SMT Solving for Non-Linear Arithmetic Theories

Erika Ábrahám

in cooperation with
Florian Corzilius, Gereon Kremer, Stefan Schupp and others

RWTH Aachen University, Germany

Vienna, 17 May 2016
What is this talk about?

Satisfiability problem

The satisfiability problem is the problem of deciding whether a logical formula is satisfiable.

We focus on the automated solution of the satisfiability problem for quantifier-free first-order logic formulas over arithmetic theories.
Success story: **SAT-solving** for propositional logic

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.
Success story: SAT-solving for propositional logic

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.

Community support:

- Standardised input language, lots of benchmarks available.
- Competitions since 2002.
  - SAT-Race 2015: 3 tracks, 30 solvers.
Propositional logic is sometimes too weak for modelling.

We need more expressive logics and decision procedures for them.

Logics:
quantifier-free fragments of first-order logic over various theories.

Our focus: SAT-modulo-theories (SMT) solving.
Propositional logic is sometimes too weak for modelling.
We need more expressive logics and decision procedures for them.

Logics:
quantifier-free fragments of first-order logic over various theories.

Our focus: SAT-modulo-theories (SMT) solving.

SMT-LIB as standard input language since 2004.
Competitions since 2005.

SMT-COMP 2015 competition:
- 2 tracks, 40 logical categories, 11 teams, 30 solvers.
- 154238 benchmarks in the main track.
- QF non-linear real arithmetic: 7 solvers
  (CVC3, CVC4, CVC4(exp), SMT-RAT, Yices2-NL, [Z3], raSAT).
SMT-LIB theories

Source: http://smtlib.cs.uiowa.edu/logics.shtml
SMT-LIB theories

Source: http://smtlib.cs.uiowa.edu/logics.shtml
(Full/less) lazy SMT solving
(Full/less) lazy SMT solving

\( \varphi \) quantifier-free FO formula
(Full/less) lazy SMT solving

Boolean abstraction

Tseitin’s transformation

\( \varphi \)

quantifier-free FO formula

\( \varphi' \)

propositional logic formula in CNF
(Full/less) lazy SMT solving

Boolean abstraction

Tseitin’s transformation

\( \varphi \)

\( \varphi' \)

quantifier-free FO formula

propositional logic formula in CNF

SAT solver
(Full/less) lazy SMT solving

Boolean abstraction
Tseitin’s transformation

$\varphi$ quantifier-free FO formula

$\varphi'$ propositional logic formula in CNF

SAT solver

time constraints

Theory solver(s)
(Full/less) lazy SMT solving

Boolean abstraction
Tseitin’s transformation

quantifier-free FO formula

Boolean abstraction
Tseitin’s transformation

propositional logic formula in CNF

SAT solver

theory constraints

SAT
or
UNSAT
+ lemmas

Theory solver(s)
(Full/less) lazy SMT solving

Boolean abstraction
Tseitin's transformation

quantifier-free FO formula

propositional logic formula in CNF

SAT solver

SAT or UNSAT

theory constraints

SAT or UNSAT + lemmas

Theory solver(s)
Less lazy SMT solving
(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)
Less lazy SMT solving

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

\[
\downarrow
\]

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

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

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

\[\neg a\]
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

$\neg a, \ b$

Theory solver(s)
Less lazy SMT solving

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

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

\[-a, b\]

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

(a ∨ b) ∧ (c ∨ d)

SAT solver

-x, b

x ≥ 0, x > 2

Theory solver(s)

SAT

Less lazy SMT solving
Less lazy SMT solving

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

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

\(x \geq 0, x > 2\)

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

Theory solver(s)
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(s)

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

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

Theory solver(s)

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

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

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

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

SAT solver

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

\(x \geq 0, x > 2, x^2 \neq 1, x^2 < 0\)   
UNSAT: \((-x^2 < 0)\)

Theory solver(s)
Less lazy SMT solving

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

\[\text{SAT solver}\]

\[\text{Theory solver(s)}\]
Less lazy SMT solving

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

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

\(\neg d\)

