Satisfiability Modulo Theories for Arithmetic Problems

... and a lot of references

Stanford University

Contains mostly other people’s work!
Satisfiability Modulo Theories

\[ \exists x. \varphi(x) \]

Is an existential first-order formula satisfiable?
$\exists \overline{x}. \varphi(\overline{x})$

Is an existential first-order formula satisfiable?

Theories:
- uninterpreted functions
- arrays
- bit-vectors
- floating-point numbers
- arithmetic
- datatypes
- strings
- ...
$\exists x. \varphi(x)$

Is an existential first-order formula satisfiable?

Theories:
- uninterpreted functions
- arrays
- bit-vectors
- floating-point numbers
- arithmetic
- datatypes
- strings
- ...

Extensions:
- model generation
- unsat cores
- quantifiers
- optimization queries
- interpolants
- formal proofs
- ...
SMT solving – CDCL(T)

$\varphi$

SAT solver

SAT or UNSAT

SAT or UNSAT

Theory solver

SAT + witness

or

UNSAT + reason

theory constraints
\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \]

SMT solving – CDCL(T)

SAT solver → SAT or UNSAT

theory constraints

SAT or UNSAT

SAT + witness or

UNSAT + reason

Theory solver
$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3)$

SAT solver

SAT or UNSAT

Theory solver

SAT + witness or UNSAT + reason

$\{x > 0, x^2 > 0\}$
SMT solving – CDCL(T)

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \]

\{ x > 0, x^2 > 0 \} \quad \text{SAT + } x \mapsto 1

SAT solver \quad \text{SAT or } \text{UNSAT}

Theory solver
SMT solving – CDCL(T)

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \]

SAT solver

\{x > 0, x^2 > 0, x^3 < 0\}

SAT or UNSAT

Theory solver

SAT + \(x \mapsto 1\)
SAT solver

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \]

SAT or UNSAT

Theory solver

\{ x > 0, x^2 > 0, x^3 < 0 \}  \quad UNSAT + \{ x > 0, x^3 < 0 \}
$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0)$$
SMT solving – CDCL(T)

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \]

\[
\{x > 0, \neg x^3 < 0, x = 3\} \quad \text{UNSAT} + \{x > 0, x^3 < 0\}
\]
\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \]

SMT solving – CDCL(T)

\{x > 0, \neg x^3 < 0, x = 3\}

SAT or UNSAT

SAT + x \mapsto 3
SMT solving – CDCL(T)

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \]

SAT solver \rightarrow SAT or UNSAT

\{ x > 0, \neg x^3 < 0, x = 3, x^2 > 0 \}

SAT + \ x \mapsto 3
\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \]

\[
\{ x > 0, \neg x^3 < 0, x = 3, x^2 > 0 \} \quad \text{SAT + } x \mapsto 3
\]

\[
\begin{array}{c}
\text{SAT solver} \\
\text{SAT, } x \mapsto 3
\end{array}
\]

\[
\begin{array}{c}
\text{Theory solver} \\
\text{SAT + } x \mapsto 3
\end{array}
\]
\( x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \)

SAT solving – CDCL(T)

SAT, \( x \mapsto 3 \)

\( \{x > 0, \neg x^3 < 0, x = 3, x^2 > 0\} \)

SAT + \( x \mapsto 3 \)

nonlinear real arithmetic
SMT solving – CDCL(T)

\[ x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0) \]

\[ \{x > 0, \neg x^3 < 0, x = 3, x^2 > 0\} \]

Also: NLSAT/MCSAT [Jovanović et al. 2012] [Moura et al. 2013]
Nonlinear Real Arithmetic:

- **real variables** \( v \) := \( x_i \in \mathbb{R} \)
- **constants** \( c \) := \( q \in \mathbb{Z} \)
- **terms** \( t \) := \( v \mid c \mid t \mid t \mid t \cdot t \)
- **atoms** \( a \) := \( t \sim 0 \), \( \sim \in \{<, >, \leq, \geq, =, \neq\} \)

**Intuition:** polynomials over real variables compared to zero.

**Does cover:**
- \( t > t \)
- rational constants, division (encoding with auxiliary variables)

**Does not cover:** transcendental constants, non-polynomial functions

**Linear arithmetic:** essentially a solved problem.

Use Simplex (or sometimes Fourier-Motzkin)

- SMT for NRA
- SMT for Nonlinear Real Arithmetic
Here: Theory of the Reals

Nonlinear Real Arithmetic:

- real variables $v := x_i \in \mathbb{R}$
- constants $c := q \in \mathbb{Z}$
- terms $t := v \mid c \mid t + t \mid t \cdot t$
- atoms $a := t \sim 0, \sim \in \{<,>,\leq,\geq,=,\neq\}$
Here: Theory of the Reals

Nonlinear Real Arithmetic:

- real variables $v := x_i \in \mathbb{R}$
- constants $c := q \in \mathbb{Z}$
- terms $t := v \mid c \mid t + t \mid t \cdot t$
- atoms $a := t \sim 0$, $\sim \in \{<, >, \leq, \geq, =, \neq\}$

Intuition: polynomials over real variables compared to zero.
Here: Theory of the Reals

Nonlinear Real Arithmetic:

- real variables \( v := x_i \in \mathbb{R} \)
- constants \( c := q \in \mathbb{Z} \)
- terms \( t := v \mid c \mid t + t \mid t \cdot t \)
- atoms \( a := t \sim 0, \sim \in \{<, >, \leq, \geq, =, \neq\} \)

Intuition: polynomials over real variables compared to zero.

Does cover: \( t > t \), rational constants, division (encoding with auxiliary variables)

Does not cover: transcendental constants, non-polynomial functions
Here: Theory of the Reals

Nonlinear Real Arithmetic:

- real variables $v := x_i \in \mathbb{R}$
- constants $c := q \in \mathbb{Z}$
- terms $t := v | c | t + t | t \cdot t$
- atoms $a := t \sim 0$, $\sim \in \{<, >, \leq, \geq, =, \neq\}$

Intuition: polynomials over real variables compared to zero.

