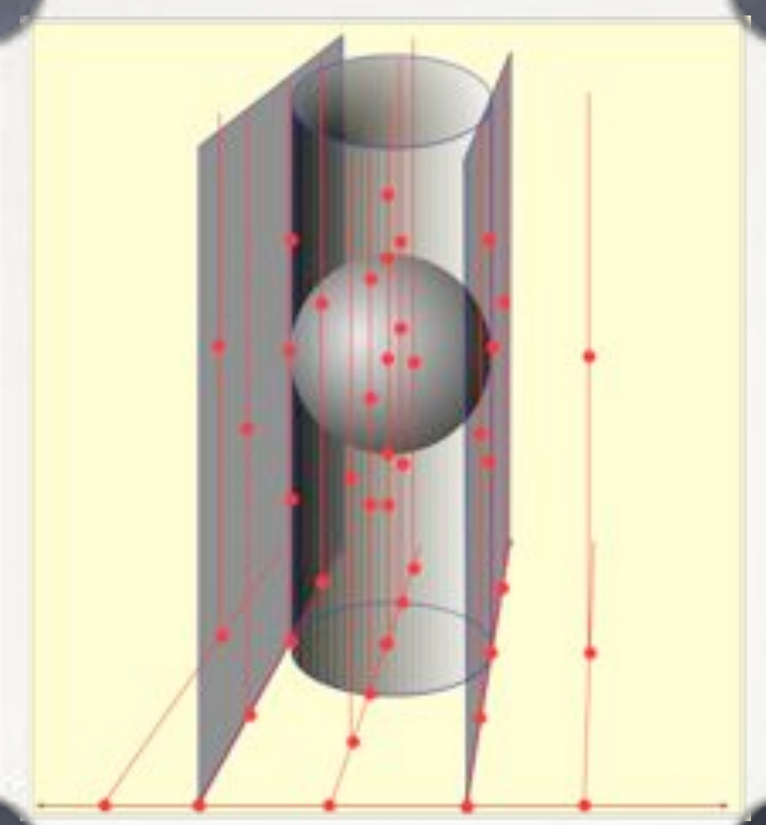
$$P \subset \mathbb{Z}[x_1, \ldots, x_n]$$

P-invariant **CAD**

a CAD of $\mathbb{R}^n$ s.t. for all cells c_i, all $p \in P$

$$\forall \vec{r} \in c_i (p(\vec{r}) = 0) \quad \vee$$
$$\forall \vec{r} \in c_i (p(\vec{r}) > 0) \quad \vee$$
$$\forall \vec{r} \in c_i (p(\vec{r}) < 0) \quad .$$





CAD sphere diagrams: C. Brown and QEPCAD-B

CAD Phase I: Projection

$$Proj_{i+1} : \mathbb{Z}[x_1, \ldots, x_{i+1}] \to \mathbb{Z}[x_1, \ldots, x_i]$$

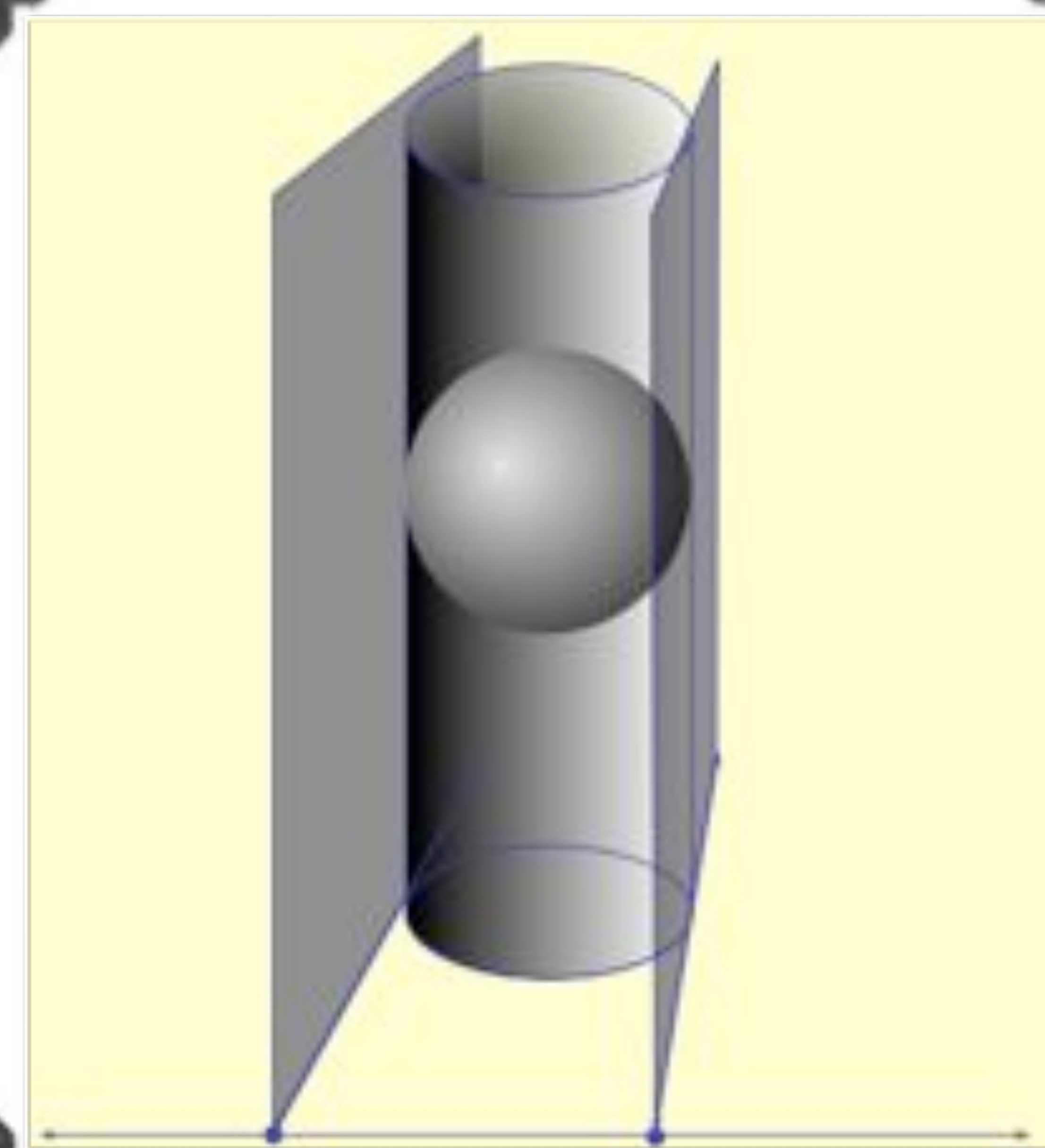
Inductive Property:

A (P_{i+1})-invariant CAD for R^{i+1}
can be constructed from
a (P_i)-invariant CAD of R^i.

$$P_n = P \subset \mathbb{Z}[x_1, \ldots, x_n]$$
$$P_{n-1} = Proj(P_n) \subset \mathbb{Z}[x_1, \ldots, x_{n-1}]$$
$$\vdots$$
$$P_2 = Proj(P_3) \subset \mathbb{Z}[x_1, x_2]$$
$$P_1 = Proj(P_2) \subset \mathbb{Z}[x_1]$$

