Polya

A Heuristic Procedure for Reasoning with Real Inequalities

Robert Y. Lewis

Carnegie Mellon University

November 24, 2014

Table of Contents

Computers in mathematics

There are two broad ways in which computers are used to construct mathematical proofs:

- Automated reasoning
- Interactive theorem proving

The two aren't independent!

Automated reasoning

There is a long history of mathematicians searching for decision procedures for various domains:

- Boolean satisfiability
- Linear arithmetic
- Real closed fields

A computer runs these algorithms with no guidance from the mathematician.

Interactive theorem proving

In interactive theorem proving, the user describes a proof to the computer, which fills in the gaps.

Some popular systems:

- Mizar
- Isabelle
- Coq
- Lean

Some significant formalizations:

- Kepler conjecture (Hales et al, 2014)
- Feit-Thompson theorem (Gonthier et al, 2013)
- Four-color theorem (Gonthier et al, 2005)

Interactive theorem proving

Interactive proofs often involve calls to automated reasoning tools, to fill in larger gaps.

These tactics save the user the trouble of writing out long, tedious calculations.

But, they must be proof-producing to be of any use.

Real closed fields

Consider the language L with:

- Constants 0 and 1
- Operations $+$ and $\cdot$
- Relations $>$ and $=$

The theory of $\mathbb{R}$ under L is known as the theory of real closed fields.

Theorem

For all x and y, if $0 < x < 1$ and $0 < y < 1$, then $x^{500} + y^{500} > x^{500} \cdot y^{500}$

Real closed fields

The theory of real closed fields admits quantifier elimination, and is thus decidable, by Tarski (1948).

But, Tarski's algorithm is extremely inefficient. Modern alternatives (cylindrical algebraic decomposition) are better, but still unintuitively complex.

Furthermore, it is difficult to extend these methods beyond the language L, or to make them proof-producing.

A motivating example

Consider the following implication:

$$0 < x < y, \ u < v$$

$$\Longrightarrow$$

$$2u + \exp(1 + x + x^4) < 2v + \exp(1 + y + y^4)$$

A motivating example

Consider the following implication:

$$0 < x < y, \;\; u < v$$
$$\Longrightarrow$$
$$2u + \exp(1 + x + x^4) < 2v + \exp(1 + y + y^4)$$

- This inference is not contained in linear arithmetic or real closed fields.
- This inference is tight: symbolic or numeric approximations to exp are not useful.
- Backchaining using monotonicity properties suggests many equally plausible subgoals.
- But, the inference is completely straightforward.

A new method

We propose and implement a method based on this type of heuristically guided forward reasoning. Our method:

- Verifies inequalities on which other procedures fail.
- Is amenable to producing proof terms.
- Captures natural, human-like inferences.

A new method

We propose and implement a method based on this type of heuristically guided forward reasoning. Our method:

- Verifies inequalities on which other procedures fail.
- Is amenable to producing proof terms.
- Captures natural, human-like inferences.

- Is not complete.
- Is not guaranteed to terminate.

A new method

We propose and implement a method based on this type of heuristically guided forward reasoning. Our method:

- Verifies inequalities on which other procedures fail.
- Is amenable to producing proof terms.
- Captures natural, human-like inferences.

- Is not complete.
- Is not guaranteed to terminate.

We envision it as a complement, not a replacement, to other verification procedures.

A prototype implementation of the algorithm in Python, named Polya, shows that the method compares favorably to other techniques.

The code is open-source and available online.

Background

Our system verifies inequalities between real variables using:

- Operations $+$ and $\cdot$
- Multiplication and exponentiation by rational constants
- Arbitrary function symbols
- Relations $<$ and $=$

All functions are assumed to be total. $1/0$, $\sqrt{-1}$, etc. exist, but no assumptions are made about their values.

As is common for resolution theorem proving, we establish a theorem by negating the conclusion and deriving a contradiction.

