Automation for Interactive Proof

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. This proposal involves adding first-order automation to the interactive theorem provers Isabelle and HOL. Paulson's early work on LCF contributed much (both code and concepts) to HOL. Paulson introduced Isabelle in 1986, under the Alvey programme, and has been improving it ever since. Isabelle is a generic system, with support for 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 [7, 8]. The present proposal has many of the same technical issues in common with this previous research.

The designated Research Assistant, Dr. Joe Hurd, did a small-scale version [4] of this project as an exercise in the first year of his PhD. study. His experience is directly relevant. During his years as a PhD. student, he proved to be capable, imaginative and hard-working. Earlier he earned a First in the Cambridge Mathematics Tripos and Distinctions in Part III Mathematics and the Diploma in Computer Science.

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.

Several past projects at Cambridge involve Isabelle:

- *Combining HOL and Isabelle* (SERC ref. GR/H40570), 1992-95. This project applied Isabelle to HOL-style problems, the main application being proof support for Lamport's TLA (Temporal Logic of Actions).
- *Verifying ML Programs using Evaluation Logic* (SERC ref. GR/G53279), 1991–95. This project has clarified some of the subtle interactions that occur when references to a store interact with higher-order functions.

- *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, which can be used to analyze a great variety of protocols in depth. The results of this project have been highly influential.

- *Mechanising Temporal Reasoning* (EPSRC ref. GR/K57381), 1995-99. This project investigated the verification of reactive systems using logics such as TLA and UNITY. An Isabelle/HOL proof environment for UNITY was developed and distributed.

- *Compositional Proofs of Concurrent Programs* (EPSRC ref. GR/M75440), 2000-03. This project continues *Mechanising Temporal Reasoning*. It concentrates on UNITY and the guarantees-calculus of Sanders and Chandy. The research assistant, Sidi Ehmety, started in September 2000. Much effort has gone into the problem of formalizing states; towards this objective, an untyped UNITY environment has been developed within Isabelle/ZF. A number of journal and conference papers have appeared.

- *Verifying Electronic Commerce Protocols* (EPSRC ref. GR/R 01156/01), 2000-03. This project continues *Authentication Logics*. The objectives have largely been achieved already: most of the huge SET protocol has been analyzed, and some vulnerabilities found. The research assistant, Giampaolo Bella, has investigated the Zhou-Gollmann non-repudiation protocol and is currently examining a protocol for secure electronic mail. A visitor to Cambridge, Frédéric Blanqui, has recently analyzed some novel protocols for agent-based shopping.

2 Description of Proposed Research

Interactive proof tools support rich, expressive formalisms. The user can express a complex model of, say, a CPU design, and develop its formal theory consisting of hundreds of theorems. The user constructs proofs by applying *tactics*: commands based upon sound inference rules. Unfortunately, interactive proof construction is extremely tedious. Verifying a significant system requires months of an expert's time, making the cost prohibitive. At the opposite extreme, resolution systems are automatic and powerful. However, they are restricted to first-order logic with equality. Can we combine the advantages of these two technologies?

Today's interactive tools are much better than those of 20 years ago, but the improvements are incremental and largely stem from better hardware. Many of today's decision procedures and rewriting heuristics were in use 20 years ago. Predicate calculus theorem proving techniques are the main novelty, having only recently been incorporated into interactive tools. They are highly successful: Isabelle's `blast` [8] and `fast` [7] methods can prove complicated theorems, while HOL users accomplish much using `MESON_TAC`. This proposal aims to go much further, combining interactive proof tools with a leading resolution theorem prover.

Twenty years ago, when many users had to share a single computer, a proof command could realistically take at most a few seconds of processor time. Now that 2GHz processors are commonplace, we should abandon the traditional mode of interaction, where the proof tool does nothing until the user types a command. Background processes (perhaps on several computers) should try to prove the outstanding subgoals. Better automation could make formal verification affordable.

The theoretical aspect of this proposal concerns the relationships between logical formalisms. For example, translating a first-order resolution proof into a higher-order logic natural deduction proof is difficult. Resolution relies on Skolemization. While Skolemization does not presuppose the axiom of choice (AC), all known ways of importing resolution proofs into an interactive proof tool require the latter to assume a strong form of AC, so-called global choice. Some recent formal theories are inconsistent with AC, which suggests that we seek a different approach to proof translation.