SAT solver

Theory solver(s)
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(s)
Less lazy SMT solving

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

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

SAT solver

\[\neg d, c\]

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

Theory solver(s)
Less lazy SMT solving

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

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

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

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

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

SAT solver

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

Theory solver(s)

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

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

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

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

( a ∨ b ) ∧ ( c ∨ d ) ∧ (¬d)

x^2 ≥ 0, x^2 = 1, x ≥ 0, x > 2

SAT solver

¬d, c, ¬a, b

Theory solver(s)
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

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

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

UNSAT: \((x^2 = 1 \land x > 2)\)

Theory solver(s)
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

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

Theory solver(s)

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

UNSAT: \[-(x^2 = 1 \land x > 2)\]
Less lazy SMT solving

\[(x < 0 \lor x > 2) \land (x^2 = 1 \lor x^2 < 0)\]
Less lazy SMT solving

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

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

SAT solver \rightarrow \neg d, c, \neg b

Theory solver(s)

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

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

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

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

SAT solver \rightarrow Theory solver(s) \rightarrow \neg d, c, \neg b, a
Less lazy SMT solving

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

\[
\begin{align*}
(a \lor b) \land (c \lor d) \land (\neg d) \land (\neg c \lor \neg b)
\end{align*}
\]

SAT solver

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

Theory solver(s)

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

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

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

\(x^2 \geq 0, \ x^2 = 1, \ x \leq 2, \ x < 0\)

SAT

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

Theory solver(s)

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

(a ∨ b) ∧ (c ∨ d) ∧ (¬d) ∧ (¬c ∨ ¬b)
Some theory solver candidates for arithmetic theories

Linear real arithmetic:
- Simplex
- Ellipsoid method
- Fourier-Motzkin variable elimination (mostly preprocessing)
- Interval constraint propagation (incomplete)

Linear integer arithmetic:
- Cutting planes, Gomory cuts
- Branch-and-bound (incomplete)
- Bit-blasting (eager)
- Interval constraint propagation (incomplete)

Non-linear real arithmetic:
- Cylindrical algebraic decomposition
- Gröbner bases (mostly preprocessing/simplification)
- Virtual substitution (focus on low degrees)
- Interval constraint propagation (incomplete)

Non-linear integer arithmetic:
- Generalised branch-and-bound (incomplete)
- Bit-blasting (eager, incomplete)
Some corresponding implementations in CAS

<table>
<thead>
<tr>
<th>Gröbner bases</th>
</tr>
</thead>
<tbody>
<tr>
<td>- CoCoA, F4, Maple, Mathematica, Maxima, Singular, Reduce, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cylindrical algebraic decomposition (CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Mathematica, QEPCAD, Reduce, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual substitution (VS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Reduce, ...</td>
</tr>
</tbody>
</table>

Strength: Efficient for conjunctions of real constraints.
Some corresponding implementations in CAS

<table>
<thead>
<tr>
<th>Gröbner bases</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CoCoA, F4, Maple, Mathematica, Maxima, Singular, Reduce, ...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cylindrical algebraic decomposition (CAD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematica, QEPCAD, Reduce, ...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual substitution (VS)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce, ...</td>
<td></td>
</tr>
</tbody>
</table>

Strength: Efficient for conjunctions of real constraints.

So why don’t we just plug in an algebraic decision procedure as theory solver into an SMT solver?
Why not use CAS out of the box?

- Theory solvers should be **SMT-compliant**, i.e., they should work **incrementally**, generate **lemmas** explaining inconsistencies, and be able to **backtrack**.
Why not use CAS out of the box?

- Theory solvers should be **SMT-compliant**, i.e., they should work incrementally, generate **lemmas** explaining inconsistencies, and be able to **backtrack**.

- Originally, the mentioned methods are not **SMT-compliant**, they are seldomly available as **libraries**, and are usually not **thread-safe**.
Why not use CAS out of the box?

- Theory solvers should be **SMT-compliant**, i.e., they should work **incrementally**, generate **lemmas** explaining inconsistencies, and be able to **backtrack**.