**Does cover:** $t > t$, rational constants, division (encoding with auxiliary variables)
**Does not cover:** transcendental constants, non-polynomial functions

Linear arithmetic: essentially a solved problem.
Use Simplex (or sometimes Fourier-Motzkin)
Theory of the Reals in a nutshell

- **complete** (we have decision procedures that are sound and complete)
- admits **quantifier elimination** (quantifiers are conceptually easy)
complete (we have decision procedures that are sound and complete)

admits quantifier elimination (quantifiers are conceptually easy)

Some methods:

[Tarski 1951] Tarski: first complete method, non-elementary complexity

[Buchberger 1965] Gröbner bases: limited applicability, standard tool in CA

[Collins 1974] CAD: complete, doubly exponential complexity

[Weispfenning 1988] VS: up to bounded degree, singly exponential complexity


[Fontaine et al. 2017] Subtropical satisfiability: incomplete reduction to LRA

[Irfan 2018] Linearization: incomplete, axiom instantiation

[Ábrahám et al. 2021] CDCAC: conflict-driven CAD

and some more...
SC-Square

SC²
Satisfiability Checking and Symbolic Computation
Bridging Two Communities to Solve Real Problems

Consortium of the EU-CSA project

University of Bath
RWTH Aachen
Fondazione Bruno Kessler
Università degli Studi di Genova
Maplesoft Europe Ltd
Université de Lorraine (LORIA)
Coventry University
University of Oxford
Universität Kassel
Max Planck Institut für Informatik
Universität Linz

James Davenport; Russell Bradford
Erika Ábrahám; Viktor Levandovskyy
Alberto Griggio; Alessandro Cimatti
Anna Biagelli
Jürgen Gerhard; Stephen Forrest
Pascal Fontaine
Matthew England
Daniel Kroening; Martin Brain
Werner Selen; John Abbott
Thomas Sturm
Tudor Jebelean; Bruno Buchberger; Wolfgang Windsteiger; Roxana-Maria Holom
Overview

1. SMT for NRA
2. Linearization
3. Interval Constraint Propagation
4. Subtropical Satisfiability
5. Gröbner Bases
6. Virtual Substitution
7. Cylindrical Algebraic Decomposition
8. Conflict-Driven Cylindrical Algebraic Coverings
9. Related topics
Incremental linearization

implicitly linearize: $x \cdot y \leadsto a_{x \cdot y}$

$$x > 2 \land y > -1 \land x \cdot y < 2$$
implicitly linearize: \( x \cdot y \rightsquigarrow a_{x \cdot y} \)

\[
x > 2 \land y > -1 \land x \cdot y < 2
\]

Model: \( x \mapsto 3, y \mapsto 0, x \cdot y \mapsto 1 \)

Lemma: \( y = 0 \Rightarrow x \cdot y = 0 \)
Implicit linearization: \( x \cdot y \sim a_{x \cdot y} \)

\[
x > 2 \land y > -1 \land x \cdot y < 2
\]

Model: \( x \mapsto 3, y \mapsto 0, x \cdot y \mapsto 1 \)

Lemma: \( y = 0 \Rightarrow x \cdot y = 0 \)

Model: \( x \mapsto 3, y \mapsto 1, x \cdot y \mapsto 0 \)

Lemma: \( (x = 3 \land y = 1) \Rightarrow x \cdot y = 3 \)
implicitly linearize: \( x \cdot y \leadsto a_{x \cdot y} \)

\[
x > 2 \land y > -1 \land x \cdot y < 2
\]

Model: \( x \mapsto 3, y \mapsto 0, x \cdot y \mapsto 1 \)

Lemma: \( y = 0 \Rightarrow x \cdot y = 0 \)

Model: \( x \mapsto 3, y \mapsto 1, x \cdot y \mapsto 0 \)

Lemma: \( (x = 3 \land y = 1) \Rightarrow x \cdot y = 3 \)
Incremental linearization

implicitly linearize: \( x \cdot y \mapsto a_{x \cdot y} \)

\[
x > 2 \land y > -1 \land x \cdot y < 2
\]
Model: \( x \mapsto 3, y \mapsto 0, x \cdot y \mapsto 1 \)
Lemma: \( y = 0 \Rightarrow x \cdot y = 0 \)
Model: \( x \mapsto 3, y \mapsto 1, x \cdot y \mapsto 0 \)
Lemma: \( (x = 3 \land y = 1) \Rightarrow x \cdot y = 3 \)

[Cimatti et al. 2018]
implicitly linearize: $x \cdot y \mapsto a_{x \cdot y}$

$$x > 2 \land y > -1 \land x \cdot y < 2$$

Model: $x \mapsto 3, y \mapsto 0, x \cdot y \mapsto 1$

Lemma: $y = 0 \implies x \cdot y = 0$

Model: $x \mapsto 3, y \mapsto 1, x \cdot y \mapsto 0$

Lemma: $(x = 3 \land y = 1) \implies x \cdot y = 3$

$$(x \leq 3 \land y \leq 1) \lor (x \geq 3 \land y \geq 1)$$

$\iff (x \cdot y \geq 1 \cdot x + 3 \cdot y - 3 \cdot 1)$
Incremental linearization – schemas

\[ \top \Rightarrow (t = 0 \lor t \neq 0) \]

sign

\[ x > 0 \land y > 0 \Rightarrow xy > 0 \]
\[ x = 0 \Rightarrow xyz = 0 \]

magnitude

\[ |x| > |y| \Rightarrow |xz| > |yz| \]
\[ |z| > |y| \land |u| > |w| \land |x| \geq 1 \Rightarrow |zuwx| > |yw| \]

bounds

resolution bounds

\[ x > 0 \land y > z + w \Rightarrow xy > x(z + w) \]

\[ y \geq 0 \land s \leq xz \land xy \leq t \Rightarrow ys \leq zt \]

\[ (x \leq 3 \land y \leq 1) \lor (x \geq 3 \land y \geq 1) \Rightarrow xy \geq x + 3y - 3 \]
Intuition: *iteratively teach the linear solver* about the nonlinear parts, add lemmas that cut away unsatisfiable regions.
Intuition: *iteratively teach the linear solver* about the nonlinear parts, add lemmas that cut away unsatisfiable regions.