Proving the unsatisfiability of

$$\left(\bigwedge_{i=1}^{k} \varphi_i(x_1, \ldots, x_n)\right)$$

is logically equivalent to proving the validity of

$$(\forall x_1) \ldots (\forall x_n) \left(\bigvee_{i=1}^{k} \neg\varphi_i(x_1, \ldots, x_n)\right)$$

or equivalently

$$(\forall x_1) \ldots (\forall x_n) \left(\bigwedge_{i=1}^{k-1} \varphi_i(x_1, \ldots, x_n) \to \neg\varphi_k(x_1, \ldots, x_n)\right)$$

Terms and normal forms

The key ideas behind the algorithm:

- Normal forms for terms and named subterms of different types.
- Decentralized modules communicating with a central database.

Terms and normal forms

The inequality

$$15 < 3(3y + 5x + 4xy)^2 f(u+v)^{-1}$$

is expressed canonically as

$$\underbrace{\mathbf{1}}_{t_0} \leq 5 \cdot (\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2})^2 f(\underbrace{u}_{t_4} + \underbrace{v}_{t_5})^{-1}$$

Terms and normal forms

The inequality

$$15 < 3(3y + 5x + 4xy)^2 f(u + v)^{-1}$$

is expressed canonically as

$$\underbrace{\mathbf{1}}_{t_0} \leq 5 \cdot (\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})^{-1}$$

Terms and normal forms

The inequality

$$15 < 3(3y + 5x + 4xy)^2 f(u+v)^{-1}$$

is expressed canonically as

$$\underbrace{\mathbf{1}}_{t_0} \leq 5 \cdot \Big(\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3}\Big)^2 \underbrace{f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})}_{t_8 = f(t_7)}^{-1}$$

Terms and normal forms

The inequality

$$15 < 3(3y + 5x + 4xy)^2 f(u + v)^{-1}$$

is expressed canonically as

$$\underbrace{\mathbf{1}}_{t_0} \leq 5 \cdot \underbrace{(\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 f(\underbrace{\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5}}_{t_8 = f(t_7)})^{-1}}_{t_9 = t_6^2 t_8^{-1}}$$

Terms and normal forms

The inequality

$$15 < 3(3y + 5x + 4xy)^2 f(u + v)^{-1}$$

is expressed canonically as

$$\underbrace{\underbrace{\mathbf{1}}_{t_0} \leq 5 \cdot \underbrace{\underbrace{(\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 \underbrace{f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})}_{t_8 = f(t_7)}{}^{-1}}_{t_9 = t_6^2 t_8^{-1}}}}_{t_0 \leq 5 t_9}$$

Modules and database

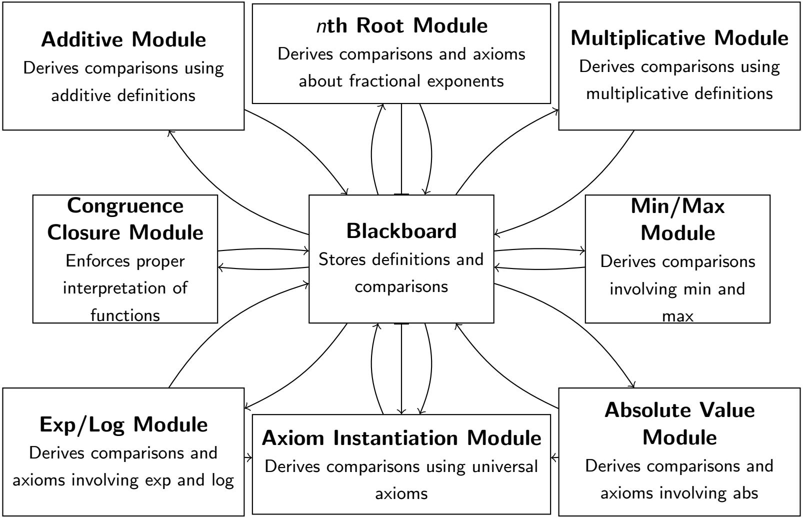
Any comparison between canonical terms can be expressed as $t_i \bowtie 0$ or $t_i \bowtie c \cdot t_j$, where $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$. This is in the common language of addition and multiplication.

