A Very Mathematical Dilemma

Alan Bundy

University of Edinburgh

A.Bundy@ed.ac.uk

Abstract

Mathematics is facing a dilemma at its heart: the nature of mathematical proof. We have known since Church and Turing independently showed that mathematical provability was undecidable, that there are theorems whose shortest proofs are enormous. Within the last half century we have discovered practical examples of such theorems: the classification of all finite simple groups, the Four Colour Theorem and Kepler's Conjecture. These theorems were only proved with the aid of a computer. But computer proof is very controversial, with many mathematicians refusing to accept a proof that has not been thoroughly checked and understood by a human. The choice seems to be between either abandoning the investigation of theorems whose only proofs are enormous or changing traditional mathematical practice to include computer-aided proofs. Or is there a way to make large computer proofs more accessible to human mathematicians?

Introduction

Mathematicians strive to achieve proofs which are short, simple and elegant. Unfortunately, these ideals are not always attainable. There are theorems whose shortest proof is enormous. Mathematics faces a dilemma: either these theorems must be ignored or computers must be used to assist with their proof.

Short, simple and elegant proofs are desirable because mathematics is a social process. Mathematicians need to understand proofs not only to ensure that they are correct but also to internalise the key ideas contained within them. Enormous proofs make such understanding difficult or impossible.

Since the late 1950s, there has been a steady stream of work on the automation of mathematical reasoning. Mathematical theories and conjectures are formalised using logic and represented in a computer. An automated theorem prover combines the rules and axioms of the theories in order to prove the conjectures. The abilities of these automated theorem provers has steadily improved with improvements in both the underlying theory and the speed of the hardware on which they are implemented. They are now capable of tackling open conjectures, even in totally automated mode, and, with the aid of human interaction, of solving large and complex problems. However, most mathematicians have shown little or no interest in automated proof.

The proofs produced by automated theorem provers consist of a series of low-level logical steps. They tend to be very long and complicated, which obscures the underlying ideas and makes them difficult to understand. Human mathematical proofs, on the other hand, use formalisation sparingly, so are much shorter and easier to understand. Such human proofs are sometimes described as *rigorous,* to contrast them with logical proofs. On the whole, mathematicians are uninterested in the formal foundations of their subject or of logical proof. By construction, automated proofs are free of error (modulo the soundness of the logical calculus and the faithfulness of the implementation to this calculus). Human proofs often contain errors, and mathematicians are surprisingly tolerant of these, provided they are not fatal to the key ideas of the proof. These differences must be reconciled if mathematicians are to use automated provers to tackle enormous proofs, as advocated in this paper.

In recognition of these issues, the Royal Society held a meeting in London in October 2004 entitled "The Nature of Mathematical Proof" [Bundy et al 2005]. I was involved as both a speaker and an organiser. Many of the ideas presented below arose from the presentations at this meeting and the subsequent discussions.

The Inevitability of Enormous Proofs

The inevitability of theorems whose shortest proof is enormous arises from a result in mathematical logic. In 1936, Alonzo Church and Alan Turing independently proved that predicate calculus provability is undecidable, i.e. that there was no algorithm that would determine whether or not a conjecture in predicate calculus was provable [Church 1936a, Church 1936b, Turing 1936]. This undecidablity of provability extends to nearly all nontrivial areas of mathematics.

The inevitability of enormous proofs is a simple corollary of the Church/Turing result. Suppose, instead, that there *was* a limit to the size of proofs. To be concrete, suppose that there was an arithmetic function that set an upper limit on the size of a theorem's proof given the size of the theorem, e.g. measured by the number of symbols in its statement. We *could* now design an algorithm for the provability of a conjecture, contrary to Church/Turing. Using ideas from mathematical logic, it is easy to define a grammar from which all the proofs in a theory can be generated. If we take some care with the grammar and what it means for two proofs to be equivalent, then there will only be finitely many proofs of any given size. Given a conjecture, we first determine its size and hence the maximum size of any proof of that conjecture. We now generate all proofs in the theory up to that maximum size. This requires an astronomical, but finite, amount of work. When the process terminates, if no proof has been found then the conjecture is unprovable. So, since assuming an upper limit to proof size contradicts Church/Turing's proof there can be no such upper limit. There must be theorems whose shortest proof is enormous.