**Problems:** difficult to identify models (linear solver only finds corners), linear solver only finds rational assignments \((x^2 = 2)\)
Intuition: **iteratively teach the linear solver** about the nonlinear parts, add lemmas that cut away unsatisfiable regions.

**Problems:** difficult to identify models (linear solver only finds corners), linear solver only finds rational assignments \((x^2 = 2)\)

**Extensions:**
- Repair model (if easily possible)
- Transcendental functions (\(\sin, \cos, \ldots\))
- extended operators in general

**Question**
Better linearization lemmas? Linearization lemmas for other functions?
\[ y > x^2 \land y < -x^2 + 2x \land y \leq 1 - x \land x \times y \]
\( y > x^2 \land y < -x^2 + 2x \land y \leq 1 - x \)
\( y > x^2 \Rightarrow y \in (0, \infty) \)
\( x \times y \Rightarrow (\infty, \infty) \times (0, \infty) \)
Interval Constraint Propagation

\[ y > x^2 \land y < -x^2 + 2x \land y \leq 1 - x \]
\[ y > x^2 \Rightarrow y \in (0, \infty) \]
\[ x > 0.5x^2 + y \Rightarrow x \in (0, \infty) \]

\[ x \times y \]
\[ (-\infty, \infty) \times (0, \infty) \]
\[ (0, \infty) \times (0, \infty) \]
Interval Constraint Propagation

\[ y > x^2 \land y < -x^2 + 2x \land y \leq 1 - x \]
\[ y > x^2 \Rightarrow y \in (0, \infty) \]
\[ x > 0.5x^2 + y \Rightarrow x \in (0, \infty) \]
\[ x \leq -y + 1 \Rightarrow x \in (0, 1) \]
\[ x \times y \]
\[ (-\infty, \infty) \times (0, \infty) \]
\[ (0, \infty) \times (0, \infty) \]
\[ (0, 1) \times (0, \infty) \]
Interval Constraint Propagation

\[ y > x^2 \land y < -x^2 + 2x \land y \leq 1 - x \]

\[ x \times y \]

\[ (\infty, \infty) \times (0, \infty) \]

\[ y > x^2 \Rightarrow y \in (0, \infty) \]

\[ (0, \infty) \times (0, \infty) \]

\[ x \geq 0.5x^2 + y \Rightarrow x \in (0, \infty) \]

\[ (0, 1) \times (0, \infty) \]

\[ x \leq -y + 1 \Rightarrow x \in (0, 1) \]

\[ (0, 1) \times (0, \infty) \]

\[ y \leq -x + 1 \Rightarrow y \in (0, 1) \]

\[ (0, 1) \times (0, 1) \]
\begin{align*}
y > x^2 & \land y < -x^2 + 2x \land y \leq 1 - x \\
y > x^2 & \implies y \in (0, \infty) \\
x > 0.5x^2 + y & \implies x \in (0, \infty) \\
x \leq -y + 1 & \implies x \in (0, 1) \\
y \leq -x + 1 & \implies y \in (0, 1) \\
guess \ midpoint & \ (0.5, 0.5) \in (0, 1) \times (0, 1)
\end{align*}
Core idea:

- Maintain interval assignment (that represents the current box)
- Perform over-approximating contractions until
  - the current box is empty (UNSAT),
  - we can guess a model (SAT), or
  - we reach a threshold.
- When reaching a threshold
  - we terminate with unknown or
  - split: $x \in [0, 5] \leadsto (x < 3 \lor x \geq 3)$
Core idea:

- Maintain **interval assignment** (that represents the current box)
- Perform **over-approximating** contractions until
  - the current box is **empty** (UNSAT),
  - we can **guess a model** (SAT), or
  - we reach a **threshold**.
- **Incomplete solving procedure**
- Used as **preprocessor** for other techniques [Loup et al. 2013]
- **Delicate tuning** of heuristics (splitting, thresholds, model guessing)

**Question**

Sensible initial bounds? Better propagation schemas?
Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) - x$
Core idea: reduce \( p = 0 \) to a linear problem in the exponents of \( p \)

- Assume \( p(1, \ldots, 1) < 0 \) (otherwise consider \( -p \))
- Find \( x \in \mathbb{R}^n_+ \) such that \( p(x) > 0 \)
- Solve \( p(y) = 0 \) with \( y \) on the line \((1, \ldots, 1) - x\)
- Incomplete (no such \( y \) exists, though \( p = 0 \) has a solution)
Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) \mapsto x$
- Incomplete (no such $y$ exists, though $p = 0$ has a solution)

Core problem: How to find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$?
Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) \leftarrow x$
- Incomplete (no such $y$ exists, though $p = 0$ has a solution)

Core problem: How to find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$?

For $n = 1$: $\lim_{x \to \infty} p(x) = \infty$ if $\text{lcoeff}(p) > 0$. Increase $x$ as necessary.
Core idea: reduce \( p = 0 \) to a linear problem in the exponents of \( p 

- Assume \( p(1, \ldots, 1) < 0 \) (otherwise consider \( -p \))
- Find \( x \in \mathbb{R}^n_+ \) such that \( p(x) > 0 \)
- Solve \( p(y) = 0 \) with \( y \) on the line \((1, \ldots, 1) - x \)
- Incomplete (no such \( y \) exists, though \( p = 0 \) has a solution)

Core problem: How to find \( x \in \mathbb{R}^n_+ \) such that \( p(x) > 0 \)?

For \( n = 1 \): \( \lim_{x \to \infty} p(x) = \infty \) if \( \text{lcoeff}(p) > 0 \). Increase \( x \) as necessary.
For \( n \geq 2 \): search direction in exponent space such that the largest exponent in this direction is positive. Increase \( x \) in this direction as necessary.
Subtropical satisfiability

Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) - x$
- Incomplete (no such $y$ exists, though $p = 0$ has a solution)

Core problem: How to find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$?