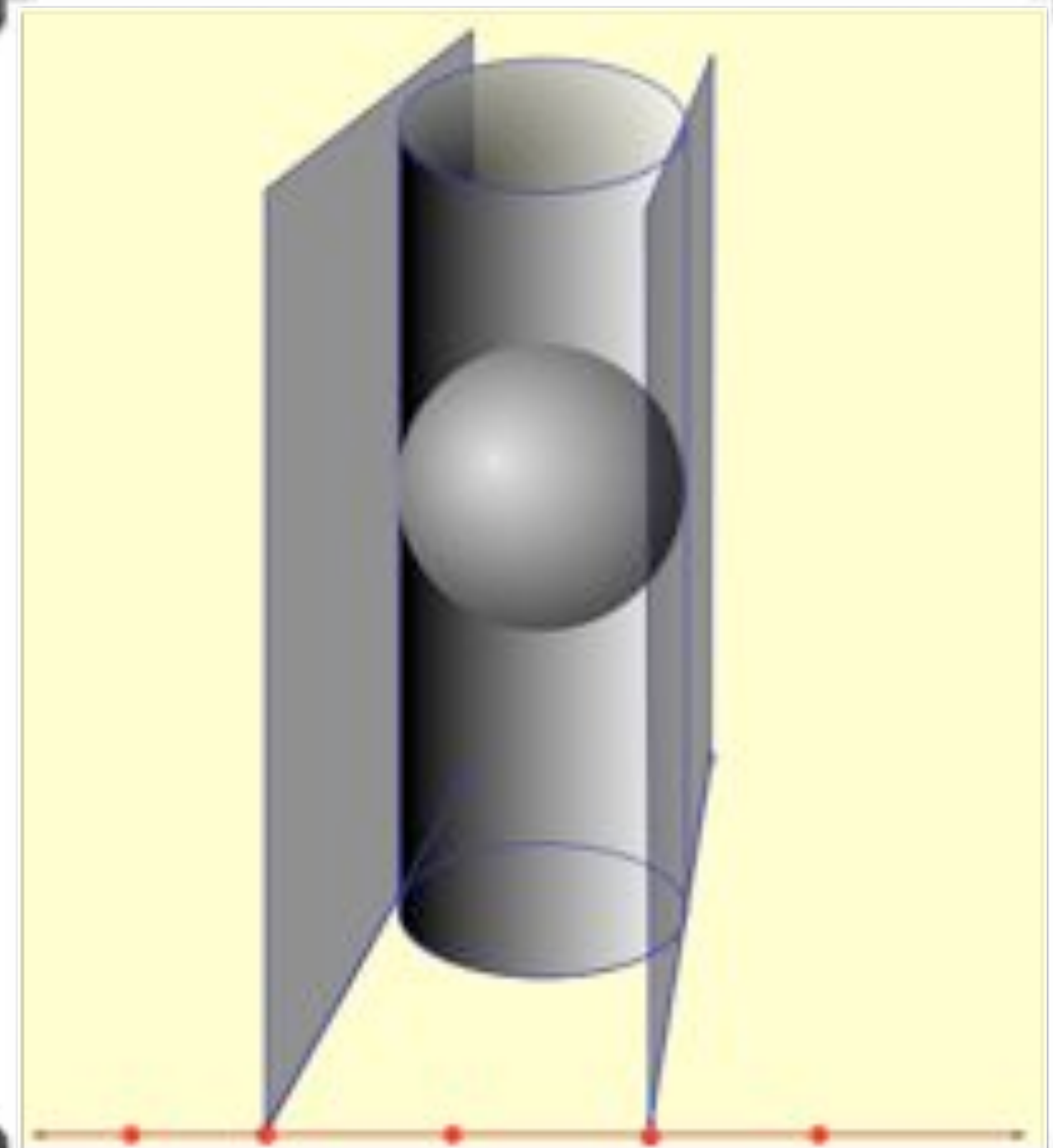
Projection sets

$$P_3 = \{x_1^2 + x_2^2 + x_3^2 - 4\}$$
$$P_2 = \{x_2^2 + x_1^2 - 4\}$$
$$P_1 = \{x_1 + 2, x_1 - 2\}$$



Projection sets

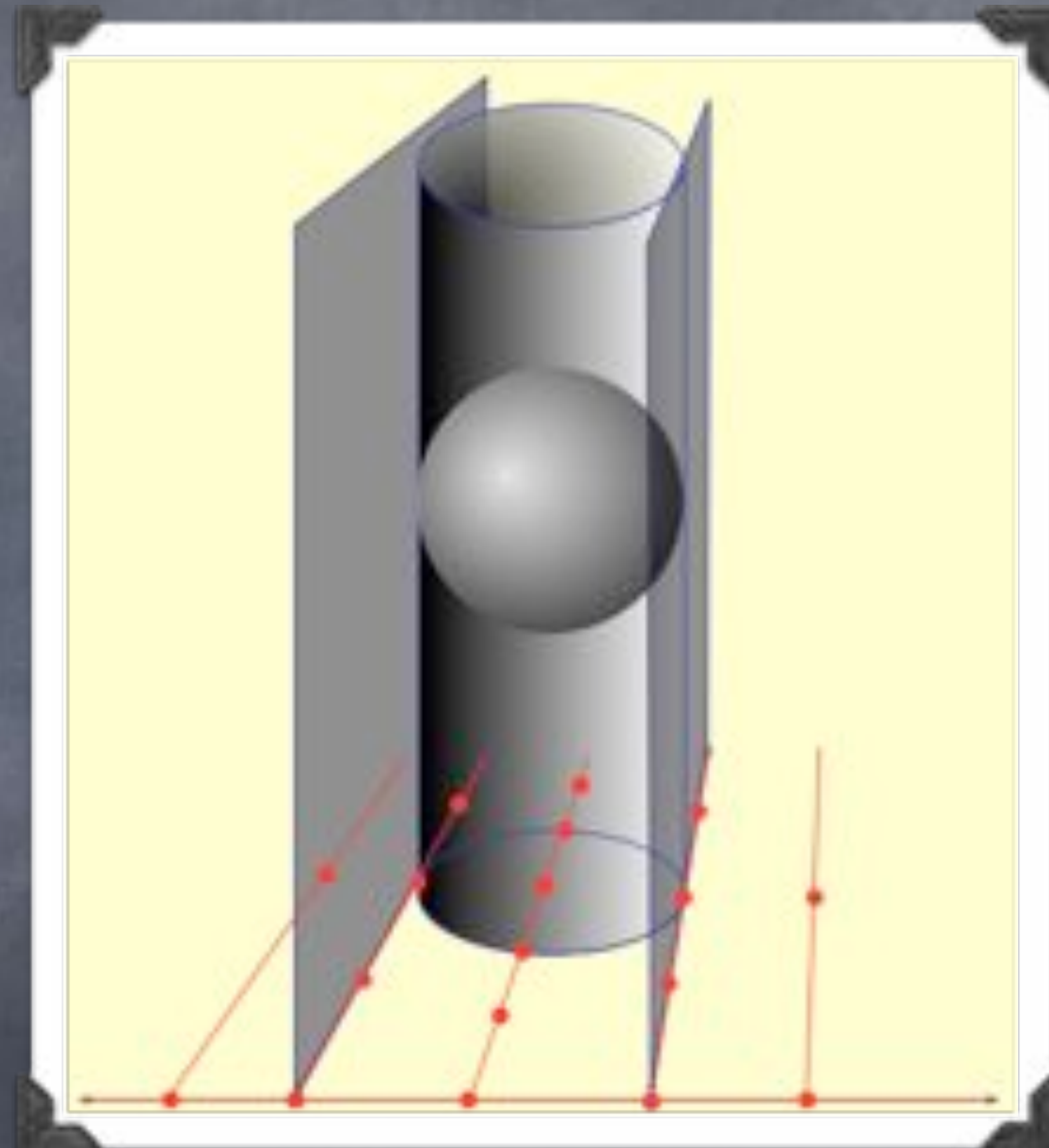
$$P_3 = \{x_1^2 + x_2^2 + x_3^2 - 4\}$$
$$P_2 = \{x_2^2 + x_1^2 - 4\}$$
$$P_1 = \{x_1 + 2, x_1 - 2\}$$



