# Computers, reasoning and mathematical practice \*

# Ursula Martin

um@dcs.st-and.ac.uk University of St Andrews

Computer aided formal reasoning, mathematical assistants which check complex arguments and automated proofs of new and interesting mathematical results have been part of the dream of computational logic for many years. This dream is in part being realised by the success of endeavours such as the Mizar project [110], which has produced many volumes of formalised mathematics, and McCune's recent proof of the Robbins conjecture [79], cited, along with autonomous vehicle guidance and the chess program Deep Blue, as one of the five most significant achievements of artificial intelligence [114]. Yet it is still the case that few mathematicians use such programs, and their impact outside certain specialised communities has been less than might have been hoped.

The main impact of computation on mathematics has been through experiment and computations used as part of a traditional mathematical proof: techniques such as numerical computation, visualisation of complex topological surfaces, the specialised algorithms used in the classification of finite simple groups or the use of computers to perform case analyses such as in the proof of the four colour theorem. In addition general tools such as computer algebra systems and software for mathematical typesetting and teaching are used very widely.

Indeed this widespread use of computation for experimental work has lead to new journals such as Experimental Mathematics and claims from some that the nature of mathematics is being fundamentally changed with a paradigm shift from rigour to "semi-rigorous" or "speculative work". Most mathematicians would question these claims: the discipline of proof remains unchanged and computation has attracted attention where it has been used precisely because this is unusual. In the major achievements of 20th century mathematics, such as the recent proof of Fermat's theorem or the developments in topology and geometry, the role of computation has been slight. In the case of the classification of finite simple groups, some were constructed computationally but could then be investigated theoretically, and the 5000 pages or so of the classification theorem is, essentially, a hand proof that there are no others. Computation has transformed biology, chemistry, neuroscience and physics: without computational mathematics we would have no Mars pathfinder robot, no human genome project and no CAT scans. We might still have the proof of Fermat's theorem.

Nevertheless computational techniques are becoming increasingly important in some areas of mathematics, and it is our purpose here to examine the opportunities they create for computational logic research. We first analyse current

<sup>\*</sup> To appear, Computational Logic, (Marktoberdorf, 1997), NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci., Springer, Berlin, 1998, ed H Schwichtenberg

mathematical practice, which we identify with producing conjectural mathematical knowledge by means of speculation, heuristic arguments, examples and experiments, which may then be confirmed as theorems by producing proofs in accordance with a community standard of rigour, which may be read by the community in a variety of ways. The goal may be individual theorems, such as Fermat's conjecture, or more general bodies of knowledge such as topology and geometry, addressing the structure of surfaces. We note that while notions of rigour and quality (what makes a good proof) may have changed, practice itself has remained remarkably robust over the centuries. We observe that computation has proved important for speculation and experiment in several areas, and that, despite vigorous debate over matters such as the four colour theorem, computation, particularly well understood domain specific algorithms, is regularly accepted as part of a proof. As with acceptance of other kinds of proof, community acceptance of the computation is often governed by matters such as the perceived competence of the software developer or the public availability of the sources rather than line by line verification of the code.

We then consider computer aided reasoning, an academic endeavour that has grown up largely independently of research mathematics, and assess its application to research mathematics in the light of our description of practice. We consider two strands: automated proof by techniques such as resolution, and full formal development of large bodies of mathematics. We argue that the former is indeed useful for proving new theorems, somewhat comparable to domain specific algorithms in its relation to practice, and should be regarded as a useful technique best pursued in collaboration with those mathematicians who can make use of it. However the latter is at odds with mathematical practice: many mathematicians view formal proof development unsympathetically and it is hard to see it being incorporated except in expensive reworkings of well-understood material.

In the light of the previous analysis we identify three main objectives for computer aided reasoning: enhancing the techniques available for mathematical experimentation, developing community standards for experiment and modelling and developing methods which will make computation more acceptable as part of a proof. We discuss three areas of research which address these: the use of theorem proving techniques to enhance or extend mathematical software systems, support for formal methods techniques to increase the reliability of such systems, and the use of computer aided formal reasoning in support of mathematical practice. This last includes activities such as formalising systems for computational mathematics or visualisation so that they can still be used informally but generate a formal development, and developing techniques to provide assistance in the initial stages of developing a new theory.

By mathematics here we mean the activities of working research mathematicians, producing new results in pure or applied mathematics, although we touch briefly on some questions concerning the applications of computational mathematics and simulation in research science and engineering. We have left out several other related areas entirely: logical questions of decidability, sound-

 $\mathbf{2}$ 

ness or completeness, theoretical computer science issues of semantics, computability or complexity, foundational issues such as constructivity, computer aided proofs about software and hardware, and the use of computers in mathematical education at all levels and in heuristic discovery. In particular foundational questions about computation have transformed mathematical logic, computation has made constructive proof feasible, and effective notions of practice for proofs about hardware and software are by no means well understood. However these matters fall outside the scope of this paper.

# 1 Some twentieth century mathematical achievements

To set the scene we provide the roughest of outlines of some recent mathematical achievements. Research mathematics is difficult and requires specialised knowledge: a precise statement of most of the theorems indicated below requires concepts treated at the graduate level of mathematical education: a full understanding of the proofs is probably restricted, when they first appear, to a handful of experts.

#### 1.1 Geometry, number theory and Fermat's theorem

Fermat conjectured in 1637 that the equation

$$x^n + y^n = z^n$$

has no positive integer solution if n is an integer greater than 2, famously suggesting that he had a proof which the margin was too small to contain. In the nineteenth century mathematicians such as Hilbert developed number theory, algebraic geometry, Galois theory and the theory of ideals, investigating equations (i.e. curves and surfaces) in zero and non-zero characteristic by investigating the algebraic structure and symmetries of the fields generated by their solutions.

Twentieth century mathematics, particularly the development by Grothendieck and his school of commutative algebra and cohomology, provided a unifying framework for topology, differential geometry and algebraic geometry, replacing computations in specific curves or surfaces with very general homological and categorical machinery. This enabled the translation of results from number theory into function fields and led to a stream of extremely significant results, such as Deligne's proof of an analogue of the Riemann Hypothesis for certain varieties, and Falting's proof of Fermat's conjecture for all but finitely many cases. Fermat's conjecture, in this terminology, concerns the Galois theory of certain elliptic curves. The Shimura-Taniyama conjecture asserted a link between modular forms and elliptic curves, and it was Wiles's proof of Fermat's theorem. See Wiles [118], and Singh [100] for an attempt at a popular account. Wiles's proof draws on a vast and continuing body of difficult mathematical material, developed over two centuries.

 $\mathbf{3}$ 

#### 1.2 The classification of finite simple groups

A set of elements closed under a binary associative operation and inversion forms a group (for example the integers modulo n under addition, or the symmetries of a cube under composition): a group with no proper quotient groups is called simple (for example the integers modulo a prime). Groups derived from symmetries and equations were studied by Lagrange and Galois: by 1960 or so mathematicians were aware of 16 infinite families of finite simple groups, and 5 isolated "sporadic" simple groups discovered by Mathieu in the last century. By 1980 a further 21 sporadic groups had been predicted and then constructed, often by using a computer to search for certain matrices over finite fields: the largest, the "monster" has around  $10^{54}$  elements. The classification theorem states that there are no other finite simple groups.

Little was known until mid-century about general finite simple groups. In 1954 Richard Brauer suggested a strategy for classifying simple groups of even order: Burnside had conjectured 50 years before that all finite non-commutative simple groups had even order. Feit and Thompson proved Burnside's conjecture in 1963 (in a 254 page paper) and in 1972 Gorenstein drew up a sixteen point plan based on Brauer's ideas to prove the classification theorem, executed by 1983 by his coworkers in around five thousand journal pages. Solomon [104] observes that the output "simply overwhelmed the digestive system of the group theory community", and he remarks that a key unpublished eight hundred page paper by Mason, in which several errors were found and corrected in 1992 "is an extreme point on the spectrum of incompletely assimilated manuscripts from the later years of the classification".

Such concerns have led to the revision project, planned to appear in twelve volumes, some of which are now complete. The classification theorem has stimulated new areas of research: perhaps one of the most intriguing, affectionately called "moonshine" by the experts, is John McKay's observation that the number 196883, an invariant of the monster is also associated to a certain modular form. The consequence of this is a connection between this extraordinary combinatorial object and deep results in number theory and quantum field theory. See Solomon's survey [104] for more details.

## 1.3 Dynamical systems

Non-linear dynamical systems, in the guise of "chaos theory" accompanied by fractal images, have become a popular metaphor, although literary theorists who equate chaos theory and post-modernism do not inspire confidence that they have entirely understood the underlying mathematical concepts.

Around one hundred years ago Poincaré proved that in two dimensions "typical" dynamical systems, for example objects in motion governed by differential equations, have "essentially" two kinds of limiting behaviour or attractors : either they stop (like rocks or footballs) or they settle down into periodic motion (like the planets or the tides). The words "typical" and "essential" here hide some elaborate topology and differential geometry. In three and higher dimensions

however a third kind of attractor, the "strange attractor" appears. This is characterised by two things: we know that the particle settles down in a certain region, generally a particularly complicated one topologically, and we cannot predict its trajectory within that region, where it is subject to "extreme sensitivity to initial condition". This phenomenon was identified by Poincaré: it means that very small changes in the initial conditions can have very large effects on the outcome, so that we would have to know the initial position of the particle exactly to predict it. The asteroid belt between Mars and Jupiter appears to exhibit such chaotic phenomena: claims about for example, the stock market, are rather harder to verify.

Despite the popular attention "chaos theory" has not affected numerical computation as much as one might suppose, except to give a greater appreciation of the difficulties of things like long range weather forecasting. One reason for this is that as yet we have nothing like a classification of strange attractors, only an ever growing catalogue of examples which it would be very hard to study without the aid of computational simulations and investigations. See Stewart [105] for more details.

## 1.4 The four colour theorem

In 1852 the British mathematician Francis Guthrie conjectured that any map could be coloured with four colours: the flaw in a proof by Kempe in 1879 remained undiscovered for 11 years and then the problem remained unsolved until Appel and Haken [2] announced their computer proof in 1976. The main idea of the proof was to identify a set of 1478 configurations and show that each these occurred in a minimal counterexample (this was a mammoth hand proof) and then to use a computer to check that in fact none of these configurations did appear in a minimal counterexample. Robertson and Seymour [92], motivated by the difficulty of understanding the hand part of the proof, have recently provided a new proof reducing the number of configurations to 633, providing a machine checkable formalisation of the hand part of the proof and a new implementation for the machine checked part. As yet no hand proof has been found, although the 5 colour case is very easy. There is an admirable account of the history of the proof and the debate it generated in Mackenzie's survey "Slaying the Kraken" [76].

This work is a little isolated from other areas of contemporary mathematics, and is more accessible to non-experts than the rest of the material we have described above. However once we start to consider not just maps, that is graphs drawn in the plane, but graphs drawn on more elaborate surfaces we come upon advanced matters in geometry and topology. For example any map on a sphere with n handles (think of a doughnut with n holes) can be coloured with  $\lfloor 7 + \sqrt{(48n + 1)} \rfloor$  colours.

# 2 What is mathematical practice?

In assessing the impact of computation on mathematical practice it is as well first to ask what mathematical practice is. After a hint at the philosophical issues we concentrate on five aspects of the question which are important for the discussion of the role of computation: proof, rigour, taste, discovery and exposition. We will identify mathematical practice with

producing conjectural mathematical knowledge by means of speculation, heuristic arguments, examples and experiments, which may then be confirmed as theorems by producing proofs in accordance with a community standard of rigour, which may be read by the community in a variety of ways.

Mathematics is about developing concepts, definitions, theories, methods and conjectures, as well as verifying some of those conjectures by proving theorems. Mathematicians' understanding of this, and the way in which they go about it, has remained remarkably unchanged over several centuries, despite debates about foundation or the advent of computation. While notions of, for example, acceptable standards of rigour, have changed over time, the general notion of mathematical practice has not. For Littlewood, a mathematician at Trinity College in Cambridge in mid-century, the Greek mathematicians "are not clever schoolboys or 'scholarship candidates', but 'Fellows of another college'." Major areas of mathematics such as topology, algebra or number theory have developed continuously and cumulatively over several centuries. Lakatos in his book Proofs and Refutations gives a detailed account of the history of Euler's theorem about polyhedra that covers a hundred and fifty years of mathematical history: an account of other fields would reveal the same pattern.