For $n = 1$: $\lim_{x \to \infty} p(x) = \infty$ if $\text{lcoeff}(p) > 0$. Increase $x$ as necessary.
For $n \geq 2$: search direction in exponent space such that the largest exponent in this direction is positive. Increase $x$ in this direction as necessary.

$$p = -y + 2xy^3 - 3x^2y^2 - x^3 - 4x^3y^3$$
Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}_+^n$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) - x$
- Incomplete (no such $y$ exists, though $p = 0$ has a solution)

Core problem: How to find $x \in \mathbb{R}_+^n$ such that $p(x) > 0$?

For $n = 1$: \( \lim_{x \to \infty} p(x) = \infty \) if $\text{lcoeff}(p) > 0$. Increase $x$ as necessary.  
For $n \geq 2$: search direction in exponent space such that the largest exponent in this direction is positive. Increase $x$ in this direction as necessary.

\[
p = -y + 2xy^3 - 3x^2y^2 - x^3 - 4x^3y^3
\]

Find hyperplane that separates a positive node
Core idea: reduce $p = 0$ to a linear problem in the exponents of $p$

- Assume $p(1, \ldots, 1) < 0$ (otherwise consider $-p$)
- Find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$
- Solve $p(y) = 0$ with $y$ on the line $(1, \ldots, 1) - x$
- Incomplete (no such $y$ exists, though $p = 0$ has a solution)

Core problem: How to find $x \in \mathbb{R}^n_+$ such that $p(x) > 0$?

For $n = 1$: $\lim_{x \to \infty} p(x) = \infty$ if $\text{lcoeff}(p) > 0$. Increase $x$ as necessary.

For $n \geq 2$: search direction in exponent space such that the largest exponent in this direction is positive. Increase $x$ in this direction as necessary.

$$p = -y + 2xy^3 - 3x^2y^2 - x^3 - 4x^3y^3$$

Find hyperplane that separates a positive node

Encoding in QF_LRA

Growing degree only impacts coefficient size
Gröbner basis

- Canonical generators for a polynomial ideal
- For us: Normal form for sets of polynomials
- Maintains set of common complex roots
- The workhorse of computer algebra for polynomial equalities
- Mature implementations (every CAS)
- Doubly exponential in worst case, but usually much faster.

Relevant for SMT:
\[ \exists x \in \mathbb{C}^n. p(x) = 0 \]

But: What about inequalities? How to go from \( \mathbb{C} \) to \( \mathbb{R} \)?

see [Junges 2012] for some approaches.

Question: How to construct models? How to obtain infeasible subsets?
Gröbner basis

- Canonical generators for a polynomial ideal
- For us: Normal form for sets of polynomials
- Maintains set of common complex roots
- The workhorse of computer algebra for polynomial equalities
- Mature implementations (every CAS)
- Doubly exponential in worst case, but usually much faster.

Relevant for SMT: \( \exists x \in \mathbb{C}^n. p(x) = 0 \)
Gröbner basis

- Canonical generators for a polynomial ideal
- For us: Normal form for sets of polynomials
- Maintains set of common complex roots
- The workhorse of computer algebra for polynomial equalities
- Mature implementations (every CAS)
- Doubly exponential in worst case, but usually much faster.

Relevant for SMT: $\exists x \in \mathbb{C}^n. p(x) = 0$

But: What about inequalities? How to go from $\mathbb{C}$ to $\mathbb{R}$? see [Junges 2012] for some approaches.

Question

How to construct models? How to obtain infeasible subsets?
Core idea:

- Use **solution formula** to solve polynomial equation for \( x \)
- **Substitute value** for \( x \) into remaining equations
- Repeat for remaining variables
Core idea:
- Use solution formula to solve polynomial equation for $x$
- Substitute value for $x$ into remaining equations
- Repeat for remaining variables

What about inequalities?
- Construct test candidates for all sign-invariant regions in $x$
- Always try the roots and the smallest values of the intermediate intervals
- Introduces special terms $t + \varepsilon$ and $-\infty$
Algorithmic core: a collection of substitution rules

Example: Substitute $e + \varepsilon$ for $x$ into $a \cdot x^2 + b \cdot x + c > 0$:

\[
\begin{align*}
\lor & \quad (a x^2 + bx + c > 0)[e//x] \\
\lor & \quad (a x^2 + bx + c = 0)[e//x] \land (2ax + b > 0)[e//x] \\
\lor & \quad (a x^2 + bx + c = 0)[e//x] \land (2ax + b = 0)[e//x] \land (2a > 0)[e//x]
\end{align*}
\]
Algorithmic core: a collection of substitution rules

Example: Substitute $e + \varepsilon$ for $x$ into $a \cdot x^2 + b \cdot x + c > 0$:

\[
\begin{align*}
\vee & \quad (ax^2 + bx + c > 0)[e//x] \\
\vee & \quad (ax^2 + bx + c = 0)[e//x] \land (2ax + b > 0)[e//x] \\
\vee & \quad (ax^2 + bx + c = 0)[e//x] \land (2ax + b = 0)[e//x] \land (2a > 0)[e//x]
\end{align*}
\]

Not always applicable:

- Solution formulas only exist up to degree four
- The above rule may introduce a degree growth
- Efficient if applicable
- [Košta et al. 2015] uses FO formulas, allows arbitrary but fixed degrees (needs precomputed substitution rules obtained by quantifier elimination)
The core idea: sign-invariance (or rather truth-table equivalence)

$$\text{sgn}(p(a)) = \text{sgn}(p(b)) \quad \forall p \in \varphi \quad \Rightarrow \quad \varphi(a) = \varphi(b)$$

For our purpose, $a$ and $b$ are equivalent!
The core idea: sign-invariance (or rather truth-table equivalence)

\[ \text{sgn}(p(a)) = \text{sgn}(p(b)) \quad \forall p \in \varphi \quad \Rightarrow \quad \varphi(a) = \varphi(b) \]

For our purpose, \( a \) and \( b \) are equivalent!

