A heuristic prover for real inequalities

Jeremy Avigad

Department of Philosophy and
Department of Mathematical Sciences
Carnegie Mellon University

joint work with Robert Y. Lewis and Cody Roux
(many slides borrowed from Rob's presentations)

June 2015

Computer support for mathematics

Two variables:

- level of user interaction
- strength of correctness guarantees

Both are continua.

User interaction

In interactive theorem provers (Lean, Coq, Isabelle, HOL-light), users construct proofs interactively.

Computer algebra systems allow interactive exploration.

Automated reasoning systems (resolution thoerem provers, SMT solvers, . . .) provide "push-button" service.

Numerical software packages (linear and nonlinear optimization packages, statistical packages, scientific computing) compute largely autonomously.

Some interactive theorem provers (Isabelle, ACL2) make strong use of automation.

Correctness guarantees

Interactive theorem provers construct *formal axiomatic proofs* to support their claims. This provides a high degree of certainty.

Other mathematical software relies on careful programming and testing.

In some domains, users recognize and accept shortcomings:

- loose semantics of algebraic computation
- roundoff error in numeric computation

In other domains, we may be perfectly happy with experimental results.

Automation and Correctness

	unverified	verified
automated	automated theorem provers, numeric and symbolic computation	verified theorem provers, verified computation
interactive	computer algebra systems, "notebook" environments	interactive theorem proving

Polya

Polya provides automated methods to establish real-valued inequalities.

Desiderata:

- Automatically verify inequalities that come up in interactive theorem proving.
- Construct formal axiomatic proofs backing them up.

Inequalities

Mathematics is largely about measurement, and comparing quantities.

Show $\Gamma \vdash s \leq t$.

- pure mathematics: analysis, number theory, combinatorics, geometry, probability
- applied mathematics: engineering, statistics, data analysis, modeling and prediction, ...

One can use symbolic or numeric methods.

Linear inequalities

Consider entailments of the form

$$t_1 \geq 0, t_2 \geq 0, \ldots, t_n \geq 0 \Rightarrow s > 0$$

The entailment is valid iff the constraints

$$t_1 \geq 0, t_2 \geq 0, \ldots, t_n \geq 0, -s \geq 0$$

are infeasible.

Linear programming methods can handle hundreds of thousands of variables and constraints.

The methods are exact, and it is not hard to extract certificates / proofs.

Nonlinear inequalities

In the 1930's, Tarski proved the decidability of the first-order theory of the reals as an ordered ring.

- symbolic method
- proof-producing (in principle)

George Collins gave a *practical* method in 1975, "cylindrical algebraic decomposition."

It is implemented in Mathematica, Maple, Qepcad (now in Sage), Reduce (Redlog), RAHD.

The procedure has double-exponential complexity, and succeeds only on small problems

Numerical methods

There are a variety of numerical methods, including:

- Interval methods, and interval constraint propagation.
- Convex programming techniques.
- Optimization, stochastic methods.

Combination methods

Paulson's *MetiTarski*:

- uses a resolution theorem prover to guide search.
- replaces transcendental function by bounding rational functions.
- uses RCF back ends (Qepcad, Z3, Mathematica)

SMT solvers combine linear integer / real arithmetic methods, real-closed fields.

Parillo has used semidefinite programming methods to verify positivity of RCF expressions.

Verified numerical methods

Thomas Hales' proof of the Kepler conjecture required verifying hundred of inequalities. Most were of the form:

$$\forall \vec{x}\,, \vec{x} \in D \Rightarrow f(\vec{x}) > 0 \vee \ldots \vee f(\vec{x}) > 0$$

where D is a rectangular box.

Solovyev's 2012 thesis (under Hales' supervision) developed methods for handling these.

- Use interval methods based on Taylor series approximations.
- Also use interval methods to detect monotonicity along coordinates.

Polya

Polya is a heuristic theorem prover for real-valued inequalities.

- It is designed as automated support for interactive theorem proving.
- It is (will be) proof producing.
- It uses heuristic, symbolic methods.

An 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 are not useful. (Also, the domain is unbounded.)
- But, the inference is completely straightforward.

Backchaining

One idea is a apply rules like $u < v, w \leq x \Rightarrow u + w < v + x$ backwards.

But we can prove $a + b + c < d + e$ by proving

- $a < d, c \leq e$
- $a + c \leq e,\ b < d$
- $a + c + 5 \leq e,\ b - 5 < d,$
- ...

Another example

Here's an inequality that comes up in Shapiro's presentation of the Selberg proof of the prime number theorem.