While a hundred years ago Poincaré or Hilbert could comprehend all that was significant in mathematics it is hard nowadays for someone to be an expert in more than one field. Talking about mathematicians as one community is a little like talking about Europeans or sports players as one community: mathematics is rather split into a number of disparate tight communities, so that for example an active group theorist anywhere in the world might receive almost daily email from the "group-pub" email list, while having no technical contact beyond an occasional chat in the tea-room based on common knowledge of master's level material with the analyst in the office across the hall. Indeed, it is very unusual to find a modern research paper that can be read from scratch by a research mathematician from a different field: the overhead of implicit knowledge and dependence on earlier work is just too great. Technical mathematics is extremely difficult, and little shame attaches to saying that one doesn't understand more than a fraction of the published papers even in one's own field.

# 2.1 A few philosophical pointers

What is mathematical practice? Is its purpose to prove theorems, or more generally to advance mathematical understanding or mathematical knowledge? Does

mathematical achievement consist in the proof of particular theorems, like Fermat's theorem , or rather in the development of the underlying theories, like number theory. Indeed, what is "mathematics", "mathematical understanding" or "mathematical knowledge"? A full answer to this question involves not just Frege and Gödel but also Aristotle and Kant, and awaits the resolution of several millennia of debate in the philosophy and foundations of mathematics and science.

Science provides the best accredited knowledge we have of the world, yet what is the connection of theories and models to reality? How do we account for what Wigner has called the "unreasonable effectiveness of mathematics in the natural sciences"? What is going on when we test or compare our theories and models through experiment or simulation: are we dealing with Kuhnian paradigm shifts, Popperian theory change or Mertonian social construction? Is what we are doing just empirical (it's true because it works) or cultural (it's true because we have agreed it is)? Or is there some more fundamental notion of truth involved?

Philosophies of mathematics are sometimes characterised as 'empiricism', 'formalism', 'intuitionism' and 'Platonism'. Other philosophical accounts which attempt to combine Platonism and empiricism, looking at mathematical practice as well as questions of foundation, can be found in the 'humanism' of Hersh [50] or the 'realism' of Kitcher [63]. Russell caricatured mathematics as "the science in which we never know what we are talking about or if what we are saying is true": yet a view of mathematics as only a formal game with symbols and axioms seems to ignore the notion of meaning. Simpson suggests [99] that "most mathematicians and mathematical logicians lean toward an uneasy mixture of formalism and Platonism". Some axiom systems are more interesting and useful than others: they are arrived at initially by trying to capture objects that seem to be "out there", like equations or groups or strange attractors, and seem to model some kind of reality, even if as a theory becomes more developed the guiding intuitions may become hard to pick out from the technical details.

It is probably not unfair to say that this debate, particularly those aspects concerned with epistemology (what is mathematical knowledge) rather than methodology or practice, is viewed with detachment by most mathematicians today. Typically mathematicians spend rather little time thinking about foundations, may well have read little mathematical philosophy, have no training in formal logic beyond informal accounts of the Peano Axioms, quantifiers and the work of Gödel and Turing, and be unaware of non-classical logic. This is in contrast to the early part of the century, when leading mathematicians such as Hilbert and Poincaré were actively engaged in foundational matters.

## 2.2 Proof in mathematical practice

The cartoon vision of a mathematician is of a dishevelled individual suitably armed with paper, pencils, and caffeine whose sole occupation is that of producing rigorous chains of formal deduction along the lines of Principia Mathematica. The truth is perhaps closer to the leading geometer Thurston's description of

current practice [109]: "Within any field there are certain theorems and certain techniques that are generally known and generally accepted. When you write a paper you refer to these without proof. You look at other papers in the field, and you see what facts they quote without proof, and what they cite in their bibliography. You learn from other people some idea of the proofs. Then you're free to quote the same theorem and cite the same citations. You don't necessarily have to read the full papers or books that are in your bibliography. Many of the things that are generally known are things for which there may be no known written source. As long as people in the field are comfortable that the idea works, it doesn't need to have a formal written source". While some fields may be more cavalier than others about using "ideas with no written source" as part of a definitive proof, this approach is true of many areas of mathematics.

Research papers are typically written for a community of experts, who are presumed to share the prevailing assumptions: other mathematicians then take the judgements of these experts on trust. So for example only a handful of mathematicians were in a position to immediately understand Wiles's paper. and to find the flaw in early versions. Rather than checking every line of an argument an expert "critical reader" will often accept or reject it on the basis of the general strategy of the proof and a good understanding of the domain and where potential flaws might lie. Thus a journal referee might reject a paper because the argument is "unclear", "hard to follow" or "confused" even if no error or counterexample has been found. The goal is rigorous proof, proofs are generally believed to be formalisable in principle, but the methods of checking the proof are social rather than formal. Thurston again "reliability does not come primarily from mathematicians formally checking formal arguments: it comes from mathematicians thinking carefully and critically about mathematical ideas". These considerations are part of the notion of "surveyability" in the philosophy of mathematics.

Thus "mathematical knowledge", or the things "generally known" by the experts, is felt to be not just a list of theorems and conjectures, but rather a more general understanding of mathematical structures, techniques and phenomena, experience of where errors are likely to be made in proofs and insight into a repertoire of examples which are likely sources of counter-examples to statements which seem suspicious. So statements like "Fermat's theorem is true", or "the only finite simple groups are ...", subsume a vast body of knowledge about number theory or finite groups, all of which is brought into play by experts judging a proof.

The experts may express doubts, or take some time to be convinced. I heard John Thompson point out in a seminar around 1980 that there might easily be a mistake in the classification of finite simple groups, because so many of the proofs were proofs by contradiction, and an error in the argument might also give a contradiction. Gorenstein [37] discusses at some length whether the theorem can be trusted "...it seems beyond human capacity to present a closely reasoned several hundred page argument with absolute accuracy. I am not speaking of the inevitable typographical errors, or the overall conceptual basis for the proof,

but of "local" arguments which are not quite right — a misstatement, a gap, what have you." and concludes that, while a simple group has probably not been missed because many individuals with different perspectives have been working on the problem for fifteen years, "it clearly indicates the strong need for continual re-examination of the "proofs" ... especially on that day when the final classification is announced and the exodus to more fertile lands takes place. Some of the faithful must remain behind to improve the text."

This understanding of mathematical practice is not new: consider Hume the empiricist [55] writing in 1739 "There is no ... mathematician so expert ... as to place entire confidence in his proof immediately on his discovery of it, or regard it as anything, but a mere probability. Every time he runs over his proofs his confidence encreases; but still more by the approbation of his friends; and is rais'd to its utmost perfection by the universal assent and applauses of the learned world." or Whitehead [117] in 1926 "But when a piece of mathematics has been revised, and has been before the expert world for some time, the chance of a casual error is almost negligible." or the number theorist Hardy [44], writing in 1929 "proofs are what ...I call gas, rhetorical flourishes designed to affect psychology". Discussions of program proof in the 1970s echoed this point: in a widely cited paper entitled "Social processes and proofs of programs" DeMillo, Lipton and Perlis [26] suggest that formal proof of a program is not a substitute for its being inspected by other people.

#### 2.3 Debates about rigour

What some perceive as acceptable informality others may regard as unacceptable lack of rigour. There are well-known examples of incorrect proofs (of both false and correct results), of standards of proof in a field changing, and of vigorous argument as to what is acceptable, for example in the eighteenth century over analytic (Cartesian) and synthetic (Euclidean) geometrical proofs.

A recent debate concerned speculative or heuristic arguments in mathematical physics: for example an argument might assume the existence of a threshold constant or scaling factor which has been identified in computations but not proved to exist analytically: a proof of its existence is likely to be extremely hard. Jaffe and Quinn [57] suggested that such "speculative mathematics" be identified as such and due credit be given to those who eventually produced a rigorous proof. In response several of the world's most renowned mathematicians produced justifications of current mathematical practice, stressing the value of speculative or heuristic investigations in leading to significant advances: for example the eminent topologist Atiyah [3], commenting on the interactions between geometry and physics

"My fundamental objection is that Jaffe and Quinn present a sanitized view of mathematics which condemns the subject to an arthritic old age. They see an inexorable increase in standards of rigour and are embarrassed by earlier periods of sloppy reasoning. But if mathematics is to rejuvenate itself and break exciting new ground it will have to allow

for the exploration of new ideas and techniques which, in their creative phase, are likely to be as dubious as in some of the great eras of the past. Perhaps we now have high standards of proof to aim at but, in the early stages of new developments, we must be prepared to act in more buccaneering style.

The history of mathematics is full of instances of happy inspiration triumphing over a lack of rigour. Euler's use of wildly divergent series or Ramanujan's insights are among the more obvious, and mathematics would have been poorer if the Jaffe-Quinn view had prevailed at the time. The marvelous formulae emerging at present from heuristic physical arguments are the modern counterparts of Euler and Ramanujan, and they should be accepted in the same spirit of gratitude tempered with caution....

What is unusual about the current interaction is that it involves frontline ideas both in theoretical physics and in geometry. This greatly increases its interest to both parties, but Jaffe-Quinn want to emphasize the dangers. They point out that geometers are inexperienced in dealing with physicists and are perhaps being led astray. I think most geometers find this attitude a little patronizing: we feel we are perfectly capable of defending our virtue.

What we are now witnessing on the geometry/physics frontier is, in my opinion, one of the most refreshing events in the mathematics of the 20th century. The ramifications are vast and the ultimate nature and scope of what is being developed can barely be glimpsed. It might well come to dominate the mathematics of the 21st century. No wonder the younger generation is being attracted, but Jaffe and Quinn are right to issue warning signs to potential students. For those who are looking for a solid, safe PhD thesis, this field is hazardous, but for those who want excitement and action it must be irresistible."

Such concerns about an increase in rigour are also not new: Weyl [116] commented on the enervating effects of the formalist-intuitionist controversy of the early part of the century "It has directed my interests to fields I considered relatively "safe" and has been a constant drain on the enthusiasm and determination with which I pursued my research work.": criticising formalism in the sense of Carnap's program the philosopher of mathematics Lakatos observed [69] "On those terms Newton had to wait four centuries until Peano, Russell, and Quine helped him into heaven by formalising the calculus." Jaffe and Quinn, summarising the responses to their article, commented that "mathematics that has been successfully rigorised is dead, and the real life is . . . in speculation."

#### 2.4 Mathematical discovery and research

The mathematician Hadamard [43] observed in his book "The Psychology of Mathematical Invention" that practically all of the scientists he contacted "avoid

not only the use of mental words, but also ... the mental use of algebraic or precise signs ... they use vague images". (He also questioned them on the effects of the weather and frequent baths). Poincaré [88] is one of many mathematicians who have remarked on the importance of unconscious thought and sudden illumination in mathematical discovery. Wiles [42] described the process as follows "I start trying to find patterns. So I'm doing calculations which try to explain some little piece of mathematics. I'm trying to fit it in with some previous broad conceptual understanding of some branch of mathematics. Sometimes that'll involve going and looking up in a book to see how it's done there; sometimes it's a question of modifying things a bit, sometimes doing a little extra calculation; and sometimes you realize that nothing that's ever been done before is any use at all and you just have to find something that is completely new, and it's a mystery where it comes from. ... I decided that I really only had time for my problem and my family. When I was concentrating very hard then I found that young children provide the best possible way to relax. Talking to young children who simply aren't interested in Fermat, at this age; they want to hear a children's story, and they're not going to let you do anything else."

The practice of mathematical research is generally taken to be empirical, though personal styles may differ: "For Maclane it meant getting and understanding the needed definitions, working with them to see what could be calculated and what might be true to finally come up with new structure theorems. For Atiyah it meant thinking hard about a somewhat vague and uncertain situation, trying to guess what might be true and only then finally reaching definitions and the definitive theorems and proofs." [3]. Maclane talks of intuition, trial, error, speculation, conjecture and proof, Lakatos of the logic of conjectures and refutations, Polya of formulating a conjecture but needing to give more precise meanings to the terms to render it strictly correct.

In 'Proofs and Refutations' Lakatos provided a lengthy rational reconstruction of the historical development of the proof of Euler's theorem that for all polyhedra V - E + F = 2, where V, E, F are the numbers of vertices, edges and faces respectively. The theorem was stated by Euler in 1750, but an acceptable definition of polyhedron was only arrived at by Poincaré in 1895. As well as trying to prove conjectures Lakatos advocates attempting refutation by considering possible local and global counter-examples which if found can be used to guide modifications: the technique of Thurston's critical reader.

While one may not share his philosophical assumptions Lakatos's work is an admirable account of mathematical practice: the process of individuals or mathematical communities playing with definitions, examples and heuristic arguments to work out what might be true, of getting stuck, of identifying the key example or lemma, and working out an outline of the most significant part of a proof before going back and tidying up the details. In comparison very little time is spent actually writing out proofs: just as a good programmer probably spends very little time actually writing lines of code.

As with Euler's result the statements of theorems, arrived at by speculation, heuristics, or generalisation from examples, are often regarded as a greater

achievement than the proofs themselves: Lakatos [69] p.9 quotes Gauss "I have had my results for a long time, but I do not yet know how to arrive at them" and Riemann "If only I had the theorems! Then I should find the proofs easily enough".