Construct a sign-invariant decomposition of \( \mathbb{R}^n \):

\[ \text{cell } C \subset \mathbb{R}^n : \forall a, b \in C : \varphi(a) = \varphi(b) \]

Abstraction: \( \mathbb{R}^n \) to finite set of cells, consider a single \( a \in C \) per cell.
The core idea: sign-invariance (or rather truth-table equivalence)

\[ \text{sgn}(p(a)) = \text{sgn}(p(b)) \quad \forall p \in \varphi \Rightarrow \varphi(a) = \varphi(b) \]

For our purpose, \( a \) and \( b \) are equivalent!

Construct a sign-invariant decomposition of \( \mathbb{R}^n \):

\[
\text{cell } C \subset \mathbb{R}^n : \forall a, b \in C : \varphi(a) = \varphi(b)
\]

Abstraction: \( \mathbb{R}^n \) to finite set of cells, consider a single \( a \in C \) per cell.

\[ \varphi = (p > 0) \land (q < 0) \]

Solution space
The core idea: **sign-invariance** (or rather truth-table equivalence)

\[ \text{sgn}(p(a)) = \text{sgn}(p(b)) \quad \forall p \in \varphi \quad \Rightarrow \quad \varphi(a) = \varphi(b) \]

For our purpose, \( a \) and \( b \) are equivalent!

Construct a **sign-invariant decomposition** of \( \mathbb{R}^n \):

\[
\text{cell } C \subset \mathbb{R}^n : \forall a, b \in C : \varphi(a) = \varphi(b)
\]

Abstraction: \( \mathbb{R}^n \) to finite set of cells, consider a single \( a \in C \) per cell.

\[ \varphi = (p > 0) \land (q < 0) \]

Sign-invariant regions

### Cylindrical Algebraic Decomposition

Gereon Kremer | Stanford University | June 2, 2022

17/29
The core idea: **sign-invariance** (or rather truth-table equivalence)

\[ sgn(p(a)) = sgn(p(b)) \quad \forall p \in \varphi \quad \Rightarrow \quad \varphi(a) = \varphi(b) \]

For our purpose, \( a \) and \( b \) are equivalent!

Construct a **sign-invariant decomposition of** \( \mathbb{R}^n \):

\[
\text{cell } C \subset \mathbb{R}^n : \forall a, b \in C : \varphi(a) = \varphi(b)
\]

Abstraction: \( \mathbb{R}^n \) to finite set of cells, consider a single \( a \in C \) per cell.

\[ \varphi = (p > 0) \land (q < 0) \]

Sample points
The core idea: sign-invariance (or rather truth-table equivalence)

\[ \text{sgn}(p(a)) = \text{sgn}(p(b)) \quad \forall p \in \varphi \Rightarrow \varphi(a) = \varphi(b) \]

For our purpose, \( a \) and \( b \) are equivalent!

Construct a sign-invariant decomposition of \( \mathbb{R}^n \):

\[
\text{cell } C \subset \mathbb{R}^n : \forall a, b \in C : \varphi(a) = \varphi(b)
\]

Abstraction: \( \mathbb{R}^n \) to finite set of cells, consider a single \( a \in C \) per cell.

\[ \varphi = (p > 0) \land (q < 0) \]

Actual sample points

Arranged in cylinders
Cylindrical Algebraic Decomposition in $\mathbb{R}^2$

Proceed **dimension-wise**: project to lower-dimensional problem, lift results.
Cylindrical Algebraic Decomposition in \( \mathbb{R}^2 \)

Proceed dimension-wise: project to lower-dimensional problem, lift results.

**Intuition**

**Critical points**
Cylindrical Algebraic Decomposition in $\mathbb{R}^2$

Proceed dimension-wise: project to lower-dimensional problem, lift results.

Intuition

Critical points

Project sample
Proceed \textit{dimension-wise}: project to lower-dimensional problem, lift results.

\textbf{Intuition}

Critical points
Project sample
Solve 1-dim
Cylindrical Algebraic Decomposition in $\mathbb{R}^2$

Proceed *dimension-wise*: project to lower-dimensional problem, lift results.

**Intuition**
- Critical points
- Project sample
- Solve 1-dim
- Lift to 2-dim
Proceed **dimension-wise**: project to lower-dimensional problem, lift results.

**Intuition**
- Critical points
- Project sample
- Solve 1-dim
- Lift to 2-dim

**Implementation**
Proceed dimension-wise: project to lower-dimensional problem, lift results.

Intuition
- Critical points
- Project sample
- Solve 1-dim
- Lift to 2-dim

Implementation
- Project polynomials
- Resultant
Proceed dimension-wise: project to lower-dimensional problem, lift results.

**Intuition**
- Critical points
- Project sample
- Solve 1-dim
- Lift to 2-dim

**Implementation**
- Project polynomials
- Solve 1-dim

**Resultant**
Cylindrical Algebraic Decomposition in $\mathbb{R}^2$

Proceed dimension-wise: project to lower-dimensional problem, lift results.