- Originally, the mentioned methods are not **SMT-compliant**, they are seldomly available as **libraries**, and are usually not **thread-safe**.

- Usually, SMT-adaptations are tricky.
Our SMT-RAT library

We have developed the SMT-RAT library of theory modules.

[Sat’12, SAT’15]

https://github.com/smtrat/smtrat/wiki
SMT Solver
Strategic composition of SMT-RAT modules

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

CArL
real-arithmetic computations

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

- SAT solver
- Manager
- Strategy
- Condition
- Module
- ...
Solver modules in SMT-RAT

- Libraries for basic computations [NFM’11, CAI’11]
- SAT solver
- CNF converter
- Preprocessing/simplifying modules
- Interval constraint propagation
- Simplex
- Virtual substitution [FCT’11, PhD Corzilius]
- CAD [CADE-24, PhD Loup, PhD Kremer]
- Gröbner bases [CAI’13]
- Generalised branch-and-bound
Solver modules in SMT-RAT

- Libraries for basic computations [NFM’11, CAI’11]
- SAT solver
- CNF converter
- Preprocessing/simplifying modules
- Interval constraint propagation
- Simplex
- Virtual substitution [FCT’11, PhD Corzilius]
- CAD [CADE-24, PhD Loup, PhD Kremer]
- Gröbner bases [CAI’13]
- Generalised branch-and-bound
Quantifier-free non-linear real arithmetic (QF-NRA)

Syntax

Polynomials:  \( p ::= 0 \mid 1 \mid x \mid p + p \mid p - p \mid (p \cdot p) \)

Constraints: \( c ::= p = 0 \mid p < 0 \)

Formulas: \( \varphi ::= c \mid \neg \varphi \mid \varphi \land \varphi \)

where \( x \) is a variable.

- **Syntactic sugar:** >, ≤, ≥, ≠, ∀, →, ....
- **Normal form:** \( p = a_1 x_1^{e_{1,1}} \cdots x_n^{e_{n,1}} + \ldots + a_k x_1^{e_{1,k}} \cdots x_n^{e_{n,k}}. \)
- **\( \text{deg}(p) \):** \( \max_{1 \leq j \leq k} (\sum_{i=1}^n e_{i,j}) \) degree of \( p \).
- \( \mathbb{Z}[x_1, \ldots, x_n] \) is the set of all polynomials over variables \( x_1, \ldots, x_n \).
### Syntax

| Polynomials: | $p ::= 0 \mid 1 \mid x \mid p + p \mid p - p \mid (p \cdot p)$ |
| Constraints:  | $c ::= p = 0 \mid p < 0$ |
| Formulas:    | $\varphi ::= c \mid \neg \varphi \mid \varphi \land \varphi$ |

where $x$ is a variable.

- **Syntactic sugar:** $\rightarrow, \ldots$.
- **Normal form:** $p = a_1 x_1^{e_{1,1}} \cdots x_n^{e_{n,1}} + \ldots + a_k x_1^{e_{1,k}} \cdots x_n^{e_{n,k}}$.
- **$\text{deg}(p)$:** $\max_{1 \leq j \leq k} (\sum_{i=1}^{n} e_{i,j})$ degree of $p$.
- **$\mathbb{Z}[x_1, \ldots, x_n]$** is the set of all polynomials over variables $x_1, \ldots, x_n$.
- Assume in the following:
  - **constraints** $C = \{p_1 \sim_1 0, \ldots, p_k \sim_k 0\}$
  - **polynomials** $P = \{p_1, \ldots, p_k\} \subseteq \mathbb{Z}[x_1, \ldots, x_n]$
  - **sign condition** $\sigma = (\sigma_1, \ldots, \sigma_k)$ with $\sigma_i = \begin{cases} -1 & \text{if } \sim_i \text{ is } < \\ 0 & \text{if } \sim_i \text{ is } = \\ 1 & \text{if } \sim_i \text{ is } > \end{cases}$
Solution sets and $P$-sign-invariant regions