Base Phase: R^1

Projection sets

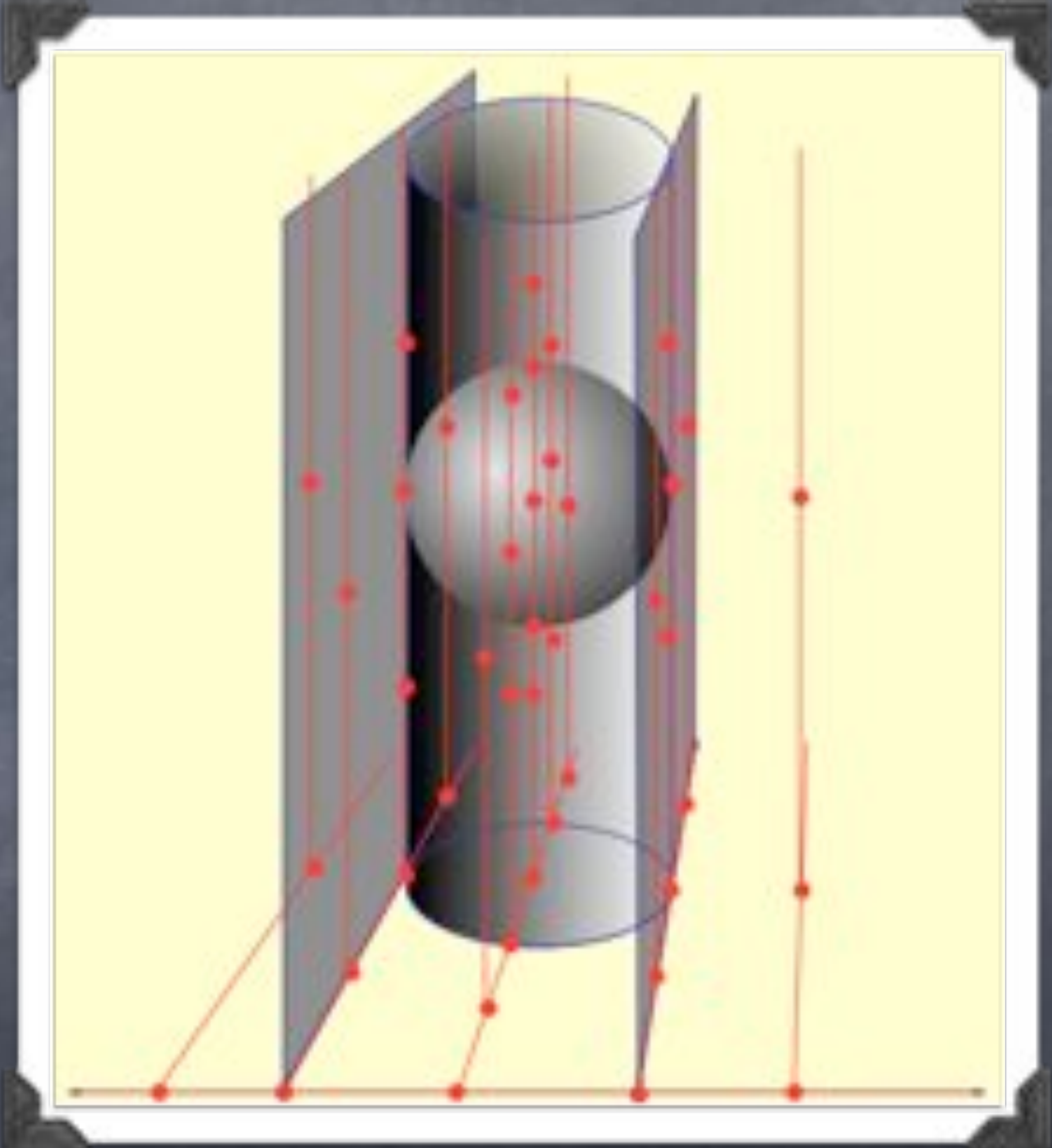
$$P_3 = \{x_1^2 + x_2^2 + x_3^2 - 4\}$$
$$P_2 = \{x_2^2 + x_1^2 - 4\}$$
$$P_1 = \{x_1 + 2, x_1 - 2\}$$



Lifting Phase: R^1 -> R^2

Projection sets

$$P_3 = \{x_1^2 + x_2^2 + x_3^2 - 4\}$$
$$P_2 = \{x_2^2 + x_1^2 - 4\}$$
$$P_1 = \{x_1 + 2, x_1 - 2\}$$



Lifting Phase: R^2 -> R^3

A CAD-based Approach to Optimization

$$\begin{array}{ll} \underset{\vec{x}}{\text{minimize}} & f(\vec{x}) \\ \text{subject to} & \bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \end{array}$$

Step 1: New coordinate function y

$$F(\vec{x}, y) \triangleq \left(y = f(\vec{x}) \wedge \bigwedge_{i=1}^{m} f_i(\vec{x}) \leq b_i \right)$$

Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$

$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$

$$\vdots$$

$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$

$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$

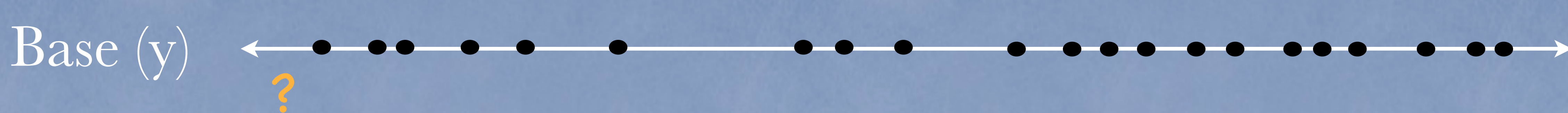
$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$