## 2.5 Mathematical taste and style

Community standards of rigour determine what is acceptable in a proof. Community standards, often expressed in terms of taste or style, also affect what is found significant, and are used as a justification of the direction which academic leaders choose to take. The things "generally known" in a field will typically include heuristic arguments as to why a conjecture, sub-area, theorem or proof is "worthwhile", "deep", "challenging", "hard", "uninteresting" or "routine".

Hardy [44], writing in 1940, argued that "there is no place in the world for ugly mathematics" and claimed that the mathematical aesthetic was not confined to a few eccentrics but shared by anyone who played chess. Hardy combines "beauty" with "seriousness" and "depth", which he claims distinguishes mathematical theories from chess problems and make mathematics worthy of the attention of "first-rate minds". Fermat's theorem or the classification of simple groups would be "deep" because they relate to structures with an architectural quality and have many ramifications and connections with other deep structures: the four colour theorem as it stands lacks this architectural quality and is "shallow", although opinions might change if, for example, it could be connected to the theory of manifolds. Mathematicians sometimes draw on musical metaphors to try and convey this notion: it corresponds to the difference between Beethoven and lesser composers. The analyst Dieudonné [23], a member of the Bourbaki group, wrote a book called "Mathematics, the music of reason" in which he classifies mathematical problems as untreatable, where no progress has ever been made, sterile, such as the four-colour problem and many problems of elementary geometry, whose solution has not led to new developments, and prolific. Deep problems such as Fermat's theorem, or the Poincaré conjectures, are the prolific problems which continue to produce new, unexpected results and connections, like McKay's identification of the "moonshine" number.

Good theorems should have good proofs. To paraphrase von Neumann [112] a "good" theorem should not just consist of an enumeration of special cases but have some unifying element: a "good" proof will be elegant rather than a rote computation (by hand or machine) and give some sense of why a result is true, and the whole will exhibit some overall architectural structure which reduces complexities to simple guiding notions. In particular a proof with these qualities is more likely to be readily surveyable by the critical reader.

Notice that these notions of "good proof" are independent of, and may be at odds with, the ideas of reductive proof theory, investigated by Feferman, [33] who conjectured that his system W suffices for all of scientifically applicable mathematics, or Simpson [99] who argues for the use of finitism for a partial realisation of Hilbert's program. A "good" proof in the above sense may well

involve abstraction from the original problem, obtaining the result as a consequence of a more general theory, or structuring the argument using notions which are higher order or more abstract, to make it more surveyable than a different proof using more restricted foundational notions. Abstraction in itself is not the goal: for Whitehead [117] "it is the large generalisation, limited by a happy particularity, which is the fruitful conception."

As an example consider the theorem in ring theory, which states that if R is a ring, f(x) is a polynomial over R and f(r) = 0 for every element of r of R then R is commutative. Special cases of this, for example f(x) is  $x^2 - x$  or  $x^3 - x$ , can be given a first order proof in a few lines of symbol manipulation. The usual proof of the general result [20] (which takes a semester's postgraduate course to develop from scratch) is a corollary of other results: we prove that rings satisfying the condition are semi-simple artinian, apply a theorem which shows that all such rings are matrix rings over division rings, and eventually obtain the result by showing that all finite division rings are fields, and hence commutative. This displays von Neumann's architectural qualities: it is "deep" in a way in which the symbol manipulation is not.

As a more approachable example consider the proof of

$$1 + 2 + \ldots + (n - 1) + n = n(n + 1)/2$$

This has a routine induction proof. It can also be proved by the "trick" of rearranging

$$2 * (1 + 2 + \ldots + (n - 1) + n) = (1 + 2 + \ldots + n) + (n + (n - 1) + \ldots + 2 + 1) = (1 + n) + (2 + n - 1) + \ldots + ((n - 1) + 2) + (n + 1) = n(n + 1)$$

Done formally this still involves induction: the informal argument hides the induction in the "obviously true" rearrangement, leading to an apparently more "intuitive" proof of the result. But the difference is that this informal argument not only works out the answer, rather than requiring us to know it before we start, but can also be said to explain why the answer is true. Similar examples are used in developing an aesthetic and a repertoire of clever techniques in students of mathematics: for example in the text Concrete Mathematics [38] Graham, Knuth and Patashnik outline 8 ways of working out sums of the form

$$f(1) + \ldots + f(n),$$

only one of which makes the induction explicit.

The eighth technique is Gosper's algorithm, which requires no cleverness, just accuracy, and will find closed form sums completely automatically for extremely complicated terms: for example given

$$\sum_{j=0}^{j=n} 2^{n-k-2j} \binom{n}{j} \binom{n-j}{j+k}$$

the algorithm returns

$$\binom{2n}{n+k}$$

In which case, the exasperated student might ask, why does one need the other seven techniques? The reason is not just to keep students busy: the notion of "good" proof will depend on the context. It may be the case that the exact value of the sum doesn't matter to us, if we are only using it to get an upper bound on some quantity for example, or that finding this sum is a fairly incidental piece of some much larger argument (a page in the 5000 or so about simple groups), in which case application of routine technique is probably preferable to the reader. On the other hand if (hypothetically) the series were to turn out to be the key point in a new proof of the four colour theorem then a proof which explained why it was true, somehow linking the terms in the series to colourings of graphs, would, while not being "more correct" certainly be "better" in the sense above. Our first example of a series would be a routine computation in support of a routine result, our second a routine computation in support of a more significant result. There are of course infinitely many series summable in this routine way by Gosper's algorithm: it is the context which determines whether the theorem thus proved is significant or not.

The choice of proof in, for example, a text-book exposition may bring other factors into play such as the overall structure of the work or the expected background of the reader. For example in presenting linear algebra it may be preferable for expository purposes to develop the explicit definition of a determinant as a polynomial in the entries of a matrix rather than the implicit version which associates an alternating bilinear form to a linear transformation. The former would be judged by Hardy's standards as "ugly" (and constructive), the latter as a "better" (and non-constructive) exposition since it abstracts away from the explicit formula for the determinant and provides a coordinate-free explanation as to why determinants behave as they do. However experienced teachers know that the former is comprehensible by the average student whereas the latter is considerably more demanding! Lang [71] compares the expository style of a student text and a more advanced work by saying "The purpose of the latter is to jazz things up as much as possible. The purpose of the former is to educate someone in the first steps which might eventually culminate in his knowing the jazz too, if his tastes allow him that path."

The early proofs of a result may be superseded by later "better" proofs in this sense: an "ugly" valid proof will be accepted while recognising that the search for a "better" one is worthwhile. Sometimes the "better" proof will be completely different: the result might be shown to be a consequence of something far more general which explains why it is true. For example it is conceivable that the four colour theorem could turn out to be a consequence of a much more general result about the embeddability of graphs on manifolds. Sometimes the "better" proof will be broadly similar but manifest more insight, as in the reworking of the classification of finite simple groups currently under way [104].

While a mathematical aesthetic, rather than notions of applicability or usefulness, has often been used to guide what areas are worthy of future investigation,

there has frequently been vigorous debate as to what does constitute "deep" or "beautiful" mathematics. One might interpret eighteenth century arguments about replacing geometrical arguments by algebraic manipulation as arguments about style rather than acceptable rigour. In this reading Newton and Leibniz argued that geometry provided insight and compactness of representation while algebra provided quantities of output but obscured the sources of discovery, others that algebra provided greater power and economy of thought [8].

Similar arguments surrounded twentieth century developments in number theory and geometry, as described by Lang [71]. An integral part in this development was played by Lang's book Diophantine Geometry, published in 1962, which gave an exposition of earlier work of Mordell, Siegel and others within the new framework of Grothendick's theory, which by then occupied about 10,000 published pages. The older Mordell in a savage review lamented what he regarded as the quite unnecessary generalisation and abstraction of Lang's approach "The reviewer was reminded of Rip van Winkle . . . who woke up to a state of affairs and a civilisation completely different from that to which he had been accustomed." In a widely read letter to Mordell Siegel remarked

Thank you for the copy of your review of Lang's book. When I first saw this book, about a year ago, I was disgusted with the way in which my own contributions to the subject had been disfigured and made unintelligible. My feeling is very well expressed when you mention Rip van Winkle!

The whole style of the author contradicts the sense for simplicity and honesty which we admire in the works of the masters in number theory-Lagrange, Gauss, or on a smaller scale, Hardy, Landau. Just now Lang has published another book on algebraic numbers which, in my opinion, is still worse than the former one. I see a pig broken into a beautiful garden and rooting up all flowers and trees. ...

I am afraid that mathematics will perish before the end of this century if the present trend for senseless abstraction-as I call it: theory of the empty set-cannot be blocked up. Let us hope that your review may be helpful...".

For Siegel the flowers and trees were evidence of prolific conjectures, whereas the abstraction rendered them sterile. Siegel's concerns were unfounded: these are the techniques that led to enormous developments in number theory including Fermat's theorem, and Lang observes that while Mordell and Siegel were great mathematicians their lack of understanding obstructed the development of mathematics in their own countries.

Further investigation of mathematical aesthetics again leads us to philosophical considerations. Von Neumann draws attention to the dangers of degeneracy, of the "classical" turning into the "baroque" if a branch of mathematics becomes governed entirely by an internal aesthetic without reference to an empirical source. Others have been rather more harsh about the inward looking tendencies of mathematicians: the computer scientist Dijkstra [24] observes "informality is the hallmark of the Mathematical Guild, whose members - like poor programmers - derive their intellectual excitement from not quite knowing what they are doing and prefer to be thrilled by the marvel of the human mind (in particular their own). For them, the dream of Leibniz is a nightmare"

For some mathematicians the progress of mathematics in the light of such an aesthetic is regarded as "natural" or "inevitable": others find this more problematic. One might analyse the mathematical aesthetic culturally in terms of the power structures and negotiation of boundaries underlying bland terms like "community culture". One might also take into account the personal psychology of research mathematicians: an extraordinary collection of individuals, who sometimes, like climbers, seem to choose their goals for the difficulties and challenges they offer, which may be unrelated to the flowers growing on the scree or the view obtained from the top.

## 2.6 Mathematical exposition

We have described usual mathematical practice but mathematical exposition is generally very different.

Mathematicians write and read a variety of texts. Working notes while thinking about a problem or conjecture may consist of a mixture of partially worked out ideas, conjectures, definitions that weren't quite right, proofs abandoned because the conjecture turned out to be wrong or the author couldn't see how to do the crucial step, blind alleys and illustrative examples and counter-examples recorded in a form comprehensible to the author alone. These may be "written down" to form a more permanent record, including the examples and blind alleys (in an attempt to avoid going down them again). The author may present the results as a seminar or colloquium, or publish a research announcement or working paper on the internet. At a suitable point (maybe the definitive result has been proved, or maybe the author is reporting progress to date in response to professional pressures or loss of time or enthusiasm) the work may be "written up" in a form suitable for research journal publication. The working notes will be reworked, probably abbreviated for reasons of length or style, to produce a chain of argument of acceptable rigour from the hypotheses to the conclusion shorn of blind alleys, motivation, examples or counter-examples. If the work has attracted sufficient attention it may be reworked later with other results in the field as part of a survey article or a text book giving more space to motivation and background.

By the time the work reaches a wider audience of text-book readers or undergraduates the experiments, false starts, and guesses will probably have disappeared, giving the impression that the textbook contains a historical account of what the mathematician did to prove the theorem. Lakatos's presentation in "Proofs and Refutations" is much closer to what actually happens, and he argues for more widespread use of his technique of "heuristic exposition" instead of this deductivist style of "authoritarian mysticism" which presents only the completed argument: Thurston [109] makes a similar suggestion.

Just as there is a variety of mathematical writings there is a variety of mathematical readers (I have not been able to find a source for the statement that a mathematical paper is read on average by 3 people). For research papers these might be close associates reading a draft of the work, or Thurston's "critical reader" (who might be a referee or reviewer) assessing the proof for correctness, or a mathematician who is prepared to take the proof on trust but is reading it to get an idea of the techniques used, or is only interested in the statements of the theorems or the "big ideas", or a casual reader flipping the pages in the library, or a beginner wrestling with the work as a rite of passage. For seminars the audience might be more eclectic: following closely or somewhat bemused.

Current sociological studies consider scientific texts and their role in the "manufacture of knowledge". A full account would consider not just what I have outlined above but referees reports, papers that weren't accepted, proofs that were discredited, the reception in Mathematical Reviews or other commentaries of a significant result: even grant proposals, letters and, today, email messages.