Example

\[ p_1 = (x - 2)^2 + (y - 2)^2 - 1 \]

\[ p_2 = x - y \]

\[ C = \{ p_1 < 0, \ p_2 = 0 \} \]

\[ P = \{ p_1, \ p_2 \} \]

\[ \sigma = (-1, 0) \]
Solution sets and $P$-sign-invariant regions

Example

\[ p_1 = (x - 2)^2 + (y - 2)^2 - 1 \]
\[ p_2 = x - y \]

\[ C = \{ p_1 < 0, \ p_2 = 0 \} \]
\[ P = \{ p_1, \ p_2 \} \]
\[ \sigma = (-1, 0) \]

Solution set: \( S_\sigma(P) = \{ (a, a) \mid a \in (2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}) \} \)
Solution sets and $P$-sign-invariant regions

Example

\[ p_1 = (x - 2)^2 + (y - 2)^2 - 1 \]
\[ p_2 = x - y \]

\[ C = \{ p_1 < 0, \ p_2 = 0 \} \]
\[ P = \{ p_1, \ p_2 \} \]
\[ \sigma = (-1, 0) \]

Solution set: \( S_\sigma(P) = \{ (a, a) \mid a \in (2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}) \} \)

Solution set \( \equiv \) finite union of \( P \)-sign-invariant regions
A CAD for a set $P$ of polynomials from $\mathbb{Z}[x_1, \ldots, x_n]$ splits $\mathbb{R}^n$ into a finite number of $P$-sign-invariant regions.
Delineability on an example

\[ P = \{(x-2)^2 + (y-2)^2 - 1, x-y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x-2)^2 + (y-2)^2 - 1, x - y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x-2)^2 + (y-2)^2 - 1, x - y}\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \left\{ \left( x - 2 \right)^2 + \left( y - 2 \right)^2 - 1, x - y \right\} \]

\[ \sigma = \left( -1, 0 \right) \]
Delineability on an example

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\}, \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x-2)^2 + (y-2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \left\{ (x - 2)^2 + (y - 2)^2 - 1, x - y \right\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x-2)^2 + (y-2)^2 - 1, x-y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[
P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\}
\]

\[
\sigma = (-1, 0)
\]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ \begin{align*}
\mathbb{P} &= \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \\
\sigma &= (-1, 0)
\end{align*} \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, x - y \} \]

\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, \ x - y \} \]
\[ \sigma = (-1, 0) \]
Delineability on an example

\[ P = \{ (x - 2)^2 + (y - 2)^2 - 1, \ x - y \} \]
\[ \sigma = (-1, 0) \]
A CAD for a set $P$ of polynomials from $\mathbb{Z}[x_1, \ldots, x_n]$ splits $\mathbb{R}^n$ into a finite number of $P$-sign-invariant regions.
Example: CAD projection

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]
Example: CAD projection

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

Projection example (Brown-McCallum)

\[
\text{proj}(P) = \{2x^2 - 8x + 7, \ res(p_1, p_2) \\
x^2 - 4x + 3, \ disc(p_1) \\
\ldots \}
\]
Example: CAD projection

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

Projection example (Brown-McCallum)

\[
\text{proj}(P) = \{2x^2 - 8x + 7, \ res(p_1, p_2) \\
\quad x^2 - 4x + 3, \ disc(p_1) \\
\quad \ldots \} \]

\[
\{2 - \frac{\sqrt{2}}{2}, \ 2 + \frac{\sqrt{2}}{2}\} \]

\[
\{1, \ 3\} \quad \{\}
\]

...its real roots
Example: CAD projection

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

Projection example (Brown-McCallum)

\[
\begin{align*}
\text{proj}(P) &= \{2x^2 - 8x + 7, \text{res}(p_1, p_2) \}, \\
x^2 - 4x + 3, \text{disc}(p_1) \\
\ldots \}
\end{align*}
\]

...its real roots

\[
\{2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}\} \\
\{1, 3\} \\
\{\} 
\]
Example: CAD projection

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, x - y\} \]

