LibPoly: A Library for Reasoning about Polynomials

Dejan Jovanović    Bruno Dutertre

SRI International

SMT Workshop 2017
OUTLINE

INTRODUCTION

LIBPOLY
- Working with Polynomials
- Constructing a Sign Table
- Cylindrical Algebraic Decomposition

CONCLUSION
Outline

Introduction

LibPoly
- Working with Polynomials
- Constructing a Sign Table
- Cylindrical Algebraic Decomposition

Conclusion
Non-Linear Reasoning

Many applications

\[
T^X_1(t) = 3.2484 + 270.7t + 433.12t^2 - 324.83999t^3
\]

\[
T^Y_1(t) = 15.1592 + 108.28t + 121.2736t^2 - 649.67999t^3
\]

\[
T^Z_1(t) = 38980.8 + 5414t - 21656t^2 + 32484t^3
\]

\[
T^X_2(t) = 1.0828 - 135.35t + 234.9676t^2 + 3248.4t^3
\]

\[
T^Y_2(t) = 18.40759 - 230.6364t - 121.2736t^2 - 649.67999t^3
\]

\[
T^Z_2(t) = 40280.15999 - 10828t + 24061.9816t^2 - 32484t^3
\]

\[
D = 5 \quad H = 1000 \quad 0 \leq t \leq \frac{1}{20}
\]

\[
|T^Z_1(t) - T^Z_2(t)| \leq H \quad (T^X_1(t) - T^X_2(t))^2 + (T^Y_1(t) - T^Y_2(t))^2 \leq D^2
\]

Non-linear reasoning

Many applications

\[ T_1^X(t) = 3.2484 + 270.7t + 433.12t^2 - 324.8399t^3 \]
\[ T_1^Y(t) = 15.1592 + 108.28t + 121.2736t^2 - 649.67999t^3 \]
\[ T_1^Z(t) = 38980.8 + 5414t - 21656t^2 + 32484t^3 \]

**Run SMT solver**

\[ t \mapsto \frac{319}{16384} \approx 0.019470215 \]

\[ x^2 + 3248.4t^3 \]
\[ 36t^2 - 649.67999t^3 \]
\[ 9816t^2 - 32484t^3 \]

\[ D = 5 \quad H = 1000 \quad 0 \leq t \leq \frac{1}{20} \]

\[ |T_1^X(t) - T_2^X(t)| \leq H \quad (T_1^X(t) - T_2^X(t))^2 + (T_1^Y(t) - T_2^Y(t))^2 \leq D^2 \]

Example from Narkawicz, Muž, Formal Verification of Conflict Detection Algorithms for Arbitrary Trajectories, 2012
Popular techniques in SMT (QF_NRA):

- Interval reasoning: RASAT
- Linear reasoning + model-based refinement: cvc4
- DPLL(T) + VTS: veriT
- DPLL(T) + CAD: smtrat, veriT
- MCSAT + CAD: Z3, yices2
Non-Linear Reasoning

SMT Techniques

Popular techniques in SMT (QF_NRA):

- Interval reasoning: RASAT
- Linear reasoning + model-based refinement: CVC4
- DPLL(T) + VTS: VERIT
- DPLL(T) + CAD: SMTRAT, VERIT
- MCSAT + CAD: Z3, YICES2

Cylindrical Algebraic Decomposition (CAD):

- complete method, currently state-of-the-art;
- requires advanced polynomial operations.
The graph compares the cumulative time (in seconds) for solving benchmarks with various SMT solvers. The solvers include yices2, z3, verit+rasat+redlog, cvc4, and smtrat. The x-axis represents the number of benchmarks solved, and the y-axis shows the cumulative time. The data is presented on a logarithmic scale, indicating that the time increases significantly as the number of benchmarks grows.
Non-Linear Reasoning
CAD-based reasoning

1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanations:
   - principal subresultant coefficients.
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**HOW TO GET THESE?**
- Use an existing library
Non-Linear Reasoning

CAD-based reasoning

1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

How to get these?
- Use an existing library 😞
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**How to get these?**
- Use an existing library 😞
- Use a computer algebra system
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**How to get these?**
- Use an existing library 😞
- Use a computer algebra system 😞
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**How to get these?**
- Use an existing library 😞
- Use a computer algebra system 😞
- Borrow and adapt code
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**HOW TO GET THESE?**

- Use an existing library 😞
- Use a computer algebra system 😞
- Borrow and adapt code 😞
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**HOW TO GET THESE?**

- Use an existing library 😞
- Use a computer algebra system 😞
- Borrow and adapt code 😞
- Implement yourself 😞
1. Representation of polynomials.
2. Basic operations:
   - variables, variable ordering;
   - arithmetic (addition, multiplication, ...);
   - GCD computation;
   - some factorization.
3. Solving and model representation:
   - Sturm sequences;
   - interval reasoning;
   - root isolation (multivariate);
   - resultants;
   - computation with algebraic numbers.
4. Projection and symbolic explanation:
   - principal subresultant coefficients.