The *Project Objectives*, therefore, are both practical and theoretical:

1. to give interactive proof tools greatly improved automation
2. to develop the concept of an interactive proof tool using background processing
3. to explore the formal relationships between first- and higher-order logic, from the perspective of mechanical theorem proving

Background

Vampire [9] is one of the world's leading resolution theorem provers. In each of the past few years, Vampire came first in some category of the System Competition held at the Conference on Automated Deduction (CADE). Prof. Voronkov has

suggested integrating his system with Isabelle, and he is eager to co-operate with this project (see attached letter of support). Voronkov has received EPSRC funding for the development of Vampire, with integration as an objective.

There have been several previous attempts at integrating interactive and automatic theorem provers. Ahrendt et al. [1] combined KIV with the tableau prover ${}_3T^AP$. Their results were disappointing, but ${}_3T^AP$ was an ill-advised choice, having never ranked in the first category of automatic provers.

John Harrison implemented a model elimination prover and integrated it with a version of the (interactive) HOL system [3]. This was a success: `MESON_TAC` is among the most powerful of HOL tools. However, the user must collect the relevant lemmas manually, and support for equality is limited.

Joe Hurd [4] integrated the HOL system with the prover Gandalf. His system translated the Gandalf proofs into HOL proofs to ensure soundness. The results were disappointing: `MESON_TAC` was usually faster. However, this was a brief experiment undertaken by a first-year research student. More recently, Hurd has made simple implementations of some first-order calculi, comparing their performance on subgoals arising in interactive proof. The main result so far is the production of a generic interface for converting higher-order logic goals to first-order clauses, and for translating first-order refutations back to higher-order logic proofs [5].

Isabelle's `blast` is a successful instance of integration. A major difference from the present proposal is that `blast`'s prover is designed specifically to simplify the integration, including the translation into Isabelle proofs. That prover uses simplistic tableau methods. It cannot tackle problems that are trivial for resolution provers, and its treatment of equality is rudimentary. Resolution theorem provers are powerful but less easy to modify. We must work harder at the integration.

Programme and Methodology

Most of the tasks are similar to those done during the implementation of `blast`.

Task 1: First experiments (no equality). The objective is to make a standard resolution prover undertake proofs previously performed interactively. The procedure is to take existing Isabelle proofs, including those using the `blast` tactic, and attempt to reproduce them using Vampire. This involves dumping Isabelle's current context of default classical rules and translating them into first-order clauses; we must omit the many rules for basic logical reasoning, which Vampire provides already. The initial experiments will be basic: some of the translations may even be done by hand and Vampire input files prepared manually.

Task 2: First experiments (equality). The objective is to exploit a resolution prover's treatment of equality. The procedure is to identify Isabelle proofs that require a combination of equality and classical reasoning, attempting to reproduce them using Vampire.

Interactive tools such as Isabelle and HOL use separate tactics for simplification and predicate calculus reasoning. Therefore, even an elementary fact involv-

ing equality may require a lengthy proof script. A resolution prover may find such proofs automatically, since these provers integrate equality with their other inference methods.

The outcome of these two tasks will include test suites derived from actual Isabelle proofs.

Task 3: Basic functionality (untyped). The objective is to implement a tactic that transfers a subgoal and its context from a proof assistant to a resolution prover and reports the outcome. The procedure is to write code to automate the steps performed manually in the previous two tasks.

We shall use Isabelle/ZF, which provides untyped set theory. Many difficult proofs are available for experimentation, and the lack of types is an important simplification. The "context" mentioned above will include a list of potentially useful lemmas. Some lemmas could be chosen by the user, but the great majority would be chosen automatically — using the existing labelling in Isabelle libraries — as being good for simplification or classical reasoning.

Network programming will allow Isabelle to communicate with a Vampire process, possibly running on another machine. Hurd's HOL-Gandalf integration uses PROSPER [2], a toolkit for allowing various proof tools to communicate. We shall build upon the lessons learned from PROSPER.

Task 4: Basic functionality (typed). This is the same as the previous task, with the added complication of types. The objective is to obtain sound reasoning in a simply typed formalism using an untyped resolution prover.