$$\vdots$$

$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$

$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

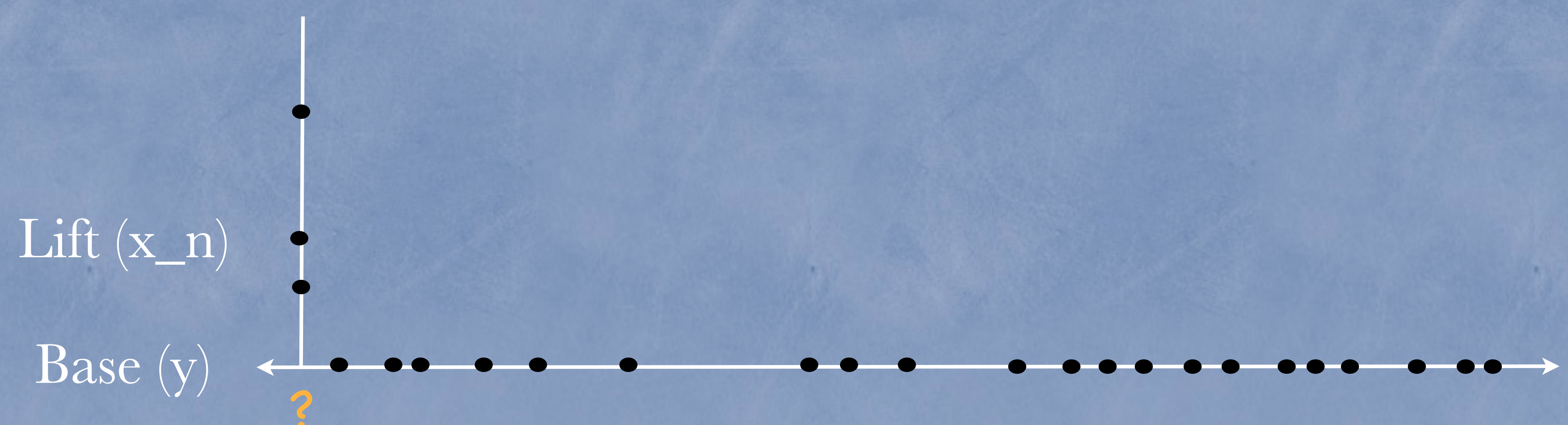
Step 3: CAD Base and Lifting (depth-first) from L to R



Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$
$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$
$$\vdots$$
$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$
$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

Step 3: CAD Base and Lifting (depth-first) from L to R



Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$

$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$

$$\vdots$$

$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$

$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

Step 3: CAD Base and Lifting (depth-first) from L to R

Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$

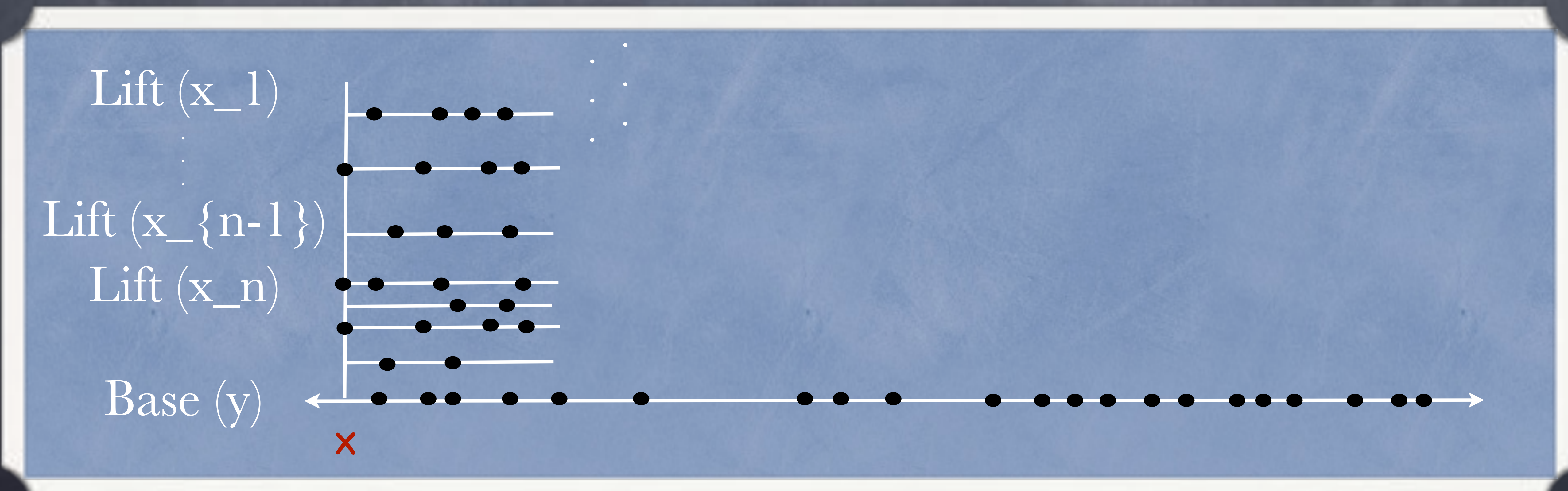
$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$

$$\vdots$$

$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$

$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

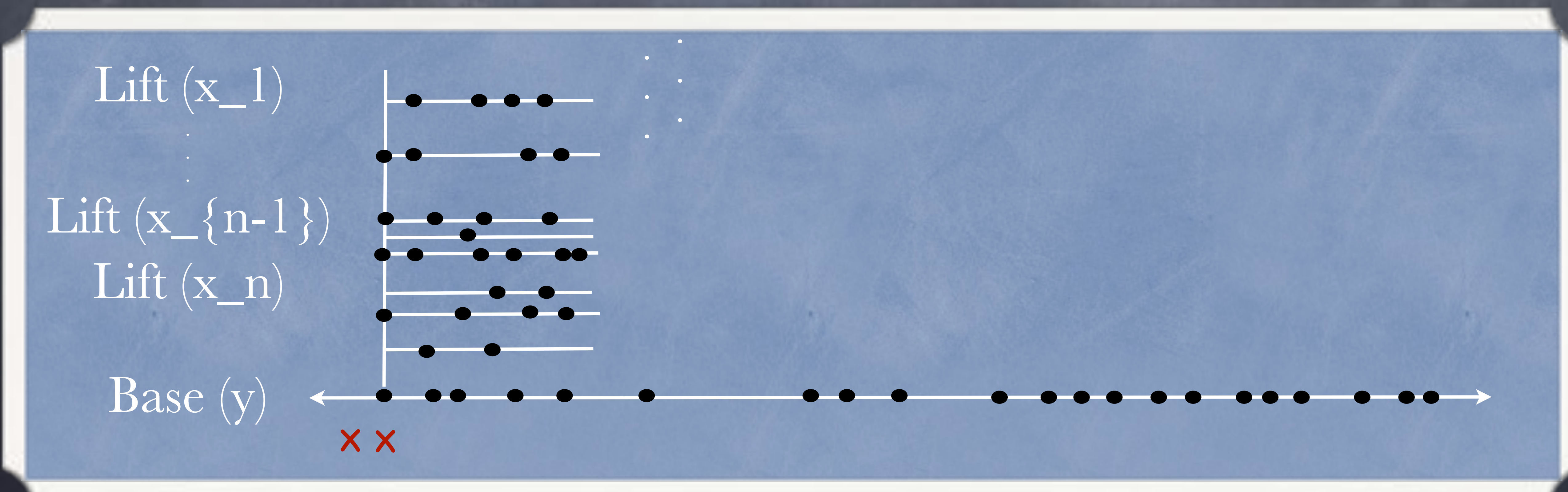
Step 3: CAD Base and Lifting (depth-first) from L to R



Step 2: CAD projection (with y lowest variable)

$$P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} \subset \mathbb{Z}[y, x_1, \ldots, x_n]$$
$$P_n = Proj(P_{n+1}) \subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}]$$
$$\vdots$$
$$P_2 = Proj(P_3) \subset \mathbb{Z}[y, x_1]$$
$$P_1 = Proj(P_2) \subset \mathbb{Z}[y]$$