**Intuition**
- Critical points
- Project sample
- Solve 1-dim
- Lift to 2-dim

**Implementation**
- Project polynomials
- Solve 1-dim
- Lift to 2-dim

resultant
Cylindrical Algebraic Decomposition in $\mathbb{R}^n$

Theory atoms $\rightarrow P_n \subset \mathbb{Z}[x_1..x_n]$ $\rightarrow$ Solutions

$S_n \subset S_{n-1} \times \mathbb{R}$

$P_{n-1} \subset \mathbb{Z}[x_1..x_{n-1}]$

$P_1 \subset \mathbb{Z}[x_1]$ $\rightarrow$ $S_1 \subset \mathbb{R}$

$S_2 \subset S_1 \times \mathbb{R}$
Cylindrical Algebraic Decomposition in $\mathbb{R}^n$

Theory atoms $\rightarrow P_n \subset \mathbb{Z}[x_1..x_n] \rightarrow S_n \subset S_{n-1}\times\mathbb{R} \rightarrow$ Solutions

$P_{n-1} \subset \mathbb{Z}[x_1..x_{n-1}] \rightarrow \cdots$ $\downarrow$

$Lift(P_k, S_{k-1})$ $\downarrow$

$P_1 \subset \mathbb{Z}[x_1] \rightarrow S_1 \subset \mathbb{R}$
Cylindrical Algebraic Decomposition in $\mathbb{R}^n$

Theory atoms $\rightarrow P_n \subset \mathbb{Z}[x_1..x_n]$

$P_{n-1} \subset \mathbb{Z}[x_1..x_{n-1}]$

$\text{Proj}(P_k)$

$P_1 \subset \mathbb{Z}[x_1]$

Solutions $\rightarrow S_n \subset S_{n-1} \times \mathbb{R}$

$Lift(P_k, S_{k-1})$

$\vdots$

$S_2 \subset S_1 \times \mathbb{R}$

$S_1 \subset \mathbb{R}$

Projection:

- Intersections (resultants)
- Flipping points (discriminants)
- Singularities (coefficients)
Cylindrical Algebraic Decomposition in $\mathbb{R}^n$

Theory atoms \( P_n \subset \mathbb{Z}[x_1..x_n] \)

\( P_{n-1} \subset \mathbb{Z}[x_1..x_{n-1}] \)

\( Proj(P_k) \)

\( P_1 \subset \mathbb{Z}[x_1] \)

Solutions

\( S_n \subset S_{n-1} \times \mathbb{R} \)

\( S_{n-1} \subset S_{n-2} \times \mathbb{R} \)

\( \ldots \)

\( Lift(P_k, S_{k-1}) \)

\( S_2 \subset S_1 \times \mathbb{R} \)

\( S_1 \subset \mathbb{R} \)

Projection:

- Intersections (resultants)
- Flipping points (discriminants)
- Singularities (coefficients)

Lifting:

- Substitution \( s \in S_k, \ p \in P_{k+1} \)
  \[ p(s) \rightarrow p' \in \mathbb{Z}[x_{k+1}] \]
- Isolate real roots of \( p' \)
Asymptotic complexity: \((n \cdot m)^{2^r}\) (\(r\) variables, \(m\) polynomials of degree \(n\))

Oftentimes way faster, but worst-case occurs in practice!

Best complete method that is known and implemented. [Hong 1991]

Active research:

- **Projection** [McCallum 1984] [McCallum 1988] [Hong 1990] [Lazard 1994] [Brown 2001] [McCallum 2001] [McCallum et al. 2016] [McCallum et al. 2019]; [Strzeboński 2000] [Seidl et al. 2003] [Jovanović et al. 2012] [Brown 2013] [Strzeboński 2014] [Brown et al. 2015]

- **Lifting** [Collins 1974] [Lazard 1994] [McCallum et al. 2016] [McCallum et al. 2019]

- **Equational constraints** [Collins 1998] [McCallum 1999] [McCallum 2001] [England et al. 2015] [Haehn et al. 2018] [Nair et al. 2019]

- **Variable ordering** [England et al. 2014] [Huang et al. 2014] [Nalbach et al. 2019] [Florescu et al. 2019]

- **Adaptions** [Jovanović et al. 2012] [Brown 2013] [Brown 2015] [Ábrahám et al. 2021]

Implementation needs groundwork: polynomial computation (resultants, multivariate gcd, optionally multivariate factorization), real algebraic numbers (representation, multivariate root isolation)
Core idea: use CAD techniques in a conflict-driven way.
My intuition: MCSAT turned into a theory solver.
Core idea: use **CAD techniques in a conflict-driven way**.

My intuition: MCSAT turned into a theory solver.

- Fix a **variable ordering**
- For the $k$th variable
  - Use constraints to **exclude unsatisfiable intervals**
  - **Guess** a value for the $k$th variable
  - Recurse to $k + 1$st variable and obtain
    - a full variable assignment ($\rightarrow$ return SAT)
    - or a covering for the $k + 1$st variable
  - Use **CAD machinery** to infer an interval for the $k$th variable
- Until the collected intervals form a **covering** for the $k$th variable
An example

\[ c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2 \]
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 \( x \)
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 \( x \mapsto 0 \)
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 \( x \mapsto 0 \)

\( c_1 \rightarrow y \notin (-1, \infty) \)
An example

\begin{align*}
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
\end{align*}

No constraint for $x$

Guess $x \mapsto 0$

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

$c_2 \rightarrow y \notin (-\infty, 0.75)$
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 \( x \rightarrow 0 \)

\[ c_1 \rightarrow y \notin (-1, \infty) \]
\[ c_2 \rightarrow y \notin (-\infty, 0.75) \]
\[ c_3 \rightarrow y \notin (-\infty, 0.5) \]
An example

$c_1 : 4 \cdot y < x^2 - 4$  \hspace{1cm}  $c_2 : 4 \cdot y > 4 - (x - 1)^2$  \hspace{1cm}  $c_3 : 4 \cdot y > x + 2$

No constraint for \( x \)

Guess \( x \mapsto 0 \)

\( c_1 \rightarrow y \notin (-1, \infty) \)

\( c_2 \rightarrow y \notin (-\infty, 0.75) \)

\( c_3 \rightarrow y \notin (-\infty, 0.5) \)

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 \( x \mapsto 0 \)
\[ c_1 \rightarrow y \notin (-1, \infty) \]
\[ c_2 \rightarrow y \notin (-\infty, 0.75) \]
\[ c_3 \rightarrow y \notin (-\infty, 0.5) \]
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 \( x \mapsto 0 \)

\[ c_1 \rightarrow y \notin (-1, \infty) \]
\[ c_2 \rightarrow y \notin (-\infty, 0.75) \]
\[ c_3 \rightarrow y \notin (-\infty, 0.5) \]

Construct covering

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

Construct interval for \( x \)

