Satisfiability checking for propositional logic

Success story: SAT-solving. Frequently used in different research areas for, e.g., analysis, synthesis and optimization. Also massively used in industry for, e.g., digital circuit verification. Competitions since 2002.

2014 SAT Competition: 3 categories, 79 participants with 137 solvers.
Success story: SAT-solving.

Frequently used in different research areas for, e.g., analysis, synthesis and optimization.

Also massively used in industry for, e.g., digital circuit verification.
Success story: SAT-solving.

Frequently used in different research areas for, e.g., analysis, synthesis and optimization.

Also massively used in industry for, e.g., digital circuit verification.

Competitions since 2002.

2014 SAT Competition: 3 categories, 79 participants with 137 solvers.
Satisfiability checking for other logics

Propositional logic is sometimes too weak for modeling. We need more expressive logics and decision procedures for them.

Logics:
- quantifier-free fragments of FO logic over different theories.

A class of decision procedures:
- SAT -modulo-theories (SMT) solving.

Competitions since 2005.

SMT-COMP 2014 competition:
- 34 logical categories,
- 20 solvers

Uninterpreted functions (QF UF) (since 2005):
- 7 solvers

Linear real arithmetic (QF LRA) (since 2005):
- 6 solvers

Non-linear real arithmetic (QF NRA) (since 2010):
- 4 solvers
  (Z3/non-competitive, CVC3, CVC4, raSAT)
Satisfiability checking for other logics

- Propositional logic is sometimes too weak for modeling.
- We need more expressive logics and decision procedures for them.
- Logics: quantifier-free fragments of FO logic over different theories.
- A class of decision procedures: SAT-modulo-theories (SMT) solving.
Satisfiability checking for other logics

- Propositional logic is sometimes too weak for modeling.
- We need more expressive logics and decision procedures for them.
- Logics: quantifier-free fragments of FO logic over different theories.
- A class of decision procedures: SAT-modulo-theories (SMT) solving.
- Competitions since 2005.
- SMT-COMP 2014 competition:
  - 34 logical categories, 20 solvers
  - Uninterpreted functions (QF_UF) (since 2005): 7 solvers
  - Linear real arithmetic (QF_LRA) (since 2005): 6 solvers
  - Non-linear real arithmetic (QF_NRA) (since 2010): 4 solvers (Z3/non-competitive, CVC3, CVC4, raSAT)
Real arithmetic ($\mathbb{R}$, $+$, $\times$, $<$, 0, 1)
Real arithmetic \((\mathbb{R}, +, \times, <, 0, 1)\)

Theoretical results:

- decidable [Tarski, 1948] with
- an upper bound \(2^{2^n}\) for time complexity for \(n\) variables
  [Weispfenning, 1988] [Davenport and Heintz, 1998] [Brown and Davenport, 2007] and
- an upper bound \(2^n\) for time complexity for the existential fragment
  [Heintz, Roy and Solernó, 1993].
Real arithmetic \((\mathbb{R}, +, \times, <, 0, 1)\)

Theoretical results:

- Decidable [Tarski, 1948] with
- An upper bound \(2^{2^n}\) for time complexity for \(n\) variables
  [Weispfenning, 1988] [Davenport and Heintz, 1998] [Brown and Davenport, 2007] and
- An upper bound \(2^n\) for time complexity for the existential fragment
  [Heintz, Roy and Solernó, 1993].

Theoretically (decision procedures) and practically (implementation) highly challenging!
SMT-solvers for real arithmetic

Incomplete SMT-solvers for real arithmetic:
- CVC3/CVC4 (New York, Iowa / USA)
- MiniSmt (Innsbruck / Austria)
- HySat/iSAT (Oldenburg, Freiburg / Germany)

Complete SMT-solvers for real arithmetic:
- Z3/nlsat (NYU, Microsoft Research / USA)
- SMT-RAT (RWTH Aachen University / Germany)
Incomplete SMT-solvers for real arithmetic:
- CVC3/CVC4 (New York, Iowa / USA)
- MiniSmt (Innsbruck / Austria)
- HySat/iSAT (Oldenburg, Freiburg / Germany)
- ...
Incomplete SMT-solvers for real arithmetic:
- CVC3/CVC4 (New York, Iowa / USA)
- MiniSmt (Innsbruck / Austria)
- HySat/iSAT (Oldenburg, Freiburg / Germany)
- ...

