# A Heuristic Prover for Real Inequalities

Jeremy Avigad, Robert Y. Lewis, and Cody Roux

Carnegie Mellon University

July 16, 2014

## 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 $=$

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

## Terms and normal forms

The term

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

is expressed canonically as

$$75 \cdot (\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2})^2 f(\underbrace{u}_{t_4} + \underbrace{v}_{t_5})^{-1}$$

## Terms and normal forms

The term

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

is expressed canonically as

$$75 \cdot (\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})^{-1}$$

## Terms and normal forms

The term

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

is expressed canonically as

$$75 \cdot (\underbrace{\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 \underbrace{f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})}_{t_8 = f(t_7)}{}^{-1}$$

## Terms and normal forms

The term

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

is expressed canonically as

$$75 \cdot \underbrace{\underbrace{(\underbrace{x}_{t_1} + \frac{3}{5} \cdot \underbrace{y}_{t_2} + \frac{4}{5} \cdot \underbrace{xy}_{t_3 = t_1 t_2}}_{t_6 = t_1 + \frac{3}{5} t_2 + \frac{4}{5} t_3})^2 \underbrace{f(\underbrace{\underbrace{u}_{t_4} + \underbrace{v}_{t_5}}_{t_7 = t_4 + t_5})}_{t_8 = f(t_7)}{}^{-1}}_{t_9 = t_6^2 t_8^{-1}}$$

## Modules and database

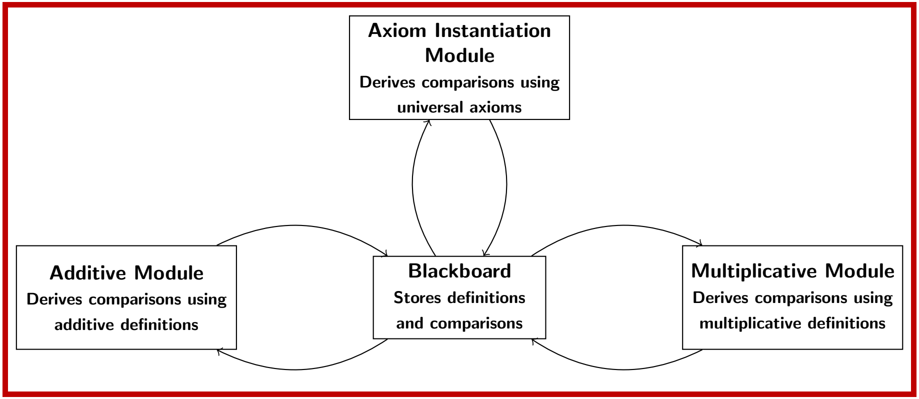
Any comparison between canonical terms can be expressed as $t_i \bowtie 0$ or $t_i \bowtie c \cdot t_j$, where $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$. This is in the common language of addition and multiplication.

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.

## Polya: an outline


Axiom Instantiation Module
Derives comparisons using universal axioms
Additive Module
Derives comparisons using additive definitions
Blackboard
Stores definitions and comparisons
Multiplicative Module
Derives comparisons using multiplicative definitions

## 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 $\{t_i \bowtie 0\}$:

- Substitute the equations into the comparisons.
- 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 $t_3$:

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

## Fourier-Motzkin additive module

To find comparisons between $t_1$ and $t_2$, eliminate $t_3$:

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

## Fourier-Motzkin additive module

To find comparisons between $t_1$ and $t_2$, eliminate $t_3$:

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

## 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 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 $t_i^3 < 2t_j^2$.

## 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.

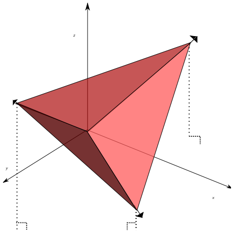
In practice, this approach works surprisingly well. But one can construct examples where the complexity leads to significant slowdown.

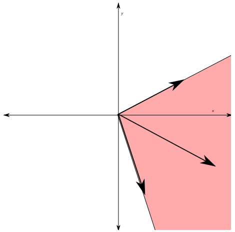
## Geometric additive module

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 language of RCF.

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

$$\begin{aligned} \text{Using axiom: } & (\forall x)(0 < f(x) < 1) \\ \text{Prove: } & f(a)^3 + f(b)^3 > f(a)^3 \cdot f(b)^3 \end{aligned}$$

## Axiom instantiation module

Unification must happen modulo equalities:

$$\begin{array}{l} t_1 = t_2 + t_3 \\ t_4 = 2t_3 - t_5 \\ t_6 = f(t_1 + t_4) \end{array} \quad , \quad \begin{array}{l} v_1 \mapsto t_2 - t_5 \\ v_2 \mapsto 3t_3 \end{array} \quad \Longrightarrow \quad f(v_1 + v_2) \equiv t_6$$

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. This lets the module selectively add terms of interest to the blackboard.

## Successes

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

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

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

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

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

## Limitations

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

$$x > 0, y < z \implies xy < xz \tag{5}$$

$$x^2 + 2x + 1 \geq 0 \tag{6}$$

## 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.
  - Exponentials and logarithms
  - Minima and maxima
  - Arbitrary summations, products, integrals

## Thank you!

Our code, a collection of 50 examples, and comparison data is available at:

`https://github.com/avigad/polya`