Projection example (Brown-McCallum)

\[ \text{proj}(P) = \{2x^2 - 8x + 7, \ res(p_1, p_2) \]
\[ x^2 - 4x + 3, \ \text{disc}(p_1) \]
\[ \ldots \} \]

\[ \{2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}\} \]
\[ \{1, 3\} \]
\[ \{\} \]

... its real roots
Example: CAD sample construction

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\} \]

Samples for \( \text{proj}(P) \):

\[ \{1, 2 - \frac{\sqrt{2}}{2}, \ 2 + \frac{\sqrt{2}}{2}, \ 3\} \]
Example: CAD sample construction

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\} \]

Samples for \( \text{proj}(P) \):

\{1, 2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}, 3\}
\{0.5, 1.15, 2, 2.85, 3.5\}
Example: CAD sample construction

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\} \]

Samples for \( \text{proj}(P) \):

\[ \{1, 2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}, 3\} \]

\[ \{0.5, 1.15, 2, 2.85, 3.5\} \]
Example: CAD sample construction

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\} \]

Samples for \( \text{proj}(P) \): 
\[ \{1, 2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}, 3\} \]
\[ \{0.5, 1.15, 2, 2.85, 3.5\} \]

Sample construction: substitute the \( x \)-samples in \( P \) and determine the zeros of the resulting polynomials

Example sample construction for \( x = 2 \)

- \((2 - 2)^2 + (y - 2)^2 - 1\) yields \((2, 1)\) and \((2, 3)\).
- \(2 - y\) yields 2.
Example: CAD sample construction

\[
P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\}
\]

Samples for \(\text{proj}(P)\):

\{1, 2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}, 3\}
\{0.5, 1.15, 2, 2.85, 3.5\}

Sample construction: substitute the \(x\)-samples in \(P\) and determine the zeros of the resulting polynomials

Example sample construction for \(x = 2\)

- \((2 - 2)^2 + (y - 2)^2 - 1\) yields \((2, 1)\) and \((2, 3)\).
- \(2 - y\) yields 2.
Example: CAD sample construction

\[ P = \{(x - 2)^2 + (y - 2)^2 - 1, \ x - y\} \]

Samples for \( \text{proj}(P) \):

\{1, 2 - \frac{\sqrt{2}}{2}, 2 + \frac{\sqrt{2}}{2}, 3\}
\{0.5, 1.15, 2, 2.85, 3.5\}

Sample construction: substitute the \( x \)-samples in \( P \) and determine the zeros of the resulting polynomials

Example sample construction for \( x = 2 \)

- \((2 - 2)^2 + (y - 2)^2 - 1\) yields (2, 1) and (2, 3).
- \(2 - y\) yields 2.
Incrementality and backtracking

Polynomials $P = P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

Construction phase

CAD for $\mathbb{R}^n$

CAD for $\mathbb{R}^{n-1}$

CAD for $\mathbb{R}^2$

CAD for $\mathbb{R}^1$
Incrementality and backtracking

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$ for $n \geq 1$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

Construction phase

Projection phase

CAD for $R_1$

CAD for $R_2$

...  

CAD for $R_{n-1}$

CAD for $R_n$
Incrementality and backtracking

Projection phase

Construction phase

$P = \text{P}_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

$P_n - 1 \subseteq \mathbb{Z}[x_1, \ldots, x_n - 1]$

$\vdots$

$P_2 \subseteq \mathbb{Z}[x_1, x_2]$

$P_1 \subseteq \mathbb{Z}[x_1]$
Incrementality and backtracking

Polynomials

$P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$.

Polynomials

$P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$.

Polynomials

$P_2 \subseteq \mathbb{Z}[x_1, x_2]$.

Polynomials

$P_1 \subseteq \mathbb{Z}[x_1]$.

Projection phase

Construction phase
Incrementality and backtracking

Projection phase

Construction phase

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

CAD for $R_n$
Incrementality and backtracking

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

... 

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

Projection phase

Construction phase
Incrementality and backtracking

Projection phase

Construction phase
Incrementality and backtracking

Projection phase

\[ \mathbb{P} = \mathbb{P}_n \subseteq \mathbb{Z}[x_1, \ldots, x_n] \]

Polynomials

\[ \mathbb{P}_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}] \]

... 

