Beyond Linear Arithmetic: Automatic Proof Procedures for the Reals

Prof. Lawrence C. Paulson
Computer Laboratory, University of Cambridge

1 Previous Research and Track Record

Lawrence C. Paulson is Professor of Computational Logic at the University of Cambridge, where he has held established posts since 1983. Paulson's early work on LCF contributed much (both code and concepts) to HOL. Paulson introduced Isabelle in 1986 and has been building upon it ever since. Isabelle is a generic proof tool that supports higher-order logic (HOL), Zermelo-Fraenkel set theory (ZF) and other formalisms. Isabelle is widely used in research. Many developments are due to Prof. Tobias Nipkow's group at the Technical University of Munich. Automatic proof search, one of Isabelle's particular strengths, is however due to Paulson [10, 11].

The present proposal involves developing automatic proof procedures for the theory of the reals, including transcendental functions. It builds upon Paulson's existing experience of developing heuristic proof procedures and building proof tools, as well as his extensive knowledge of the automated reasoning literature.

The work will be done within the Cambridge Automated Reasoning Group. Hardware verification was pioneered here by Prof. M. J. C. Gordon and his students. They introduced what have become standard techniques, such as the use of higher-order logic to model hardware and software systems. The group's work continues to attract worldwide attention. Former members such as Dr. John Harrison have taken formal verification to Intel and other companies. The group has built two of the most important proof environments used today, namely HOL and Isabelle. Institutes using Isabelle as a basis for their research include the University of Edinburgh, Carnegie-Mellon University and Australia's Defence Science and Technology Organisation (DSTO).

Several past projects at Cambridge involve Isabelle. Recent projects include the following:

- *Authentication Logics: New Theory and Implementations* (EPSRC ref. GR/K77051), 1996–99. This project was concerned with proving the correctness of security protocols. It produced the *inductive method* for protocol verification, whereby detailed protocol models are proved to be correct or shown to be flawed. The results of this project have been widely cited and have been incorporated into many of today's protocol verification tools.

- *Verifying Electronic Commerce Protocols* (EPSRC ref. GR/R 01156/01), 2000–03. This continued the previous project, with the objective of verifying security protocols of industrial complexity. The huge SET protocol suite was analysed and some vulnerabilities found. The research assistant, Giampaolo Bella, investigated the Zhou-Gollmann non-repudiation protocol and a certified electronic mail protocol designed by Abadi et al. This project produced numerous publications.

- *Compositional Proofs of Concurrent Programs* (EPSRC ref. GR/M75440), 2000–03. This project investigated the verification of reactive systems using UNITY and the guarantees-calculus of Sanders and Chandy. It made much progress on the problem of formalizing states; toward this objective, it produced an untyped UNITY environment, formalized within Isabelle/ZF. A number of journal and conference papers have appeared.

- *Automation for Interactive Proof* (EPSRC ref. GR/S57198/01), 2004–07. This project aims to give interactive proof tools improved automation through an effective combination of interactive and automatic tools. It is developing techniques for transferring subgoals from an interactive prover to an automatic one and for transferring proofs in the opposite direction. The returned proofs are reconstructed (and thus verified) within the interactive prover. This project has been underway for less than a year, but a prototype incorporating these elements is already completed, linking Isabelle to the resolution provers Vampire and SPASS.

The designated Research Assistant, Jia Meng, is currently working on *Automation for Interactive Proof* as a project student. She therefore has experience with Isabelle and with resolution provers. She will have finished her PhD by the time this project starts. She is capable, hard-working and enthusiastic, and already has two research publications.

2 Description of Proposed Research

A primary goal of formal verification research is improved automation. Many verification tasks involve proving equations and inequations over the reals. For example, Denney et al. [4] describe an application of automated theorem proving to aerospace software; they intend that all proof obligations should be eliminated by simplification or other means, leaving none to be presented to the user. An example involving interactive proof is the formalization in Isabelle/HOL of the Prime Number Theorem [1], where the authors spent much time proving simple facts involving transcendental functions.

Linear arithmetic is well understood, and many proof tools provide efficient decision procedures, at least for the quantifier-free case. Such procedures can automatically prove propositional combinations of inequalities between linear combinations of variables. Unfortunately, linear arithmetic restricts us to the language of $=$, $<$, $\leq$, $+$ and multiplication by integer constants. How can we extend this language?

