

# Automated Deduction Systems for Real Mathematicians

Paul Cairns and Jeremy Gow<sup>1</sup>

UCL Interaction Centre, University College London,  
26, Bedford Way, London WC1H 0AP, UK  
{p.cairns, j.gow}@ucl.ac.uk  
WWW home page: [www.ucl.ac.uk](http://www.ucl.ac.uk/ucl-icc)

**Abstract.** Automated deduction systems currently have a low uptake (or even recognition) amongst mathematicians. This paper takes a human computer interaction perspective on the role of automated deduction in mathematics. We first dismiss the fallacy that making systems easy to use will make them used by heuristically comparing Microsoft Word and  $\text{\LaTeX}$  systems in mathematical authoring. Through considering the goals of mathematicians, we propose ways to develop automated deduction systems that mathematicians actually want. There are clearly considerable technological and even philosophical barriers but it is hoped that these can be surmounted in time.

## 1 Introduction

Computer algebra systems have made considerable impact on the working practices of mathematicians, particularly applied mathematicians. Using systems such as Mathematica, a mathematician can perform many symbolic manipulations on complicated models and at the right moment convert these to numerical results for a concrete problem. This has radically changed not only the working practices of mathematicians but also those of physicists and engineers who no longer need rely on expert mathematicians to help them with their problems.

Given the wide variety of mature automated deduction systems (ADS), it would be hoped that at least one system might have already had a comparable impact on the working practices of mathematicians. Yet, with one or two notable exceptions [16], mathematicians have barely even heard of automated deduction systems let alone used them. This is not that the ADS developers have not tried to make such systems available to mathematicians: the UITP conference [2] was expressly concerned with producing good user interfaces to theorem provers; the proof transformation and presentation workshop has similar goals [21]; and this workshop too is clearly hoping to make in-roads in this area. There is clearly then some greater gap between mathematicians and ADS than can be explained by difficult user interfaces.

In this paper, we take a human-computer interaction perspective on the role of automated deduction in mathematics. Rather than considering ADS as the

solutions to mathematicians' problems, we consider what problems mathematicians might currently have and whether ADS can help out. First, we discuss why making systems easy to use does not make them fit for use by comparing the easy to use word processor Microsoft Word with the document preparation system  $\text{\LaTeX}$  [14]. Next, we take a high-level look at the needs of mathematicians as individuals and as a community and see how ADS might meet those needs. Finally, the paper proposes some ways forward to bring a user-oriented perspective to the development of ADS for mathematicians.

## 2 Easy to use $\nrightarrow$ Useful

It is a common selling point of many software systems that they are easy to use. This is all well and good but ease of use in itself does not constitute a useful system. If the software is not useful in the sense of helping users to achieve their goals then no amount of easy to use interfaces is going to make users want to use the system. With mathematicians, this can be seen most clearly in the uptake of  $\text{\LaTeX}$  above other easy to use systems such as Microsoft Word.

$\text{\LaTeX}$  is a document preparation system, that is, it allows the user to type in the content of the document and  $\text{\LaTeX}$  will format it, typeset it and generally make a good attempt to make it look good on the printed page. The user must enter content made up of the words that the user wishes to convey together with instructions for displaying symbolic equations and further instructions for the usual typographical aids such as section headings, paragraphs, footnotes, lists, references and the like. The content is then processed separately to produce a printable version of the document.

This contrasts markedly with the WYSIWYG (“What You See Is What You Get”) approach of most popular word processors. With these, the user types the words and they appear on the screen as they would on the printed document — content and presentation are inextricably linked. Toolbars and menu allow the user to change the formatting of the document and those changes appear immediately on the screen. This means the user never needs to stop entering content in order to process the document to see what it looks like and thus their work has a seamless flow from starting entering content to printing out the final document.

In terms of ease of use, WYSIWYG systems, as epitomised by Microsoft Word, are the most easy to use. The user starts the application and is immediately producing the document in its final form.  $\text{\LaTeX}$  is far from easy to use: the user must not only enter the content but must instruct the system on how to present equations, what sort of document is being produced and what presentational features to include. None of these instructions are made available to the user via the system as  $\text{\LaTeX}$  is separate from any text editor that might be used to prepare the documents. Novice users must type with the manual to hand and even expert users consult the manual to achieve difficult or unusual typesetting effects.