A central database (the blackboard) stores term definitions and comparisons of this form.

Modules use this information to learn and assert new comparisons.

The procedure has succeeded in verifying an implication when modules assert contradictory information.

Computational structure



Arithmetical modules

Two modules, for additive and multiplicative arithmetic, work together to solve arithmetical problems.

Using the known atomic comparisons and definitions, the modules saturate the blackboard with the strongest derivable atomic comparisons.

We use two techniques for this: Fourier-Motzkin variable elimination, and a geometric projection method.

Fourier-Motzkin additive module

The Fourier-Motzkin algorithm is a quantifier elimination procedure for $\langle \mathbb{R}, 0, +, < \rangle$.

Given additive equations $\{t_i = \sum_j c_j \cdot t_{k_j}\}$ and atomic comparisons $\{t_i \bowtie c \cdot t_j\}$ and $\{t_i \bowtie 0\}$:

- For each pair i, j, eliminate all variables except t_i and t_j.
- Add the strictest remaining comparisons to the blackboard.

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

$$\begin{aligned} 3t_1 + 2t_2 - t_3 &> 0 \\ 4t_1 + t_2 + t_3 &\geq 0 \\ 2t_1 - t_2 - 2t_3 &\geq 0 \\ -2t_2 - t_3 &> 0 \end{aligned}$$

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

$$\left.\begin{aligned} 3t_1 + 2t_2 - t_3 &> 0 \\ 4t_1 + t_2 + t_3 &\geq 0 \\ 2t_1 - t_2 - 2t_3 &\geq 0 \\ -2t_2 - t_3 &> 0 \end{aligned}\right. \implies 7t_1 + 3t_2 > 0$$

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

$$\begin{aligned} 3t_1 + 2t_2 - t_3 &> 0 \\ 4t_1 + t_2 + t_3 &\geq 0 \\ 2t_1 - t_2 - 2t_3 &\geq 0 \\ -2t_2 - t_3 &> 0 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} 7t_1 + 3t_2 &> 0 \\ 10t_1 + t_2 &\geq 0 \end{aligned}$$

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

$$\begin{array}{r} 3t_1 + 2t_2 - t_3 > 0 \\ 4t_1 + t_2 + t_3 \geq 0 \\ 2t_1 - t_2 - 2t_3 \geq 0 \\ -2t_2 - t_3 > 0 \end{array} \implies \begin{array}{r} 7t_1 + 3t_2 > 0 \\ 10t_1 + t_2 \geq 0 \\ 4t_1 - t_2 > 0 \end{array}$$

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, find the strongest pair:

$$\begin{aligned} 3t_1 + 2t_2 - t_3 &> 0 \\ 4t_1 + t_2 + t_3 &\geq 0 \\ 2t_1 - t_2 - 2t_3 &\geq 0 \\ -2t_2 - t_3 &> 0 \end{aligned} \implies \begin{aligned} 7t_1 + 3t_2 &> 0 \\ 10t_1 + t_2 &\geq 0 \\ 4t_1 - t_2 &> 0 \end{aligned} \implies \begin{aligned} t_1 &> -\frac{3}{7}t_2 \\ t_1 &\geq -\frac{1}{10}t_2 \\ t_1 &> \frac{1}{4}t_2 \end{aligned}$$

Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, find the strongest pair:

$$\begin{aligned} 3t_1 + 2t_2 - t_3 &> 0 \\ 4t_1 + t_2 + t_3 &\geq 0 \\ 2t_1 - t_2 - 2t_3 &\geq 0 \\ -2t_2 - t_3 &> 0 \end{aligned} \implies \begin{aligned} 7t_1 + 3t_2 &> 0 \\ 10t_1 + t_2 &\geq 0 \\ 4t_1 - t_2 &> 0 \end{aligned} \implies \begin{aligned} t_1 &> -\frac{3}{7}t_2 \\ t_1 &\geq -\frac{1}{10}t_2 \\ t_1 &> \frac{1}{4}t_2 \end{aligned}$$

Fourier-Motzkin multiplicative module

By the map $x \mapsto e^x$, we see that $\langle \mathbb{R}, 0, +, < \rangle \cong \langle \mathbb{R}^+, 1, \cdot, < \rangle$.