The alternatives are to employ more general decision procedures or to seek heuristic methods. More general decision procedures do exist, because the theory of real closed fields is decidable: thus, our language can be extended with multiplication and quantifiers. Dolzmann et al. [5] survey the main algorithms, which work by quantifier elimination. Heuristic methods tend to be extensions of elementary variable elimination methods. In the 1980s, Boyer and Moore [3] showed how such a procedure could share information with other parts of a theorem prover. Hunt et al. [8] describe a more elaborate procedure that can handle some problems involving polynomials (and thus, products of variables). Tiwari [14] describes a different heuristic procedure for polynomials, using ideas from Gröbner basis methods.

Decision procedures interest mathematicians, since they represent solutions to decision problems, while heuristic methods may seem to be ad-hoc. For verification purposes, heuristic methods are typically broader in scope and more efficient than decision procedures. Tiwari [14] shows that for many problems, his heuristic procedure easily outperforms QEPCAD [7], a decision procedure that performs partial cylindrical algebraic decomposition (and whose complexity is doubly exponential). Richardson's theorem tells us not to expect a useful decision procedure for any theory that includes the transcendental functions [16]. Weispfenning [15] has published a decision procedure for linear arithmetic augmented with a single, well-behaved transcendental function; although the work may contribute useful insights, this type of highly specialized procedure does not address our objectives. Heuristic methods are therefore appropriate.

This project will focus on heuristic procedures to prove arithmetic statements that may refer to transcendental functions and polynomials. The statements must be quantifier-free, with free variables being implicitly universally quantified. The project will include two other components:

- A *disproof* procedure will identify invalid arithmetic statements so that further proof attempts can be abandoned. A successful disproof procedure for functional programs already exists for Isabelle [2].

- A *problem library* for real arithmetic will assist the research community by allowing comparisons between different implementations and procedures. These problems will be published via TPTP (Thousands of Problems for Theorem Provers) [13], if possible, and otherwise on a dedicated website.[1]

The *Project Objectives* are pragmatic, and focus equally on interactive and automatic verification.

1. To develop heuristic automatic proof procedures for the theory of real arithmetic with transcendental functions.

2. To develop disproof procedures for this theory.

3. To publish a problem library suitable for evaluating such procedures.

The aim is to develop procedures that are sound and effective against real-world examples.

Background
Most heuristic proof procedures for arithmetic [3, 8, 14] extend the Fourier-Motzkin decision procedure. They use proof by contradiction: the conjecture is negated and converted to disjunctive normal form (DNF). The original conjecture is true if and only if all disjuncts are unsatisfiable. Each disjunct is a conjunction of inequalities, so the proof procedures work by refuting such conjunctions. Disproving the conjecture, conversely, requires showing that one of the conjunctions is satisfiable.

The Fourier-Motzkin procedure works by repeated variable elimination. For example, given $2x < 3y - 1 \land 2y - x < 0 \land y \geq -1$, we obtain $4y < 2x$ from the second conjunct and thus $4y < 3y - 1$ from the first conjunct by transitivity. Thus $y < -1$, and from the third conjunct we obtain $-1 < -1$, contradiction.

Hunt et al. [8] and Tiwari [14] both extend this procedure to polynomials by including mechanisms to eliminate nonlinear leading terms, while the earlier procedure of Boyer and Moore [3] is able to eliminate certain function applications by using stored facts about those functions. Looking further afield, Ratschan is undertaking a programme of research [12] that may be relevant to both proof and disproof procedures, though set in the context of constraint programming.

PROOF PROCEDURE OUTLINE
The proposed extension of Fourier-Motzkin starts with the observation that most of the well-known functions on the reals are continuous. Moreover, a typical function's domain can be segmented such that the function is monotonic or anti-monotonic on each interval. Functions such as $\sqrt{x}$, e^x, $\log(x)$, and $K \cdot x$ (product

[1]Geoff Sutcliffe has already indicated his willingness to include such problems in TPTP.

by a variable known to be positive) are monotonic over their entire domain, while other functions are monotonic on intervals: $\sin(x)$ is monotonic on $[-\pi/2, \pi/2]$. If f is a monotonic and continuous segment of a function, then f^{-1} will also be monotonic and continuous with a suitable domain and range. The existence of inverses will assist in eliminating occurrences of f.