This argument might be dismissed as an interesting theoretical result with no practical consequences. Maybe all the theorems with enormous proofs are of no mathematical or practical interest. However, there have been several examples in the last half-century of mathematically interesting and simply-stated theorems with very large proofs. In order to prove these theorems it has been necessary to enlist the aid of computers. This use of computers has proven highly controversial. The controversy has focussed both on the correctness of the computer-generated proofs and on their

understandability. I will discuss three examples: the classification of finite simple groups, the Four-Colour Theorem, and Kepler's Conjecture. I have classified these proofs as merely "very large" as opposed to "enormous" because, despite their size, they are only in the foothills of the unbounded proof sizes predicted by Church/Turing.

Three Examples of Simple Theorems with Very Large Proofs

Groups are a simple algebraic structure with a very wide range of applications. **The classification of finite, simple groups** has been an enterprise by a large number of group theorists to identify the kinds of groups that can exist. Michael Aschbacher, one of the principal researchers involved, has estimated this enterprise as consisting of "perhaps 10,000 pages in hundreds of papers" [Aschbacher 2005, p2401]. Understanding the full detail of all this proof is beyond the capacity of a single human mathematician. As yet, no single-volume summary of the proof exists. Moreover, Aschbacher estimates that "the probability of an error in the proof is one" [Aschbacher 2005, p2402], i.e. that the proof is certainly erroneous. Nevertheless, the classification has proven enormously useful and has been widely used to support further mathematical proofs.

Computer algebra systems were used to prove the existence and/or the uniqueness of sporadic groups i.e. some of the very large groups which did not fit neatly into the mainstream classification. Currently, this use of computers has been largely eliminated. We may ask why such elimination was considered a good thing.

Imagine a map displaying different countries. Suppose you wanted to colour this map so that no two adjacent countries had the same colour (see Figure 1). **The Four-Colour theorem** asks whether four colours are always sufficient to do this. This theorem has a long history covering several centuries. It has long been believed to be true but, despite its simplicity, for most of its history it resisted proof. There were many erroneous proof attempts, some of which survived for many years before their errors were discovered. In 1976, Appel and Haken finally proved the theorem with the aid of a computer [Appel & Haken 1977]. They broke down the problem into 1,936 cases, the computer being used to analyse each case in turn. Many mathematicians rejected this proof on the grounds that they could not understand it and, therefore, could not check its correctness. They argued that computer programs were notoriously buggy so that one could not be sure that each of the cases was correctly analysed. Note that Appel and Haken's program was not an automated theorem prover but an ad hoc program that they had written just for this problem.

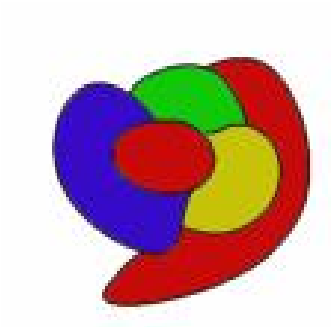

Figure 1: The Four Colour Theorem. *This map is coloured with just four colours, so that no two adjacent countries share a colour. Will four colours always suffice?*

Greengrocers pack oranges, and other spherical fruits and vegetables, in a neat pyramid, in which each orange is balanced on three others (see figure 2). In 1611, **Kepler conjectured that this was the densest way to pack spheres**. This conjecture also resisted proof until late last century. In 1998, Thomas Hales found a computer-assisted proof of this four-century-old conjecture, again using an ad hoc program to test a large number of cases. Hales submitted his proof to the Annals of Mathematics, which set up a 12-person referee team to try to verify the correctness of the computer-generated part. After months of effort the team had to admit defeat. The proof was eventually published in the Annals of Mathematics but with a disclaimer about the correctness of the computer part.