Complete SMT-solvers for real arithmetic:
- Z3/nlsat (NYU, Microsoft Research / USA)
- SMT-RAT (RWTH Aachen University / Germany)
Less lazy SMT solving
Less lazy SMT solving

\( \varphi \) quantifier-free FO formula
Less lazy SMT solving

Boolean abstraction \( \phi \) \( \Downarrow \) \( \phi' \)

quantifier-free FO formula

propositional logic formula
Less lazy SMT solving

Boolean abstraction

\( \phi \)

quantifier-free FO formula

\( \phi' \)

propositional logic formula

SAT solver
Less lazy SMT solving

Boolean abstraction

\( \varphi \)

\( \varphi' \)

quantifier-free FO formula

propositional logic formula

SAT solver

theory constraints

Theory solver
Less lazy SMT solving

Boolean abstraction

$\varphi$

$\varphi'$

quantifier-free FO formula

propositional logic formula

SAT solver

SAT or

UNSAT+ (minimal) infeasible subset

theory constraints

Theory solver
Less lazy SMT solving

Boolean abstraction

quantifier-free FO formula

SAT or UNSAT

SAT solver

SAT or UNSAT

theory constraints

Theory solver

propositional logic formula

SAT or UNSAT+(minimal) infeasible subset
Less lazy SMT solving

$$(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)$$
\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[\downarrow\]

\[(a \lor b) \land (c \lor d)\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d)\]

SAT solver

Theory solver
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d)\]

SAT solver

Theory solver

\[\neg a\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[\left( a \lor b \right) \land \left( c \lor d \right)\]

SAT solver

Theory solver

\[\neg a, b\]
(x < 0 ∨ x > 2) ∧ (x^2 = 1 ∨ x^2 < 0)

(a ∨ b) ∧ (c ∨ d)

SAT solver

¬a, b

x ≥ 0, x > 2

Theory solver
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(\neg a \lor b) \land (c \lor d)\]

\(x \geq 0, x > 2\)
(x < 0 ∨ x > 2) ∧ (x^2 = 1 ∨ x^2 < 0)

(a ∨ b) ∧ (c ∨ d)

SAT solver

¬a, b, ¬c

Theory solver

x ≥ 0, x > 2
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d)\]

SAT solver

\[-a, b, -c, d\]

Theory solver

\[x \geq 0, x > 2\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d)\]

SAT solver

\[\neg a, b, \neg c, d\]

Theory solver

\[x \geq 0, x > 2, x^2 \neq 1, x^2 < 0\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d)\]

SAT solver

\(\neg a, b, \neg c, d\)

\[x \geq 0, x > 2, x^2 \neq 1, x^2 < 0\]

UNSAT: \(x^2 < 0\)

Theory solver
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d)\]

SAT solver

\[-a, b, \neg c, d\]

Theory solver

\[x \geq 0, x > 2, x^2 \neq 1, x^2 < 0\]

UNSAT: \[x^2 < 0\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d)\]
\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d)\]
(x < 0 ∨ x > 2) ∧ (x^2 = 1 ∨ x^2 < 0)

(a ∨ b) ∧ (c ∨ d) ∧ (¬d)

SAT solver

¬d, c

Theory solver
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d)\]

SAT solver

\[\neg d, c\]

Theory solver

\[x^2 \geq 0, x^2 = 1\]
(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)

(a \lor b) \land (c \lor d) \land (\neg d)