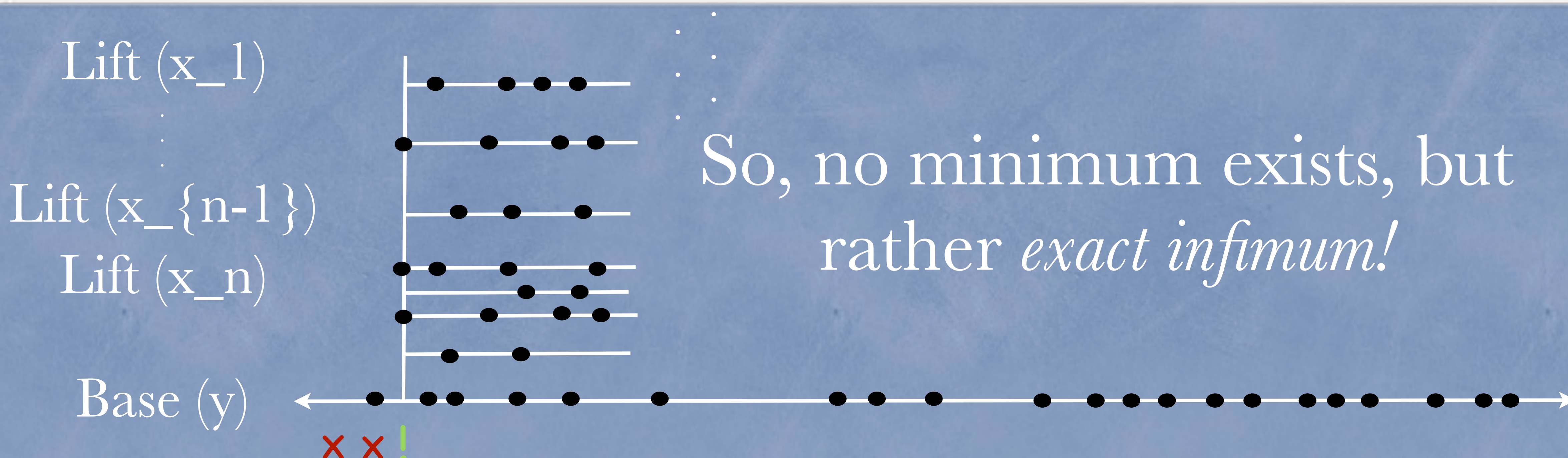
Step 3: CAD Base and Lifting (depth-first) from L to R



Step 2: CAD projection (with y lowest variable)

$$
\begin{aligned}
P_{n+1} = \{y - f(\vec{x}), f_1(\vec{x}) - b_1, \ldots, f_m(\vec{x}) - b_m\} &\subset \mathbb{Z}[y, x_1, \ldots, x_n] \\
P_n = Proj(P_{n+1}) &\subset \mathbb{Z}[y, x_1, \ldots, x_{n-1}] \\
&\vdots \\
P_2 = Proj(P_3) &\subset \mathbb{Z}[y, x_1] \\
P_1 = Proj(P_2) &\subset \mathbb{Z}[y]
\end{aligned}
$$

Step 3: CAD Base and Lifting (depth-first) from L to R



So, no minimum exists, but rather *exact infimum!*

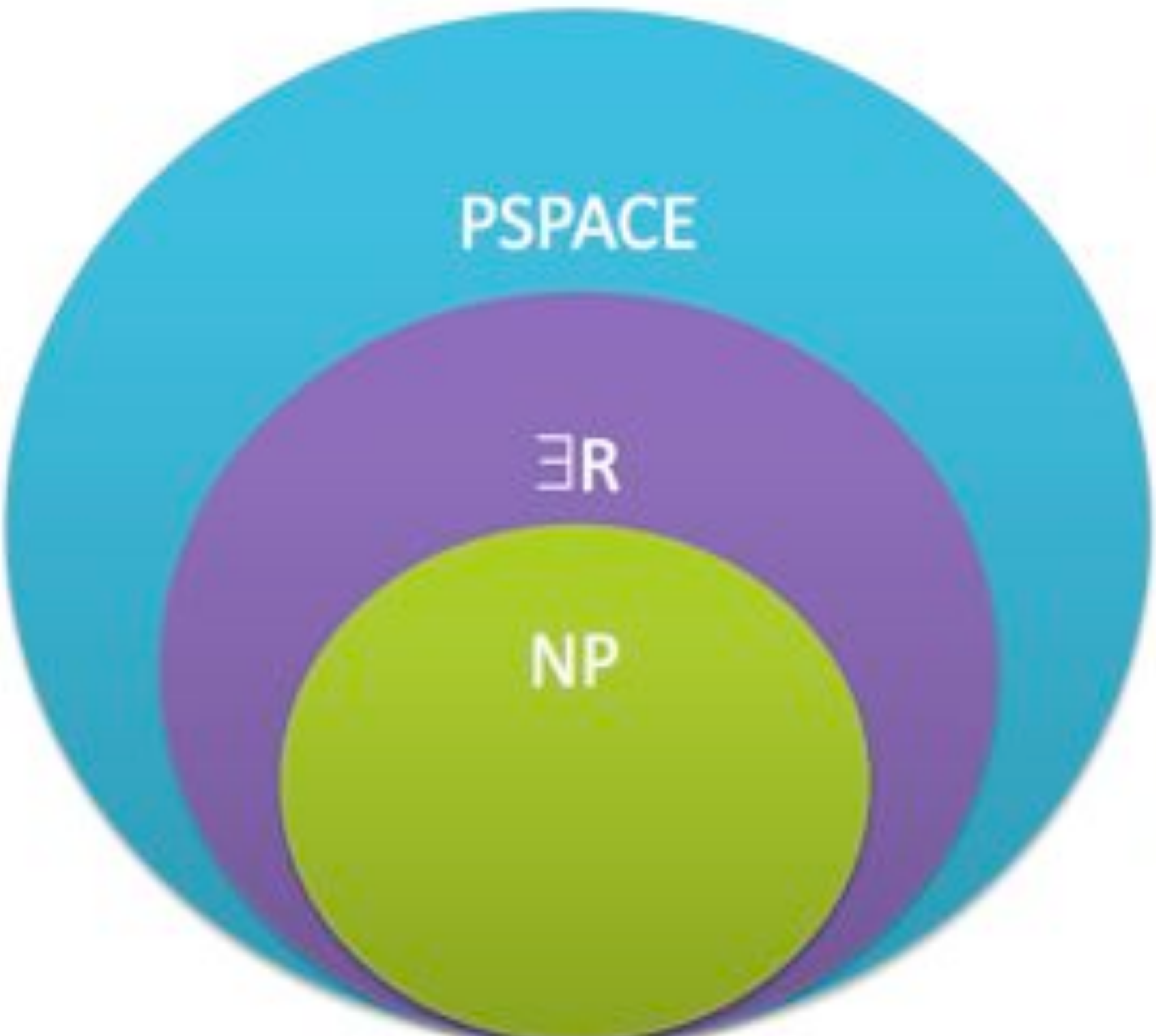
Recap of CAD-based Approach

- Used by Mathematica
- Doesn't require explicit computation of Phi(y)
- But, it is *eager* and *pessimistic:*
  FULL CAD Projection (expensive!!!)
- Our new approach is *lazy* and *optimistic*
- We build on *nlsat*, a CDCL-like approach to the Existential fragment of RCF

nlsat: CDCL-like approach to ExRCF

- Start building model for formula immediately, without first going through projection phase
- When conflict arises, use *projection on demand*
- Real-algebraic analogue of *conflict clauses* **generalize** a non-extendable partial models to rule out a *delineable* region containing them
- Non-chronological backtracking

How hard is ∃R?



