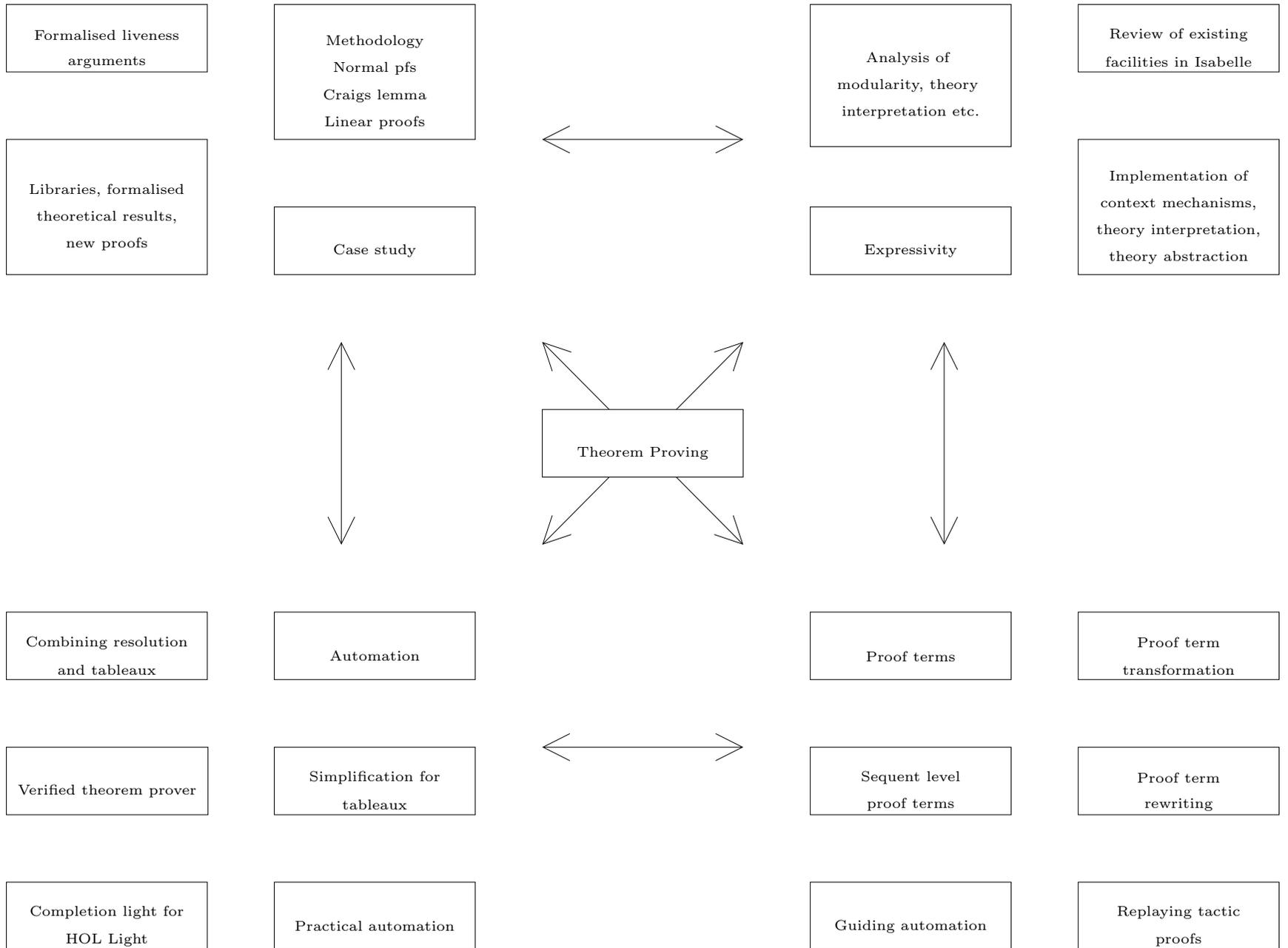


Overview Of Work

Tom Ridge

October 31, 2004



1 Theorem Proving

All our work has been squarely in the field of theorem proving. The aim is to make theorem proving practical. To this end we embarked on a *case study*, in order to identify those areas that needed attention. We came to realise that there were serious deficiencies in both the *expressivity* of current theorem provers such as Isabelle/HOL (that is, they were not practically adequate to express the form of arguments typically encountered in mathematics), and in the *automation* available to users of these systems.

In order to implement our ideas aimed at increasing the expressivity of theorem provers, it was necessary to develop the machinery of *proof terms*, for our chosen theorem prover (at this point HOL Light). On the other hand, in order to guide the automation, one frequently wanted to give the automated prover a skeleton argument, which it would proceed to flesh out itself. The form of this skeleton argument is nicely expressible as a (necessarily not fully expansive) proof term. Thus the areas of automation and expressivity met in the implementation of proof terms.

In addition to the connections detailed above, it is not too hard to imagine how one area might additionally influence another. For instance, if one is able to express theorems concisely, then the automation will have a substantially easier time than if merely stating a lemma takes half a page.

We detail some of the main achievements in each area.