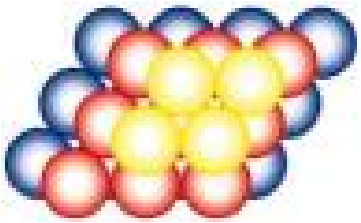

Figure 2: Kepler's Conjecture. *These spheres are packed closely together. Could they be packed more tightly?*

Hales' reaction to this experience is a key piece of evidence for this paper. He has set up a project, called Flyspeck, to replace his ad hoc use of computers with a more principled use of automated theorem provers [Hales 2005]. Since automated theorem provers carry a much higher assurance of correctness, he argues that this should counter the objections, raised by the Annals of Mathematics referee team, to the correctness of his computer-generated proof.

The ostensible objection to these computer-generated proofs has been the possibility of incorrectness caused by bugs in the programs. However, a potentially more significant, but implicit, objection is the difficulty of understanding computer-generated proofs. For instance, both the Four Colour Theorem and Kepler's Conjecture generate thousands of cases that are then checked by ad hoc computer programs. There is too much detail for a human to get a complete understanding of what is going on in each of these cases. As Aschbacher says "thus proofs are... a vehicle for arriving at a deeper understanding of mathematical reality" [Aschbacher 2005, p2403]. Hence the social process, by which mathematicians internalise then build on previous results, is denied, potentially blocking further progress in this direction.

To see how we might combine computer-generated proofs with correctness guarantees and understandability we must look more closely at the more principled use of computers: computer algebra systems and automated theorem provers.

Computer Algebra Systems

Computer algebra systems began in the 1960s with systems for symbolic integration. These systems were extended to include algorithms for manipulating algebra, matrices, groups etc, and were made widely available in systems such as Macsyma and Reduce. They found keen users in the scientific and engineering communities, which needed to manipulate large mathematical expressions and which welcomed the removal of both tedium and error. Today's popular systems include Maple, Mathematica and GAP.

Computer algebra systems have also been taken up by the mathematics community. They are taught to students and used in research. For instance, as we have seen, computer algebra systems were used in the classification of finite simple groups. Not only are they used for routine manipulation of large expressions, but they can also be used for experimentation and visualisation, which is helpful for conjecture formation.

Unfortunately, computer algebra systems are notoriously unsound. For instance, they frequently fail to take into account singularities and other irregularities, especially at boundaries of domains. Uninformed use can lead to the ‘derivation’ of non-theorems. Such unsoundness does not seem to have led to the same criticisms of potential unsoundness that were levelled at Appel, Haken and Hales. Perhaps this is because their incorrectness is more predictable and, with care, can be circumvented.

Automated Theorem Provers

Automated theorem proving also began in the late 1950s and early 1960s, it built on work in mathematical logic dating back to the late 19th-century and early 20th-century. Both axioms and conjectures can be represented as logical formulae. Rules of inference can be represented as programs for manipulating these formulae. The provers build a sequence of formulae, each of which is either an axiom or follows from earlier formulae by a rule of inference. When the last formula of the sequence is the conjecture, then the sequence represents its proof. Unfortunately, many different sequences must be explored in the search for that proof. This search can either be conducted automatically, interactively or by a mixture of both.

A theorem prover’s inner core of axioms and rules of inference can be simply represented and readily inspected. Since the logical manipulations are done by this inner core, which can be given a sound logical interpretation, then the correctness of the prover comes with a high level of assurance. The outer layers of search control, human-computer interface, etc, may be complex, impenetrable and buggy, but they can only request those inner logical manipulations and cannot influence their correctness.

After half a century, automated theorem provers are in a fairly mature state. A number of totally automated systems, such as Vampire, Spass, Otter and E, and a number of interactive systems, such as Isabelle, HOL, Coq, PVS, ACL2 and Nuprl, have a wide user base and a good track record in solving difficult problems.

However, the use of the systems, even the totally automated ones, remains a highly skilled and time-consuming task. These act as a barrier to their wider adoption. In particular, and unlike computer algebra systems, automated theorem provers have never been popular amongst mathematicians.