**How to get these?**
- Use an existing library 😞
- Use a computer algebra system 😞
- Borrow and adapt code 😞
- Implement yourself 😞
- Use LIBPOLY 😊.
INTRODUCTION

LibPoly
- Working with Polynomials
- Constructing a Sign Table
- Cylindrical Algebraic Decomposition

CONCLUSION
LibPoly

- Open source: https://github.com/SRI-CSL/libpoly.
- Permissive License: LGLP
- Lightweight: Implemented in C, 15KLOC.
- Only depends on GMP.
- Basis for non-linear reasoning in YICES2.
Polynomial Basics

- Polynomials with coefficients over \(\mathbb{Z}\).
- \(\mathbb{Z}[x_1, \ldots, x_n]\) are polynomials over variables \(\vec{x} = \langle x_1, \ldots, x_n \rangle\).
- For \(f \in \mathbb{Z}[\vec{y}, x]\):

\[
    f(\vec{y}, x) = a_m \cdot x^{d_m} + a_{m-1} \cdot x^{d_{m-1}} + \cdots + a_1 \cdot x^{d_1} + a_0
\]

- \(a_m \neq 0, a_i \in \mathbb{Z}[\vec{y}], d_m > \cdots > d_1 > 0\)
- \(x\) is the top variable
- \(d_m\) is the degree of \(f\)
- \(a_m\) is the leading coefficient
ASSIGNMENT AND EVALUATION

An assignment assigns variables to values

\[ m = \{ x \mapsto 1, y \mapsto 2, z \mapsto 3 \} . \]

We can evaluate the sign of a polynomial \( f \in \mathbb{Z}[x, y, z] \)

\[ \text{sgn}(f, m) \in \{ +1, 0, -1 \} . \]
Zeros of a Polynomial (Root Isolation)

**Root Isolation**

For $f \in \mathbb{Z}[\vec{y}, x]$ and an assignment $\vec{y} \mapsto \vec{v}$, find solutions to $f(\vec{v}, x) = 0$.

**Example**

- $m_1 = \{\}$
- $m_2 = \{x \mapsto 1\}$
- $m_3 = \{x \mapsto 1, y \mapsto \sqrt{2}\}$

- $f_1(x) = x - 1$
- $f_2(x, y) = y^2 - 2x$
- $f_3(x, y, z) = z^3 - y^2 - x$
### Example (Sign Table)

<table>
<thead>
<tr>
<th></th>
<th>$(-\infty,-1)$</th>
<th>$[-1]$</th>
<th>$(-1,0)$</th>
<th>$[0]$</th>
<th>$(0,1)$</th>
<th>$[1]$</th>
<th>$(1,2)$</th>
<th>$[2]$</th>
<th>$(2,\infty)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2 - 1$</td>
<td>$+$</td>
<td>$0$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$0$</td>
<td>$+$</td>
<td>$+$</td>
<td>$+$</td>
</tr>
<tr>
<td>$x(x-2)$</td>
<td>$+$</td>
<td>$+$</td>
<td>$+$</td>
<td>$0$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$0$</td>
<td>$+$</td>
</tr>
</tbody>
</table>
**Sign Table: What is it?**

**Example (Sign Table)**

<table>
<thead>
<tr>
<th></th>
<th>$(-\infty, -1)$</th>
<th>$[-1]$</th>
<th>$(-1, 0)$</th>
<th>$[0]$</th>
<th>$(0, 1)$</th>
<th>$[1]$</th>
<th>$(1, 2)$</th>
<th>$[2]$</th>
<th>$(2, +\infty)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2 - 1$</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>$x(x - 2)$</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Sign Table**

- Partition of $\mathbb{R}$ into intervals $I_1, \ldots I_n$.
- Picking an **arbitrary** sample value $v \in I_k$ is enough to evaluate signs.
- It completely characterizes the behavior of the polynomials.
Can we do multivariate?

Example (Multivariate)

\[ x^2 + y^2 - 1 \leq 0 \quad , \quad (x - 1)^2 + y^2 - 1 \leq 0 . \]
Can we do multivariate?