SAT solver

\neg d, c

x^2 \geq 0, x^2 = 1
(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)

(a \lor b) \land (c \lor d) \land (\neg d)

x^2 \geq 0, \quad x^2 = 1

\neg d, \quad c, \quad \neg a

SAT solver

Theory solver
(x < 0 ∨ x > 2) ∧ (x^2 = 1 ∨ x^2 < 0)

(a ∨ b) ∧ (c ∨ d) ∧ (¬d)

x^2 ≥ 0, x^2 = 1

¬d, c, ¬a, b
Less lazy SMT solving

\[(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)\]

\[(a \vee b) \wedge (c \vee d) \wedge (\neg d)\]

\[x^2 \geq 0, \ x^2 = 1, \ x \geq 0, \ x > 2\]

SAT solver

\[\neg d, \ c, \ \neg a, \ b\]

Theory solver
(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)

( a \lor b ) \land ( c \lor d ) \land (\neg d)

SAT solver

\neg d, c, \neg a, b

x^2 \geq 0, x^2 = 1, x \geq 0, x > 2

UNSAT: x^2 = 1, x > 2
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d) \land (\neg c \lor \neg b)\]

SAT solver

Theory solver

\[x^2 \geq 0, \ x^2 = 1, \ x \geq 0, \ x > 2\]  

UNSAT: \[x^2 = 1, \ x > 2\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[
( a \lor b ) \land ( c \lor d ) \land (\neg d) \land (\neg c \lor \neg b)
\]

SAT solver

Theory solver

\[x^2 \geq 0, \ x^2 = 1\]

\[\neg d, \ c\]
(x < 0 ∨ x > 2) ∧ (x^2 = 1 ∨ x^2 < 0)

(a ∨ b) ∧ (c ∨ d) ∧ (¬d) ∧ (¬c ∨ ¬b)

SAT solver

Theory solver

¬d, c, ¬b

x^2 ≥ 0, x^2 = 1
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d) \land (\neg c \lor \neg b)\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[
(\ a \ \lor \ b \ ) \land (\ c \ \lor \ d \ ) \land (\neg d) \land (\neg c \lor \neg b)
\]

\[x^2 \geq 0, \ x^2 = 1, \ x \leq 2, \ x < 0\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]

\[(a \lor b) \land (c \lor d) \land (\neg d) \land (\neg c \lor \neg b)\]

\[x^2 \geq 0, \ x^2 = 1, \ x \leq 2, \ x < 0\]
Less lazy SMT solving

\((x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\)

\(((a \lor b) \land (c \lor d) \land (\neg d) \land (\neg c \lor \neg b))\)

SAT solver

Theory solver

SAT
Theory decision procedures: Algebraic methods

Cylindrical algebraic decomposition (CAD)  
QEPCAD (North Carolina), Reduce (Stanford, worldwide), . . .

Virtual substitution (VS)  
Reduce

Gröbner bases  
Maple, Mathematica, Singular, Maxima, CoCoA, Reduce, . . .

Efficient for conjunctions of real constraints.
Theory decision procedures: Algebraic methods

Cylindrical algebraic decomposition (CAD)

- QEPCAD (North Carolina), Reduce (Stanford, worldwide), ...
Theory decision procedures: Algebraic methods

Cylindrical algebraic decomposition (CAD)

- QEPCAD (North Carolina), Reduce (Stanford, worldwide), ...

Virtual substitution (VS)

- Reduce
Theory decision procedures: Algebraic methods

Cylindrical algebraic decomposition (CAD)
- QEPCAD (North Carolina), Reduce (Stanford, worldwide), ...

Virtual substitution (VS)
- Reduce

Gröbner bases
- Maple, Mathematica, Singular, Maxima, CoCoA, Reduce, ...
Theory decision procedures: Algebraic methods

Cylindrical algebraic decomposition (CAD)

- QEPCAD (North Carolina), Reduce (Stanford, worldwide), ...