\( x \notin (-2, 3) \)
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 \( x \mapsto 0 \)

\[ c_1 \rightarrow y \not\in (-1, \infty) \]
\[ c_2 \rightarrow y \not\in (-\infty, 0.75) \]
\[ c_3 \rightarrow y \not\in (-\infty, 0.5) \]

Construct covering \(( -\infty, 0.5 ), (-1, \infty)\)

Construct interval for \( x \)

\( x \not\in (-2, 3) \)

New guess for \( x \)
function get_unsat_cover((s_1, \ldots, s_{i-1}))

\[ \mathbb{I} := \text{get_unsat_intervals}(s) \]

while \( \bigcup_{I \in \mathbb{I}} I \neq \mathbb{R} \) do

\[ s_i := \text{sample_outside}(\mathbb{I}) \]

if \( i = n \) then return \((\text{SAT}, (s_1, \ldots, s_{i-1}, s_i))\)

\((f, O) := \text{get_unsat_cover}((s_1, \ldots, s_{i-1}, s_i))\)

if \( f = \text{SAT} \) then return \((\text{SAT}, O)\)

else if \( f = \text{UNSAT} \) then

\[ R := \text{construct_characterization}((s_1, \ldots, s_{i-1}, s_i), O) \]

\[ J := \text{interval_from_characterization}((s_1, \ldots, s_{i-1}), s_i, R) \]

\[ \mathbb{I} := \mathbb{I} \cup \{J\} \]

end

end

return \((\text{UNSAT}, \mathbb{I})\)
The main algorithm

function get_unsat_cover((s₁, ..., sᵢ₋₁))

\( \mathbb{I} := \text{get_unsat_intervals}(s) \)

while \( \bigcup_{I \in \mathbb{I}} I \neq \mathbb{R} \) do

\( s_i := \text{sample_outside}(\mathbb{I}) \)

if \( i = n \) then return (SAT, (s₁, ..., sᵢ₋₁, sᵢ))

\( (f, O) := \text{get_unsat_cover}((s₁, ..., sᵢ₋₁, sᵢ)) \)

if \( f = \text{SAT} \) then return (SAT, O)

else if \( f = \text{UNSAT} \) then

\( R := \text{construct_characterization}((s₁, ..., sᵢ₋₁, sᵢ), O) \)

\( J := \text{interval_from_characterization}((s₁, ..., sᵢ₋₁), sᵢ, R) \)

\( \mathbb{I} := \mathbb{I} \cup \{J\} \)

end

end

return (UNSAT, \( \mathbb{I} \))
function get_unsat_cover((s_1, ..., s_{i-1}))

\[
\begin{align*}
\mathbb{I} & := \text{get_unsat_intervals}(s) \\
\text{while} \bigcup_{I \in \mathbb{I}} I \neq \mathbb{R} \text{ do} \\
& \quad s_i := \text{sample_outside}(\mathbb{I}) \\
& \quad \text{if } i = n \text{ then return } (\text{SAT}, (s_1, ..., s_{i-1}, s_i)) \\
& \quad (f, O) := \text{get_unsat_cover}((s_1, ..., s_{i-1}, s_i)) \\
& \quad \text{if } f = \text{SAT} \text{ then return } (\text{SAT}, O) \\
& \quad \text{else if } f = \text{UNSAT} \text{ then} \\
& \quad \quad R := \text{construct_characterization}((s_1, ..., s_{i-1}, s_i), O) \\
& \quad \quad J := \text{interval_from_characterization}((s_1, ..., s_{i-1}), s_i, R) \\
& \quad \quad \mathbb{I} := \mathbb{I} \cup \{J\} \\
& \quad \text{end} \\
& \text{end} \\
& \text{return } (\text{UNSAT}, \mathbb{I})
\end{align*}
\]

Real root isolation over a partial sample point
Select sample from \( \mathbb{R} \setminus \mathbb{I} \)
The main algorithm

function get_unsat_cover((s_1, \ldots, s_{i-1}))

\[ \mathbb{I} := \text{get_unsat_intervals}(s) \]

while \( \bigcup_{I \in \mathbb{I}} I \neq \mathbb{R} \) do

\[ s_i := \text{sample_outside}(\mathbb{I}) \]

if \( i = n \) then return (SAT, (s_1, \ldots, s_{i-1}, s_i))

\((f, O) := \text{get_unsat_cover}((s_1, \ldots, s_{i-1}, s_i))\)

if \( f = \text{SAT} \) then return (SAT, O)
else if \( f = \text{UNSAT} \) then

\[ R := \text{construct_characterization}((s_1, \ldots, s_{i-1}, s_i), O) \]
\[ J := \text{interval_from_characterization}((s_1, \ldots, s_{i-1}), s_i, R) \]
\[ \mathbb{I} := \mathbb{I} \cup \{J\} \]

end

end

return (UNSAT, \mathbb{I})
The main algorithm

\begin{verbatim}
function get_unsat_cover((s_1, \ldots, s_{i-1}))

\[ \mathbb{I} := \text{get_unsat_intervals}(s) \]

while \( \bigcup_{I \in \mathbb{I}} I \neq \mathbb{R} \) do

\[ s_i := \text{sample_outside}(\mathbb{I}) \]

if \( i = n \) then return (SAT, (s_1, \ldots, s_{i-1}, s_i))

\[ (f, O) := \text{get_unsat_cover}((s_1, \ldots, s_{i-1}, s_i)) \]

if \( f = \text{SAT} \) then return (SAT, O)

else if \( f = \text{UNSAT} \) then

\[ R := \text{construct_characterization}((s_1, \ldots, s_{i-1}, s_i)) \]

\[ J := \text{interval_from_characterization}((s_1, \ldots, s_{i-1}), s_i, R) \]

\[ \mathbb{I} := \mathbb{I} \cup \{ J \} \]

end

end

return (UNSAT, \mathbb{I})
\end{verbatim}

- Real root isolation over a partial sample point
- Select sample from \( \mathbb{R} \setminus \mathbb{I} \)
- Recurse to next variable
- CAD-style projection: Roots of polynomials restrict where covering is still applicable
The main algorithm

