Challenges and Opportunities for Automated Reasoning

John Harrison

Intel Corporation

10th October 2012 (15:50–16:35)

Summary of talk

- Motivation: the need for dependable proof
  - LCF-style theorem proving
  - Intel verification work
  - The Flyspeck project
- Combining tools and certifying results
  - Why is this important?
  - Focus on nonlinear arithmetic
- Beyond standard geometric decision procedures:
  - Without loss of generality
  - Decision procedures for vector spaces

0: Motivation

Motivation: dependable proof

We are interested in machine-checked and machine generated *formal proof*

- *Not* just a ‘yes’ or ‘no’ from a complex decision procedure
- A real step-by-step proof using basic rules of formal logic

Motivation: dependable proof

We are interested in machine-checked and machine generated *formal proof*

- *Not* just a ‘yes’ or ‘no’ from a complex decision procedure
- A real step-by-step proof using basic rules of formal logic

Why?

- High reliability
- Independent checkability

Motivation: dependable proof

We are interested in machine-checked and machine generated *formal proof*

- *Not* just a ‘yes’ or ‘no’ from a complex decision procedure
- A real step-by-step proof using basic rules of formal logic

Why?

- High reliability
- Independent checkability

How?

- LCF approach à la Milner

Motivation 1: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors

Motivation 1: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- Very rarely encountered, but was hit by a mathematician doing research in number theory.

Motivation 1: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- Very rarely encountered, but was hit by a mathematician doing research in number theory.
- Intel eventually set aside US $475 million to cover the costs.

Motivation 1: the FDIV bug

One of the most serious problems that Intel has ever encountered:

- Error in the floating-point division (FDIV) instruction on some early Intel®Pentium® processors
- Very rarely encountered, but was hit by a mathematician doing research in number theory.
- Intel eventually set aside US $475 million to cover the costs.

A very powerful motivation for performing rigorous proofs of numerical algorithms!

Motivation 2: the Kepler conjecture

- States that no arrangement of identical balls in ordinary 3-dimensional space has a higher packing density than the obvious ‘cannonball’ arrangement.

Motivation 2: the Kepler conjecture

- States that no arrangement of identical balls in ordinary 3-dimensional space has a higher packing density than the obvious ‘cannonball’ arrangement.
- Hales, working with Ferguson, arrived at a proof in 1998, consisting of 300 pages of mathematics plus 40,000 lines of supporting computer code: graph enumeration, nonlinear optimization and linear programming.

Motivation 2: the Kepler conjecture

- States that no arrangement of identical balls in ordinary 3-dimensional space has a higher packing density than the obvious ‘cannonball’ arrangement.
- Hales, working with Ferguson, arrived at a proof in 1998, consisting of 300 pages of mathematics plus 40,000 lines of supporting computer code: graph enumeration, nonlinear optimization and linear programming.
- Hales submitted his proof to *Annals of Mathematics* . . .

The response of the reviewers

After a full four years of deliberation, the reviewers returned:

> *"The news from the referees is bad, from my perspective. They have not been able to certify the correctness of the proof, and will not be able to certify it in the future, because they have run out of energy to devote to the problem. This is not what I had hoped for.*
> *Fejes Toth thinks that this situation will occur more and more often in mathematics. He says it is similar to the situation in experimental science — other scientists acting as referees can't certify the correctness of an experiment, they can only subject the paper to consistency checks. He thinks that the mathematical community will have to get used to this state of affairs."*

The birth of Flyspeck

- Hales's proof was eventually published, and no significant error has been found in it. Nevertheless, the verdict is disappointingly lacking in clarity and finality.

The birth of Flyspeck

- Hales's proof was eventually published, and no significant error has been found in it. Nevertheless, the verdict is disappointingly lacking in clarity and finality.
- As a result of this experience, the journal changed its editorial policy on computer proof so that it will no longer even try to check the correctness of computer code.

The birth of Flyspeck

- Hales's proof was eventually published, and no significant error has been found in it. Nevertheless, the verdict is disappointingly lacking in clarity and finality.
- As a result of this experience, the journal changed its editorial policy on computer proof so that it will no longer even try to check the correctness of computer code.
- Dissatisfied with this state of affairs, Hales initiated a project called *Flyspeck* to completely formalize the proof.