Virtual substitution (VS)

- Reduce

Gröbner bases

- Maple, Mathematica, Singular, Maxima, CoCoA, Reduce, ...

Efficient for conjunctions of real constraints.
Why don't we just plug in an algebraic decision procedure as theory solver into an SMT-solver? Theory solvers should work incrementally, generate (minimal) infeasible subsets and be able to backtrack. Originally, the mentioned methods do not have these functionalities. Our goal: develop theory solvers which support these features; make their implementation publicly available as a toolbox; embed them into an SMT-solver; combine the theory solvers heuristically to achieve wider applicability.
Why don’t we just plug in an algebraic decision procedure as theory solver into an SMT-solver?
SMT-solving for real arithmetic

- Why don’t we just plug in an algebraic decision procedure as theory solver into an SMT-solver?
- Theory solvers should
  - work incrementally,
  - generate (minimal) infeasible subsets and
  - be able to backtrack.
Why don’t we just plug in an algebraic decision procedure as theory solver into an SMT-solver?

Theory solvers should

- work incrementally,
- generate (minimal) infeasible subsets and
- be able to backtrack.

Originally, the mentioned methods do not have these functionalities.
Why don’t we just plug in an algebraic decision procedure as theory solver into an SMT-solver?

Theory solvers should

- work incrementally,
- generate (minimal) infeasible subsets and
- be able to backtrack.

Originally, the mentioned methods do not have these functionalities.

Our goal:

- develop **theory solvers** which support these features;
- make their implementation publicly available as a **toolbox**;
- embed them into an **SMT-solver**;
- combine the theory solvers heuristically to achieve wider applicability.
SMT Solver
Strategic composition of SMT-RAT modules

SMT-RAT
(SMT real-algebraic toolbox)
preprocessing modules
SAT and theory solvers

CArL
real-algebraic computations

gmp, Eigen3, boost
SMT Solver
Strategic composition of SMT-RAT modules

SMT-RAT
(SMT real-algebraic toolbox)
preprocessing modules
SAT and theory solvers

CArL
real-algebraic computations

gmp, Eigen3, boost
Assume a set of univariate polynomial constraints $p_1 \sim 1_0, \ldots, p_k \sim k_0$ with $p_i \in \mathbb{Z}[x]$ and $\sim_i \in \{<, \leq, =, \neq, \geq, >\}$.

Cauchy bound $\Rightarrow$ Interval $C$ containing all real roots of $p_1, \ldots, p_k$.

Sturm sequence $\Rightarrow$ count the real roots of each $p_i$ in an interval.

Split $C$ until each sub-interval $I$ contains at most one real root.
Assume a set of univariate polynomial constraints

\[ p_1 \sim_1 0, \ldots, p_k \sim_k 0 \] with \( p_i \in \mathbb{Z}[x] \) and \( \sim_i \in \{<, \leq, =, \neq, \geq, >\} \).
Assume a set of univariate polynomial constraints

\[ p_1 \sim_1 0, \ldots, p_k \sim_k 0 \] with \( p_i \in \mathbb{Z}[x] \) and \( \sim_i \in \{<, \leq, =, \neq, \geq, >\} \).

Cauchy bound \( \Rightarrow \) Interval \( C \) containing all real roots of \( p_1, \ldots, p_k \).
CAD for $\mathbb{R}$

- Assume a set of univariate polynomial constraints $p_1 \sim_1 0, \ldots, p_k \sim_k 0$ with $p_i \in \mathbb{Z}[x]$ and $\sim_i \in \{<, \leq, =, \neq, \geq, >\}$.
- Cauchy bound $\Rightarrow$ Interval $C$ containing all real roots of $p_1, \ldots, p_k$.
- Sturm sequence $\Rightarrow$ count the real roots of each $p_i$ in an interval.

$$C$$
Assume a set of univariate polynomial constraints
\[ p_1 \sim_1 0, \ldots, p_k \sim_k 0 \] with \( p_i \in \mathbb{Z}[x] \) and \( \sim_i \in \{ <, \leq, =, \neq, \geq, > \} \).