Polynomials

\[ \mathbb{P}_2 \subseteq \mathbb{Z}[x_1, x_2] \]

Polynomials

\[ \mathbb{P}_1 \subseteq \mathbb{Z}[x_1] \]

Construction phase

CAD for \( R_1 \)

CAD for \( R_2 \)

... 

CAD for \( R_{n-1} \)

CAD for \( R_n \)
Incrementality and backtracking

Projection phase

Construction phase

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

$P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

$\ldots$

$P_2 \subseteq \mathbb{Z}[x_1, x_2]$

$P_1 \subseteq \mathbb{Z}[x_1]$
Incrementality and backtracking

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]

\ldots

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]

Construction phase

Projection phase
Incrementality and backtracking

Polynomials $P = P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

...  

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

Construction phase

Projection phase
Incrementality and backtracking

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

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$

Construction phase

Projection phase
Incrementality and backtracking

Projection phase

Construction phase

Polynomials

\[ \mathbb{P}_n \subseteq \mathbb{Z}[x_1, \ldots, x_n] \]

Polynomials

\[ \mathbb{P}_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}] \]

Polynomials

\[ \mathbb{P}_2 \subseteq \mathbb{Z}[x_1, x_2] \]

Polynomials

\[ \mathbb{P}_1 \subseteq \mathbb{Z}[x_1] \]
Incrementality and backtracking

Projection phase

Construction phase

Polynomials $P = P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$
Incrementality and backtracking

Projection phase

Construction phase
Incrementality and backtracking

Projection phase

Construction phase

Polynomials $P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

...
Incrementality and backtracking

Projection phase

Construction phase

Polynomials

\[ P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n] \]

Polynomials

\[ P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}] \]

...
Incrementality and backtracking

**Projection phase**

**Construction phase**

Polynomials $P = P_n \subseteq \mathbb{Z}[x_1, \ldots, x_n]$

Polynomials $P_{n-1} \subseteq \mathbb{Z}[x_1, \ldots, x_{n-1}]$

...  

Polynomials $P_2 \subseteq \mathbb{Z}[x_1, x_2]$

Polynomials $P_1 \subseteq \mathbb{Z}[x_1]$
Explanations

Projection phase

Construction phase
Explanations

Projection phase

Construction phase

Explanation for unsatisfiability: covering set
Explanations

Projection phase

Construction phase

Explanation for unsatisfiability: covering set
Explanations

Projection phase

Construction phase

Explanation for unsatisfiability:
covering set $S$
Heuristics

Projection phase

Construction phase
Heuristics

Projection phase

Construction phase

variable ordering
polynomial selection
sample point selection
polynomial selection
Heuristics

Projection phase

Construction phase

variable ordering
Heuristics

Projection phase

variable ordering
polynomial selection

Construction phase
Heuristics

Projection phase

Construction phase

variable ordering
polynomial selection
Heuristics

Projection phase

variable ordering
polynomial selection
...

Construction phase
Heuristics

Projection phase

variable ordering

polynomial selection

...

...
Heuristics

Projection phase

variable ordering
polynomial selection

...
Heuristics

Projection phase

Construction phase

variable ordering
polynomial selection

polynomial selection
Heuristics

Projection phase

variable ordering
polynomial selection

...)

Construction phase

polynomial selection
Heuristics

Projection phase

- variable ordering
- polynomial selection
- . . .

Construction phase

- polynomial selection
- . . .
Heuristics

Projection phase

- variable ordering
- polynomial selection

Construction phase

- sample point selection
- polynomial selection

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Heuristics

Projection phase

- variable ordering
- polynomial selection
-...

Construction phase

- sample point selection
- polynomial selection
-...

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Heuristics

Projection phase

variable ordering
polynomial selection

... 