PSPACE membership
Canny – 1988,
Grigor'ev – 1988

NP-hardness

x is "Boolean" $\rightarrow$ $x(x-1) = 0$

x or y or z $\rightarrow$ $x + y + z > 0$

nlsat: CDCL-like approach to ExRCF

Two kinds of decision

1. case-analysis (Boolean)
2. model construction (CAD lifting)

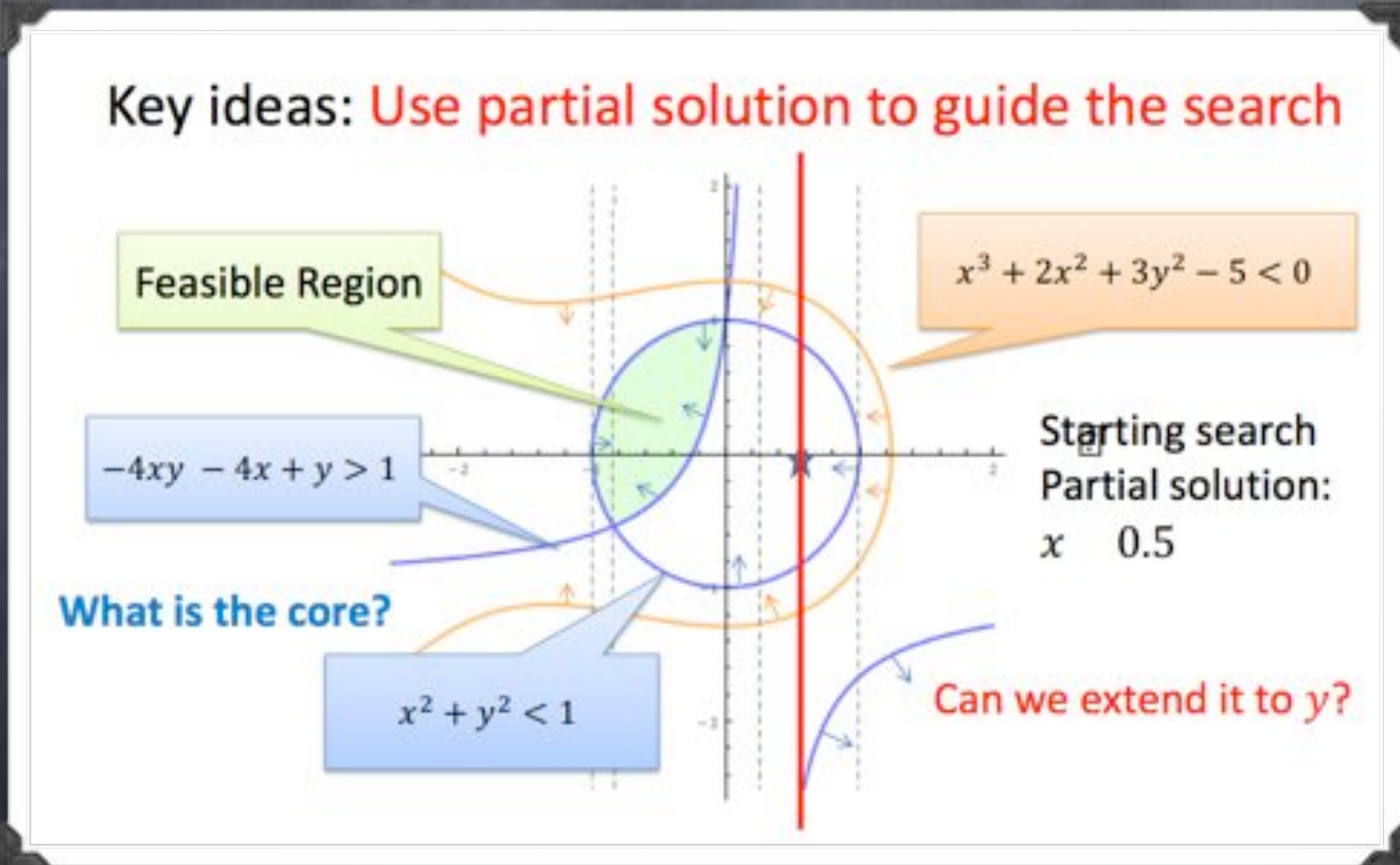
Parametric calculus: $explain(F, M)$

Finite basis explanation function

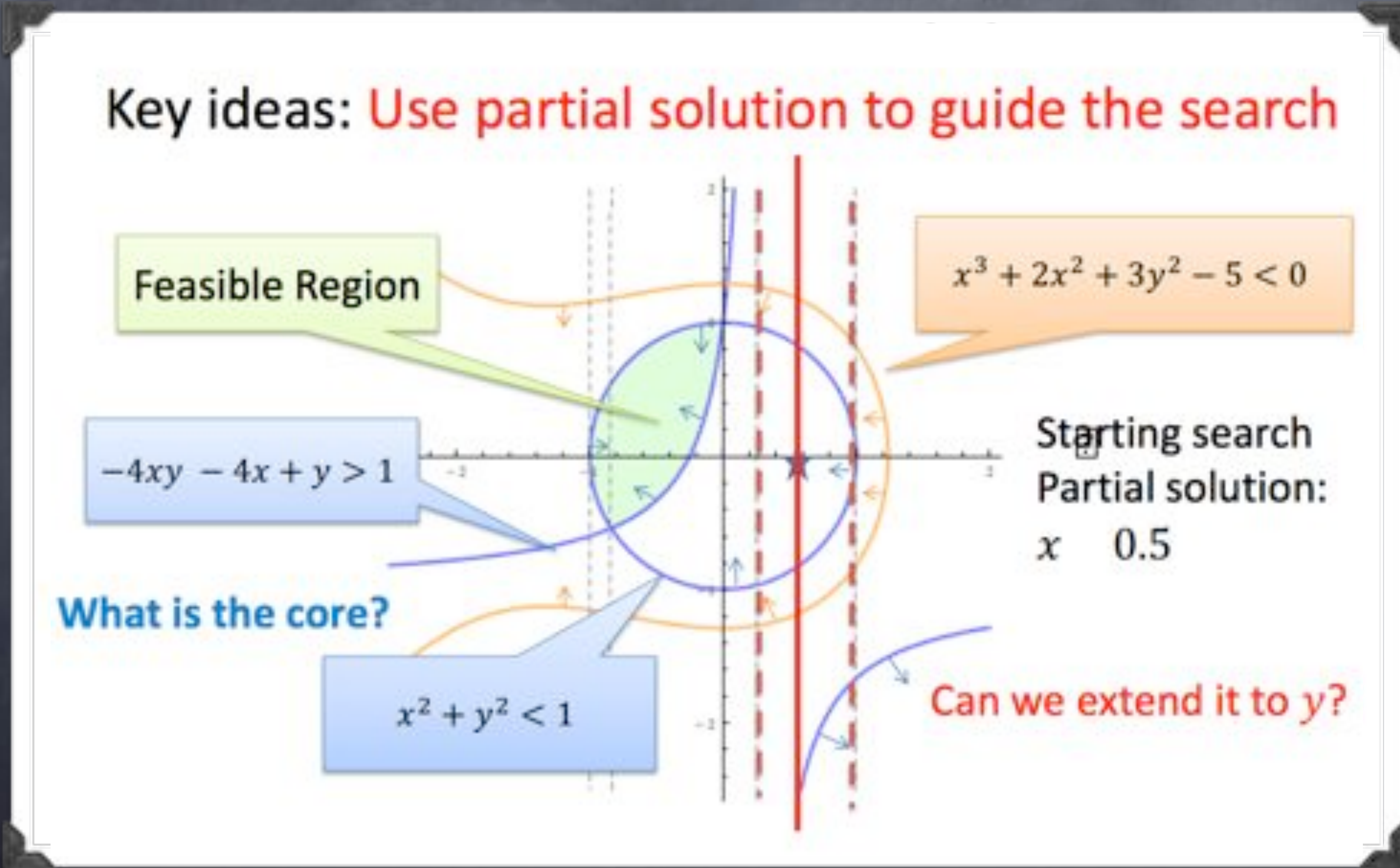
Explanations may contain new literals

They evaluate to false in the current state

nlsat: CDCL-like approach to ExRCF



nlsat: CDCL-like approach to ExRCF



nlsat: CDCL-like approach to ExRCF

Key ideas: Solution based Project/Saturate

$$\mathrm{P}_c(A, x) = \bigcup_{f \in A} \mathrm{coeff}(f, x) \cup \bigcup_{\substack{f \in A \\ g \in \mathrm{R}(f,x)}} \mathrm{psc}(g, g'_x, x) \cup \bigcup_{\substack{i<j \\ g_i \in \mathrm{R}(f_i,x) \\ g_j \in \mathrm{R}(f_j,x)}} \mathrm{psc}(g_i, g_j, x)$$

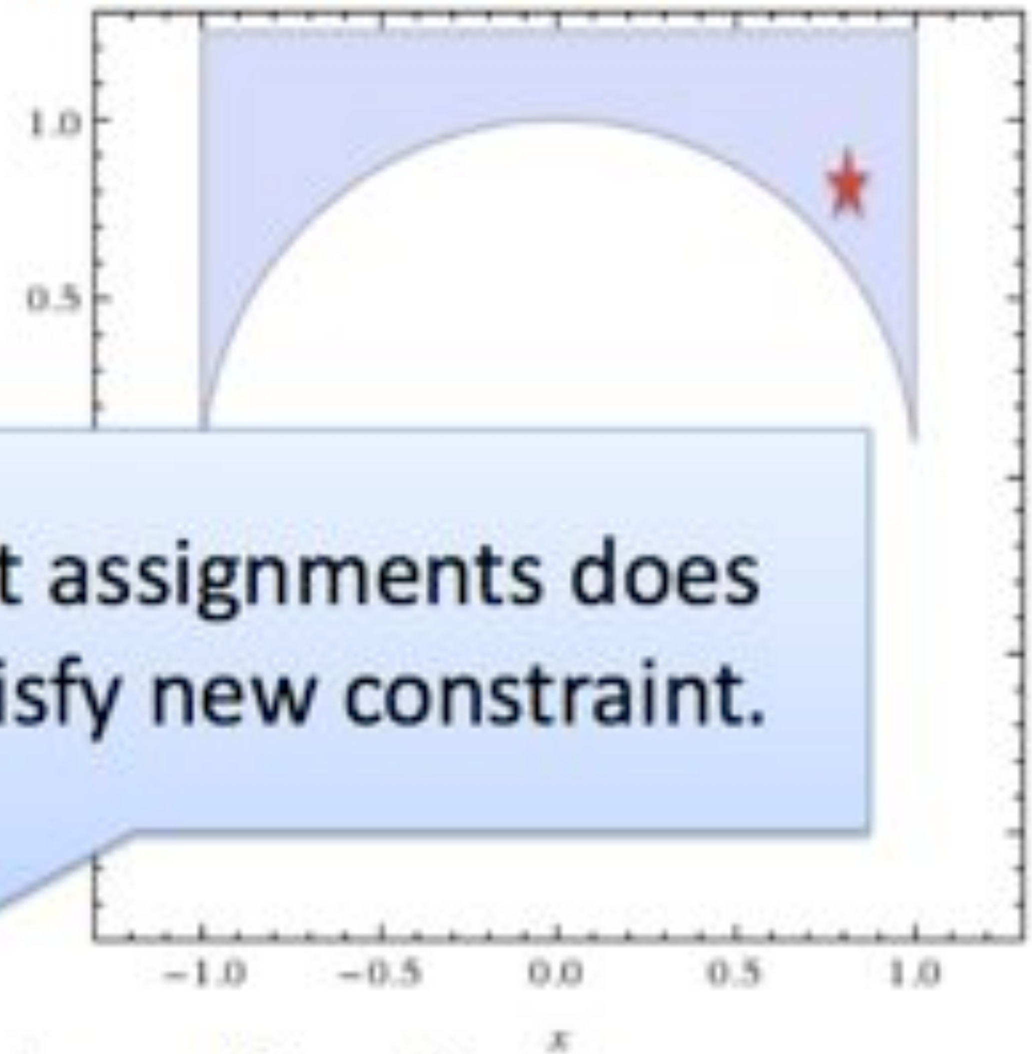
Standard project operators are pessimistic.