The heuristic procedure has several ways to eliminate occurrences of functions. It can appeal to monotonicity, for example reducing $f(A) < f(B)$ to $A < B$, or reducing $f(A) < B$ to $A < f^{-1}(B)$; such steps require checking, through a recursive call to the procedure, that A and B lie within the same monotonic segment of f. If there is no other way to proceed, function occurrences can be eliminated by appealing to linear upper or lower bounds. For typical functions, these can be obtained from Taylor expansions around a suitable point. Examples of such bounds include $e^x \geq x + 1$; $\log(x) \leq x - 1$ for $x \geq 1$; $x^2 \geq 0$; $x^2 \geq x$ for $x \geq 1$; $\sin(x) \leq x$ for $x \geq 0$; $1/x < 2 - x$ if $|x - 1| < 1$. The procedure would probably not compute bounds during its execution, but would be equipped with suitable information prior to execution. Users would be able to augment the procedure with new information about functions used in their application.

For example, the proof of $u \leq v \land x \leq y \to e^u + e^x \leq e^v + e^y$ would involve refuting the set of inequalities $u \leq v$, $x \leq y$, $e^v + e^y < e^u + e^x$. To isolate y, the use of monotonicity and inverses would yield $y < \log(e^u + e^x - e^v)$, concluding $x < \log(e^u + e^x - e^v)$ by transitivity. To isolate x, the pattern $f^{-1}(\cdots f(x) \cdots)$ suggests applying f to both sides, here yielding $e^x < e^u + e^x - e^v$ and therefore $e^v < e^u$. By monotonicity we obtain $v < u$, which contradicts $u \leq v$ and we are done.

Clearly a practical procedure will require many further elements. Few functions are as well-behaved as e^x. We need to take into account the division of a function into monotonic segments, on intervals that may be open, closed or unbounded at either end. Moreover, many examples seem to require additional knowledge such as $e^0 = 1$ and $\sin(0) = 0$.

Disproof Procedure Outline

The disproof procedure will accept the same input language as the proof procedure, and it will also begin by transforming the negated conjecture into DNF. The obvious approach to disproving a conjecture is simply to generate random variable assignments. This can be done separately on each conjunctive part of the DNF: satisfying one of them yields a counterexample to the conjecture.

Floating-point arithmetic, however, could give incorrect results due to rounding errors. False positives (incorrect “counterexamples”) are unacceptable: they could lead to the abandonment of a valid goal. False negatives (overlooked counterexamples) are less serious, but they are still failures. If the conjecture is an equality, say $f(x) = g(x)$, then a floating-point calculation from a random value x could erroneously “disprove” the equality: a false positive. We should never accept a claimed counterexample unless all conjuncts evaluate to **true** using an exact real arithmetic package such as iRRAM [9].

The counterexample search does not have to be entirely random. We can separate the set of conjuncts into a linear part and a non-linear part. The Fourier-Motzkin procedure can be used to generate variable assignments satisfying the linear part. These can then be tested against the non-linear part.

Programme and Methodology
Task 1: Gathering of test data. Several sources of sample problems are available, including Isabelle's formal development of the reals and Avigad's formalization of the Prime Number Theorem. Avigad has offered to contribute such problems; see the attached letter of support. Other examples could be obtained from published PVS theories and by making enquiries over various mailing lists. We shall organize them, put them into a uniform format and publish them, probably via the TPTP. *Estimated time*: 2 person-months.

Task 2: Preliminary development of heuristic proof procedures. The outline has been given above, but many details need to be fleshed out, driven by experimentation with our corpus of test data. *Estimated time*: 12 person-months.

Task 3: Evaluation of exact real arithmetic packages. There exist many packages for exact real arithmetic. Criteria for preferring one to another include reliability, support for transcendental functions, efficiency and likelihood of continued development. We may decide to develop a communication mechanism so that the disproof procedure can be developed independently from any particular arithmetic package. *Estimated time*: 2 person-months.

Task 4: Preliminary development of disproof procedures. This task may appear to be straightforward, but many questions require investigation. For example, should we use floating-point arithmetic in the initial search for counterexamples, and only use exact real arithmetic to confirm a claimed counterexample? Such an approach avoids false positives, and the use of floating-point arithmetic makes it more efficient, but it leaves open the risk of false negatives. *Estimated time*: 9 person-months.