We shall use Isabelle/HOL, which provides higher-order logic. Overloading and polymorphism are complications. For example, the equality symbol ($=$) is polymorphic: that is, any two terms of the same type can be compared for equality. But equality between sets is special: we can prove $A = B$ by proving $A \subseteq B$ and $B \subseteq A$. Equality between booleans is also special: we can prove $A = B$ by proving $A \to B$ and $B \to A$. Unless the resolution prover is given information about Isabelle's type system, it will have no way of knowing which inference rules are applicable.

This problem arose in the development of `blast`, and a similar solution will be tried. Overloaded constants will carry their type as an additional argument. The target prover will not need to have a type system: Isabelle types will be represented as terms. First-order unification will propagate type constraints.

Task 5: Full modelling of Isabelle types. The objective is to extend the tactic developed in the previous task to handle order-sorted polymorphism. The procedure is to encode the relevant concepts, such as type classes, in first-order logic.

Order-sorted polymorphism originated with the Haskell programming language and is also available in Isabelle. For example, the theorem `order_refl` asserts $x \leq x$. At first sight, this theorem is polymorphic, holding regardless of the type of x. But, more precisely, it holds only if x's type belongs to the *type*

class of partial orders [10]. To prevent unsound resolution proofs, we must model type classes in Vampire. If we represent types as constants, then membership in type classes can be expressed by atomic facts or by implications. The Isabelle declarations that enter types into classes must be communicated to Vampire.

Task 6: New reductions to first-order logic. The objective is to increase the scope of resolution by introducing translations from higher-order logic into first-order logic. Most proof tools allow variable binding within terms, but resolution provers do not. Even Isabelle/ZF, which is first-order set theory, admits terms such as $\bigcup_{x\in A} B(x)$. Sometimes the problematical term can be removed by transformation. For example, $u \in \bigcup_{x\in A} B(x)$ is equivalent to $\exists x\,[x \in A \wedge u \in B(x)]$. The formula $\phi(\bigcup_{x\in A} B(x))$ is equivalent (in ZF) to

$$\exists v\,[\phi(v) \wedge \forall u\,[u \in v \leftrightarrow u \in \bigcup_{x\in A} B(x)]],$$

allowing the previous equivalence to remove the union. More generally, Joe Hurd has tried translating λ-abstractions into combinatory expressions. Trials can show whether such transformations are practical. An alternative is to allow some of the background processes to run a higher-order theorem prover, such as LEO [6].

Task 7: Transparency. The objective is to minimize the requirement for the user to trust the resolution prover. The many proof tools descended from Edinburgh LCF enforce soundness by allowing only a small part of the code to generate theorems. To follow this principle requires translating the resolution proof into one acceptable to the tool. (Having the translated proof would also let us re-run the verification without the background resolution processes.) Automatic proof translation is therefore the ideal solution. A preliminary, minimal objective would be for Vampire to return the precise list of lemmas used, which would make it much easier for the user to find an alternative proof, e.g. via `blast`.

Task 8: Transferring the technology. The objective is to show that the techniques developed above are general, and not restricted to specific tools. At this stage, we should understand how to overcome the differences between an interactive proof tool for higher-order logic and an automatic theorem prover for first-order logic. We shall demonstrate this understanding by linking the HOL system (probably hol4) to Vampire and perhaps to other resolution provers.

Task 9: Performance refinements. Even when all functionality has been achieved, tuning will be needed in order to obtain high performance. Experimentation will determine how to obtain the best results from the available machine resources. For example, a background job can be allowed to run for a long time, but many shorter runs using a variety of prover settings may be more effective. Multiple processors can work on separate subgoals or on a single subgoal with different prover settings.

Relevance to Beneficiaries
The beneficiaries can be found in several academic disciplines as well as industry.

- Users of Isabelle and HOL will benefit from a much higher degree of automation. Our techniques will be valuable to the developers of other interactive proof tools.
- Methods for translating resolution proofs into natural deduction proofs will interest logicians.
- Interactive verification will become cheaper and thus more widely available. Over time, the verification community will grow, also in industry.

Dissemination and Exploitation
The improved versions of Isabelle and HOL arising from this project will be freely distributed via the Internet. The solutions to the technical problems will be presented in journal and conference papers and in lectures.