We can therefore use the same elimination procedure on multiplicative terms, with some caveats:

- Sign information is needed for all variables.
- Constants become irrational under the transformation.
- Deduced comparisons can have the form $t_i^3 < 2t_j^2$.

Preprocessing techniques to infer sign information can help with the first point.

Fourier-Motzkin arithmetical modules

The FM algorithm can require doubly-exponential time in the number of variables.

In a problem with n subterms, one pass of the additive module requires $O(n^2)$ instances of FM with up to n variables in each.

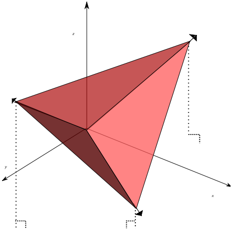
In practice, this approach works surprisingly well. But one can construct examples where the complexity leads to significant slowdown.

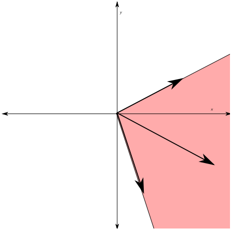
Geometric additive module

An alternative approach uses geometric insights.

A homogeneous linear equality in n variables defines an $(n-1)$-dimensional hyperplane through the origin in $\mathbb{R}^n$. An inequality defines a half-space. A conjunction of inequalities defines a polyhedron.

By projecting this polyhedron to the $t_i t_j$ plane, one can find the strongest implied comparisons between t_i and t_j.





Geometric arithmetical modules

We use the computational geometry packages cdd and lrs for the conversion from half-plane representation to vertex representation.

This approach scales better than the Fourier-Motzkin procedure.

Geometric multiplicative module

Translating this procedure to the multiplicative setting introduces a new problem:

$$5t_2^2t_1^4 \mapsto \underbrace{\log(5)}_{\notin\mathbb{Q}} + 2\log(t_2) + 4\log(t_1)$$

To avoid this, we introduce new variables

$$p_2 = \log(2), p_3 = \log(3), p_5 = \log(5), \ldots$$

as necessary.

Axiom instantiation module

A highlight of our approach is its ability to prove theorems outside the theory of real closed fields.

An axiom instantiation module takes universally quantified axioms about function symbols and selectively instantiates them with subterms from the problem.

Example

$$\text{Using axiom: } (\forall x)(0 < f(x) < 1)$$

$$\text{Prove: } f(a)^3 + f(b)^3 > f(a)^3 \cdot f(b)^3$$

Axiom instantiation module

Unification must happen modulo equalities:

Example

Unify $f(v_1 + v_2)$:

Definitions	Assignments	Result
$t_1 = t_2 + t_3$	$v_1 \mapsto t_2 - t_5$	$f(v_1 + v_2) \equiv t_6$
$t_4 = 2t_3 - t_5$	$v_2 \mapsto 3t_3$	
$t_6 = f(t_1 + t_4)$		

We combine a standard unification algorithm with a Gaussian elimination procedure to find relevant substitutions.

Trigger terms can be specified by the user or picked by default.

Built-in functions

In addition to the arithmetic and axiom modules, Polya has modules that interpret specific functions.

- Exponentials and logarithms
- Minima and maxima
- Absolute values
- Various others

Exponential and logarithm module

The exponential module asserts axioms that say $\exp(x)$ is positive and increasing.

It also adds identities of the forms

$$\exp(c \cdot t) = \exp(t)^c$$

$$\exp\left(\sum c_i t_i\right) = \prod \exp\left(c_i t_i\right).$$

It adds similar axioms and identities for $\log(x)$, conditional on $x > 0$.