Task 5: Integration with interactive proof tools. One objective of this project is to support interactive proofs in arithmetic. We intend to work with Prof. Tobias Nipkow (of the Technical University of Munich) to extend his arithmetic proof procedure to use the methods developed in the project. The main difficulty concerns reconstructing the proofs in the host prover's formal system, but the heuristic proof methods will be designed to make proof reconstruction as simple as possible. We shall also co-operate with interested developers of other proof tools; in particular, many developers of the HOL system are based at Cambridge. Integration of the disproof procedures with other proof tools is simpler, because there are no proofs to be reconstructed.

Estimated time: 9 person-months.

Task 6: Testing and refinement. The proof and disproof procedures will undergo continual testing with new problems, and will be refined accordingly. We also intend to evaluate them in their role of supporting interactive proof.

Estimated time: 9 person-months.

The total effort of 43 person-months allows 36 for the research assistant and 7 for the principal investigator (who will dedicate 20% of his time to the project).

Relevance to Beneficiaries

The project is designed to benefit a broad range of parties, both academic and industrial.

- Developers of proof tools, both interactive and automatic, will be able to provide increased automation to their users.
- The many research groups using arithmetic reasoning (for example, to verify hybrid systems) will benefit from increased automation.
- Formal arithmetic reasoning will increasingly benefit mathematicians. Already the Flyspeck Project [6] exists to produce a machine-assisted formal proof of the Kepler Conjecture, because the existing proof is too complicated for human referees.

Dissemination and Exploitation

The source code arising from this project will be freely distributed via the Internet. The techniques and the experimental results will be presented in journal and conference papers.

Justification of Resources

Staff. Paulson will work part-time on the project, specifically to support tasks 2, 4 and 5. He proposes to supervise one full-time research assistant, Jia Meng. She will undertake all of the project tasks, with day-to-day guidance from Prof. Paulson. We are requesting the top scale point because Meng is already receiving job offers from overseas: we need to be able to offer a competitive salary. The project also requires a Computer Officer (10% time) to maintain the hardware and software used in the research.

Travel and Subsistence. Conference attendance is essential to keep abreast of developments and to disseminate results. We are requesting funds to attend some of the main conferences, such as CADE, CAV, IJCAR and TPHOLs. We may wish to attend relevant workshops at Schloß Dagstuhl and elsewhere. We are also requesting funds for visits to other institutions such as TUM, as detailed on the application form.

Consumables. The figure shown comprises UNIX-based computers for Paulson and Meng, one laptop (to be shared), and various other computing-related costs.

References

[1] J. Avigad. Notes on a formalization of the prime number theorem. On the Internet at `http://www.andrew.cmu.edu/user/avigad/isabelle/pntnotes_a4.pdf`, Sept. 2004.

[2] S. Berghofer and T. Nipkow. Random testing in Isabelle/HOL. In *Software Engineering and Formal Methods (SEFM'04)*, pages 230–239. IEEE Computer Society, 2004.

[3] R. S. Boyer and J. S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, pages 83–124. Oxford University Press, 1988.

[4] E. Denney, B. Fischer, and J. Schumann. Using automated theorem provers to certify auto-generated aerospace software. In D. Basin and M. Rusinowitch, editors, *Automated Reasoning — Second International Joint Conference, IJCAR 2004*, LNAI 3097, pages 198–212. Springer, 2004.

[5] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. Technical Report MIP-9720, Universität Passau, D-94030, Germany, 1997.

[6] T. C. Hales. The flyspeck project fact sheet. On the Internet at `http://www.math.pitt.edu/~thales/flyspeck/`.

[7] H. Hong. QEPCAD — quantifier elimination by partial cylindrical algebraic decomposition. On the Internet at `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`. Web site includes sources and documentation.

[8] W. A. Hunt, Jr., R. B. Krug, and J. Moore. Linear and nonlinear arithmetic in ACL2. In D. Geist and E. Tronci, editors, *Correct Hardware Design and Verification Methods (CHARME)*, LNCS 2860, pages 319–333, 2003.

[9] N. Mueller et al. iRRAM — exact arithmetic in C++. Sources and documentation available at `http://www.informatik.uni-trier.de/iRRAM/`, 2004.