Assuming

$$n \leq (K/2)x$$
$$0 < C$$
$$0 < \varepsilon < 1$$

we have

$$\left(1 + \frac{\varepsilon}{3(C+3)}\right) \cdot n < Kx$$

Decision procedures for real closed fields seem the wrong way to go.

Our method

We have developed an approach, that:

- verifies inequalities that other procedures don't
- extends beyond the language of RCF
- is amenable to producing proof terms
- captures natural patterns of inference

But:

- It is not complete.
- It not guaranteed to terminate.

It is designed to complement other procedures.

Overview

The main ideas:

- Use forward reasoning (guided by the structure of the problem).
- Show "hypotheses $\Rightarrow$ conclusion" by negating the conclusion and deriving a contradiction.
- As much as possible, put terms in canonical "normal forms," e.g. to recognize that $3(x + y)$ is a multiple of $2y + 2x$.
- Derive relationships between "terms of interest," including subterms of the original problem.
- Different modules contribute bits of information, based on their expertise.

Language

Our system verifies inequalities between real variables using:

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

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

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

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

Terms and normal forms

The inequality

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

is expressed canonically as

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

$$t_9 = t_6^2 t_8^{-1}$$

Terms and normal forms

The inequality

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

is expressed canonically as

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

Modules and the "blackboard"

Any comparison between canonical terms can be expressed as $t_i \bowtie 0$ or $t_i \bowtie c \cdot t_j$, where $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$.

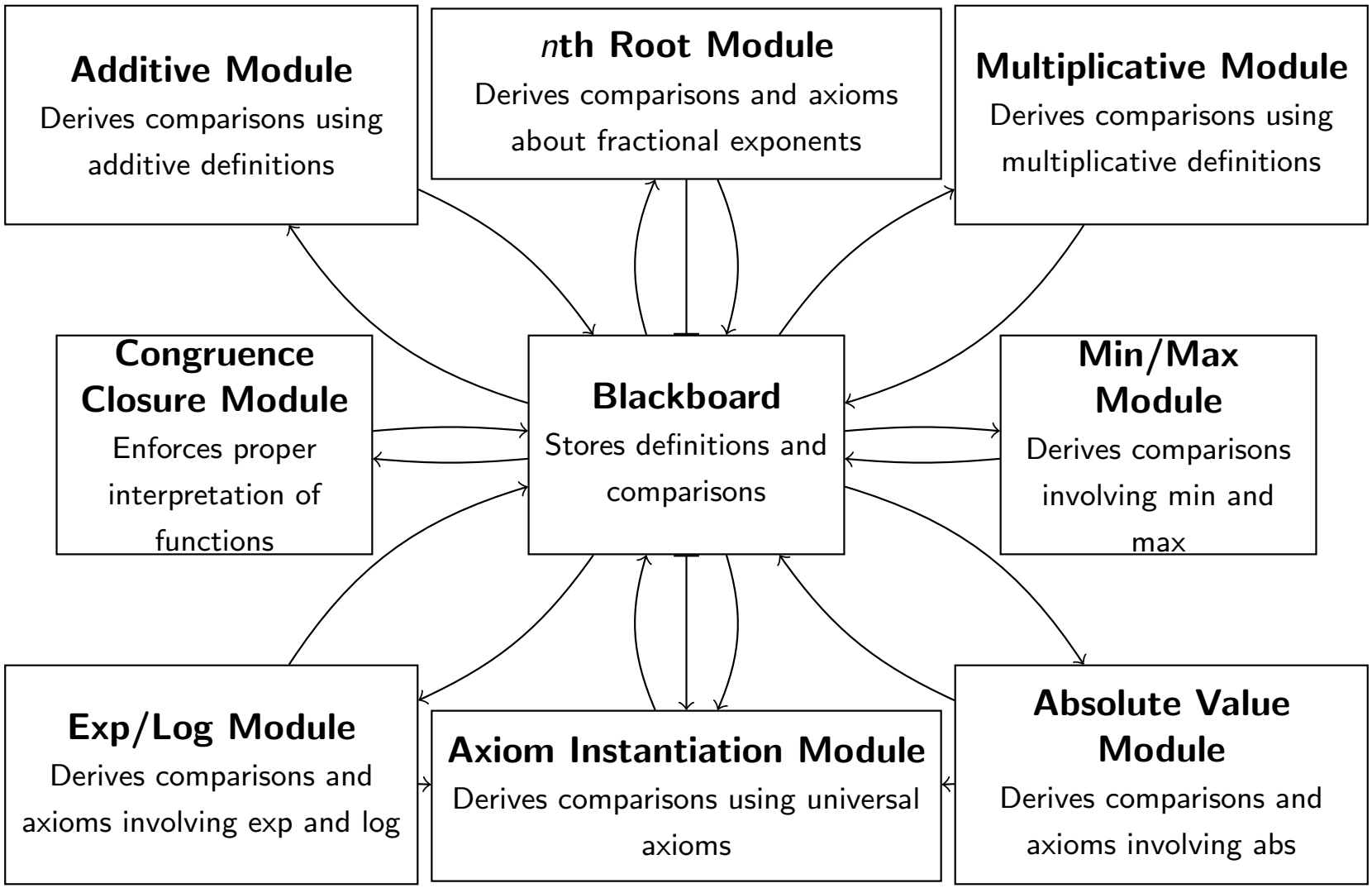
All modules are assumed to "understand" these relationships.