Coefficients can vanish!

nlsat: CDCL-like approach to ExRCF

Key ideas: Lemma Learning

Prevent a **Conflict** from happening again.

Current assignment
$x \rightarrow 0.75$
$y \rightarrow 0.75$



Current assignments does not satisfy new constraint.

Conflict
$x^2 + y^2 + z^2 < 1$

Lemma
$$-1 < x < 1 \ \wedge\ y > root_2(1 - \tilde{y}^2 - x^2) \ \Rightarrow \bot$$

CAD-based optimization + nlsat = ?

CAD-based Optimization uses projection eagerly and pessimistically

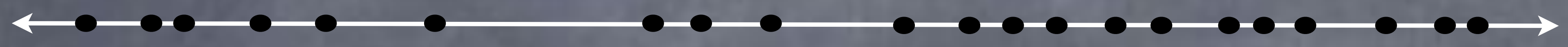
nlsat solves Exists RCF by using projection lazily and optimistically

...but how can we combine the two?

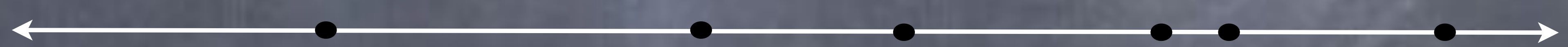
Key difficulty: Sweeping L to R

- We use coordinate function y to represent the objective function
- Then, we need to *sweep* along all possible values of y from **Left to Right**
- After CAD projection, we can do this
- But, what about with nlsat?

Key difficulty: Sweeping L to R



vs



Key difficulty: Sweeping L to R

vs

where to start? how to move to `next’ region?