Techniques of literary theory applied to mathematics are more contentious: for example Rotman [94] claims "Mathematics ... is a process: an ongoing, openended, highly controlled, and specific form of written intersubjectivity." and "Mathematical reasoning is thus an irreducibly tripartite activity in which the Person (Dreamer awake) observes the Subject (Dreamer) imagining a proxy-the Agent (Imago)-of him/herself, and, on the basis of the likeness between Subject and Agent, comes to be persuaded that what the Agent experiences is what the Subject would experience were he or she to carry out the unidealized versions of the activities in question." Mathematicians who are tempted to give simplistic or ironic accounts of their discipline may find their words appear to support such arguments, however unsympathetic they find them, rather more readily than they support traditional philosophies of mathematics.

# 3 Computation for mathematical research

In this section we identify some themes in the use of computers in mathematical practice. We describe below utilities, numerical methods, symbolic computation, computer aided reasoning and the use of computation in constructions, in proofs such as the four colour theorem and in support of speculative reasoning in geometry and combinatorics.

## 3.1 Utilities

One might start, prosaically, by pointing out that the greatest impact of computation on the day to day work of the mathematician has been in tools for day to day mathematical processes:

- mathematical type setting with associated standards for input and output to other software such as computer algebra, graphics and statistical systems
- internet services such as email and newsgroups, giving immediate access to the community and exchange of technical information. For example today nearly every mathematical community will have an electronic mailing list, enabling rapid circulation of new results and response to queries.
  - 17

- services such as Mathematical Reviews on line through MathSciNet (bloated bibliographies a breeze!), citation indices, electronic journals, the centralised mathematics pre-print server [15] and repositories of papers, often searchable and with downloadable output in a variety of formats
- electronic sources of data for example
  - Sloane's Sequence Seeker [103]: given a positive integer sequence this will apply a collection of transformations and attempt to suggest a generating function
  - Fateman's web database of integrals [31], which uses a sophisticated table look-up to account for degenerate and special cases
  - the Atlas [22] containing large datasets related to the finite simple groups

# 3.2 Numerical methods

The most widespread computational techniques are those of numerical computation, mathematical modelling and simulation. Numerical methods have been part of applied mathematics and the physical sciences for the past fifty years, widely available through standard libraries such as NAG [82] and providing the basis for large software systems, usually written in FORTRAN or C and used in chemical, physical or astronomical research as well as in practical fields like engineering, meteorology and aeronautics and increasingly today in visualisation and animation. While one can find horror stories of catastrophic bugs in numeric code (for example the military plane which turned over as it crossed the equator) matters such as testing, error analysis, and simulation are on the whole reliable: in part perhaps because such systems are largely developed by scientists with a good understanding of a mature body of technical material.

Sometimes such software might be an implementation of a theory about physical phenomena, with the results it produces from given inputs being tested against experiment or measurement. Today measurement itself is computer mediated, though it was not always so: Michelson and Morley had no computers. With theory testing as with program testing Dijkstra's maxim [24] that "Testing can prove the presence of bugs but never their absence" holds true: such tests provide in a Popperian sense possible falsifications of the theory.

In other contexts software might be an implementation of a generally accepted theory and the output of the program regarded as a prediction, with estimates of error being provided by mathematical analysis in the light of the theory and the reliability of the data. In some cases predictions be easy to check: like times of tides, where the mathematics is such as to make reliable prediction possible, or weather, where the mathematics of dynamical systems theory tells us reliable prediction is more difficult. However in other cases it may be hard or impossible to check the predictions: for example safety thresholds for aircraft loads or discharge of pollutants.

If trusted enough, the software may be relied upon as an implementation of the theory that is used in turn to stand for the physical reality in further experiments or simulations, and thus itself to suggest possible further conjectures or theories. A sequence of numerical results or a visualisation of the phase space

of an equation may suggest a closed form solution or a new chaotic phenomenon for example. The investigation of chaotic phenomena in the asteroid belt was carried out using, not astronomical observation, but the "Digital Orrery", a custom built parallel computer which simulated 200 million years of the motion of the solar system [105].

This is a simplified account and there are many additional factors. We may be testing rival theories which are not immediately comparable in the Popperian sense using the quality of their predictions. The same software may be used in one context to test a novel theory and in another as the standard implementation of an accepted theory. It may be too simplistic to assume the software implements one of our best guesses at a theory, as to test one part of a theory it may be necessary to make wildly simplifying assumptions about another: a program may combine quantum and classical mechanics for example.

All the usual concerns about software correctness may be raised for such systems. How do we know the software is a correct implementation of the underlying theory? Which aspects of its behaviour are artefacts of the implementation (for example random number generation) rather than consequences of the theory? What hidden or explicit assumptions have been made at different stages of the processes above and how do they affect the uses to which the system has been put: for example, if the system is used in a new application and predicts that x > 3, is this a consequence of the theory, or of some implementation decision being called upon outside its domain of validity? A particular issue in numerical work is correctness of floating point implementation and convergence criteria: is the implementation robust enough to produce the same answer again for the same inputs? To make matters more complicated, often much of the software comprises large legacy systems where the underlying assumptions may have varied over time, or where later implementors may not have fully understood the original assumptions, or have incorporated variations based on new results. For example the consistent handling of floating point arithmetic or the translation between machines with different word-lengths are recurring legacy problems.

Additional questions arise if the output is used as input to further systems, for example visualisation, CAD/CAM or real-time. What does "correctness" mean in the context of a surgeon using a data glove and a CAT-scan to control a laser in an operation on a patient in a different time-zone? Predictions may be used in legal or governmental processes where issues are not as clearly understood as they might be: the epistemological issues surrounding "properties of a model" and "reality" for example. An excellent account from the point of view of environmental predictions is given in Oreskes [85]. Many of the issues involving the nature of computational models of reality are similar to those raised in the debate following Fetzer's [34] notorious attack on program verification.

#### 3.3 Symbolic computation

General purpose computer algebra systems (CAS), such as AXIOM [59], Maple [49], or Mathematica [113], as well as more specialised tools such as GAP [96]

for computational discrete mathematics or the AXIOM/PoSSo library for highperformance polynomial system solving, are used by many different communities of users including educators, engineers, and researchers in both science and mathematics. The specialised systems in particular are extremely powerful. The PoSSo library has been used to compute a single Gröbner basis which (compressed) occupies more than 5GB of disk space, while GAP is routinely used to compute with groups of permutations on millions of points.

After pioneering work in the 1960s CAS have become mainstream commercial products: everyday tools not only for researchers but also for engineers and scientists: for example Aerospatiale [91] use a Maple-based system for motion planning in satellite control. The systems have become more complicated, providing languages, graphics, programming environments and diverse sophisticated algorithms for integration, factorisation and so on, to meet the needs of a variety of users, many not expert in mathematics. All the usual software engineering issues arise, such as modularity, re-use, interworking and HCI. For example NAG's AXIOM [59] is a strongly typed CAS: user and system libraries are written in the Aldor language which supports a hierarchy of built-in parameterised types and algorithms for mathematical objects such as rings, fields and polynomials. Aldor is interpreted in the AXIOM kernel which provides basic routines such as simplification and evaluation: code developed in Aldor may be compiled to C for export to other products.

Such systems are widely used in mathematical research particularly for heuristic investigations and trusted implementations of standard algorithms. For example the GAP [96] system, developed almost entirely by researchers in discrete mathematics, contains a large library of built in examples and implementations of procedures for computing the structure of given groups. Old papers involving lengthy hand calculations are confirmed in a few seconds by GAP: for example the nilpotency class of a topological group. GAP computations of this kind are accepted in research papers: but such computations might not feature in the final deductive presentation of a result because they are investigations which form part of the proof development but not of the final proof. A computation may also suggest a more general proof: the author was told of a long GAP computation involving the number 7: on inspection it turned out that replacing 7 by p throughout produced an argument valid for all primes p which was then published: however GAP itself cannot handle arbitrary primes in this way, and was not mentioned in the final publication.

CAS have been used similarly in mathematical research: both for routine computations such as simplifying or factorisation accepted as part of a proof, and for preliminary investigations and formulating conjectures. We cite for example

- Labelle's [68] investigation of new power series identities and asymptotic estimates, of the kind that might be used in complexity investigations, subsequently proved "by hand": the paper is part of a volume of the Journal of Symbolic Computation devoted to these methods.
- Brown's [11] computation of homotopy groups in AXIOM, which automates complex computations in non-commutative algebra which it would be very

time-consuming and difficult to do by hand.

- Gosper's algorithm, and a number of variants due to Zeilberger and others [120], have been implemented in Maple. They allow the automatic identification and verification of closed form sums for a certain class of expressions involving binomial and hypergeometric identities, and work by producing a "certificate" which represents each summand using a recurrence, so that when the summands are summed the sum collapses to a closed form, or showing that no such certificate exists. Thus they transform the "cleverness" typically required to solve identities like

$$\binom{2n}{n+k} = \sum_{j=0}^{j=n} 2^{n-k-2j} \binom{n}{j} \binom{n-j}{j+k}$$

to a completely routine procedure.

As before concerns about correctness, floating point and reproducibility arise. CAS can contain bugs: for example Clarke [17] notes that Mathematica handles  $0^0$  incorrectly.

However there are many more subtle problems which can be read as "features" rather than "bugs" but none-the-less cause confusion for the unwary. Fateman [31] has a survey. Some CAS are "cautious" only giving an answer when certain pre-conditions are satisfied, others attempt to return an answer whenever they can. Furthermore some of the problems are a consequence of implementation decisions driven by different user expectations of CAS. Some users see them as "intelligent paper" doing purely formal manipulation of expressions, and take upon themselves the responsibility of ensuring that those manipulations are appropriate to the underlying problem: ignoring side-conditions in a speculative investigation for example. Others wish to clearly express the assumptions of their work, and then expect the CAS to use only appropriate manipulations. Still others, naively, may not be aware that there is a problem, and dangerously assume that a system acting in the first mode is acting in the second. These concerns mirror the debates about rigour in mathematical practice described above. To give some examples:

- There is no satisfactory canonical representation of elementary functions over the reals, and the *ad hoc* methods used by some systems can mean that expressions like atan(x-b) + atan(1/(x-b)), which is undefined at *b* and  $sgn(x-b)\pi/2$  otherwise, are not handled correctly.
- Type systems are not unproblematic: for example the consistent handling of coercions such as (2x+2)/2 1. AXIOM is strongly typed: it treats FLOAT as a ring although it is obviously not one.
- Side conditions cause problems. A CAS may be able to compute an answer on a large class of inputs, be sound on only a subclass of those inputs and be able to check soundness easily on a smaller subclass still. For example consider a deeply nested division by an expression (x-3). What value should the CAS return, what explanation might it provide to the user of what it has done
  - 21

and what warnings should it flag? Most CAS ignore such side-conditions, leading to potentially incorrect output.

- The use of parameters adds to these difficulties: for example the matrix

$$\left(\begin{array}{cc}2&2a\\(a+1)&2a\end{array}\right)$$

over a field F has rank 0 if F has characteristic 2 and a = 1, rank 2 if F does not have characteristic 2 and  $a \neq 0, 1$  and rank 1 otherwise.

 More generally many computer algebra algorithms are only valid if preconditions are satisfied which it is generally not possible to check at run-time: if the function is continuous for example. Again users may be misled if they apply procedures inappropriately.

We should not conclude from the above that all computer algebra systems are problematic all of the time: there are many areas, especially where the machines are doing straightforward symbol manipulation or relying on efficient implementations of well-understood algorithms, where matters are fairly unproblematic and experts use the systems with confidence.

#### 3.4 Computation and mathematical experiment

We group together in this section three phenomena involving numerical computation in support of mathematical endeavour: visualisation, computation as an aid to investigation and computations forming part of a proof.

The Mandelbrot set is one of the best known computational visualisations: and although Poincaré predicted chaotic phenomena more than a century ago modern dynamical systems theory has only been developed in the past thirty years or so. It is tempting to suppose that it would not have been possible without the use of computer simulations and visualisations to investigate chaotic phenomena such as strange attractors. Since such phenomena are a manifestation of extreme sensitivity to initial conditions, great care is needed to ensure that they are "really there" and not subject to implementation bias. In fact Ruelle [95] suggests that there were other barriers: the necessary pure mathematical tools of ergodic theory were not yet available to investigate extreme sensitivity, and developments in quantum theory meant that researchers looked there rather than to classical mathematics for explanations of chance and randomness.

Modern geometry has benefited from the ability to visualise unusual surfaces whose properties can then be investigated analytically: for example Hoffman [52] discovered a new surface this way in a class thought to have been well understood since the eighteenth century and Thurston's famous video 'Outside In' [73] demonstrates how a sphere can be turned inside out. Such programs are generally a combination of computational geometry, symbolic and numeric computation, in some cases essentially performing geometric reasoning calculations with variants of Gröbner basis algorithms.