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

Modules for additive and multiplicative arithmetic work together to solve arithmetical problems.

They "saturate" the blackboard with the strongest derivable atomic comparisons.

We use two techniques for this:

- *Fourier-Motzkin* variable elimination
- a geometric projection method

The 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.

The Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

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

The Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

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

The Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

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

The Fourier-Motzkin additive module

To find comparisons between t_1 and t_2, eliminate t_3:

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

The Fourier-Motzkin additive module

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

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

The Fourier-Motzkin additive module

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

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

The 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.

The 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.

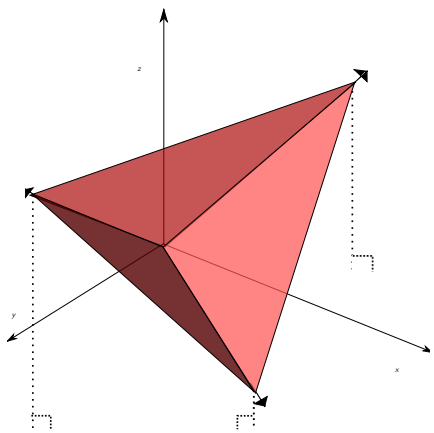
In practice, this approach works surprisingly well. But one can construct examples where the complexity leads to significant slowdown.

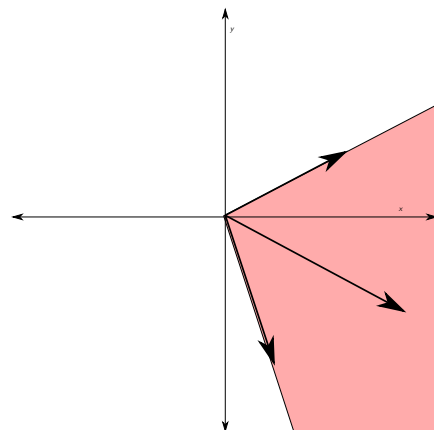
The Geometric additive module

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:

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

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.

For example: we can unify $f(v_1 + v_2)$:

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

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.

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:

$$(\forall x)(-1 \leq \mathit{sin}(x) \leq 1)$$

$$(\forall x)(-1 \leq \mathit{cos}(x) \leq 1)$$

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

$$\vdots$$

Successes

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

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

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

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

Successes

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

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

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

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

KeYmaera

André Platzer, at Carnegie Mellon, is developing a tool, KeYmaera, for verifying hybrid system.

The system generates problems like this one:

```
Hypothesis:  ru10**2 == (1/3)*x1u0**2
Hypothesis:  x1u0 <= 0
Hypothesis:  ru10 > 0
Hypothesis:  d1 == -1 * om * (h2 + -1*x2)
Hypothesis:  d2 == om * (h1 + -1*x1)
Hypothesis:  (h1 + -1*x1)**2 + (h2 + -1*x2)**2 ==
r**2
Hypothesis:  1 != ru10**-1 * ru10
Conclusion:  False
```

KeYmaera

On a collection of 4442 problems generated automatically by KeYmaera, we solve over 95% with a 10-second timeout.

- 10 minutes using geometric packages
- 18 using Fourier-Motzkin

Limitations

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

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

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

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

Plans

Increase scope:

- Handle trigonometric functions, etc.
- Handle integer constraints.
- Handle second-order constructions: sums, products, integrals.

Draw on other methods:

- Make use of numerical information.
- Interact with numerical procedures (dReal, interval constraint propagation).
- Make use of additional symbolic information (derivatives, gradients, convexity).
- Make use of algebraic procedures for real-closed fields.
- Make use of methods and systems for convex analysis.

Plans

Improve performance and search:

- Handle case splits and bactracking.
- Make the procedures incremental.

Make it proof-producing:

- Implement it in Lean.

References

For more information, see:

1. Avigad, Lewis, and Roux. A heuristic prover for real inequalities, *Interactive Theorem Proving* (Springer LNCS), 2014.
2. Our revised, expanded version of the same paper.
3. Lewis' 2014 MS thesis.
4. `https://github.com/avigad/polya`