Minimum module

For any term $t := \min(c_1 t_1, \ldots, c_n t_n)$, the minimum module asserts

$$t \leq c_i t_i$$

$$(\forall z)\left(\bigwedge_i (z \leq c_i t_i) \rightarrow z \leq t\right).$$

Since $\max(c_1 t_1, \ldots, c_n t_n) = -\min(-c_1 t_1, \ldots, -c_n t_n)$, it does not need to be handled separately.

Absolute value module

The absolute value module asserts axioms

$$(\forall x)\,(|x| \geq 0 \wedge |x| \geq x \wedge |x| \geq -x)$$

$$(\forall x)\,(x \geq 0 \rightarrow |x| = x)$$

$$(\forall x)\,(x \leq 0 \rightarrow |x| = -x)$$

and looks for appropriate instantiations of the triangle inequality

$$\left|\sum c_i t_i\right| \leq \sum |c_i t_i|\,.$$

Builtins module

The builtin functions module asserts miscellaneous axioms about various functions:

$$(\forall x)(-1 \leq \sin(x) \leq 1)$$

$$(\forall x)(-1 \leq \cos(x) \leq 1)$$

$$(\forall x)(x - 1 < \lfloor x \rfloor \leq x)$$

$$\vdots$$

Successes

Our implementation in Python successfully proves many theorems, some of which are not proved by other systems.

$$0 < x < 1 \implies 1/(1-x) > 1/(1-x^2) \tag{1}$$

$$0 < u,\ u < v,\ 0 < z,\ z+1 < w \implies (u+v+z)^3 < (u+v+w)^5 \tag{2}$$

$$(\forall x, y.\ x \leq y \to f(x) \leq f(y)),\ u < v,\ 1 < v,\ x \leq y \implies u + f(x) \leq v^2 + f(y) \tag{3}$$

Successes

$$(\forall x, y.\ f(x+y) = f(x)f(y)),\ f(a+b) > 2,\ f(c+d) > 2 \implies f(a+b+c+d) > 4 \tag{4}$$

$$u > 0,\ v > 1 \implies \sqrt[3]{u^9 v^4} > u^3 v \tag{5}$$

$$x < y,\ u \leq v \implies u + \min(x+2u, y+2v) \leq x + 3v \tag{6}$$

$$y > \max(2, 3x),\ x > 0 \implies \exp(4y - 3x) > \exp(6) \tag{7}$$

Limitations

Since our method is incomplete, it fails on a wide class of problems where other methods succeed.

$$x > 0,\ xyz < 0,\ xw > 0 \implies w > yz \tag{8}$$

$$x^2 + 2x + 1 \geq 0 \tag{9}$$

$$\begin{aligned} 4 \leq x_i \leq 6.3504 \implies & \\ & x_1x_4(-x_1 + x_2 + x_3 - x_4 + x_5 + x_6) \\ & + x_2x_5(x_1 - x_2 + x_3 + x_4 - x_5 + x_6) \\ & + x_3x_6(x_1 + x_2 - x_3 + x_4 + x_5 + -x_6) \\ & - x_2x_3x_4 - x_1x_3x_5 - x_1x_2x_6 - x_4x_5x_6 > 0 \end{aligned} \tag{10}$$

Future work

There are a number of directions for improvement:

- Improved case-splitting and CDCL.
- Backtracking and incrementality.
- Proof-producing implementation.
- Heuristically handle distribution.
- More modules for more tasks.

Future work

There are a number of directions for improvement:

- Improved case-splitting and CDCL.
- Backtracking and incrementality.

- Proof-producing implementation.
- Heuristically handle distribution.
- More modules for more tasks.
    - Trigonometry, integers
    - Arbitrary summations, products, integrals

Future work

There are a number of directions for improvement:

- Improved case-splitting and CDCL.
- Backtracking and incrementality.
- Proof-producing implementation.
- Heuristically handle distribution.
- More modules for more tasks.
  - Trigonometry, integers
  - Arbitrary summations, products, integrals
- Extending the idea beyond real inequalities?

Thank you!

Our code, a collection of 80 examples, and comparison data is available at:

https://github.com/avigad/polya