Computation has long been used to explore ideas or to gather support for conjectures, or find counterexamples. The purpose may be to aid in developing a theory, or to convince a mathematician of the truth of a result before time is invested in a hand proof. For example Wiles's proof of Fermat's theorem appears not to involve computation at all. However a related and still unproved conjecture due to Birch and Swinnerton-Dyer [108] was subject to much computational experiment in the 1970s, giving mathematicians more confidence that was worth pursuing the lines of enquiry that eventually led to Wiles's proof.

Recent work on threshold functions in satisfiability problems [97] is a good example of computational experiment in support of a conjecture. Numerous simulations have identified an apparent threshold in the likelihood of satisfiability of a problem expressed in clausal form with fixed clause length as the ratio of clauses to variables varies, and similar techniques have been applied to many related problems. It seems probable that it would be very hard to prove this analytically: similar problems in percolation theory have involved the development of an enormous amount of probability theory just to prove that the function being analysed is defined and continuous at the appropriate point [39]. However all the same questions arise as in the previous section: how can we trust the implementation, how can we be sure that what we observe is a consequence of the theory rather than of bias in the implementation, how do we know our statistical analyses are correct (see Slaney [102] for a discussion of this) and how far are we justified in using the results of such computations in further work: we might for instance want to use the information about the threshold function in further algorithm design.

Computers can be used to construct discrete mathematical objects: all but the smallest of the 26 sporadic finite simple groups were first conjectured, then constructed computationally by Conway and others [22] using a combination of brute force search and clever ideas to find certain matrices over finite fields. Brendan McKay has investigated various Ramsey numbers [81]: the Ramsey number R(n, k) is the smallest number of vertices a graph must have to ensure that either it contains a complete subgraph on n vertices or its complement contains a complete subgraph on k vertices. It is useful to note the difference between establishing an upper and a lower bound for a Ramsey number. The former involves intense computational effort to find a graph containing a certain configuration of edges and non-edges which can be checked very easily once found, and the latter involves equally intense computational effort to show that no suitable graph exists. In the former case the final proof, which consists of exhibiting a graph with certain properties which can be readily checked by hand, can be presented independently from the computation: in the latter case one can regard the computation as a proof or as evidence for the truth of the results and the existence of a non-computer proof. Similar remarks apply to Lam's [70] demonstration by exhaustive search that there no projective planes of order 10.

The proof of the four colour theorem [2] by Appel and Haken in 1977 was the earliest and most famous use of a computer in a proof: scrutinised particularly because of the long history of failed proofs of the result and because it consisted of a large case analysis which was hard to check and gave little insight. We return to this below.

# 4 The effect of computation on mathematical practice

Thus far, ignoring epistemological difficulties, debates on foundational matters and the temptations of an ironic post-modern account of "mathematics", we have identified mathematical practice with the business of producing conjectural mathematical knowledge by means of speculation, heuristic arguments, examples and experiments, which may then be confirmed as theorems by producing proofs in accordance with a community standard of rigour, which may be read by the community in a variety of ways, and we have noted that while notions of rigour and so on may have changed practice itself has remained remarkably robust over the centuries.

As we have seen computation is used both in speculative activities and, in the form of standard algorithms (either stand alone or in systems such as Maple) or exhaustive search, as part of proofs. Most of the work we describe has involved substantial software development over a long period with attention to matters at the forefront of computer science such as algorithms, data structures, graphics and memory management: the problems being addressed are often at the limits of computational power. While little if any of the software involved has been formally proved correct the community is on the whole well aware of the possibility of error and much of it has been subject to intense scrutiny by other scientists who understand the underlying mathematics and are quick to point out bugs through, for example, internet newsgroups like sci.math.symbolic. Community acceptance has a lot to do with the perceived reliability of the software developers: for example GAP is trusted because it has been produced by leaders in the field who make sources freely available and adhere to strict protocols about distributing material produced by others with the GAP software. Much of the lively debate over the merits of rival CAS concerns the perceived mathematical competence and reliability of their implementors. In other words trust in software, like trust in proofs, is in part a social process.

The speculative investigation of chaotic phenomena, threshold functions, surfaces and the like, which provide evidence for the truth of theorems without being proofs, can be described as experimental mathematics, defined by Borwein [10] as "that branch of mathematics which concerns itself ultimately with the codification and transmission of results in the mathematical community through the use of experimental exploration of conjectures and more informal beliefs and a careful analysis of the data acquired in this pursuit". The new journal Experimental Mathematics encourages publication of accounts of such mathematical experiments, particularly computational ones, the editors observing in the spirit of Lakatos that "It is to our loss that most of the mathematical community are almost always unaware of how new results are discovered ... The early sharing of insights increases the possibility that they will lead to theorems", [30] while acknowledging hostility from some quarters that the journal should be renamed "the Journal of Unproved Theorems" [28]. Borwein argues that the growth of such experimentation requires also the development of community standards as to experimental methodology: matters such as correctness of the implementation, implementation bias, reproducibility, statistical and simulation concerns

and the presentation of such experimental results.

The use of computation as part of a proof also raises debate: is it correct and even if it is, is it a "good" proof. Perhaps the least problematic are computations where the results, once obtained, can be readily checked very quickly by hand or another program. Thus a proposed indefinite integral can be checked by differentiation, a proposed factorisation by multiplying the factors up again and proofs using Gosper's algorithm can be checked by hand. Producing the matrices that generated a sporadic simple group was a tremendous computational challenge at the time: a hand proof confirmed that they did indeed generate the required group.

Mackenzie's [76] study of the four colour theorem addresses the issues of correctness and quality. Those questioning the correctness of the proof of the four colour theorem raised concerns unsurprising to the automated reasoning community: the hardware or software might have an error and the proof was (in this case) hard to repeat. The computer scientist David Gries, an expert in program correctness, inspected the code looking for an error: although he thought the code was very ugly the only error he found was a "safe" one, that is the program considered an unnecessary extra case. Some easily corrected errors were found in the hand part of the proof. Those questioning the quality argued that even if printed out the proof was not surveyable: it could only be checked formally but not by Thurston's critical reader, and it gave no insight into why the result was true [111, 77]. These were countered by the argument that such criticisms could be levelled at lengthy hand proofs (such as the hand part of Appel and Haken's proof), particularly those which were only accessible to a small group of specialists [2]. Within ten years or so the debate had largely died down. The new proof by Robertson and Seymour, which automated both parts of the proof, required consideration of a smaller number of cases and, because computers had got much faster and programs more portable in the intervening years, could be re-run on a variety of machines. However as Robertson commented [76] "nobody likes very much a proof that uses a computer".

In other words a proof by a computation, whether hand or machine, may be accepted as a rigorous proof, but, if the result is a significant one, it is not a "good" proof. It may not be a good proof, but it may be a considerably more reliable one, involving much less effort for the author. Thus for a less significant result, like many of the computations done by computer algebra systems as part of a larger endeavour for example, the community seems to find no problem with admitting a computation as part of a proof provided the authors are trusted to have used the system properly. Thurston [109], who has considerable computing experience, suggests, in a marked contrast to some of the mathematical hostility to the very idea of computation, that "mathematics as we practice it is much more formally complete and precise than other sciences, but is much less formally complete and precise for its content than computer programs". Even if doubts remain, the "social effect" of a widely accepted computer proof (particularly one that is difficult or costly to replicate) inside a field is, generally, that the attention of the community shifts to other problems.

Yet if the objection to machine computations being part of a proof is that they provide no insight, what of pictures and simulations? Pictures or videos like "Outside in", produced for exposition and making no claim to be a proof, provide far greater insight than the corresponding verbal proof: verbal arguments involving stretching and cutting rubber sheets are already acceptable in certain branches of geometry and topology and it will be interesting to see if conventions evolve for making visualisations acceptable, perhaps by constructing a symbolic argument automatically in parallel with a visual argument.

Some argue that computational activity will replace traditional mathematics: Zeilberger suggests "The computer has already started doing to mathematics what the telescope and microscope did to astronomy and biology. In the future, not all mathematicians will care about absolute certainty, since there will be so many exciting new facts to discover." [120], stimulating journalistic discussion of "The Death of Proof" [54]. Others, such as Krantz [65], regard such debates as a distraction or indeed a serious threat to mathematical research, noting that funding pressures, current fads and cultural relativism combine to favour computational over theoretical work.

How do we analyse mathematical practice in the light of computation? For the many mathematicians who still use their computers only for Latex and internet access one could argue that it has hardly changed at all! The impact of computation on most of the major areas of twentieth century mathematical research has been slight (except logic, but that's another story). For example computers were used to construct some of the sporadic simple groups, but only after they had been conjectured theoretically, so that constructing them reduced to a search for a particular configuration of matrices: the vast bulk of the classification theorem is essentially a hand proof that there are no other simple groups. Computation had little impact on the vast body of work in number theory and geometry leading to Fermat's theorem, or on work in differential geometry or topology which might lead to a proof of the Poincaré conjecture or a classification of three-manifolds. Indeed, one might argue that of all the sciences pure mathematics has been affected least by twentieth century developments in computation.

The computer has become a valuable tool in certain areas, particularly for speculative work and automating routine calculations. Some areas, like dynamical systems or the study of threshold phenomena, would not have developed in the same way without computational investigation and experiment, using numerical, symbolic and graphical techniques. Certain operations, whether speculative or part of a proof, which used to involve complex hand calculations, such as manipulating group presentations, summing series amenable to Gosper's algorithm or, as we shall see, first order resolution proofs in the style of OTTER, can be routinely automated. Certain proofs like the four colour theorem or the nonexistence of projective planes of order 10 would have been impossible without a computer: the computation seems, albeit unwillingly to have been accepted.

Even in those areas where computation has been used mathematical practice has not changed, as we see by reading Mackenzie's account of the four colour

theorem: as a result of speculation and experiment mathematicians come up with conjectures and proofs of their conjectures which are then assessed by community standards of rigour. Standards of rigour for computer proofs, as with hand ones, concern community judgements of the quality and reliability of the software and the use made of it. Thus a claimed computer proof might be rejected because of lack of rigour, either because there was a bug in it or because the referees did not have sufficient confidence in the software or the account given of it.

Computer science may yet transform mathematical practice entirely, as we discuss below. However we might pick out from the above discussion three challenges from current practice:

- can computers do more for speculation, modelling and experiment?
- can computers do more for proof?
- in both cases can we increase our trust in them or make them more reliable?

# 5 Computer aided reasoning

Computer aided reasoning means the use of computers to produce formal proofs in a given logical system – a practical version of Hilbert's programme. It ranges from implementations of decision procedures, semi-decision procedures, and collections of strategies for the machine to try which will not necessarily find a proof, to programs which check line by line whether or not the input generated by the user or another program is a valid proof. It is not always clear where to draw the line between theorem proving programs and other programs that are used to generate mathematical results. A theorem proving program usually involves symbol manipulation rather than explicit calculation, allows the user to generate a full listing of all the steps needed in a proof and is concerned with logic rather than a specific application domain such as group theory, number theory or differential equations: there is some borderline activity as we shall indicate. Mackenzie [75] has given a valuable historical survey.

The rise of formalism in computer science since the mid 1960s has led to particular interest in, and development of, appropriate logics and type theories, and to many applications of reasoning in specification and verification, that is in proving that programs, distributed systems, hardware devices, or the descriptions of these in some formal system have certain properties.

This work has necessitated the formalisation of relevant parts of mathematics, such as real numbers or floating point arithmetic, but the use of computer aided reasoning techniques in mathematical research has been negligible, with a few notable exceptions that we refer to below. After describing the background we loosely categorise this activity as follows

- proof of new mathematical results using techniques such as resolution, rewriting, geometry theorem proving or constraint solving which are not that far from those domain specific techniques such as computer algebra systems
  - 27

- formalising mathematical definitions, theorems and proofs in a suitable logical system
- hybrid systems which combine computer aided reasoning with other technologies, in particular
  - embedded verification techniques, which increase the power of computer algebra systems and numerical software by the incorporation of new techniques drawn from automated reasoning, particularly for speculative or experimental work, but do not necessarily address the underlying rigour of such systems
  - formal methods for mathematical software, which increase the reliability computer algebra systems and numerical software, and help to develop community standards for modelling and experiment
  - focused formalisation techniques, which address concerns that mathematical software is unreliable and not rigorous enough to include in a proof by producing a more formal development to support informal computation, and also make formalised mathematics available to mathematicians in a way that supports current mathematical practice

For each area we describe the work that has been done, assess it against our model of mathematical practice and set out grounds for future development in the light of this.

#### 5.1 Background

W S Jevons, a Manchester economist, built a "Reasoning piano" in the 1870's which looked like an old-fashioned mechanical cash register, and could be used to determine whether a Boolean expression was a tautology by pressing down a suitable combination of keys. The machine is in the Oxford Science Museum.