The birth of Flyspeck

- Hales's proof was eventually published, and no significant error has been found in it. Nevertheless, the verdict is disappointingly lacking in clarity and finality.
- As a result of this experience, the journal changed its editorial policy on computer proof so that it will no longer even try to check the correctness of computer code.
- Dissatisfied with this state of affairs, Hales initiated a project called *Flyspeck* to completely formalize the proof.
- "Flyspeck" = "Formal proof of the Kepler Conjecture"

1: Combining tools and certifying results

Combining tools and certifying results: Why?

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers

Combining tools and certifying results: Why?

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- The Kepler proof uses linear programming, nonlinear optimization, and other more ad hoc algorithms

Combining tools and certifying results: Why?

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- The Kepler proof uses linear programming, nonlinear optimization, and other more ad hoc algorithms
- Many powerful facilities in computer algebra systems that we'd like to exploit

Combining tools and certifying results: Why?

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- The Kepler proof uses linear programming, nonlinear optimization, and other more ad hoc algorithms
- Many powerful facilities in computer algebra systems that we'd like to exploit
- May want to combine work done in different theorem provers, e.g. ACL2, Coq, HOL, Isabelle.

Diversity at Intel

Intel is best known as a hardware company, and hardware is still the core of the company's business. However this entails much more:

- Microcode
- Firmware
- Protocols
- Software

Diversity at Intel

Intel is best known as a hardware company, and hardware is still the core of the company's business. However this entails much more:

- Microcode
- Firmware
- Protocols
- Software

If the Intel® Software and Services Group (SSG) were split off as a separate company, it would be in the top 10 software companies worldwide.

A diversity of verification problems

This gives rise to a corresponding diversity of verification problems, and of verification solutions.

- Propositional tautology/equivalence checking (FEV)
- Symbolic simulation
- Symbolic trajectory evaluation (STE)
- Temporal logic model checking
- Combined decision procedures (SMT)
- First order automated theorem proving
- Interactive theorem proving

Integrating all these is a challenge!

Flyspeck: a diversity of methods

The Flyspeck proof combines large amounts of pure mathematics, optimization programs and special-purpose programs:

- Standard mathematics including Euclidean geometry and measure theory
- More specialized theoretical results on *hypermaps*, *fans* and packing.
- Enumeration procedure for ‘tame’ graphs
- Many linear programming problems.
- Many nonlinear programming problems.

Certificates for linear arithmetic

- Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.

Certificates for linear arithmetic

- Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.
- The key is Farkas's Lemma, which implies that for any unsatisfiable set of inequalities, there's a linear combination of them that's 'obviously false' like $1 < 0$.

Certificates for linear arithmetic

- Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.
- The key is Farkas's Lemma, which implies that for any unsatisfiable set of inequalities, there's a linear combination of them that's 'obviously false' like $1 < 0$.
- Alexey Solovyev's highly optimized implementation of this is essential for Flyspeck.

Certificates for universal theory of reals (1)

- There is an analogous way of certifying nonlinear universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):

Certificates for universal theory of reals (1)

- There is an analogous way of certifying nonlinear universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):
- The polynomial equations $p_1(\overline{x}) = 0, \ldots, p_k(\overline{x}) = 0$ in a real closed closed field have *no* common solution iff there are polynomials $q_1(\overline{x}), \ldots, q_k(\overline{x}), s_1(\overline{x}), \ldots, s_m(\overline{x})$ such that

$$q_1(\overline{x}) \cdot p_1(\overline{x}) + \cdots + q_k(\overline{x}) \cdot p_k(\overline{x}) + s_1(\overline{x})^2 + \cdots + s_m(\overline{x})^2 = -1$$

Certificates for universal theory of reals (1)

- There is an analogous way of certifying nonlinear universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):
- The polynomial equations $p_1(\overline{x}) = 0, \ldots, p_k(\overline{x}) = 0$ in a real closed closed field have *no* common solution iff there are polynomials $q_1(\overline{x}), \ldots, q_k(\overline{x}), s_1(\overline{x}), \ldots, s_m(\overline{x})$ such that

$$q_1(\overline{x}) \cdot p_1(\overline{x}) + \cdots + q_k(\overline{x}) \cdot p_k(\overline{x}) + s_1(\overline{x})^2 + \cdots + s_m(\overline{x})^2 = -1$$

- The similar but more intricate Positivstellensatz generalizes this to inequalities of all kinds.