At face value then mathematicians, like many other professionals, should prefer using Word over  $\text{\LaTeX}$ . But this is clearly not the case. Many journals are produced exclusively in  $\text{\LaTeX}$  to the point that they even produce style sheets so that the authors can achieve the journal style on their submissions [24]. Publishing companies, such as Springer, produce their mathematical texts entirely in  $\text{\LaTeX}$ . And the mathematical community produces most of its work using  $\text{\LaTeX}$  to the point that in talks, when introducing a new symbol, the speaker will sometimes give the  $\text{\LaTeX}$  code for the symbol so that the listeners know what it is.

The reason for this apparent irrationality in an otherwise logical community must be that whilst Word is easy to use it is not easy to do what mathematicians want to do. For instance, consider the following anecdotal evidence.

Mathematicians commonly want to put equations into their text so, for example, to include the following sort of a equation (taken from [6]) should be a routine task.

$$C = \bigcup_{f \in 2^\omega} \bigcap_{n \in \omega} \overline{U_{f|_n}}$$

In  $\text{\LaTeX}$ , this is done with the somewhat complicated instructions:

```
$C = \bigcup_{f \in 2^{\omega}} \bigcap_{n \in \omega} \overline{U_{f|_n}}$
```

Only an expert user or a novice user sitting with the manual would know how to do this. In Word, though, it is enough to go to the menu `Insert|Object` and select the equation editor. The user is then taken from the Word document into the special editor and presented with a palette of buttons. The icons on the buttons clearly correspond to different types of symbol that could be used in equations and it is fairly easy to find the exact symbol that is required and so build up the equation. However, already the WYSIWYG paradigm has been broken, the user has been taken from the flow of producing a document to go to a special new process for equations. Whereas, in  $\text{\LaTeX}$ , typing the equation was done in the same way that other typesetting is set up — by typing instructions. Even so, for the novice, the task was reasonably painless in Word and probably a lot less difficult than  $\text{\LaTeX}$ .

However, the important point in this scenario is that this is what mathematicians *commonly* do. Very quickly, whichever system is being used, the mathematician user is going to get used to entering equations and producing certain symbols that they use frequently. Yet with Word, the only way to enter equations is through the editor and the only way to produce symbols is through the palette of buttons. There are no shortcuts to getting to, say,  $\bigcup$  quickly. Expert users must go at the same pace as novices, violating a well-known usability principle that there must be ways for experts to improve their performance [18]. And for every equation, the user must interrupt their current mode of use to enter the special editor but, for mathematicians, entering equations should be the normal mode of use. In  $\text{\LaTeX}$ , there is a single mode of use and the codes

for symbols become quicker to recall with practice. As time goes by, expert users will automatically speed up and come to write equations as rapidly as they write ordinary sentences.

This is only one aspect of preparing mathematical documents and the analysis is only heuristic rather than empirically demonstrated. However,  $\text{\LaTeX}$  is undoubtedly the most successful tool in this area. There may be many reasons why this is so, including historical serendipity, but the lesson stands that ease of use in itself will not persuade mathematicians to use a tool. With regards to the role of ADS in mathematics, simply making ADS easier to use will not necessarily change their current value to mathematicians nor will it suddenly make mathematicians realise what they are missing out on. Either ADS must offer something new that mathematicians really want or they must support what mathematicians currently do with tangible benefits over the existing ways of doing things. In both cases, this requires a deep understanding of the activities and needs of mathematicians.

### 3 What do mathematicians want?

It is impossible to define what mathematicians want without extensive research into a broad cross-section of the mathematical community. And even then, it may be impossible to come up with general indications that do not exclude the majority of mathematicians. However, there are some broad, high-level goals that it is easy to assert that mathematicians really do want to achieve. Additionally, it is worth making a distinction between what individuals want and what the community as a whole wants.

As individuals, some simple goals of mathematicians are:

- Do (good) maths
- Publish results/achieve recognition
- Don't feel stupid

Mathematicians by definition want to do maths. It is not even clear that they want to do maths that would be recognised universally as good maths but at the very least they want to do maths that is good for them. Also, at some point, the majority of mathematicians want to receive recognition for their work. The usual channel for this is the traditional academic route of papers in journals or conferences but other approaches will do.

The third goal of not feeling stupid is one made emphatically by Cooper, and it is not particular to mathematicians but to everyone [8]. None of us want to look stupid and this is such a personal goal that it is almost never made explicit. Yet it can greatly clarify and motivate a lot of human activity. For mathematicians, not feeling stupid could mean producing correct proofs, or at least not obviously wrong ones, having mastery of a discipline or even just being able to ask interesting questions.