With the advent of electronic computers several logicians experimented with decision procedures: Martin Davis implemented Presburger's decision procedure for additive number theory in 1954, and a few years later Hao Wang implemented a decision procedure for the ' $\forall \exists$  predicate calculus with equality' which was able to prove many such theorems from Whitehead and Russell's Principia Mathematica.

There has always been an interest in programs which produce proofs by modelling human thought processes in some way, or are claimed to do so. Newell, Shaw and Simon's 'Logic Theorist' [84] was an early (1956) program of this nature. A critique and analysis of this approach can be found in [41] but we shall not pursue it here.

Several approaches to general first order theorem proving were tried in the early years, based on various semi-decision procedures, but the problem with all of them was that of dealing with substitutions for quantified variables. An automatic procedure which enumerates all possible ground substitutions in some way will soon lead to combinatorial explosion. The resolution method of Robinson [93], gave one solution to this problem, and became the main area of research in automated theorem proving for some time. Today theorem provers based on

resolution, particularly OTTER [119] have been among the most successful in producing proofs of unsolved mathematical results. Resolution is, of course, only a semi-decision procedure and the prover is essentially engaged on a potentially non-terminating search. Much attention has been paid to efficient data structures and the constraining of search by the use of a sophisticated user-controlled weighting strategy.

Most interesting mathematics is neither first order nor semi-decidable, and an alternative approach was the development of proof checkers, such as de Bruijn's AUTOMATH [12], developed in 1967, which was used by Jutting to check the proofs from Landau's Grundlagen der Analysis. Theorem provers in the LCF tradition automated this notion, using tactics to build up proofs automatically, and the development of these ideas using various logics and type theories has led to widely used systems such as HOL[36], Isabelle [87], NuPrl [56] and Coq [6].

There has been much research also into particular aspects such as type systems, induction, rewriting, equational reasoning and matching. Hardware verification has proved a particularly important application, leading to interest in model checking techniques based on BDDs, efficient data-structures for Boolean functions. More recently systems such as PVS [86] have combined various techniques in one platform: we describe below some hybrid systems which combine theorem provers with other applications such as Maple.

For any such system to be widely used on anything but small examples it is also important that it address systems issues such as efficiency, data structures, memory management, and so on. Reasoning systems are not easy to use and often require a good deal of insight and experience from the user, whether in tuning the parameters to constrain search in an OTTER-like system or in developing exactly the right chain of lemmas in an LCF-like one. Thus attention to user interface questions and to matters such as proof management and re-use, the organisation of a database of lemmas, parameterisation of theories and the manipulation of partial or unfinished proofs is important.

And how do you trust the theorem prover? The more features are built in to help the user the more bugs are likely to creep in, but also the more such features the less amenable the prover becomes to formal verification. Pollack [89] develops the suggestions of Cohn [19] and Slaney [101] for independent checks via a trusted (perhaps fully verified) simple proof checker, building up shared repositories of accepted material and replacing community acceptance of the proof by community acceptance of the checker. A similar idea for verification is suggested in the JAVA community: write once/run anywhere becomes prove once/check anywhere. Foundational matters might turn out to be less important than confidence in a mature technology: for example trusting the developers, having access to the code, or being able to combine the formal development with an informal account.

#### 5.2 Mathematical applications

Proving new theorems, with a long term aim of becoming more widely used by mathematicians, is an obvious goal of automated reasoning research. As well as the benefit to the mathematical community, these challenge problems provide a significant test of our systems and techniques and, incorporated into collections of standard examples such as TPTP [107] and events such as the CADE theorem proving competitions, provide valuable benchmarks for developers.

In principle any of the systems we have described could be used for proving new theorems, but in practice the most impressive results in classical logic have come from systems using decision or semi-decision procedures such as resolution or rewriting. As examples we cite:

McCune's proof in 1997 of the Robbins conjectures [80] using EQP, a variant of the resolution theorem prover OTTER, made the front page of the New York Times, and has been cited as one of the top five achievements in AI [114]. The result, conjectured by Robbins in the 1930s, states that the equation

$$n(n(x) + y) + n(n(x) + n(y)) = x$$

follows from the equations

$$\begin{aligned} x + y &= y + x \\ (x + y) + z &= x + (y + z) \\ n(n(x + y) + n(x + n(y))) &= x. \end{aligned}$$

The automated proof took 8 days on an RS/6000 processor and used about 30 megabytes of memory: one can extract from the output the final proof which is 15 lines long and easily checked by hand.

- Similarly McCune and Padmanabhan's [79] work on cubic curves used OT-TER, and Fujita and Slaney's [101] work on the existence of certain quasigroups used OTTER together with model elimination and constraint programming techniques: both involved complex search over enormous search spaces.
- Equational reasoning based on rewriting and completion is a technique used both by algebraists and in theorem provers such as LP [40]: the first paper on completion, by Knuth and Bendix [64] was published in a group theory conference proceedings, and the group theorist Holt [29] has implemented a fast completion engine for strings. In recent studies Martin, Shand and Linton [74, 78] compared this with various theorem provers and concluded that while the former was often several orders of magnitude faster, because of various built in enhancements, the latter offered the benefits of a more flexible object language which allowed some automation that the group theory community had not considered before, such as completion over terms, rather than strings, and proofs by induction.
- An early challenge problem was the construction of single axiomatisations for groups, in answer to a question of Tarski. Neumann [51] constructed the first such example by hand: subsequent work of Kunen [67] constructed a large number of examples by running completion on candidates generated automatically. Completion has also been used by Kapur [121] to show that a ring satisfying a law of the form  $x^{2n} = x$  is commutative for certain values of n.

- Geometry theorem proving [16], using various combinations of axiomatisations, Gröbner basis techniques, and quantifier elimination by Collin's algorithm has been particularly successful in proving new results and in applications to problems in graphics and robotics. Euclidean geometry statements about lines, ellipses and so on which are true in a general case sometimes fail for certain degenerate cases (for example if the ellipse is a point): these techniques identified several such missed cases in the usual statements of well-known results.
- The automation of induction proofs has received much attention, particularly the automated choice of induction hypotheses. A powerful theory has been developed by Bundy [13] and applied for example to proofs similar to those obtained with Gosper's algorithm.

These achievements are impressive. A variety of first order, equational and induction proofs can be produced, sometimes completely automatically and sometimes with user guidance in the form of weights or orderings. The output is a proof which can be checked, in principle at least, using other software or, in some cases, like the Robbins proof, distilled to produce output which can be checked by hand. Large examples like this, or Slaney's work, often stretch our systems to their limits (that is why they are unsolved!) in terms of size, speed and search, requiring high-quality implementations of specialised decision or semi-decision procedures, which may not be that far from standard techniques of mathematical computation.

Perhaps the best comparison is with techniques of computational group theory, or McKay's graph algorithms: comparable sophisticated implementations which run completely automatically on small examples and have been run for weeks on large examples, where the search is controlled by subtle user settings.

Such automated reasoning techniques can be used in speculative or experimental work: for rapidly checking many different versions of a conjecture for example. Their use in proofs can again be assessed in terms of rigour and quality. In cases where the output can be readily checked, by hand or another program, such as in the proof of Robbins' result, rigour is unproblematic. One might suppose that in general the use of such systems would be regarded as more rigorous than the use of, for example, a computer algebra system. However the underlying logical system of a theorem prover is often combined with many other features whose correctness may be no more certain than that of a CAS, with, in the eyes of a specialist user community, less sense of ownership or community checking of the code.

The debates about quality are no different in kind from those outlined when discussing Gosper's algorithm above: much depends on the context as to whether an automatic proof is considered a "good" proof or not. A particularly striking example is given by the results about rings satisfying  $x^{2n} = x$ . The first order proof lacks the insight of the higher order proof involving division rings. In the case of the Robbins result, as with the four colour theorem, a different proof might have more explanatory power, but as yet no-one has managed to find one.

Such systems are powerful and effective when used on the right problems, but

they are not yet much used by mathematicians. We conjecture that the reasons are largely historical: as we have seen computation is any case not widely used in mathematical research, and where it is the programs are likely to have been developed over the years by academics in a particular research area developing domain specific algorithms, with a particular emphasis on efficiency. Thus for example our own work on automating inductive proofs in group theory was an eye-opener to this community, despite its impressive record in computational methods. In general mathematicians do not seem to know very much about automated reasoning techniques, or have judged them to be irrelevant on the basis of over-enthusiastic reports of proofs of very simple results or sterile conjectures. Few papers on automated reasoning appear in general mathematical journals, or are reviewed in Mathematical Reviews.

Even if such programs were more widely known one should be realistic about the possible scope for their application. First order, equational or induction problems of the kind that can be successfully automated form a very small part of the repertoire of techniques used in most research areas. So, as with Gosper's algorithm or computer algebra algorithms, these should be regarded as specialised techniques for particular situations, although ones independent of application domain. Enthusiasts for automated reasoning are not likely to find these potential applications unaided: matters are not as simple as opening a mathematical journal at random or asking the nearest mathematician for an unsolved problem, then typing some hypotheses and conclusions into our favourite theorem prover! Even when we find such a candidate problem we should remember that not every conjecture is interesting: conjectures may remain unproved because no-one capable of proving them has judged them worth trying to prove.

Thus such endeavours are most profitably pursued in collaboration with mathematicians who can provide context, insights and new undocumented problems. Crucially also mathematical collaborators can provide the introduction to the mathematical community which is necessary if these techniques are to be taken up by them in any serious way.

#### 5.3 Formalised mathematics

There has been much work in the theorem proving community on the development of formalised mathematics, with computational assistance in the form of proof checking or development of proofs through high level tactics and plans in the LCF tradition.

For example Mizar [110] uses natural deduction and Tarski-Grothendieck set theory, and has been used to formalise a large cumulative body of material at text-book and research journal level recorded over the past ten years in the journal Formalised Mathematics [61]. This is impressive: more formalised mathematics than has ever been produced by mathematicians. Other examples include foundational material in analysis and algebra such as analysis in AUTO-MATH [12], computer algebra algorithms in NuPrl [56] and constructive reals in LEGO [60], and achievements such as Harrison's development of integration [46] in HOL or Shankar's proof of Gödel's theorem in Boyer and Moore's NQTHM

[98]. Constructive proofs may improve on the classical version: unlike the classical proof, Murthy's [21] version of Higman's Lemma gives an estimate of how far along a sequence of words we must look to find a self-embedding. There are big differences in both the logical foundations and capabilities of these systems, but these need not concern us here. In principle such systems could be used for proving new results, and indeed they routinely are in, for example, the hardware verification community.

It would seem to be the case that with enough work (as a rough guide many of the automated proofs we have mentioned have occupied a beginning PhD student for a couple of years, or an experienced one for several months) it is possible to mechanise just about any piece of sufficiently developed existing mathematics, filling in the details of whatever outline is provided, assuming that that outline is consistent with the logical foundation of the system we are using. The difficulties come in the sheer amount of work involved, in managing the size, scale and efficiency of the operation, in developing the theory in such a way that precisely the right lemmas are available when needed, in "bending" the theory to fit a logical foundation which may not be ideally suited to it, in being certain we have indeed implemented the hand proof we intended to implement, in going back, sometimes a long way, when the proof doesn't work because of some subtle mis-phrasing, and in getting efficient implementations of general procedures such as factorisation or real arithmetic.

Just about any piece of sufficiently well-developed piece of mathematics can be formalised, but unlike logicians and automated reasoning researchers, few practising mathematicians seem to be interested in doing so.

One reason is that, as with the techniques we described in the previous section, very few mathematicians know about such systems. Even the terminology and concepts that form part of the most basic computer education in the subject are unfamiliar: types, lambda calculus or the subtleties of partial functions are not part of mathematical education even at graduate level, and I know of no part of classical mathematics outside mathematical logic where the concepts are made explicit. But this cannot be the whole answer: even theoretical computer scientists whose output is mathematical, and who know the concepts and work alongside theorem proving experts, do not generally use these tools, although they might employ a researcher to verify parts of an algorithm for example. Constable's account of automata theory in this volume is a pleasing exception rather than the norm.

We suggest that the main reason they are not used, and are not likely to be used in their present form, is because the contribution of this endeavour to current mathematical practice is marginal. Recall that we identified mathematical practice with "producing conjectural mathematical knowledge by means of speculation, heuristic arguments, examples and experiments, which may then be confirmed as theorems by producing proofs in accordance with a community standard of rigour, which may be read by the community in a variety of ways." and we noted that it had remained relatively unaffected by computation. Computer aided formalised mathematics has at present little to offer speculation,

heuristic arguments, examples and experiments. For a piece of mathematics to be "sufficiently well developed" to formalise it needs to be fairly close to a "proof in accordance with a community standard of rigour": in other words we need to have something pretty close to a traditional hand proof already. Thus if mathematical practice is not to change, and it seems extremely resistant to change, an expensive and difficult formalised proof needs to be produced in addition to all the work that is currently been done, with the benefit that, depending on our degree of trust in the prover, it gives us more certainty of freedom from error.