The Application of Automated Theorem Proving to Mathematics

To show how automated theorem provers might be of assistance to mathematicians, consider the following five case studies.

In 1996, **Bill McCune**, from the Argonne National Laboratory, used the EQP automatic prover to confirm the long-standing **Robbins Conjecture**, that every Robbins Algebra is boolean [McCune 1997]. In 1933, Robbins had provided a set of axioms that he conjectured were an alternative axiomatization of Boolean Algebra. Despite being tackled by a series of leading mathematicians, including Tarski, this conjecture resisted solution until the EQP proof. The Argonne group had been interested in this conjecture for several years. They used a succession of totally automatic (i.e. non-interactive) provers to break the problem into sub-problems and to solve them in succession. The EQP proof completed this process. As the first important open conjecture to be solved automatically, this result was reported worldwide.

For his Cambridge PhD, **Jacques Fleuriot** formalised part of **Newton's Principia** in the Isabelle theorem prover [Fleuriot 2001]. In particular, he analysed Newton's derivation of **Kepler's Laws of planetary motion**. Since Newton reasoned diagrammatically and with infinitesimal quantities, it was necessary to formalise these within Isabelle. This formalisation uncovered a previously undiscovered error in Newton's proof. The error, the cancellation of an infinitesimal quantity on either side of an equation, was one that Newton himself had elsewhere highlighted but had unwittingly made himself on this occasion. Fleuriot was able to correct the error and complete the automation of the proof. It is remarkable that three centuries of analysis of this keynote Principia proof failed to uncover this error, but that it was readily discovered on the first attempt to automate the proof. This case study illustrates the potential of automated theorem provers to detect errors and to help correct them.

For her Edinburgh undergraduate project, supervised by Jacques Fleuriot, **Laura Meikle** automated **Hilbert's Grundlagen der Geometrie** [Meikle & Fleuriot 2003]. Hilbert's purpose in this work was to show how geometric reasoning could be fully formalised without the appeal to geometric intuition. However, Meikle's work revealed that Hilbert *had* appealed to geometric intuition, but that this lapse had remained undetected for most of the 20th-century. Again, Meikle was able to correct the error and to produce a fully formal proof. In the case of Newton, one could argue that his error was minor and easily corrected, so of no real interest to mathematicians. But, in the case of Hilbert, the appeal to geometric intuition undermined the main purpose of his work, so must be regarded as a much more serious error.

In reaction to the controversy over the correctness of **Appel and Haken's proof of the Four Colour theorem, Benjamin Werner and Georges Gonthier** have formalised and automated this proof in the Coq theorem prover [Gonthier 2005]. This

work provides a computer proof with a very high assurance of correctness. However, the proof still consists of a very large number of complex cases; it is still very hard to understand.

Finally, as already mentioned, **Thomas Hales' Flyspeck project** is an attempt to formalise and automate the proof of **Kepler's Conjecture** [Hales 2005]. Hales has estimated that this will require 20 person-years of work. There has been widespread and enthusiastic engagement from the automated theorem proving community, with many researchers contributing to the Flyspeck enterprise. As a result, different parts of the proof have been automated in different theorem provers, including HOL-Light, Coq and Isabelle. This diversity is unfortunate, as it serves to undermine the assurance of correctness and to make the proof more difficult to understand. The project is still its early stages but has recorded an initial success with the automation of the Jordan Curve Theorem.

Can Automated Theorem Proving Address the Enormous Proof Problem?

Now we have looked at the field of automated theorem proving, it is time to return to the problem of enormous mathematical proofs. Can automated theorem proving help address this problem?

Firstly, we might ask whether computers are necessary at all. We have already seen an example, the classification finite simple groups, in which a very large proof has been largely human executed. Even this proof's few uses of computer algebra systems have now been largely eliminated. It seems that very large proofs can be tackled with solely human effort. However, note that the major problems of very large proofs still occur. Not only is there no guarantee of correctness, but Aschbacher, one of the major authorities on this proof, estimates the probability of error as one. Moreover, the complexity of the proof denies full human understanding. So, although computers play only a minor role in the construction of the proof they might still have a role in ensuring its correctness. Could they also help with understandability?