1.1 Case Study

We chose to mechanise a paper [EM03] from the area of distributed algorithms. The contributions here are the standard contributions one would expect from any large, sustained mechanisation attempt. In addition, we have been studious in recording our mechanisation methodology. Where possible, we motivate our approach through connections with results in modern logic. For instance, Craig's Interpolation Lemma states that, given several subtheories T_i of a main theory T , each with language L_i , then for any theorem $\vdash A$, there exists sentences A_i in the language L_i , such that each A_i is a theorem of T_i , and in T one has $\vdash \bigwedge A_i \rightarrow A$. This suggests that it is profitable to break the proof of a theorem A into separate proofs of A_i in each subtheory T_i , and then combine the results in the main theory.

1.2 Expressivity

The case study led us to appreciate that there were severe practical problems facing any mechanisation attempt. The most glaring omission in HOL systems is any mechanism to handle modularity and context. We give an analysis of the problems in this area, review the deficiencies of current approaches, and then suggest our own approach, based on the manipulation

of proof terms. We implemented these ideas in HOL Light, using a very basic implementation of proof terms from elsewhere. For technical reasons, these proof terms were inadequate, so we were also led to develop our own implementation of proof terms.

1.3 Automation

We also experienced grave problems with the existing automation in Isabelle and HOL Light. To this end, we investigated tableaux and resolution systems. The motivation here was to investigate powerful systems of automation that dovetail well with interactive proof. On top of this, equality reasoning is all pervasive in mathematics, so it was also necessary to take account of the special properties of equality. To this end, we implemented various subsystems of automation, such as a completion procedure. We combined this with a tableaux theorem prover, such that completion was effected dynamically, during the proof search. This was sufficient to automatically prove all the lemmas from the first half of our case study. These lemmas are first order provable, so in a sense this is not surprising, but existing automation failed to handle equality reasoning adequately, and the goal was in any case to produce automated techniques that would integrate well with interactive proving.

One of the requirements of interactive proving is to have visibility into the automatic tactics, in order that they can be debugged readily. Our automation executes directly on top of the tactic mechanism, and one can step through it one step at a time. However, if one wants to guide the automated tactics, one needs some way of giving them a skeleton proof. This led back to the notion of proof terms. Specifically, we sought a way to detail the order of the case splits that were required for each theorem, in the form of a sequent level proof term. The requirement for a sequent level proof term is absolutely necessary if one wants to use the proof term to interact with the automation.

In related work, we took the ideas from [Avr93] and described the formal rule systems that allow a combination of resolution and tableaux theorem proving. We also formalised a tableaux theorem prover, resulting in the worlds first verified theorem prover, [Rid04a]. This was favourably received.

Many, many thanks for this contribution! – Larry Paulson

A very impressive piece of work! – John Harrison

We also noticed that during a mechanisation, it was often the case that the first step one wants to apply is simplification. The obvious tactic, of always applying simplification before applying logical rules, seems to be very effective. To justify this theoretically, we gave a proof that a certain tableaux system, extended with powerful simplification steps, retains completeness,

in [Rid04b]. In retrospect the proofs in this paper are rather over complicated, and the work as a whole did not provide enough new insights to be worthwhile.

1.4 Proof Terms

The implementation of proof terms was driven primarily by our requirements as far as automation and expressivity were concerned. We implemented a version of proof terms that was already a HOL standard. For technical reasons, this was inadequate, so we then implemented a version of proof terms at the sequent level (i.e. the level at which the tactics work). As far as we know, this is the first implementation of sequent level proof terms for a theorem prover. With access to the proof term, one can reconstruct a tactic script, or the tactic itself, and replay the proof against the original goal. One also has the ability to inspect the proof. This is extremely useful as far as proof planing, proof presentation and so on are concerned. As an experiment, we implemented proof term transformation by higher order rewriting, enabling us to dramatically reduce the size of many proof terms, whilst still retaining the ability to replay the (modified) proofs at the tactic level. We are also implementing a system to take skeleton proof terms and integrate them with our automation, thereby giving the user variable levels of control over the automation; specifically we aim to allow the user to dictate the order of case splits and so on that occur during a proof. This knowledge is typically available to the user (indeed, the user often wants to dictate the order of the case splits), and can vastly decrease the search space as far as automation is concerned.

References

- [Avr93] Arnon Avron. Gentzen-type systems, resolution and tableaux. *Journal of Automated Reasoning*, 10(2):265–281, 1993.
- [EM03] Javier Esparza and Monika Maidl. Simple representative instantiations for multicast protocols. In *TACAS 2003*, pages 128–143. Springer-Verlag LNCS 2619, 2003. Source for mechanisation available at <http://homepages.inf.ed.ac.uk/s0128214/doc/monjav-fullversion.ps.gz>. Current version available at <http://homepages.inf.ed.ac.uk/monika/index.html>.
- [Rid] Tom Ridge. Informatics homepage. <http://homepages.inf.ed.ac.uk/s0128214/>.
- [Rid04a] Tom Ridge. A mechanically verified, efficient, sound and complete theorem prover for first order logic. *Archive of Formal Proofs*, 2004.

[Rid04b] Tom Ridge. Simplification and proof search. 2004. Published online at [Rid].