Certificates for universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example $23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$ or

$$\forall a\ b\ c\ x.\ ax^2 + bx + c = 0 \Rightarrow b^2 - 4ac \geq 0$$

because

$$b^2 - 4ac = (2ax + b)^2 - 4a(ax^2 + bx + c)$$

Certificates for universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example $23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$ or

$$\forall a\ b\ c\ x.\ ax^2 + bx + c = 0 \Rightarrow b^2 - 4ac \geq 0$$

because

$$b^2 - 4ac = (2ax + b)^2 - 4a(ax^2 + bx + c)$$

However, most standard nonlinear solvers do not return such certificates, and this approach does not obviously generalize to formulas with richer quantifier structure.

Results on floating-point verification

- Many floating-point algorithms need a proven bound on the difference between a mathematical function $f(x)$ and a polynomial $p(x)$.

Results on floating-point verification

- Many floating-point algorithms need a proven bound on the difference between a mathematical function $f(x)$ and a polynomial $p(x)$.
- Use an intermediate, very accurate, Taylor series $t(x)$ and $|f(x) - p(x)| \leq |f(x) - t(x)| + |t(x) - p(x)|$.

Results on floating-point verification

- Many floating-point algorithms need a proven bound on the difference between a mathematical function $f(x)$ and a polynomial $p(x)$.
- Use an intermediate, very accurate, Taylor series $t(x)$ and $|f(x) - p(x)| \leq |f(x) - t(x)| + |t(x) - p(x)|$.
- Core problem becomes bounding polynomial $t(x) - p(x)$ on an interval.

Results on floating-point verification

- Many floating-point algorithms need a proven bound on the difference between a mathematical function $f(x)$ and a polynomial $p(x)$.
- Use an intermediate, very accurate, Taylor series $t(x)$ and $|f(x) - p(x)| \leq |f(x) - t(x)| + |t(x) - p(x)|$.
- Core problem becomes bounding polynomial $t(x) - p(x)$ on an interval.
- SOS works very easily in this univariate case: can generate accurate certificates using a more direct method.

Results on Flyspeck

Some simple Flyspeck inequalities, after being expressed componentwise, can be proved efficiently by SOS certification, e.g. this one in HOL Light syntax:

```
!u v w:real^3.dist(u,v) >= &2 /\
     dist(u,w) >= &2 /\
     dist(v,w) >= &2 /\
     norm(u - v) < sqrt(&8)
     ==> norm(w - &1 / &2 % (u + v))
         > norm(u - v) / &2
```

Results on Flyspeck

Some simple Flyspeck inequalities, after being expressed componentwise, can be proved efficiently by SOS certification, e.g. this one in HOL Light syntax:

```
!u v w:real^3.dist(u,v) >= &2 /\
     dist(u,w) >= &2 /\
     dist(v,w) >= &2 /\
     norm(u - v) < sqrt(&8)
     ==> norm(w - &1 / &2 % (u + v))
         > norm(u - v) / &2
```

However, some of the more complex ones seem to be out of reach of current SOS implementations.

Alternative approaches

- Alternative algorithms for real quantifier elimination
  - CAD — efficient but apparently difficult to certify
  - Cohen/Hörmander — possible but apparently inefficient
  - Others? . . .