Suppose for example that Gorenstein's reworking of the classification of finite simple groups were to be automated, as it no doubt could be with sufficient resource made available to the task. We would be somewhat more certain of the classification, in particular meeting Gorenstein's concern that some special cases had not been missed. However the community would still want something similar to the twelve volumes of hand proof currently being produced, even if only as a commentary on the machine checked proof, for reading in the various ways that mathematical texts are read: as a reference, for teaching, for explanation and for understanding. And it might prefer to spend the enormous cost of formalising the endeavour on creating some new mathematics: Atiyah's "excitement and action".

Bob Boyer [9], developer of the ACL2 prover, goes further and advocates building a library of verified results as a cultural object "Like the great pyramids, the effort required (especially early on) may be great; but the rewards can be even more staggering than this effort." Kreisel [66] is characteristically trenchant "Let us remember that Hilbert's programme, as he originally intended it, has failed, though it appeared plausible to someone with his insight. In view of the obvious connection with automatic proof procedures, we must guard against similar misjudgements here since, with all due respect, people working in automatic proof theory cannot be expected to be superior to Hilbert."

Harrison [45], who has considerable experience of formalising mathematics in the HOL system, outlines some of the challenges in embarking on such a program: choosing a foundational system, how to handle definitions or partial functions, how much automation, what kind of user interface and how to handle large databases of lemmas or the propagation of apparently minor changes through a large body of formalised material. Thurston [109], one of the few mathematicians who has considered computer aided formal reasoning, has a full appreciation of these difficulties, and while foreseeing the routine use of theorem provers in-thesmall echoes similar concerns about the number of choices that would have to be made in establishing a definitive and widely used large body of material, the difficulty of reconciling this activity with current mathematical practice and the time that would be spent in standardisation activities and resolving controversy.

The diversity of approaches being studied in the formal reasoning community suggests we are not yet ready to commit either to a technology or a logical foundation. A further barrier is that, unlike much automated reasoning research, mathematical practice is universally based on classical logic, although the impact of computation is causing a few mathematicians to rethink this position [72].

A mere glance at an account of significant areas of 20th century mathematics, such as the classification of finite simple groups or the proof of Fermat's Theorem, is a humbling reminder of the enormous size of the task: a task for which the mathematical community has neither the interest, the inclination or the resources. Many of the same questions are being addressed in the computer science community with the more modest aims of building shared libraries of standard lemmas about common data-types for use in hardware or software verification.

# 5.4 Combination of systems

We outline below three approaches which use a combination of theorem proving and mathematical software in an attempt to improve the contribution of computation to current mathematical practice. We refer the reader to Calmet [53] for a discussion of general issues such as possible architectures for combined systems, and to endeavours such as OpenMath [1] for clarification of the practical difficulties involved in matters such as common interchange formats and interoperability, particularly for systems with different semantics or type systems.

**Embedded verification techniques** We may extend computational mathematics systems used for experiment or speculation using automated reasoning techniques. As we have seen computer algebra systems are notorious for giving results which are wrong or unexpected, particularly when singularities or branch cuts are involved. Thus such extensions should be regarded as extending the functionality available to the user for speculative work rather than necessarily improving the rigour of the system.

We might add to our computer algebra system facilities for induction, quantifier elimination, constraint solving, Boolean or first order reasoning, case splits, user defined inference rules or techniques for handling infinite sums and series. Current work in this area includes quantifier elimination in REDUCE [115] and induction in Mathematica [4]. Experiments at St Andrews involve a unifying approach to some of the problems involving side conditions, singularities and continuity by using proof planning to reorganise the existing somewhat ad-hoc methods. Notice that such experiments do not necessarily repair any unsoundness present in the underlying CAS.

A particularly successful application is Clarke's Analytica system [17, 18]. He extended Mathematica with a collection of inference rules, implemented using the built in matching, simplification and rewriting algorithms. The rules comprised natural deduction with quantifiers handled by Skolemisation, together with simple induction schema and standard identities involving inequalities, sums and series. These were used to derive automatically a collection of results in analysis [17], including a proof that

$$\sum_{n=0}^{\infty} b^n \cos\left(a^n \pi x\right)$$

with 0 < b < 1, a odd and positive and  $ab > 1+3\pi/2$ , is continuous and nowhere differentiable. An extension with harmonic numbers and trigonometric functions was used to prove identities involving finite and infinite sums from Ramanujan's notebooks [18]. The latter are described as "elementary" but would probably challenge the average graduate student.

**Formal methods for computational mathematics** Light formal methods [40] aim to provide assistance with design and documentation, particularly of interfaces, and obtain consistency of typing, avoid degenerate cases, and provide precise analysis of conditions at the specification stage without the overhead of full verification. They are valuable in addressing some of the concerns about developing a reliable discipline for mathematical experiment and modelling.

Systems like GAP and AXIOM comprise a kernel together with libraries of data and applications often contributed by advanced users. While the core code is often reliable it is hardly surprising that problems arise with interfaces, documentation and later users determining exactly what the code does. The algorithms being implemented in such systems are often complex and rely on elaborate mathematics, with many choices being possible as to the exact version of an algorithm to choose affecting, for example, the inputs on which it is valid. And machine checking that inputs are indeed valid may be infeasible if, for example, the precondition is that a function is continuous or differentiable at some point or that the underlying type is a ring with certain properties. A possibly apocryphal story concerns the six implementations of Gaussian elimination in a version of Macsyma: not that the six implementation were essentially different, but that users felt more confident developing their own than relying on the version provided being exactly applicable in their circumstances.

Modern computer algebra systems like AXIOM with advanced type systems already provide some security: Dunstan [27] addresses these issues further developing light formal methods for AXIOM in the Larch tradition [40]. They rely on annotations in a behavioural interface specification language, Larch-AXIOM, which can be manipulated using theorem proving techniques. The annotations form a partial specification which also aids reuse, debugging and checking for soundness. A verification condition generator shows the user the preconditions for a module without attempting to verify them, thus alerting the user to, for example, complex side conditions while giving them the freedom to decide what, if anything, to do about them. A static analyser similar to Nelson's [83] ESC system also uses the annotations to provide low level checks on array bounds and the like.

Formal methods techniques also have a role to play in elucidating the difficulties in mathematical modelling outlined above. As well as testing an implementation one might attempt to prove properties either directly or by proving properties of the partial specifications provided by annotations. Such proofs, while still only serving as further possible falsifications of the theory (or code), would be a further step in increasing confidence in the theory (or code). Simple techniques that kept track of assumptions, or combined existing theories in a

reliable way, might also make rather clearer the relationships between different assumptions and theories.

Such techniques can also be useful in understanding and controlling legacy code or large subroutine libraries. For example the Amphion project [106] involved the use of formal methods techniques to plan and interpret space science observations by constructing FORTRAN code from a collection of standard subroutines, each of which has been provided, post-implementation, with a first order specification. A graphical interface is used to pose the problem, then this is translated into first order logic and passed to a resolution based prover which derives a suitable specification in terms of the subroutines, and proves it correct. The required code can then be synthesised automatically.

The OpenMath [1] project addresses the interoperability of mathematical systems and provides a further application for formal methods techniques in providing precise semantics for interchange formats and addressing issues such as specification matching and type reconstruction.

**Formalising computational mathematics** While a full formal development of large areas of mathematics may not be feasible for some time, work on formalising existing computational mathematics, with the aim of getting more reliable, repeatable and checkable output and making such computations more acceptable in the mathematical community as part of a proof looks more promising.

The use of formal methods in debugging designs suggests developing their use in debugging definitions: proving well-formedness conditions, definedness and so on in developing a mathematical theory. Getting definitions right and consistent with each other is often time consuming and error prone, especially when a minor change is necessary perhaps for stylistic reasons and has to be propagated through a body of material. When the definitions are correct however checking these things is routine, and is generally glossed over in printed accounts. It would be useful to develop a methodology for using theorem provers to help in such developments.

For numerical software correctness of floating point arithmetic, formal analysis of error conditions and exact real computation are all active research areas. There have been various approaches to implementing computer algebra inside theorem provers: to do this properly requires formalising large amounts of mathematics, with a great loss in efficiency if standard algorithms such as factorisation are implemented from first principles. A 5GB Gröbner basis is a daunting prospect for any theorem prover! The resulting increase in precision, for example requiring side-conditions to be handled correctly at every stage, may be unwelcome to mathematicians who want to use the system in a fairly informal way. Various hybrid systems have been proposed, where theorem provers call computer algebra systems [53]. The theorem prover can trust the algebra system fully, using it as an oracle that acts as a special purpose decision procedure: or it can trust it not at all [48], merely using it as a suggestion for lemmas which are then proved in the theorem prover. Thus for example if the computer algebra system factorises a term this can be checked by the theorem prover proving

that the product of the factors is indeed the given term. There are a number of intermediate approaches such as arranging for the computer algebra system to provide hints or plans towards proofs of its results [62], or to trust the implementation of computer algebra algorithms while making the theorem prover do the book-keeping of checking interface definitions and so on [5].

A longer term goal is the use of theorem provers to provide the infrastructure and organisation of mathematical systems, guaranteeing secure and reliable output while providing the user with a familiar front end which can be used informally and calling on other specialised procedures as necessary. For example Harrison [47] has coded standard numerical routines in a small imperative language deeply embedded (that is via an embedding of the semantics) in HOL. The Cabri-Geometry system [14] allows the user to draw a geometrical configuration of lines and curves, apply geometrical transformations by point and click and then request a cartesian proof. Beeson's MathPert [7] system uses non-standard analysis techniques to handle some of the problems caused by continuity and parameters in computer algebra systems. Jamnik [58] has experimented with the use of the  $\Omega$ -rule to generalise from diagrammatic proofs.

More generally one can conceive a mathematical software system producing plans for execution by a theorem prover, for example a visualisation system producing input to a geometric reasoning system, or an exhaustive search producing input for a proof checker. As with other areas of artificial intelligence, usability may well come from hiding the technology rather than making it explicit.

# 6 Conclusions

Widespread use by research mathematicians is not and should not be the only or main goal of computational logic research <sup>2</sup>. However in so far as it is an objective we should be aware of the extraordinary power of current mathematical software systems, and of the nature of current mathematical practice, and address mathematical speculation and conjecture and the discipline of mathematical experiment. While there is debate in the mathematical community about the status of computation as part of a proof, mathematicians do routinely use computational techniques where appropriate: they are subject to community standards of rigour just as other proofs are.

We should investigate how our techniques can help, working with mathematicians rather than preaching at them or making mathematically naive claims about what our systems can do, starting with those who might already be sympathetic to our endeavours: computer inclined mathematicians and mathematically inclined computer scientists. As well as developing our systems and presenting our results in our own conferences and journals we should consider accessible survey articles, presentations at relevant mathematical and scientific conferences and representation in Mathematical Reviews.

 $<sup>^2\,</sup>$  A cynic might observe that research mathematicians are neither numerous nor rich!

# 7 Acknowledgments

This paper grew out of lively discussions with members of the Marktoberdorf Summer School in August 1997: I thank them and the organisers for an enjoyable and stimulating meeting, and Mike Atkinson, Alan Bundy, Wilfrid Hodges, Paul Jackson, Tony Hoare, Steve Linton, Tom Melham, Jose Meseguer, Alice Miller, Duncan Shand, Jan von Plato and Lincoln Wallen for helpful comments on earlier drafts of this paper.