- **Cauchy bound** ⇒ Interval \( C \) containing all real roots of \( p_1, \ldots, p_k \).
- **Sturm sequence** ⇒ count the real roots of each \( p_i \) in an interval.
- **Split** \( C \) until each sub-interval \( I \) contains at most one real root.
Assume a set of univariate polynomial constraints
\[ p_1 \sim_1 0, \ldots, p_k \sim_k 0 \] with \( p_i \in \mathbb{Z}[x] \) and \( \sim_i \in \{<, \leq, =, \neq, \geq, >\} \).

- **Cauchy bound** \( \Rightarrow \) Interval \( C \) containing all real roots of \( p_1, \ldots, p_k \).
- **Sturm sequence** \( \Rightarrow \) count the real roots of each \( p_i \) in an interval.
- **Split** \( C \) until each sub-interval \( I \) contains at most one real root.
Assume a set of univariate polynomial constraints $p_1 \sim_1 0, \ldots, p_k \sim_k 0$ with $p_i \in \mathbb{Z}[x]$ and $\sim_i \in \{<, \leq, =, \neq, \geq, >\}$.

- **Cauchy bound** $\Rightarrow$ Interval $C$ containing all real roots of $p_1, \ldots, p_k$.
- **Sturm sequence** $\Rightarrow$ Count the real roots of each $p_i$ in an interval.
- **Split** $C$ until each sub-interval $I$ contains at most one real root.
Assume a set of univariate polynomial constraints $p_1 \sim_1 0, \ldots, p_k \sim_k 0$ with $p_i \in \mathbb{Z}[x]$ and $\sim_i \in \{<, \leq, =, \neq, \geq, >\}$.

- **Cauchy bound** $\Rightarrow$ Interval $C$ containing all real roots of $p_1, \ldots, p_k$.
- **Sturm sequence** $\Rightarrow$ count the real roots of each $p_i$ in an interval.
- **Split** $C$ until each sub-interval $I$ contains at most one real root.
Assume a set of univariate polynomial constraints 
\[ p_1 \sim_1 0, \ldots, p_k \sim_k 0 \] with \( p_i \in \mathbb{Z}[x] \) and \( \sim_i \in \{<, \leq, =, \neq, \geq, >\} \).

Cauchy bound \( \Rightarrow \) Interval \( C \) containing all real roots of \( p_1, \ldots, p_k \).

Sturm sequence \( \Rightarrow \) count the real roots of each \( p_i \) in an interval.

Split \( C \) until each sub-interval \( I \) contains at most one real root.

\[
\begin{align*}
I_4 & & I_6 & & I_7 & & I_9 \\
| & & | & & | & & | \\
I_5 & & I_2 & & I_8 & &
\end{align*}
\]

CAD for \( \mathbb{R} \) with respect to \( p_1, \ldots, p_k \):
\[ [(p_i, I_j), (p_i, I_j)] \] for each \( I_j \) containing a real root of a \( p_i \) and open intervals between them.
Cauchy bound: $C = (-3, 3)$

Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$

Number of real roots in $(-3, 3)$:

Split $(-3, 3)$ into $(-3, 0), [0, 0], (0, 3)$

Number of real roots in $I_1 = (-3, 0)$:

Number of real roots in $I_2 = (0, 0)$:

Number of real roots in $I_3 = (0, 3)$:

CAD: $[(p_1, I_1), (p_1, I_3)], [(p_1, I_1), (p_1, I_3)], (-\infty, (p_1, I_1)), ((p_1, I_1), (p_1, I_3)), ((p_1, I_3), \infty)$
$x^2 - 2 > 0$
CAD for $\mathbb{R}$: Example