Justification of Resources
Staff. Paulson will work part-time on the project. He proposes to supervise one full-time research assistant, Dr. Joe Hurd, and one research student, Jia Meng. The number and variety of project tasks, any one of which could easily overrun its estimated time, makes two assistants necessary. Meng is able (Firsts in all three undergraduate years), diligent and enthusiastic. Concerning the allocation of tasks, Hurd is an able mathematician who can investigate the logical side of proof reconstruction, where he already has experience. Hurd can also help with the practical issues, such as inter-process communications. His HOL experience makes him ideally suited to transferring the technology to the HOL system. Meng can work on most of the early tasks, on modelling the Isabelle type system, and on getting the best performance out of resolution.

Note. By the time this project starts, Meng will have completed one year of study at her family's expense, so only two years of support are requested.

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 and TPHOLs. We may wish to attend relevant workshops at Schloß Dagstuhl and elsewhere. We are also requesting funds for visits to other institutions, as detailed on the application form.

Equipment. The RA and student each need a fast, dual-processor workstation. The second processor adds little to the price tag while doubling the computational power, allowing us to exploit parallelism. Following Prof. Voronkov's advice, each processor is to have one gigabyte of memory. A laptop computer is useful for dissemination; we expect to be able to perform convincing (if small) demonstrations at conferences. We also expect to need a disc and other small items.

Consumables. In addition to the usual small items, we request £1400 per annum for a Poly/ML support contract. This will ensure that any problems we encounter with Poly/ML are promptly addressed. It also encourages continued improvements to Poly/ML, such as the new interactive debugger, which become available to all. Poly/ML is vital: compared with Standard ML of New Jersey, it executes Isabelle with twice the speed and something like half the memory.

References

[1] W. Ahrendt, B. Beckert, R. Hähnle, W. Menzel, W. Reif, G. Schellhorn, and P. H. Schmitt. Integrating automated and interactive theorem proving. In W. Bibel and P. H. Schmitt, editors, *Automated Deduction— A Basis for Applications*, volume II. Systems and Implementation Techniques, pages 97–116. Kluwer Academic Publishers, 1998.

[2] L. A. Dennis, G. Collins, M. Norrish, R. Boulton, K. Slind, G. Robinson, M. Gordon, , and T. Melham. The PROSPER toolkit. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems 6th International Conference, TACAS 2000*, LNCS 1785, pages 78–92. Springer, 2000.

[3] J. Harrison. A Mizar mode for HOL. In J. von Wright, J. Grundy, and J. Harrison, editors, *Theorem Proving in Higher Order Logics: TPHOLs '96*, LNCS 1125, pages 203–220. Springer, 1996.

[4] J. Hurd. Integrating Gandalf and HOL. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher Order Logics: TPHOLs '99*, LNCS 1690, pages 311–321. Springer, 1999.

[5] J. Hurd. An LCF-style interface between HOL and first-order logic. In A. Voronkov, editor, *Automated Deduction — CADE-18 International Conference*, LNAI 2392, pages 134–138. Springer, 2002.

[6] M. Kohlhase. Higher-order automated theorem proving. In W. Bibel and P. H. Schmitt, editors, *Automated Deduction— A Basis for Applications*, volume I. Foundations: Calculi and Methods, pages 431–462. Kluwer Academic Publishers, 1998.

[7] 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.

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

[9] A. Riazanov and A. Voronkov. Vampire 1.1 (system description). In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning — First International Joint Conference, IJCAR 2001*, LNAI 2083, pages 376–380. Springer, 2001.

[10] M. Wenzel. Type classes and overloading in higher-order logic. In E. L. Gunter and A. Felty, editors, *Theorem Proving in Higher Order Logics: TPHOLs '97*, LNCS 1275, pages 307–322. Springer, 1997.

3 Diagrammatic project plan

The allocation of time to various tasks is approximate, but it indicates how difficult each task is and how it depends upon the others. Tasks 1–5 and 9 are mainly for the student, with Prof. Paulson's assistance. Obviously, she will devote time later to writing her PhD. thesis. Hurd will mainly work on tasks 6–8; note that Transparency is particularly difficult.