function get_unsat_cover((s₁, ..., sᵢ−1))

\[ I := \text{get_unsat_intervals}(s) \]

while \( \bigcup_{I \in I} I \neq \mathbb{R} \) do

\[ s_i := \text{sample_outside}(I) \]

if \( i = n \) then return \((\text{SAT}, (s_1, \ldots, s_{i-1}, s_i))\)

\[(f, O) := \text{get_unsat_cover}((s_1, \ldots, s_{i-1}, s_i))\]

if \( f = \text{SAT} \) then return \((\text{SAT}, O)\)

else if \( f = \text{UNSAT} \) then

\[ R := \text{construct_characterization}((s_1, \ldots, s_{i-1}, s_i)) \]

\[ J := \text{interval_from_characterization}((s_1, \ldots, s_{i-1}, s_i), R) \]

\[ I := I \cup \{J\} \]

end

end

return \((\text{UNSAT}, I)\)
The main algorithm

```python
function get_unsat_cover((s₁,...,sᵢ−1))

Ⅰ := get_unsat_intervals(s)

while $\bigcup_{I \in \Pi} I \neq \mathbb{R}$ do

| $sᵢ := \text{sample_outside}(Ⅰ)$ |
| if $i = n$ then return (SAT, (s₁,...,sᵢ−1,sᵢ)) |
| $(f,O) := \text{get_unsat_cover}((s₁,...,sᵢ−1))$ |
| if $f = \text{SAT}$ then return (SAT, O) |
| else if $f = \text{UNSAT}$ then |
| $R := \text{construct_characterization}((s₁,...,sᵢ−1,sᵢ))$ |
| $J := \text{interval_from_characterization}((s₁,...,sᵢ−1,sᵢ), R)$ |
| $Ⅰ := Ⅰ \cup \{J\}$ |
| end |
| end |

return (UNSAT, Ⅰ)
```

- Real root isolation over a partial sample point
- Select sample from $\mathbb{R} \setminus Ⅰ$
- Recurse to next variable
- CAD-style projection: Roots of polynomials restrict where covering is still applicable
- Extract interval from polynomials
Identify region around sample
Identify region around sample
construct_characterization

Identify region around sample
CAD projection:
Discriminants (and coefficients)
Resultants
Identify region around sample

CAD projection:

Discriminants (and coefficients)

Resultants

construct_characterization
Identify region around sample CAD projection:

**Discriminants (and coefficients)**

**Resultants**
Identify region around sample CAD projection:

Discriminants (and coefficients)

Resultants

Improvement over CAD:

Resultants between neighbouring intervals only!
Other methods for (QF_) NRA

- Numerical methods [Kremer 2013]: focus on good approximation, but no formal guarantees
- Tarski’s method [Tarski 1951]: theoretical breakthrough only, non-elementary complexity
- Grigor’ev and Vorobjov [Grigor’ev et al. 1988], Renegar [Renegar 1988]: singly exponentional, but impractical (see [Hong 1991])
- Basu, Pollack and Roy [Basu et al. 1996]: “realizable sign conditions”, has not been implemented (yet)
- Other CAD-based methods:
  - Regular Chains [Chen et al. 2009], NuCAD [Brown 2015]
Quantifiers:
- Theory of the Reals admits quantifier elimination
- CAD constructs $\varphi'$ for $Q_x \varphi(x, y) \iff \varphi'(y)$

Theory combination with Array, BV, FP, String, . . . [Nelson et al. 1979]

Transcendentals: extend linearization [Cimatti et al. 2018] [Irfan 2018]

Optimization: CAD can optimize for an objective [Kremer 2020]

Integers: Branch&Bound complements BitBlasting [Kremer et al. 2016]
Other approaches for (QF_)NRA:

- **MCSAT / NLSAT:**
  - Theory model construction integrated in the core solver
  - SMT-RAT, yices, z3 [Jovanović et al. 2012] [Jovanović et al. 2013] [Moura et al. 2013]
    [Nalbach et al. 2019] [Kremer 2020]

- **CAD is a stand-alone tool:**
  - Maple / RegularChains [Chen et al. 2009]
  - Mathematica [Strzeboński 2014]
  - QEPCAD B [Brown 2003]
  - Redlog / Reduce [Dolzmann et al. 1997]

These can be integrated as theory solvers [Fontaine et al. 2018] [Kremer 2018]
Some results...

Experiments on QF_NRA (11489 in total)

<table>
<thead>
<tr>
<th>QF_NRA</th>
<th>sat</th>
<th>unsat</th>
<th>solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvc5</td>
<td>5137</td>
<td>5596</td>
<td>10733</td>
</tr>
<tr>
<td>Yices2</td>
<td>4966</td>
<td>5450</td>
<td>10416</td>
</tr>
<tr>
<td>z3</td>
<td>5136</td>
<td>5207</td>
<td>10343</td>
</tr>
<tr>
<td>cvc5.cov</td>
<td>5001</td>
<td>5077</td>
<td>10078</td>
</tr>
<tr>
<td>SMT-RAT</td>
<td>4828</td>
<td>5038</td>
<td>9866</td>
</tr>
<tr>
<td>veriT</td>
<td>4522</td>
<td>5034</td>
<td>9556</td>
</tr>
<tr>
<td>MathSAT</td>
<td>3645</td>
<td>5357</td>
<td>9002</td>
</tr>
<tr>
<td>cvc5.inclin</td>
<td>3421</td>
<td>5376</td>
<td>8797</td>
</tr>
</tbody>
</table>
Thank you for your attention!
Any questions?
References I


References II


- **George E. Collins.** “Quantifier Elimination by Cylindrical Algebraic Decomposition — Twenty Years of Progress”. In: Quantifier Elimination and Cylindrical Algebraic Decomposition. 1998, pp. 8–23. doi: 10.1007/978-3-7091-9459-1_2.


References III


References IV


References V

References VI

References VII

References VIII