- $x^2 - 2 > 0$

  $p_1$

- Cauchy bound: $C = (-3, 3)$
CAD for $\mathbb{R}$: Example

- $\underbrace{x^2 - 2}_{p_1} > 0$
- Cauchy bound: $C = (-3, 3)$
- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$
CAD for $\mathbb{R}$: Example

- $x^2 - 2 > 0$
- Cauchy bound: $C = (-3, 3)$
- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$
- Number of real roots in $(-3, 3)$: 2
CAD for $\mathbb{R}$: Example

- $x^2 - 2 > 0$
- Cauchy bound: $C = (-3, 3)$
- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$
- Number of real roots in $(-3, 3)$: 2
- Split $(-3, 3)$ into $(-3, 0), [0, 0], (0, 3)$
CAD for $\mathbb{R}$: Example

- $x^2 - 2 > 0$
- Cauchy bound: $C = (-3, 3)$
- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$
- Number of real roots in $(-3, 3)$: 2
- Split $(-3, 3)$ into $(-3, 0), [0, 0], (0, 3)$
- Number of real roots in $I_1 = (-3, 0)$: 1
CAD for $\mathbb{R}$: Example

- $x^2 - 2 > 0$
  - $p_1$

- Cauchy bound: $C = (-3, 3)$

- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$

- **Number of real roots in** $(-3, 3)$: 2

- **Split** $(-3, 3)$ into $(-3, 0)$, $[0, 0]$, $(0, 3)$

- **Number of real roots in** $I_1 = (-3, 0)$: 1

- **Number of real roots in** $I_2 = (0, 0)$: 0
CAD for \( \mathbb{R} \): Example

- \( x^2 - 2 > 0 \)
- \( P_1 \)
- Cauchy bound: \( C = (-3, 3) \)
- Sturm sequence: \( P_1 = (x^2 - 2, 2x, 2) \)
- Number of real roots in \((-3, 3): 2) \)
- Split \((-3, 3) \) into \((-3, 0), [0, 0], (0, 3) \)
- Number of real roots in \( I_1 = (-3, 0): 1 \)
- Number of real roots in \( I_2 = (0, 0): 0 \)
- Number of real roots in \( I_3 = (0, 3): 1 \)

![Diagram showing intervals I₁, I₂, I₃]
CAD for $\mathbb{R}$: Example

- $\underbrace{x^2 - 2 > 0}_{p_1}$
- Cauchy bound: $C = (-3, 3)$
- Sturm sequence: $P_1 = (x^2 - 2, 2x, 2)$
- Number of real roots in $(-3, 3)$: 2
- Split $(-3, 3)$ into $(-3, 0), [0, 0], (0, 3)$
- Number of real roots in $I_1 = (-3, 0)$: 1
- Number of real roots in $I_2 = (0, 0)$: 0
- Number of real roots in $I_3 = (0, 3)$: 1
- CAD: $[(p_1, I_1), (p_1, I_1)], [(p_1, I_3), (p_1, I_3)],$ $(-\infty, (p_1, I_1)), ((p_1, I_1), (p_1, I_3)), ((p_1, I_3), \infty)$
The original method is not incremental. We achieve incrementality by refining the CAD.

Previous split:

$I_1 = (-3, 0),
I_2 = [0, 0],
I_3 = (0, 3)$

New constraint:

$x_2 - x_1 \geq p_2$

Cauchy bound:

$C_2 = (-3, 3)$

Sturm sequence:

$P_2 = (x_2 - x_1, 2x_2 - 1, 5)$

Number of real roots in $I_1 = (-3, 0)$:

$1 ($p_1, I_1)$ \neq ($p_2, I_1)$ \Rightarrow \text{split}$

Number of real roots in $I_2 = [0, 0]$: 0

Number of real roots in $I_3 = (0, 3)$:

$1 ($p_1, I_3)$ \neq ($p_2, I_3)$ \Rightarrow \text{split}$
The original method is not incremental.
The original method is not incremental.