Construction phase

sample point selection
polynomial selection

classification
polynomial selection
Heuristics

Projection phase

- variable ordering
- polynomial selection

Construction phase

- sample point selection
- polynomial selection
Heuristics

Projection phase

- variable ordering
- polynomial selection

Construction phase

- sample point selection
- polynomial selection

polynomial selection
Some experimental results

SMT-COMP 2016: Logic QF-NRA

<table>
<thead>
<tr>
<th>Solver</th>
<th>Sequential</th>
<th></th>
<th></th>
<th>Sequential (industrial)</th>
<th></th>
<th>Not solved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Err</td>
<td>Corrects</td>
<td>CPU</td>
<td>Err</td>
<td>Corrects</td>
<td>CPU</td>
</tr>
<tr>
<td>CVC3</td>
<td>0</td>
<td>3575</td>
<td>1220252.67</td>
<td>0</td>
<td>3572</td>
<td>1220206.65</td>
</tr>
<tr>
<td>CVC4</td>
<td>0</td>
<td>2694</td>
<td>1030.38</td>
<td>0</td>
<td>2693</td>
<td>1028.21</td>
</tr>
<tr>
<td>CVC4(exp)</td>
<td>0</td>
<td>2694</td>
<td>1021.11</td>
<td>0</td>
<td>2693</td>
<td>1018.93</td>
</tr>
<tr>
<td>SMT-RAT</td>
<td>0</td>
<td>8759</td>
<td>3451838.16</td>
<td>0</td>
<td>8706</td>
<td>3386768.21</td>
</tr>
<tr>
<td>Yices2-NL</td>
<td>0</td>
<td>9854</td>
<td>884237.95</td>
<td>0</td>
<td>9823</td>
<td>765719.51</td>
</tr>
<tr>
<td>[Z3]</td>
<td>0</td>
<td>10000</td>
<td>458920.89</td>
<td>0</td>
<td>9941</td>
<td>427115.80</td>
</tr>
<tr>
<td>raSAT</td>
<td>0</td>
<td>7952</td>
<td>4969778.73</td>
<td>0</td>
<td>7916</td>
<td>4863206.04</td>
</tr>
</tbody>
</table>
Satisfiability in the integer domain

\[ P = \{ (x - 4)^2 + (y - 4)^2 - 2, \ x - y \} \]
\[ \sigma = (-1, 1) \]
Satisfiability in the integer domain

\[ P = \{ (x - 4)^2 + (y - 4)^2 - 2, \ x - y \} \]

\[ \sigma = (-1, 1) \]
Satisfiability in the integer domain

Projection phase

Construction phase
Satisfiability in the integer domain

Projection phase

Construction phase

\[ \in \mathbb{Z} \]

zeros

\[ \epsilon \mathbb{Z} \]

zeros

choice → backtrack or B&B

no choice → explanation
Satisfiability in the integer domain

Projection phase

Construction phase

$\in \mathbb{Z}$

 zeros

substitute

zeros

→ backtrack or B&B

no choice

→ explanation

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Satisfiability in the integer domain

Projection phase

Construction phase

\[ \in \mathbb{Z} \]

zeros

substitute

zeros

choice \rightarrow backtrack

or B&B

no choice \rightarrow explanation
Satisfiability in the integer domain

Projection phase

Construction phase

\[ \epsilon \in \mathbb{Z} \]

zeros

substitute

zeros

substitute

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Satisfiability in the integer domain

Projection phase

Construction phase

\( \in \mathbb{Z} \)

\( \in \mathbb{Z} \)

zeros

substitute

zeros

substitute

zeros

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Satisfiability in the integer domain

Projection phase

Construction phase

$\in \mathbb{Z}$

zeros

substitute

$\in \mathbb{Z}$

does not choose → backtrack or B&B

does not choose → explanation
Satisfiability in the integer domain

Projection phase

Construction phase

\[ \in \mathbb{Z} \]

\[ \notin \mathbb{Z} \]

zeros

substitute

zeros

substitute

zeros

substitute

zeros
Satisfiability in the integer domain

Projection phase

Construction phase

\( \in \mathbb{Z} \)