Notice that there is no high-level goal of mathematicians that they want to learn about automated deduction or that they want to use ADS. In fact, giving

mathematicians a complicated system that they find difficult to understand and cannot see the benefit of will almost certainly make them feel stupid [8].

As a community, mathematicians also have high-level goals:

- Disseminate results
- Ensure the quality of published work
- Find existing results

These goals can be seen in the activity of most of the mathematical societies [15, 1]. They provide edited journals which ensure that good quality results are disseminated. Also, the American Mathematical Society publishes the Mathematical Reviews, synopses of every paper published in a mathematical journal in paper, CD ROM and web-based versions. These are so that there is a single resource for mathematicians to consult when they want to find existing mathematics. Again, ADS do not feature explicitly.

To bring automated deduction into mathematics is going to require more than just producing good ADS. Killer applications will be ones that meet at least one goal of a significant number of mathematicians.

### 3.1 Automated deduction in education

We have deliberately avoided the issue of mathematics education. This is mainly because it is not obviously a goal of individual mathematicians. However, it almost certainly is a goal of a large group of mathematicians or even the mathematical community. Melis has had some success in using proof planning methods to teach mathematicians generic proving skills [17]. Once the educational value of ADS has been recognised, it may be that mathematicians will be more generally open to ADS and their goals might begin to include specific aspects of ADS.

## 4 Ways Forward

The two sets of goals outlined above are very general. It may be that in analysing these goals in more depth it will become apparent how ADS could fit in.

In addressing the mathematicians' goal to do mathematics, it is not clear how ADS could best offer support. This is mostly because there is no real understanding of where mathematicians could use support and whether ADS are the right tools to provide it. We are about to start a project to investigate how computer-based tools might support mathematicians. The first step is to examine thoroughly what it is that mathematicians actually do when they do maths. The investigation will not only interview and observe mathematicians about their work but also look at the products of mathematical work such as text books, journals, talks and classes to understand how mathematicians communicate their ideas to each other. Also, the project will look at what sorts of interactions might support mathematicians. For example, would a tool that helps

mathematicians correct proofs as they work be useful? Currently with ADS we seem to be a long way from this goal however it would be easy to simulate such a system using a “Wizard of Oz” type set up [20]. Users may think that they are interacting with a piece of software where in reality they are interacting with a human across a network in a constrained manner. Simply performing this simulation will offer valuable insights into whether such software would be useful or degenerate into the merely irritating as with the Microsoft Word “Paper Clip”. Additionally, such simulations could suggest other ways in which ADS could be used that have not been fully explored.

As well as, going to the target users, we can consider psychological models of how people reason. There are some well-established models with good empirical evidence [12, 22]. These models are able to specify the kinds of mistake that people make generally when performing reasoning tasks. This could add to the user experience if, when working with software, the system not only affirmed correct proofs but identified and corrected common mistakes.

And there could be other auxiliary systems which would make a proof support tool more attractive such as support for finding particular results. These functions not traditionally considered an important part of automated deduction but could feasibly complement it well when integrated as part of a general support tool. Indeed, once thinking along these lines, it may be possible to come up with a myriad different tools that mathematicians might find useful. Which is why resorting back to an empirical knowledge of what mathematicians actually need is essential.

Within computer science, there is a growing recognition that mathematics is a burgeoning discipline with no real standards for communication and exploitation of results. This has led to the area of mathematical knowledge management that aims to investigate ways in which mathematics can be organised for better retrieval [5]. This will require setting standards for the representation of mathematics and applications that can convert the standard representations into forms suitable for mathematicians to use. In addition, ensuring the quality of all of this new mathematics is difficult. Here then is an ideal area in which ADS could offer support — they could help mathematicians to check existing mathematics and to ensure that conversions between different representations do not introduce errors.

Though the issues of mathematical knowledge management are not being driven by mathematicians, they address two of the goals identified earlier, namely that of disseminating results and achieving recognition for them. As such, mathematical knowledge management holds promise for developing new tools that mathematicians will want to use.

Just as mathematicians goal is to do maths, the goal of a researcher in automated deduction is to advance knowledge of automated deduction. Our sort of user investigation requires stepping back from the systems and becoming more of an HCI researcher. Is there something more immediate that HCI can offer for developers of ADS? We propose that Cooper’s notion of personas (personæ) may have potential. He advocates not designing a system for a generic user, in this

case a generic mathematician, but rather to develop an idealised but realistic mathematician persona. By targeting this persona, you are likely to produce a system that this sort of mathematician will definitely want rather than a generic system that no particular mathematician wants.