Alternative approaches

- Alternative algorithms for real quantifier elimination
  - CAD — efficient but apparently difficult to certify
  - Cohen/Hörmander — possible but apparently inefficient
  - Others? . . .
- Methods focused on restricted nonlinear optimzation
  - Bernstein polynomials (Zumkeller)
  - Interval arithmetic by proof (Solovyev)

Alternative approaches

- Alternative algorithms for real quantifier elimination
  - CAD — efficient but apparently difficult to certify
  - Cohen/Hörmander — possible but apparently inefficient
  - Others? . . .
- Methods focused on restricted nonlinear optimzation
  - Bernstein polynomials (Zumkeller)
  - Interval arithmetic by proof (Solovyev)

Solovyev's highly optimized implementation in HOL Light is already able to prove many difficult inequalities, but efficiency challenges remain.

2: Beyond standard geometric decision procedures

Beyond existing decision procedures

Many geometric problems can be solved efficiently using coordinate reduction and automated algorithms, e.g.

Beyond existing decision procedures

Many geometric problems can be solved efficiently using coordinate reduction and automated algorithms, e.g.

- Wu's algorithm or Gröbner bases for problems over algebraically closed fields.

Beyond existing decision procedures

Many geometric problems can be solved efficiently using coordinate reduction and automated algorithms, e.g.

- Wu's algorithm or Gröbner bases for problems over algebraically closed fields.
- Nonlinear real decision procedures for real-specific cases, e.g. involving inequalities.

Beyond existing decision procedures

Many geometric problems can be solved efficiently using coordinate reduction and automated algorithms, e.g.

- Wu's algorithm or Gröbner bases for problems over algebraically closed fields.
- Nonlinear real decision procedures for real-specific cases, e.g. involving inequalities.

However, these are not always efficient when applied in a straightforward manner, especially with the extra problem of generating a complete formal proof.

Without loss of generality

- Mathematical proofs sometimes state that a certain assumption can be made ‘without loss of generality’ (WLOG).

Without loss of generality

- Mathematical proofs sometimes state that a certain assumption can be made ‘without loss of generality’ (WLOG).
- Claims that proving the result in a more special case is nevertheless sufficient to justify the theorem in full generality.

Without loss of generality

- Mathematical proofs sometimes state that a certain assumption can be made ‘without loss of generality’ (WLOG).
- Claims that proving the result in a more special case is nevertheless sufficient to justify the theorem in full generality.
- Often justified by some sort of symmetry or invariance in the problem, particularly in geometry:
  - Choose a convenient origin based on invariance under translation
  - Choose convenient coordinate axes based on rotation invariance

HOL Light ‘WLOG’ tactics

- A series of HOL Light tactics that automatically allow the user to make such WLOG steps, generating a formal proof behind the scenes.

HOL Light ‘WLOG’ tactics

- A series of HOL Light tactics that automatically allow the user to make such WLOG steps, generating a formal proof behind the scenes.
- Proves automatically that a suitable transformation T exists

HOL Light ‘WLOG’ tactics

- A series of HOL Light tactics that automatically allow the user to make such WLOG steps, generating a formal proof behind the scenes.
- Proves automatically that a suitable transformation T exists
- Systematically rewrites quantifiers $\forall x.\ \phi[x]$ to $\forall x.\ \phi[T(x)]$, and likewise with other quantifiers, set abstractions etc.

HOL Light 'WLOG' tactics

- A series of HOL Light tactics that automatically allow the user to make such WLOG steps, generating a formal proof behind the scenes.
- Proves automatically that a suitable transformation T exists
- Systematically rewrites quantifiers $\forall x.\ \phi[x]$ to $\forall x.\ \phi[T(x)]$, and likewise with other quantifiers, set abstractions etc.
- Uses a stored list of 'invariance' theorems to automatically lift up and eliminate the transformation.

HOL Light 'WLOG' tactics

- A series of HOL Light tactics that automatically allow the user to make such WLOG steps, generating a formal proof behind the scenes.
- Proves automatically that a suitable transformation T exists
- Systematically rewrites quantifiers $\forall x.\ \phi[x]$ to $\forall x.\ \phi[T(x)]$, and likewise with other quantifiers, set abstractions etc.
- Uses a stored list of 'invariance' theorems to automatically lift up and eliminate the transformation.

Often allows the final coordinatewise proof to be much easier and more natural.

Avoiding coordinate reduction

- Performing a coordinate reduction is a general approach, but often unnatural and inefficient, even with a good choice of coordinates.