As we have seen, the correctness-sensitive parts of an automated theorem prover can be restricted to a small and inspectable core. By this means, automated theorem provers can provide a high level of assurance of the correctness of any proofs constructed with their aid. Unfortunately, computer proofs are typically inaccessible, due to the low level of detail in their logical representation. We have seen that understandability is extremely important to mathematicians -- arguably much more important than correctness. Mathematicians are extraordinary tolerant of errors in proof, provided that they are minor and readily corrected. This tolerance was a constant theme in the Royal Society meeting on "The Nature of Mathematical Proof", as witnessed, for instance, by the remark of Aschbacher quoted above.

The question we have to address is whether computer-generated proofs can be made accessible to mathematicians. Further, we may ask whether computer visualisation and abstraction techniques can aid the understandability of very large, or even enormous, proofs.

A standard abstraction technique in mathematics is to break a large theorem into lemmas. This provides structure. Each lemma can be made small enough to be intelligible. The combined proof will still be at least as large as the original, unless some lemmas are recycled in different parts of the proof. In the next section we further extend this kind of abstraction by chunking some proof steps together. This chunking process can be applied recursively. The top-level chunks will provide an outline or plan of the whole proof.

Proof Planning

For a decade and a half, my research group has been developing the technique of proof planning [Bundy 1991]. We attempt to formalise common patterns in mathematical proofs. The search for a proof then takes place at the level of these common patterns, which reduces the amount of search and improves the productivity of automated theorem proving systems. Common patterns of proof are automated as *tactics*: computer programs which direct the application of rules of inference to formulae. Both the preconditions under which tactics should be applied and the effects of their application are formalised. The proof planner then builds a customised, hierarchical plan for the current conjecture. The proof planner also has access to *critics*, which detect, analyse and correct failures of proof attempts.

In order to visualise our hierarchical proof plans, we have developed a technique that we call *hiproofs* [Tourlas et al 2005]. Each of the tactics in a proof plan is represented by a box; boxes being nested to represent the hierarchical structure of the proof plans. Arrows between the boxes represent the flow of subgoals between the tactics. Figure 3 shows a hiproof for the induction strategy proof plan.

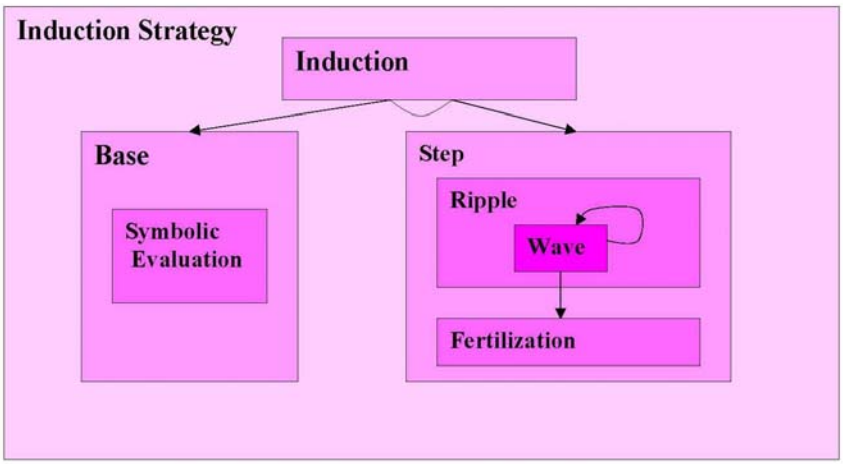

Figure 3: A hiproof for the induction strategy. *The whole induction strategy is represented by the outer box. It consists of three tactics: an application of an induction rule, followed by a base case and a step case. The base case consists of symbolic evaluation and a step case consists of ripple followed by fertilization. Ripple consists of repeated applications of the wave tactic.*