Personæ could be developed based on only a few interviews with mathematicians — Beyer and Holtzblatt [3] advocate that only fifteen users are sufficient to identify the major user activities. The key thing is to keep the persona realistic (even idiosyncratic) so that the system is more likely to address real goals of mathematicians rather than amorphous goals of an idealised user. Indeed, it may be possible to include personal anecdotes or even characteristics of famous mathematicians in the personæ. User goals are then implicitly embodied and whilst the resultant system may not suit all, or even the majority, of potential users, it has a high chance of being exactly right for some users.

## 5 Barriers between Mathematicians and ADS

Promoting the role of automated deduction in mathematics is not without barriers. A commonly occurring concern is that of formalisation. ADS naturally work at a very formal level of logic with manipulations at a symbolic level. This is undoubtedly a strength of ADS because it means that there can be a great deal of confidence in a proof that can be checked (albeit not generated) by a simple mechanistic system. Mathematicians though work at a very informal level which facilitates communication between humans but is very prone to errors. An analysis of journal articles reveals numerous small errors and even some quite substantial ones [11]. Similarly, standard textbooks suffer from mistakes [13], or if not actual mistakes, such large leaps of logic that it is almost impossible to reconstruct the original thinking [19].

To get humans and ADS to work together will require translating (and possibly correcting) the informal proofs of humans into the formal language of ADS and filling in the logical gaps. If this fails, the ADS will have to communicate back a breakdown in logic that the user may not even be aware of as relevant. There are some systems [26, 9] that propose ways of making the translation to formal proofs and others which offer ways of translating back [10, 25]. But it remains to be seen if they can be combined into a single system that can help a user correct a proof as they write it.

Formalisation, as currently done by ADS, may not be rich enough to meet the needs of mathematicians. From our own work in presenting mathematical proofs, there is also the issue of the role of examples in formal mathematics [7]. Like many formalisations of mathematics, our Polya-Lampert framework divides mathematical statements into definitions and theorems. However, examples do not fall so simply into these categories. In some sense, examples can be irrelevant to the development of a theory — illustrative but simply instantiations of existing results. In other situations, examples are crucial as providing stereotypical exemplars. They provide instances of theorems and motivations for new definitions. And in themselves they need to be defined but often have things

proven about them so do not fall naturally into either definition or theorem. Dismissing them as irrelevant to theory completely overlooks their central role to mathematicians.

The simple solution to the formal/informal mismatch would be to exhort mathematicians to become more formal [4]. But this is not the goal of mathematicians so is unlikely to happen. It may be that by revealing some of the benefits of mathematical knowledge management, mathematicians might decide it is in their interest to become more formal (much as it is in their interest to learn  $\text{\LaTeX}$ ). But for the moment, any system which aims to bridge the gap between mathematicians and machines must make some attempt to bridge between the formal and the informal.

A further issue with bringing computer-based support to mathematicians is that of creativity. Mathematicians, like all researchers, are involved in a creative process. Sometimes that process feels instantaneous, “a flash of inspiration,” and no amount of structured support is going to be available in all such circumstances. However, a lot of creativity arises a result of sustained, intentional effort and hence computers could feasibly be useful. Even in this case, a traditional HCI view of developing software to support user tasks does not really apply — there is no clear task involved in being creative, at least not one that HCI is currently able to address.

Shneiderman has developed the Genex framework that describes how computer systems might support creativity [23]. However, at the actual create step of the framework, the particular systems that might promote innovation need to be tailored for the type of creativity. It may be that ADS could play a useful role here, say, supporting the “what if” possibilities around a proof or a definition. The Genex framework does not point to ways forward in this area so, once again, it will be essential to rely on empirical investigations to find out what really supports creating new mathematics.

Even allowing for the almost philosophical issues raised by bringing automated deduction into mainstream mathematics, there are more mundane problems such as: many systems are not easily ported to a new platform; there is not a great deal of technical support for some of the systems; and there is no standard way to communicate between systems. There is going to have to be a lot of routine standardisation and development before automated deduction systems are made available to ordinary mathematicians.