We achieve incrementality by refining the CAD.
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split: \( I_1 = (-3, 0), I_2 = [0, 0], I_3 = (0, 3) \)
CAD for $\mathbb{R}$: Incrementality

- The original method is not incremental.
- We achieve incrementality by refining the CAD.
- Previous split: $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$
- New constraint: $x^2 - x - 1 > 0$

$p_2$
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split: $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$

New constraint: $\underbrace{x^2 - x - 1 > 0}_{p_2}$

Cauchy bound: $C_2 = (-3, 3)$
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split:
\[ I_1 = (-3, 0), \ I_2 = [0, 0], \ I_3 = (0, 3) \]

New constraint:
\[ x^2 - x - 1 > 0 \]

Cauchy bound:
\[ C_2 = (-3, 3) \]

Sturm sequence:
\[ P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4}) \]
CAD for $\mathbb{R}$: Incrementality

- The original method is not incremental.
- We achieve incrementality by refining the CAD.
- Previous split: $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$
- New constraint: $x^2 - x - 1 > 0$
- Cauchy bound: $C_2 = (-3, 3)$
- Sturm sequence: $P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4})$
- Number of real roots in $I_1 = (-3, 0)$: 1
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split: $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$

New constraint: $x^2 - x - 1 > 0$

Cauchy bound: $C_2 = (-3, 3)$

Sturm sequence: $P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4})$

Number of real roots in $I_1 = (-3, 0)$: 1

$(p_1, I_1) \neq (p_2, I_1) \Rightarrow$ split
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split: $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$

New constraint: $x^2 - x - 1 > 0$

Cauchy bound: $C_2 = (-3, 3)$

Sturm sequence: $P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4})$

Number of real roots in $I_1 = (-3, 0)$: 1

$(p_1, I_1) \neq (p_2, I_1) \Rightarrow$ split

Number of real roots in $I_2 = [0, 0]$: 0
The original method is not incremental.
We achieve incrementality by refining the CAD.

- **Previous split:** $I_1 = (-3, 0)$, $I_2 = [0, 0]$, $I_3 = (0, 3)$
- **New constraint:** $x^2 - x - 1 > 0$
- **Cauchy bound:** $C_2 = (-3, 3)$
- **Sturm sequence:** $P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4})$
- **Number of real roots in $I_1 = (-3, 0)$:** 1
  
  $(p_1, I_1) \neq (p_2, I_1) \Rightarrow \text{split}$
- **Number of real roots in $I_2 = [0, 0]$:** 0
- **Number of real roots in $I_3 = (0, 3)$:** 1
The original method is not incremental.

We achieve incrementality by refining the CAD.

Previous split: \( I_1 = (-3, 0) \), \( I_2 = [0, 0] \), \( I_3 = (0, 3) \)

New constraint: \( x^2 - x - 1 > 0 \)

Cauchy bound: \( C_2 = (-3, 3) \)

Sturm sequence: \( P_2 = (x^2 - x - 1, 2x - 1, \frac{5}{4}) \)

Number of real roots in \( I_1 = (-3, 0) \): 1

\((p_1, I_1) \neq (p_2, I_1) \Rightarrow \text{split}\)

Number of real roots in \( I_2 = [0, 0] \): 0

Number of real roots in \( I_3 = (0, 3) \): 1

\((p_1, I_3) \neq (p_2, I_3) \Rightarrow \text{split}\)
The original method cannot generate infeasible subsets. For \( \mathbb{R} \) we collect for each CAD interval one constraint whose sign condition is not satisfied by the interval. The case for higher dimensions is more involved, but the basic idea is still similar...
The original method cannot generate infeasible subsets.
The original method cannot generate infeasible subsets.