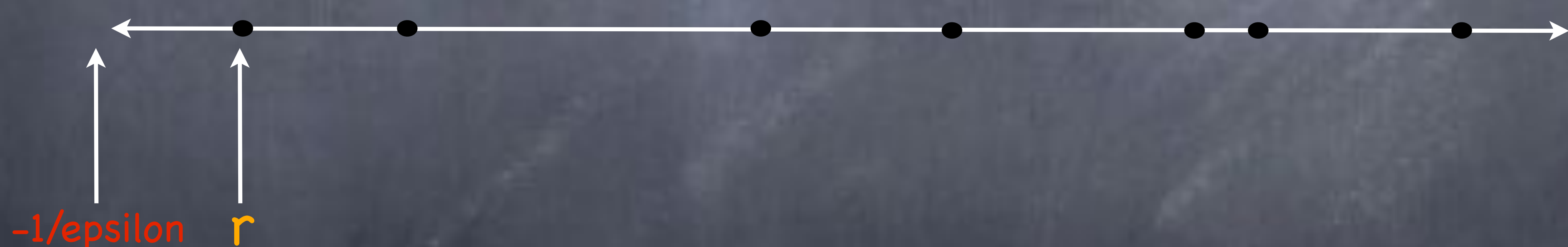
Key difficulty: Sweeping L to R

Key idea:
RCFs containing infinitesimals!

where to start? how to move to `next' region?

Key difficulty: Sweeping L to R

Key idea:
RCFs containing infinitesimals!



where to start? how to move to `next' region?

Key difficulty: Sweeping L to R

Key idea:
RCFs containing infinitesimals!

where to start? how to move to `next' region?

Satisfiability Modulo Assignment

We define the *satisfiability modulo assignment* problem as: given a formula $F[\bar{x}, \bar{y}]$ and an assignment $\{\bar{y} \mapsto \bar{v}\}$, produce one of the following outputs:

sat: if there is a $\bar{w}$ s.t. $\{\bar{x} \mapsto \bar{w}, \bar{y} \mapsto \bar{v}\}$ satisfies $F[\bar{x}, \bar{y}]$;

unsat(S): if there is no $\bar{w}$ s.t. $\{\bar{x} \mapsto \bar{w}, \bar{y} \mapsto \bar{v}\}$ satisfies $F[\bar{x}, \bar{y}]$, S is a formula that does not contain $\bar{x}$, S is implied by $F[\bar{x}, \bar{y}]$, and the assignment $\{\bar{y} \mapsto \bar{v}\}$ does not satisfy S.

Satisfiability Modulo Assignment

We define the *satisfiability modulo assignment* problem as: given a formula $F[\bar{x}, \bar{y}]$ and an assignment $\{\bar{y} \mapsto \bar{v}\}$, produce one of the following outputs:

sat: if there is a $\bar{w}$ s.t. $\{\bar{x} \mapsto \bar{w}, \bar{y} \mapsto \bar{v}\}$ satisfies $F[\bar{x}, \bar{y}]$;

unsat(S): if there is no $\bar{w}$ s.t. $\{\bar{x} \mapsto \bar{w}, \bar{y} \mapsto \bar{v}\}$ satisfies $F[\bar{x}, \bar{y}]$, S is a formula that does not contain $\bar{x}$, S is implied by $F[\bar{x}, \bar{y}]$, and the assignment $\{\bar{y} \mapsto \bar{v}\}$ does not satisfy S.

If we view the assignment $\{\bar{y} \mapsto \bar{v}\}$ as a formula $\bar{y} = \bar{v}$, then the formula S is essentially an interpolant for the formulas $F[\bar{x}, \bar{y}]$ and $\bar{y} = \bar{v}$. We can also view S as a *generalization* of why $\{\bar{y} \mapsto \bar{v}\}$ cannot be extended to a full assignment that satisfies $F[\bar{x}, \bar{y}]$.

Satisfiability Modulo Assignment

Given a procedure P for the satisfiability modulo assignment problem, we say the (not necessarily finite) sequence $[\bar{v}_1, \bar{v}_2, \ldots]$ is a *no-good sampling* for $F[\bar{x}, \bar{y}]$ if

$$
\begin{array}{lll}
 & \mathsf{P}(F[\bar{x},\bar{y}], \{\bar{y} \mapsto \bar{v}_1\}) = \mathsf{unsat}(S_1), & G_1 = S_1 \\
\bar{v}_2 \text{ satisfies } G_1, & \mathsf{P}(F[\bar{x},\bar{y}], \{\bar{y} \mapsto \bar{v}_2\}) = \mathsf{unsat}(S_2), & G_2 = G_1 \wedge S_2 \\
\ldots & & \\
\bar{v}_i \text{ satisfies } G_{i-1}, & \mathsf{P}(F[\bar{x},\bar{y}], \{\bar{y} \mapsto \bar{v}_i\}) = \mathsf{unsat}(S_i), & G_i = G_{i-1} \wedge S_i \\
\ldots & &
\end{array}
$$

Note that each formula G_i does not contain $\bar{x}$, and can be viewed as a *good region* that does not contain any of the *bad assignments* $\{\bar{v}_1, \ldots, \bar{v}_i\}$.

We say a procedure P, for the satisfiability modulo assignment problem, has the *finite decomposition property* if every *no-good sampling* sequence is finite.

```
procedure Min(F(x⃗, y))
  G := true
  ε := MkInfinitesimal() (* create an infinitesimal value *)
  loop
    r := Min_0(G)
    case r of
      unsat ⇒ return unsat
      unbounded ⇒ v := −1/ε
      (inf, a) ⇒ v := a + ε
      (min, a) ⇒ v := a
    end
    case Check(F(x⃗, y), {y ↦ v}) of
      sat ⇒ return r
      (unsat, S) ⇒ G := G ∧ S
    end
  end
```

Min_0: Procedure for Univariate Optimization Problem

Check: Procedure for SAT Modulo Assignment Problem, with support for RCFs containing *infinitesimals*, and satisfying the *finite decomposition* property.

The RCF Optimization Problem

Input: A quantifier-free RCF formula $F(\vec{x}, y)$.

Output (with 'is (un)sat' meaning 'is (un)satisfiable over $\mathbb{R}$'):

$$\begin{cases} \mathsf{unsat}, & \text{if } F(\vec{x}, y) \text{ is unsat,} \\ \mathsf{unbounded}, & \text{if for all } v \text{ exists } w < v \text{ s.t. } F(\vec{x}, w) \text{ is sat,} \\ (\mathsf{inf}, a), & \text{if for all } v \leq a,\ F(\vec{x}, a) \text{ is unsat, and} \\ & \text{for all } \epsilon > 0 \text{ exists } v \in (a, a+\epsilon) \text{ s.t. } F(\vec{x}, v) \text{ is sat,} \\ (\mathsf{min}, a), & \text{if } F(\vec{x}, a) \text{ is sat, and for all } v < a,\ F(\vec{x}, v) \text{ is unsat.} \end{cases}$$

Conclusion

- A CDCL-like approach to exact nonlinear global optimization over the real numbers (and all RCFs)
- Three main conceptual ingredients:

CAD-based approach to optimization

eager method for nonlinear optimization, in Mathematica v9.x

nlsat / existential CAD `on demand'

lazy CDCL-like approach to Exists RCF, in Z3 (Jovanović - de Moura, 2012)

computable nonstandard RCFs

computable RCFs containing infinitesimals (de Moura - Passmore, 2013)

Our main CADE talk on Wednesday!

Thank you!