## 6 Conclusions

We have considered how making mathematicians use automated deduction systems is far more than just making the systems easier to use. ADS must support mathematicians’ goals, be they personal or communal goals, if ADS are to have a broad-based uptake in the mathematical community. Mathematical knowledge management seems to be employing ADS in a goal-directed way and so has a lot of promise. However, helping mathematicians to do mathematics requires a deeper understanding of what mathematicians do than is currently available.

Also, ADS may be more be appropriate support tools only as part of a larger system that can identify mistakes or find existing results. This broader sort of proof support can only be justified through a real understanding of what mathematicians would actually want from such a system.

## 7 Acknowledgments

Many thanks to Prof. Harold Thimbleby for his comments on this paper. This work was supported through EPSRC grant GR/N29280.

## References

1. American Mathematical Society, [www.ams.org](http://www.ams.org)
2. R. Backhouse (ed.), *Proceedings of User Interfaces for Theorem Provers*, Technical University of Eindhoven Computer Science Report 98/08, 1998
3. H. Beyer & K. Holtzblatt, *Contextual Design: Defining Customer-centered Systems*, Morgan Kaufmann, 1998
4. B. Buchberger, 'Mathematical knowledge management in Theorema,' *First International Conference on Mathematical Knowledge Management*, 2001
5. B. Buchberger & O. Caprotti, *First International Workshop on Mathematical Knowledge Management*, 2001  
[www.risc.uni-linz.ac.at/institute/conferences/MKM2001](http://www.risc.uni-linz.ac.at/institute/conferences/MKM2001)
6. P. A. Cairns, *Boundary Properties and Construction Techniques in General Topology*, DPhil Thesis, University of Oxford, 1995
7. P. A. Cairns & J. Gow, 'On Dynamically Presenting a Topology Course,' *First International Conference on Mathematical Knowledge Management*, 2001
8. A. Cooper, *The Inmates are Running the Asylum*, SAMS, 1999
9. I. Dahn & G. Schwabe, 'Personalizing Textbooks with Slicing Technologies — Concept, Tools, Architecture, Collaborative Use', HICSS, 2001
10. A. Fiedler, 'P. rex: An Interactive Proof Explainer,' in R. Goré, A. Leitsch & T. Nipkow (eds.) *Automated Reasoning – 1st International Joint Conference*, LNAI 2083, Springer Verlag, p416-420, 2001
11. J. Harrison, 'Formalized Mathematics,' *Math. Universalis*, 2, 1996
12. P. Johnson-Laird & R. Byrne, *Deduction*, Psychology Press, 1991
13. L. Lamport, 'How to write a proof,' *American Mathematical Monthly*, 102(7) p600-608, 1994
14. L. Lamport, *LaTeX: A Document Preparation System*, 2nd edn, Addison-Wesley, 1994
15. London Mathematical Society, [www.lms.ac.uk](http://www.lms.ac.uk)
16. W. McCune, 'Solution of the Robbins problem,' *J. Automated Reasoning*, 19(3) p263-276, 1997
17. E. Melis, C. Glasmacher, C. Ullrich & P. Gerjets, 'Automated Proof Planning for instructional design,' in *Annual Conference of the Cognitive Science Society*, p633-638, 2001
18. J. Nielsen, *Usability Engineering*, Morgan Kaufmann, 1993
19. L. C. Paulson & K. Grabczewski, 'Mechanizing Set Theory,' *J. of Automated Reasoning*, 17 p291-323, 1996
20. J. Preece, Y. Rogers & H. Sharp, *Interaction Design*, John Wiley & Sons, 2002

21. *Proof Transformation & Presentation*, *PTP-01*, IJCAR, 2001, [www.ags.uni-sb.de/~ptp-01](http://www.ags.uni-sb.de/~ptp-01)
22. L. J. Rips, *The Psychology of Proof*, MIT Press, 1994
23. B. Shneiderman, "Creating creativity: user interfaces for supporting innovation," in J. M. Carroll (ed.) *Human-Computer Interaction in the New Millennium*, Addison Wesley, 2002
24. *Topology & Its Applications*, [www.elsevier.com/locate/latex](http://www.elsevier.com/locate/latex)
25. M. Wenzel, 'Isar — a Generic Interpretative Approach to Readable Formal Proof Documents,' in *Proceedings of TPHOLS'99*, Springer, 1999
26. C. Zinn, 'Towards the Mechanical Verification of Textbook Proofs,' *7th Workshop on Logic, Language, Information and Computation (WOLLIC-2000)*, 2000