For $\mathbb{R}$ we collect for each CAD interval one constraint whose sign condition is not satisfied by the interval.
CAD for $\mathbb{R}$: Infeasible subsets

- The original method cannot generate infeasible subsets.
- For $\mathbb{R}$ we collect for each CAD interval one constraint whose sign condition is not satisfied by the interval.
- The case for higher dimensions is more involved, but the basic idea is still similar...
A **CAD** for a set of polynomials from $\mathbb{Z}[x_1, \ldots, x_n]$ splits $\mathbb{R}^n$ into **sign-invariant** regions.

**Projection phase**

- Polynomials over $\mathbb{Z}[x_1, \ldots, x_n]$
- Polynomials over $\mathbb{Z}[x_1, \ldots, x_{n-1}]$
- ...  
- Polynomials over $\mathbb{Z}[x_1]$

**Construction phase**

- CAD for $\mathbb{R}^n$
- CAD for $\mathbb{R}^{n-1}$
- ...  
- CAD for $\mathbb{R}^1$
The virtual substitution method

Handles polynomial constraints of bounded degree ("quadratic and beyond")

Univariate equations

\[ ax^2 + bx + c = 0, \quad a \neq 0 \]

have solution set

\[ x_1, x_2 = -\frac{b \pm \sqrt{b^2 - 4ac}}{2a} \]

if \( b^2 - 4ac \geq 0 \)

Generalize this to

\[ p_1x^2 + p_2x + p_3 \sim 0 \]
The virtual substitution method

- Handles polynomial constraints of bounded degree ("quadratic and beyond")
- Univariate equations \( ax^2 + bx + c = 0, a \neq 0 \) have solution set

\[
x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \text{if} \quad b^2 - 4ac \geq 0
\]

- Generalize this to \( p_1x^2 + p_2x + p_3 \sim 0 \)
The virtual substitution method

Repeat until all variables are eliminated:
The virtual substitution method

Repeat until all variables are eliminated:

1. **Elimination:**
   - Take one variable and find all test candidates for all constraints

But: standard substitution would lead to expressions with $\epsilon$, $\infty$, square root or division.
The virtual substitution method

Repeat until all variables are eliminated:

1. **Elimination:**
   - Take one variable and find all *test candidates* for all constraints

2. **Substitution:**
   - For each test candidate: check whether it fulfills all other constraints
   - But: standard substitution would lead to expressions with $\varepsilon$, $\infty$, square root or division
   - Virtual substitution $\hat{=}$ standard substitution
     - transformation to a real algebraic formula
   - Results in a disjunction (case splitting)
Virtual substitution tree

compute test points for $x_1$

substitute test point for $x_1$
Virtual substitution tree

- VS has a decision tree, similar to CAD

Node: $P_0$

Branches:
- $Q_1$
- $\cdots$
- $Q_n$

Subtrees:
- $R_1$, $\cdots$, $R_m$
- $S_1$, $\cdots$, $S_k$

Remarks:
- Compute test points for $x_1$
- Substitute test point for $x_1$
Virtual substitution tree

VS has a decision tree, similar to CAD

We can handle incrementality and infeasible subsets similarly as for CAD
SMT Solver
Strategic composition of SMT-RAT modules

SMT-RAT
(SMT real-algebraic toolbox)
preprocessing modules
SAT and theory solvers

CARL
real-algebraic computations

gmp, Eigen3, boost
Strategic composition of SMT-RAT modules

- SAT solver
- CNF converter
- Preprocessing modules (simplifiers)
- Interval constraint propagation (ICP)
- Simplex (LRA)
- Cylindrical algebraic decomposition (CAD)
- Virtual substitution (VS)
- Gröbner bases (GB)
- Generalized branch-and-bound (B&B)
- Under construction: equality, uninterpreted functions, bitvector arithmetic
Strategic composition of SMT-RAT modules

SAT solver

Manager

SMT solver

Strategy

Condition

Module

Condition

Module

Condition

Module

...
A new release comes soon!

If you want to use SMT-RAT before the next release, just drop an email.