Example (Multivariate)

\[ x^2 + y^2 - 1 \leq 0, \quad (x - 1)^2 + y^2 - 1 \leq 0. \]

Recursive Sign Table

1. Pick order, say \( x < y \).
2. \( P_x \): polynomials in \( x \).
3. \( P_y \): polynomials in \( x, y \).
4. Construct sign table \( T_x \) for \( P_x \).
5. For each sample \( v \in T_x \):
   - Construct sign table \( T_{v,y} \) for \( P_y \).
**Can we do multivariate?**

**Recursive Sign Table**

1. Pick order, say \( x < y \).
2. \( P_x \): polynomials in \( x \).
3. \( P_y \): polynomials in \( x, y \).
4. Construct sign table \( T_x \) for \( P_x \).
5. For each sample \( v \in T_x \):
   - Construct sign table \( T_{v,y} \) for \( P_y \).

**Example (Multivariate)**

\[
x^2 + y^2 - 1 \leq 0 , \quad (x - 1)^2 + y^2 - 1 \leq 0 .
\]
Can we do multivariate?

Example (How to Get the Extra Polynomials?)

We added extra polynomials

\[ x + 1, \quad x, \quad 2x - 1, \quad x - 1, \quad x - 2. \]

Can we find these polynomials automatically?
Given a set of polynomials $A = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[\vec{y}, x]$, the $x$-projection of $A$ is

$$P(A, x) = \bigcup_{f \in A} \text{coeff}(f, x) \cup \bigcup_{f \in A} \text{psc}(g, g', x) \cup \bigcup_{i<j} \text{psc}(g_i, g_j, x).$$
**Definition (Projection)**

Given a set of polynomials $A = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[\vec{y}, x]$, the x-projection of $A$ is

$$P(A, x) = \bigcup_{f \in A} \text{coeff}(f, x) \cup \bigcup_{f \in A} \text{psc}(g, g'_x, x) \cup \bigcup_{i<j} \text{psc}(g_i, g_j, x).$$

**coeff(f, x): Coefficients**

Signs of coefficients invariant on $S \Rightarrow$ degrees of $f \in A$ invariant on $S.$
**Definition (Projection)**

Given a set of polynomials \( A = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[\vec{y}, x] \), the x-projection of \( A \) is

\[
P(A, x) = \bigcup_{f \in A} \text{coeff}(f, x) \cup \bigcup_{f \in A, \ g \in R^*(f, x)} \text{psc}(g, g_x', x) \cup \bigcup_{i < j, \ g_i \in R^* (f_i, x), \ g_j \in R^* (f_j, x)} \text{psc}(g_i, g_j, x).
\]

**\( R^*(f, x) \): Reductums include the “right degree” polynomials**

\[
f = \sum_{k=0}^{n} a_k x^k, \quad R(f, x) = \sum_{k=0}^{n-1} a_k x^k, \quad R^*(f, x) = \{f, R(f), R(R(f)), \ldots\}.
\]
**Definition (Projection)**

Given a set of polynomials $A = \{f_1, \ldots, f_m\} \subset \mathbb{Z}[\bar{y}, x]$, the $x$-projection of $A$ is

$$P(A, x) = \bigcup_{f \in A} \text{coeff}(f, x) \cup \bigcup_{f \in A} \text{psc}(g, g'_x, x) \cup \bigcup_{i<j} \text{psc}(g_i, g_j, x).$$

**Principal Subresultant Coefficients (PSC)**

Signs of PSC invariant on $S \Rightarrow$ degree of $\gcd$ invariant on $S$. 
Given a set of polynomials $A \subseteq \mathbb{Z}[x_1, \ldots, x_n]$:

- Project variable $x_n$.
- Project variable $x_{n-1}$.
- ...
LIFTING: CONSTRUCT THE SIGN TABLE

Construct the table variable by variable:

- Isolate roots of $x_1$, pick a value in an interval.
- Isolate roots of $x_2$, pick a value in an interval.
- ...
Construct the table variable by variable:

- Isolate roots of $x_1$, pick a value in an interval.
- Isolate roots of $x_2$, pick a value in an interval.
- ...
Polynomial: $x^2 + y^2 - 1$.
Projection: $x^2 - 1$. 
INTRODUCTION

LIBPOLY
- Working with Polynomials
- Constructing a Sign Table
- Cylindrical Algebraic Decomposition

CONCLUSION
A library for non-linear reasoning:

- Permissive License: LGLP
- Ubuntu and Brew packages incoming.
- Lightweight: Implemented in C, around 15KLOC.
- Only depends on GMP.
- Basis for non-linear reasoning in YICES2.
- Both for traditional CAD and MCSAT-style CAD.
A library for non-linear reasoning:

- Permissive License: LGLP
- Ubuntu and Brew packages incoming.
- Lightweight: Implemented in C, around 15KLOC.
- Only depends on GMP.
- Basis for non-linear reasoning in YICES2.
- Both for traditional CAD and MCSAT-style CAD.

Thank you!