\( \notin \mathbb{Z} \)

zeros

substitute

zeros

substitute

zeros

substitute

zeros
Satisfiability in the integer domain

Projection phase

Construction phase

\[ \in \mathbb{Z} \]

\[ \notin \mathbb{Z} \]

zeros
substitute

choice \(\rightarrow\) backtrack or B&B

zeros
Satisfiability in the integer domain

Projection phase

Construction phase

choice → backtrack or B&B

no choice → explanation
Satisfiability in the integer domain

Projection phase

- Choice → backtrack or B&B
- No choice → explanation

Construction phase

- $\in \mathbb{Z}$
- $\notin \mathbb{Z}$
- Zeros
- Substitute

Erika Ábrahám - SMT Solving for Non-Linear Arithmetic Theories
Some experimental results

<table>
<thead>
<tr>
<th>Benchmark → Solver↓</th>
<th>AProVE (8129)</th>
<th>CAlYPTO (138)</th>
<th>LEIPZIG (167)</th>
<th>CAlYPTO∞ (138)</th>
<th>all (8572)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>time</td>
<td>#</td>
<td>time</td>
<td>#</td>
</tr>
<tr>
<td>RATₗ</td>
<td>sat</td>
<td>7283</td>
<td>2294.8</td>
<td>67</td>
<td>71.2</td>
</tr>
<tr>
<td></td>
<td>unsat</td>
<td>73</td>
<td>14.3</td>
<td>52</td>
<td>40.7</td>
</tr>
<tr>
<td>RATblast</td>
<td>sat</td>
<td>8025</td>
<td>866.3</td>
<td>21</td>
<td>35.6</td>
</tr>
<tr>
<td></td>
<td>unsat</td>
<td>12</td>
<td>0.4</td>
<td>5</td>
<td>0.1</td>
</tr>
<tr>
<td>RATblast.Z</td>
<td>sat</td>
<td>8025</td>
<td>780.7</td>
<td>79</td>
<td>122.3</td>
</tr>
<tr>
<td></td>
<td>unsat</td>
<td>71</td>
<td>42.6</td>
<td>46</td>
<td>127.5</td>
</tr>
<tr>
<td>z3</td>
<td>sat</td>
<td>7992</td>
<td>14695.5</td>
<td>78</td>
<td>19.1</td>
</tr>
<tr>
<td></td>
<td>unsat</td>
<td>102</td>
<td>595.9</td>
<td>57</td>
<td>117.6</td>
</tr>
<tr>
<td>AProVE</td>
<td>sat</td>
<td>8025</td>
<td>7052.2</td>
<td>74</td>
<td>559.1</td>
</tr>
<tr>
<td></td>
<td>unsat</td>
<td>0</td>
<td>0.0</td>
<td>0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
SMT applications

- verification (model checking, static analysis, termination analysis)
- test case generation
- controller synthesis
- predicate abstraction
- equivalence checking
- scheduling
- planning
- product design automation
- optimisation
- . . .
Bounded model checking for C/C++

CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

CBMC verifies array bounds (buffer overflows), pointer safety, exceptions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using malloc and new. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: http://www.cprover.org/cbmc/
Termination analysis

Hybrid systems reachability analysis

**dReach** is a tool for safety verification of hybrid systems.

It answers questions of the type: Can a hybrid system run into an unsafe region of its state space? This question can be encoded to SMT formulas, and answered by our SMT solver. **dReach** is able to handle general hybrid systems with nonlinear differential equations and complex discrete mode-changes.

Source: [http://dreal.github.io/dReach/](http://dreal.github.io/dReach/)
Planning

Figure 1: A GEOMETRIC ROVERS example instance, showing the starting and goal locations of the rover, areas where tasks can be performed (blue) and obstacles (orange) and a plan solving the task (green). The red box indicates the bounds of the environment.

Upcoming research directions in SMT solving

**Improve usability:**
- User-friendly models
- Dedicated SMT solvers

**Increase scalability:**
- Performance optimisation (better lemmas, heuristics, cache behaviour, . . .)
- Novel combination of decision procedures
- Parallelisation

**Extend functionality:**
- Unsatisfiable cores, proofs, interpolants
- Quantified arithmetic formulas
- Linear and non-linear (global) optimisation