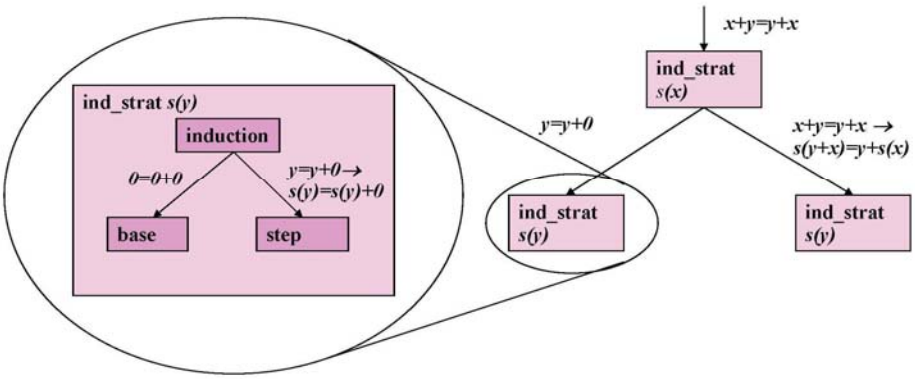

Figure 4: An exploded view of one of the tactics. *The proof of the commutativity of addition consists of three applications of the induction strategy. One of these applications has been exploded to show its inner structure, which consists of an induction tactic followed by a base and step case.*

Hiproofs enable proofs to be viewed at different levels of abstraction. We can zoom out and view only the top-level tactics or we can choose one of the tactics and zoom into its box, looking at the detail within it. Figure 4 illustrates the ability to zoom in by showing a blown-up view of one of the tactics. Figure 5 shows how a complex proof can be viewed at a high level in terms of the top-level tactics that form its overall proof plan.

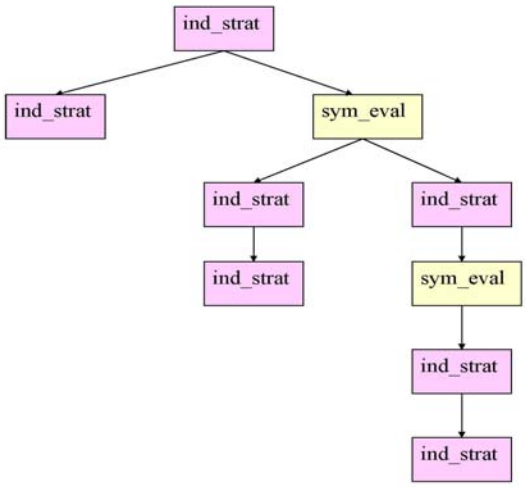

Figure 5: The overall view of a proof. *The depicted proof plan is from the verification of an electronic circuit for multiplication. The hiproof shows the top-level tactics involved in the proof. It consists of seven applications of the induction strategy interleaved with two applications of symbolic evaluation.*

Hiproofs are one candidate for assisting mathematicians to understand very large proofs. They support browsing of the structure of the proof, delving down into detail where required. They might also support the human construction of proofs, enabling the high-level structure to be mapped out, with detail being provided as it becomes

available. Of course, hiproofs do not provide a complete solution to the understanding of very large proofs. For instance, a proof with a very large number of cases, such as Appel & Haken's, remains inherently difficult to understand and, short of a completely different proof, it is not clear what can be done to solve this problem. However, if computers are to aid mathematicians in their understanding of larger and larger proofs it will be necessary to use a variety of proof visualisation techniques, hiproofs providing one candidate.