[10] L. C. Paulson. Generic automatic proof tools. In R. Veroff, editor, *Automated Reasoning and its Applications: Essays in Honor of Larry Wos*, chapter 3. MIT Press, 1997.

[11] L. C. Paulson. A generic tableau prover and its integration with Isabelle. *Journal of Universal Computer Science*, 5(3):73–87, 1999.

[12] S. Ratschan. Convergent approximate solving of first-order constraints by approximate quantifiers. *ACM Trans. Comput. Logic*, 5(2):264–281, 2004.

[13] G. Sutcliffe and C. Suttner. The TPTP problem library for automated theorem proving. On the Internet at `http://www.cs.miami.edu/~tptp/`, 2004.

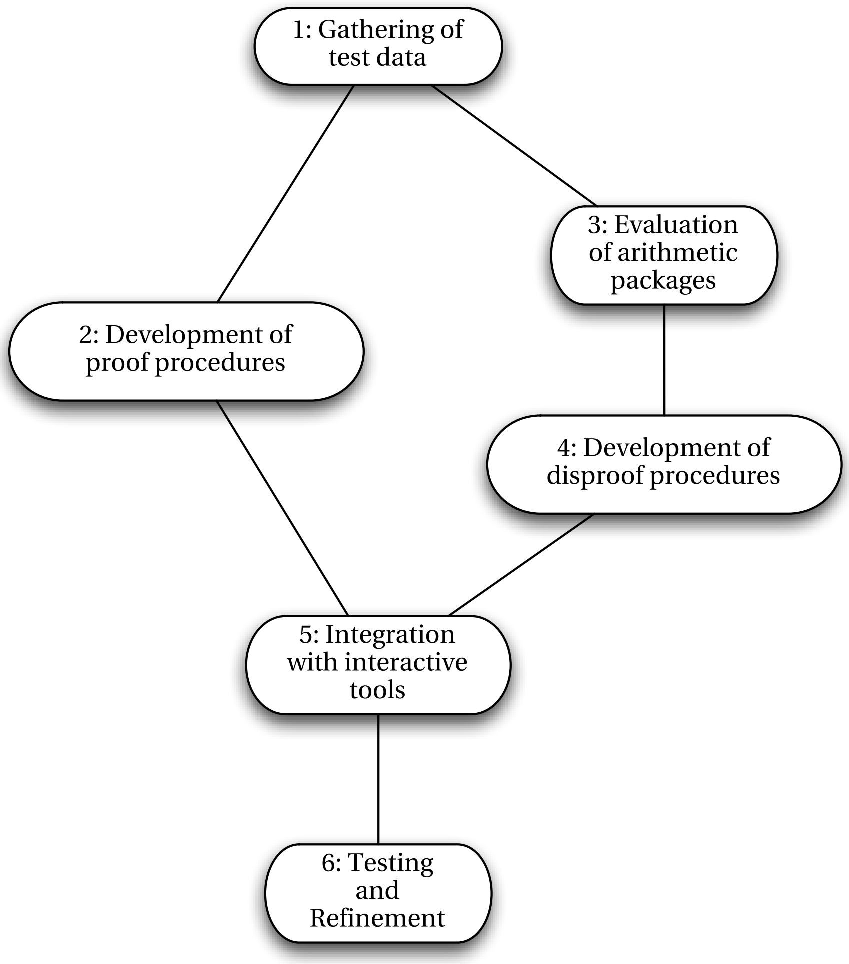
[14] A. Tiwari. Abstraction based theorem proving: An example from the theory of reals. In C. Tinelli and S. Ranise, editors, *PDPAR: Workshop on Pragmatics of Decision Procedures in Automated Deduction*, pages 40–52. INRIA, Nancy, 2003.

[15] V. Weispfenning. Deciding linear-transcendental problems. Technical Report MIP-0005, Universität Passau, D-94030 Passau, Germany, 2000.

[16] E. W. Weisstein. Richardson's theorem. From *MathWorld*—A Wolfram Web Resource, 1999. On the Internet at `http://mathworld.wolfram.com/RichardsonsTheorem.html`.

Project Plan

The chart below indicates the dependencies among the tasks. Gathering of test data will continue throughout the project.


1: Gathering of test data
3: Evaluation of arithmetic packages
2: Development of proof procedures
4: Development of disproof procedures
5: Integration with interactive tools
6: Testing and Refinement