Avoiding coordinate reduction

- Performing a coordinate reduction is a general approach, but often unnatural and inefficient, even with a good choice of coordinates.
- Attractive to consider other algorithms (e.g. the area method, bracket algebra, . . .)

Avoiding coordinate reduction

- Performing a coordinate reduction is a general approach, but often unnatural and inefficient, even with a good choice of coordinates.
- Attractive to consider other algorithms (e.g. the area method, bracket algebra, . . .)
- In collaboration with Solovay and Arthan, we considered general decision procedures for various theories of vector spaces

Avoiding coordinate reduction

- Performing a coordinate reduction is a general approach, but often unnatural and inefficient, even with a good choice of coordinates.
- Attractive to consider other algorithms (e.g. the area method, bracket algebra, . . .)
- In collaboration with Solovay and Arthan, we considered general decision procedures for various theories of vector spaces
- Many interesting results, both positive and negative, and some practically useful outcomes.

Vector space axioms

$$\forall \mathbf{u}\ \mathbf{v}\ \mathbf{w}.\ \mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$$
$$\forall \mathbf{v}\ \mathbf{w}.\ \mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$$
$$\forall \mathbf{v}.\ \mathbf{0} + \mathbf{v} = \mathbf{v}$$
$$\forall \mathbf{v}.\ -\mathbf{v} + \mathbf{v} = \mathbf{0}$$
$$\forall a\ \mathbf{v}\ \mathbf{w}.\ a(\mathbf{v} + \mathbf{w}) = a\mathbf{v} + a\mathbf{w}$$
$$\forall a\ b\ \mathbf{v}.\ (a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$$
$$\forall \mathbf{v}.\ 1\mathbf{v} = \mathbf{v}$$
$$\forall a\ b\ \mathbf{v}.\ (ab)\mathbf{v} = a(b\mathbf{v})$$

The theory of real inner product spaces

The language of vector spaces plus an inner product operation $\mathcal{V} \times \mathcal{V} \to \mathcal{S}$ written $\langle -, - \rangle$ and satisfying:

$$\forall \mathbf{v}\ \mathbf{w}.\ \langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{w}, \mathbf{v} \rangle$$
$$\forall \mathbf{u}\ \mathbf{v}\ \mathbf{w}.\ \langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$$
$$\forall a\ \mathbf{v}, \mathbf{w}.\ \langle a\mathbf{v}, \mathbf{w} \rangle = a\langle \mathbf{v}, \mathbf{w} \rangle$$
$$\forall \mathbf{v}.\ \langle \mathbf{v}, \mathbf{v} \rangle \geq 0$$
$$\forall \mathbf{v}.\ \langle \mathbf{v}, \mathbf{v} \rangle = 0 \Leftrightarrow \mathbf{v} = \mathbf{0}$$

Decidability of inner product spaces

- (Solovay): theory of real inner product spaces is decidable, and admits quantifier elimination in a language expanded with inequalities on dimension.

Decidability of inner product spaces

- (Solovay): theory of real inner product spaces is decidable, and admits quantifier elimination in a language expanded with inequalities on dimension.
- Since inner product spaces are a conservative extension of vector spaces, the theory of vector spaces is also decidable

Decidability of inner product spaces

- (Solovay): theory of real inner product spaces is decidable, and admits quantifier elimination in a language expanded with inequalities on dimension.
- Since inner product spaces are a conservative extension of vector spaces, the theory of vector spaces is also decidable
- (Arthan) a formula with k vector variables holds in all inner product spaces iff it holds in each $\mathbb{R}^n$ for $0 \leq n \leq k$.

The theory of real normed spaces

The language of vector spaces plus a norm operation $\mathcal{V} \to \mathcal{S}$ written $\| - \|$ and satisfying:

$$\forall \mathbf{v}.\ \|\mathbf{v}\| = 0 \Rightarrow \mathbf{v} = \mathbf{0}$$
$$\forall a\ \mathbf{v}.\ \|a\mathbf{v}\| = |a|\|\mathbf{v}\|$$
$$\forall \mathbf{v}\ \mathbf{w}.\ \|\mathbf{v} + \mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|$$

Normed spaces: better or worse?