Conclusion

Computers have yet to play their full potential in mathematical research. The major success story to date has been computer algebra systems. They are widely taught to students, perform routine calculations in proofs and are used for exploratory experiments. The use of ad hoc programs to analyse large numbers of cases in very large proofs has proved highly controversial. Doubts about correctness are the ostensible objection, but the difficulty of understanding the resulting proofs may be a more fundamental objection. Automated theorem provers have been largely ignored. However, recent research by Gonthier on the Four Colour Theorem and by Hales on Kepler's Conjecture are a promising new departure. Their use of automated provers suggest both that state of the art provers are capable of tackling very large proofs and also that their ability to provide a high level of correctness assurance has been recognised as a key selling point. However, automated theorem provers will not be widely adopted unless they can also aid the *understanding* of proofs, especially of very large proofs.

The presentation of proof plans with the aid of hiproofs suggests one way in which automated theorem provers might aid proof understanding. In particular, if proofs could be built top-down by using hiproofs to sketch out their high-level structure and gradually filling in more and more detail, with automation taking care of the very lowest level steps, then this might encourage the uptake of automated theorem provers by working mathematicians.

Acknowledgements

I am grateful to the organisers of, speakers at and participants in the Royal Society discussion meeting on the Nature of Mathematical Proof, for the stimulating interactions that led to this paper. I'm also grateful for feedback from the audience at my seminar on "Computational Thinking", where I presented these ideas before they were written up. I'd like especially to thank Stuart Anderson, Michael Aschbacher, Henk Barendreght, Lucas Dixon, Jeremy Gow, Alex Heneveld, Mateja Jamnik, Predrag Janicic, Andrew Ireland, Donald MacKenzie, Raul Monroy and Rodney Toper. This work was supported by EPSRC grant GR/S01771. It was dictated using the speech recognition software Dragon Naturally Speaking.

References

[Appel & Haken 1977] Appel, K.; Haken, W.; and Koch, J. "Every Planar Map is Four Colorable. I: Discharging." *Illinois J. Math.* **21**, 429-490, 1977.

[Aschbacher 2005] Aschbacher, M., *Highly complex proofs and the implications of such proofs.* In [Bundy et al 2005], pp 2401-2406, 2005.

[Bundy 1991] Bundy, A., "A science of reasoning", in *Computational Logic: essays in honor of Alan Robinson,* ed Lassez, J-L. & Plotkin, G., MIT Press, pp 178-198.

[Bundy et al 2005] *The nature of mathematical proof.* Phil Trans of the Royal Society A. Eds Bundy, A., Atiyah, M., Macintyre, A. and MacKenzie, D., Vol. 363, No. 1835, pp 2329-2461, 2005.

[Church 1936a] Church, A. *An unsolvable problem of elementary number theory*, American Journal of Mathematics, 58 (1936), pp 345-363.

[Church 1936b] Church, A. A note on the Entscheidungsproblem, Journal of Symbolic Logic, 1 (1936), pp 40-41.

[Fleuriot 2001] Fleuriot, J., *A combination of geometry theorem proving and non-standard analysis with applications to Newton's Principia.* Distinguished dissertations. Berlin Springer, 2001.

[Gonthier 2005] Gonthier, G., *A computer-checked proof of the Four Colour Theorem.* http://research.microsoft.com/~gonthier/4colproof.pdf.

[Hales 2005] Hales, T., *The Flyspeck Project Fact Sheet.* http://www.math.pitt.edu/~thales/flyspeck/.

[McCune 1997] McCune, W. "Solution of the Robbins Problem" Journal of Automated Reasoning 19(3), pp 263-276, 1997.

[Meikle & Fleuriot 2003] Meikle, L. and Fleuriot, J., "Formalising Hilbert's Grundlage in the Isabelle/Isar". In *Theorem Proving in Higher Order Logics: 16th International Conference, TPHOLs 2003*, Springer Lecture Notes in Computer Science, vol. 2758, pp 319-334. Berlin: Springer.

[Tourlas et al 2005] Tourlas, K., Power, J. and Denney, E., "Hiproofs: A hierarchical notion of proof tree", in *Procs of 21st Conference on Mathematical Foundations of Programming Semantics*, Birmingham, UK, to appear in *Electronic Notes in Theoretical Computer Science*, Elsevier, 2005.

[Turing 1936] Turing, A. M. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Series 2, 41:230-267, 1936.