# References

- J Abbott et al, Objectives of OpenMath, RIACA TU Eindhoven, Technical Report 12 (1996)
- 2. K Appel and W Haken, The four color proof suffices, Math. Intelligencer 8 (1986) 10–20
- M Atiyah et al, Responses to: A Jaffe and F Quinn, Bull. Amer. Math. Soc. 29 (1993) 1–13
- B Buchberger, Symbolic computation: computer algebra and logic, Appl. Log. Ser. 3 (1996) 193-219, Kluwer Acad. Publ
- C Ballarin et al, Theorems and Algorithms: An Interface between Isabelle and Maple International Symposium on Symbolic and Algebraic Computation, 150-157, ACM Press, 1995
- B Barras, et al, The Coq Proof Assistant Reference Manual (Version 6.1), 1996, available from ftp.inria.fr
- M Beeson, Mathpert: Computer support for learning algebra, trigonometry, and calculus, Logic Programming and Automated Reasoning, LNCS 624, Springer 1992, see also www.mathpert.com
- 8. C Boyer, A history of mathematics, John Wiley 1989
- 9. R S Boyer, The QED manifesto, CADE 12, LNCS 814, Springer 1994
- J Borwein et al, Making sense of mathematics, Math. Intelligencer 18:4 (1996) 12-18
- R Brown et al, Calculations with simplicial and cubical groups in AXIOM, J Symbolic Computation 17 (1994) 159-179
- N de Bruijn, The mathematical Language AUTOMATH, its usage, and some of its extensions, Symposium on Automatic Demonstration, Lecture Notes in Mathematics 125, Springer 1968
- A Bundy et al, Rippling: a heuristic for guiding inductive proofs, Artificial Intelligence 62 (1993) 185-253
- 14. Cabri-geometry, Texas Instruments see www.ti.com/calc/docs/cabri.htm
- 15. Centralised mathematical pre-print repository front.math.ucdavis.edu
- 16. S Chou, Mechanical geometry theorem proving, Reidel, 1988
- 17. E Clarke and X Zhao, Combining symbolic computation and theorem proving: some problems of Ramanujan, CADE 12, LNCS 814, Springer 1994
- E Clarke and X Zhao, Analytica A Theorem Prover for Mathematica, Carnegie Mellon University, School of Computer Science, CS-92-117, 1992
- A Cohn, The notion of proof in hardware verification, J Automated Reasoning 5 (1989) 127-140
- 20. P Cohn, Algebra volume 3, Wiley 1991
  - 39

- C Murthy et al, A constructive proof of Higman's lemma, Fifth Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA, 1990) 257–267
- 22. J Conway et al, Atlas of finite groups, Oxford University Press, Oxford, 1985, see also for.mat.bham.ac.uk/atlas/
- 23. J Dieudonné, Mathematics—the music of reason, Springer 1992
- 24. E Dijkstra, A discipline of programming, Prentice-Hall 1976
- 25. E Dijkstra, The tide not the waves, in Beyond computation, ed Denning, Springer 1997
- 26. R DeMillo et al, Social processes and proofs of programs CACM 22 (1979) 271-280
- 27. M Dunstan, The design and implementation of Larch-axiom, Ph D, University of St Andrews, forthcoming 1998
- 28. David Epstein, quoted in [54]
- D Epstein et al, The use of Knuth-Bendix methods to solve the word problem in automatic groups, J Symbolic Computation 12 (1991) 397-414
- 30. Journal of Experimental Mathematics, available at www.expmath.org
- R Fateman, Why Computer Algebra Systems Can't Solve Simple Equations, ACM SIGSAM Bull 30 (1996) 8-11
- 32. R Fateman, TILU Table of Integrals Look Up, www.cs.Berkeley.edu/fateman
- 33. S Feferman, What rests on what? The proof-theoretic analysis of mathematics, Philosophy of mathematics (Kirchberg am Wechsel, 1992) 147-171
- 34. J Fetzer, Program Verification: The Very Idea CACM 31 (1988) 1048-1063
- S Garland and J Guttag, An Overview of LP, The Larch Prover, RTA3, LNCS 355 137-151, Springer 1989
- 36. M Gordon and T Melham, Introduction to HOL: A theorem proving environment for higher order logic, Cambridge University Press, 1993
- D Gorenstein, Classifying the finite simple groups, Bull. Amer. Math. Soc. 14 (1986) 1–98
- 38. R Graham et al, Concrete Mathematics, Addison-Wesley, 1989
- 39. G Grimmett, Percolation, Springer 1989
- J V Guttag and J J Horning, Larch: languages and tools for formal specification, Springer 1993
- 41. J Grabiner, Computers and the nature of man: a historian's perspective on controversies about artificial intelligence, Bull. Amer. Math. Soc. 15 (1986) 113-126
- A Granville, Review of BBC Horizon Program, "Fermat's Last Theorem", Notices Amer. Math. Soc. 44 (1997) 15-16
- J Hadamard, An essay on the psychology of invention in the mathematical field, Dover 1954
- 44. G H Hardy, Mathematical Proof, Mind 38 (1929)
- 45. J Harrison, Formalized Mathematics, TUCS TR 36, 1996
- J Harrison, Constructing the Real Numbers in HOL, Formal Methods in System Design 5 (1994) 35-59
- 47. J Harrison, Floating point verification in HOL, HOL 95, LNCS 971, Springer 1995
- 48. J Harrison and L Thery, Extending the HOL Theorem Prover with a Computer Algebra System to Reason About the Reals, in HOL 93, LNCS 780, Springer 1993,
- 49. A Heck, Introduction to Maple, Springer 1993
- 50. R Hersh, What is mathematics, really? Oxford University Press 1997
- 51. G Higman, B Neumann, Groups as groupoids with one law, Publ. Math. Debrecen 2 (1952) 215-221
- D Hoffman, Computer-aided discovery of new embedded minimal surfaces, Math. Intelligencer 9 (1987) 8-21

- 53. K Homann et al, Combining theorem proving and symbolic mathematical computing, LNCS 958, Springer 1994
- 54. J Horgan, The death of proof, Sci. Amer. 269 (1993), 92-103
- 55. D Hume, Treatise on human nature, (ed Mossner), Penguin 1969, p231
- P Jackson, Exploring Abstract Algebra in Constructive Type Theory, CADE 12, LNCS 814, Springer 1994
- A Jaffe and F Quinn, "Theoretical mathematics": toward a cultural synthesis of mathematics and theoretical physics, Bull. Amer. Math. Soc. 29 (1993), 1–13
- M Jamnik et al, Automation of diagrammatic reasoning, Proceedings of the 15th IJCAI (1997) 528-533, Morgan Kaufmann
- R D Jenks and R S Sutor, axiom: The Scientific Computation System, Springer 1992
- C Jones, Completing the Rationals and Metric Spaces in LEGO, in Logical Environments, Cambridge University Press 1993
- 61. Journal of Formalised Mathematics, see mizar.uw.bialystok.pl
- M Kerber et al, Integrating Computer Algebra with Proof Planning, LNCS 1128, Springer 1996
- 63. P Kitcher, The nature of mathematical knowledge, Oxford University Press 1983
- D Knuth, P Bendix, Simple word problems in universal algebras, Computational Problems in Abstract Algebra, 263–297, Pergamon 1967
- 65. S Krantz, The immortality of proof, Notices Amer. Math. Soc. 41 (1994) 10-13
- 66. G Kreisel, Hilbert's Programme and the search for automatic proof procedures, Symposium on Automatic Demonstration, Lecture Notes in Mathematics 125, Springer 1968
- 67. K Kunen, Single axioms for groups, J. Automated Reasoning 9 (1992) 291-308
- G Labelle, Some combinatorial results first found using computer algebra, J Symbolic Computation 20 (1995) 567-594
- I Lakatos, Proofs and refutations, The logic of mathematical discovery, Cambridge University Press 1976
- 70. C Lam, How reliable is a computer-based proof? Math. Intelligencer 12 (1990) 8--12
- S Lang: Mordell's review, Siegel's letter to Mordell, Diophantine geometry, and 20th century mathematics, Notices Amer. Math. Soc. 42 (1995) 339–350
- D Epstein, S Levy: Experimentation and proof in mathematics, Notices Amer. Math. Soc. 42 (1995) 670-674
- 73. S Levy, Making waves, A guide to the ideas behind Outside in, A K Peters, 1995
- S Linton et al, Some group-theoretic examples with completion theorem provers, J Automated Reasoning 17 (1996)145-169
- 75. D Mackenzie, Knowing machines, MIT Press 1997
- D Mackenzie, Slaying the cracken: the socio-history of a mathematical proof, Preprint 1997
- 77. Ju Manin et al, How convincing is a proof? Math. Intelligencer 2 (1979) 17-24
- U Martin et al, Algebra and Automated Reasoning, CADE 13, LNCS 1102, Springer 1996
- W McCune and R Padmanabhan, Automated deduction in equational logic, LNCS 1095, Springer 1996
- W McCune, Solution of the Robbins Problem, J. Automated Reasoning 19 (1997) 263-276
- B McKay and S Radziszowski, Subgraph counting identities and Ramsey numbers, J. Combin. Theory Ser. B 69 (1997) 193–209
  - 41

- 82. The NAG Fortran library, www.nag.co.uk
- G Nelson et al, Network objects, Software Practice and Experience, 25 (1995) 87-130
- 84. A Newell et al, Empirical explorations of the Logic Theory Machine: a case study in heuristics, in Computers and Thought, McGraw Hill 1963
- 85. N Oreskes et al, Verification, validation and confirmation of numerical models in the earth sciences, Science 263 (Feb 1994) 641–646
- 86. S Owre et al, PVS: A Prototype Verification System, LNCS 607, Springer 1992
- 87. L Paulson, Logic and Computation, Cambridge University Press, 1987
- H Poincaré, Mathematical creativity, reprinted in The world of mathematics ed Newman, volume 4, Microsoft Press, 1988
- 89. R Pollack, How to believe a machine checked proof, Twenty five years of constructive type theory, ed G Sambin, Oxford University Press 1997
- K Popper, Conjectures and refutations: The growth of scientific knowledge, Basic Books 1962
- 91. P Rideau, Computer algebra and mechanics: the JAMES software, in Computer Algebra in Industry, (ed A M Cohen), Wiley 1993, 143–158
- 92. N<br/> Robertson et al, The four-colour theorem, J. Combin. Theory Ser. B<br/> 70 (1997) $2{-}44$
- J Robinson, A machine oriented logic based on the resolution principle, JACM 12 (1964), 23-41
- B Rotman, Thinking dia-grams: mathematics, writing and virtual reality, South Atlantic Quarterly 94 (1995) 389-416
- 95. D Ruelle, Chance and chaos, Penguin 1991
- Martin Schönert et.al. GAP- Groups, Algorithms, and Programming, Lehrstuhl D för Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1995, or at www-gap.st-and.ac.uk
- B Selman and S Kirkpatrick, Critical behavior in the computational cost of satisfiability testing, Artificial Intelligence 81 (1996) 273-295
- N Shankar, Metamathematics, machines, and Gödel's proof, Cambridge University Press 1994
- S Simpson, Partial realizations of Hilbert's Program, J. Symbolic Logic 53 (1988) 349-363
- 100. S Singh, Fermat's Enigma: The Epic Quest to Solve the World's Greatest Mathematical Problem, Walker and Company 1997
- J Slaney et al, Automated reasoning and exhaustive search: quasigroup existence problems, Comput. Math. Appl. 29 (1995) 115-132
- J Slaney, S Thiebaux, Phase transitions and optimality: sense and nonsense Preprint 1997
- 103. N Sloane, S Plouffe: The encyclopedia of integer sequences, Academic Press 1995, www.research.att.com/njas/sequences/
- 104. R Solomon, On finite simple groups and their classification, Notices Amer. Math. Soc. 42 (1995) 231-239
- 105. I Stewart, Does god play dice? The mathematics of chaos, Blackwell 1989
- 106. M Stickel et al, The deductive composition of astronomical software from subroutine libraries, CADE 12, LNAI 814, Springer 1994 341–355
- 107. G Sutcliffe et al, The TPTP Problem Library, CADE 12, LNAI 814, Springer 1994 252-266
- 108. H P F Swinnerton-Dyer and B Birch, Elliptic curves and modular functions, Lecture Notes in Math, 476 (1975) 2-32
  - 42

- W Thurston, On proof and progress in mathematics, Bull. Amer. Math. Soc. 30 (1994) 161-177
- A Trybulec, The Mizar-QC 6000 logic information language, ALCC bulletin 6 (1978) 136-140
- T Tymoczko, A philosophical investigation of the four-color proof, Math. Mag. 53 (1980) 131–138
- J von Neumann, The mathematician, in The Works of the Mind (1947) 180-196, University of Chicago Press
- 113. S Wolfram, The Mathematica book, Cambridge University Press 1996
- D Waltz, Artificial intelligence: realizing the ultimate promises of computing, AI magazine 18 (1997) 49-52
- V Weispfenning, Simulation and optimization by quantifier elimination, Applications of quantifier elimination, J. Symbolic Computation 24 (1997)189–208
- 116. H Weyl, Mathematics and logic, Amer Math Monthly 53 (1946) 2-13
- 117. A Whitehead, Science and the modern world, Cambridge University Press 1926
- 118. A Wiles, Modular elliptic curves and Fermat's last theorem, Ann. of Math. (2) 141 (1995) 443-551
- 119. L Wos, The Automation of Reasoning: An Experimenter's Notebook with Otter Tutorial, Academic Press (1996)
- D Zeilberger, Theorems for a price: tomorrow's semirigorous mathematical culture, Notices Amer. Math. Soc. 40 (1993) 978-981
- 121. H Zhang and D Kapur, Consider only General Superpositions in Completion Procedures, RTA 3, LNCS 355 513-527, Springer 1989

This article was processed using the LATEX macro package with LLNCS style