- (Solovay) The full theory of real normed spaces is strongly undecidable (same many-one degree as the true Π^2_1 sentences in third-order arithmetic).

Normed spaces: better or worse?

- (Solovay) The full theory of real normed spaces is strongly undecidable (same many-one degree as the true Π^2_1 sentences in third-order arithmetic).
- (Arthan) Even the purely *additive* theory of 2-dimensional normed spaces is strongly undecidable.

Normed spaces: better or worse?

- (Solovay) The full theory of real normed spaces is strongly undecidable (same many-one degree as the true Π^2_1 sentences in third-order arithmetic).
- (Arthan) Even the purely *additive* theory of 2-dimensional normed spaces is strongly undecidable.
- (Harrison) However the $\forall$ (purely universal) fragment of the theory is decidable. In the additive case, can be decided by a generalization of parametrized linear programming.

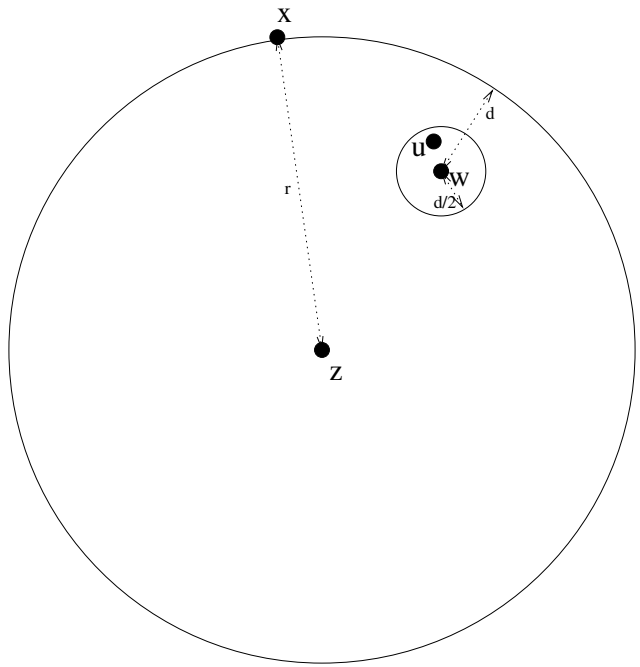
Normed spaces: better or worse?

- (Solovay) The full theory of real normed spaces is strongly undecidable (same many-one degree as the true Π^2_1 sentences in third-order arithmetic).
- (Arthan) Even the purely *additive* theory of 2-dimensional normed spaces is strongly undecidable.
- (Harrison) However the $\forall$ (purely universal) fragment of the theory is decidable. In the additive case, can be decided by a generalization of parametrized linear programming.
- (Arthan) This decidability result is quite sharp: both the $\forall\exists$ and $\exists\forall$ fragments, and even the $(\forall) \Rightarrow (\forall)$ fragments are undecidable.

Real application in formalizing complex analysis

An example where our linear normed space procedure is much more efficient than coordinate reduction:

```
|- abs(norm(w - z) - r) = d /\
   norm(u - w) < d / &2 /\
   norm(x - z) = r
   ==> d / &2 <= norm(x - u)
```



Conclusions

- Practical and efficient certification is an interesting problem for symbolic computation algorithms generally.

Conclusions

- Practical and efficient certification is an interesting problem for symbolic computation algorithms generally.
- A useful tool in soundly integrating different proof tools, which has value in verification and in mathematics

Conclusions

- Practical and efficient certification is an interesting problem for symbolic computation algorithms generally.
- A useful tool in soundly integrating different proof tools, which has value in verification and in mathematics
- Nonlinear arithmetic is a particularly challenging example for such certification, and has many potential applications.

Conclusions

- Practical and efficient certification is an interesting problem for symbolic computation algorithms generally.
- A useful tool in soundly integrating different proof tools, which has value in verification and in mathematics
- Nonlinear arithmetic is a particularly challenging example for such certification, and has many potential applications.
- There are strong motivations for looking for higher-level (more efficient or conceptual) approaches to such problems.