Computing and the cultures of proving

BY DONALD MACKENZIE

School of Social & Political Studies, University of Edinburgh, Adam Ferguson Building, Edinburgh EH8 9LL, Scotland
(d.mackenzie@ed.ac.uk)

This article discusses the relationship between mathematical proof and the digital computer from the viewpoint of the ‘sociology of proof’: that is, an understanding of what kinds of procedures and arguments count for whom, under what circumstances, as proofs. After describing briefly the first instance of litigation focusing on the nature of mathematical proof, the article describes a variety of ‘cultures of proving’ that are distinguished by whether the proofs they conduct and prefer are (i) mechanized or non-mechanized and (ii) formal proofs or ‘rigorous arguments’. Although these ‘cultures’ mostly coexist peacefully, the occasional attacks from within one on another are of interest in respect to what they reveal about presuppositions and preferences. A variety of factors underpinning the diverse cultures of proving are discussed.

Keywords: mathematical proof; sociology of proof; cultures of proving; computer-system verification; formal verification

1. Introduction

The relationship between mathematical proof and the digital computer is at the heart of a number of major scientific and technological activities: see figure 1. Proofs are conducted about computers in at least three areas: some of those software systems upon which human lives depend; key aspects of some microprocessors and some systems upon which national security depends. These ‘formal verifications’, as proofs about the ‘correctness’ (correspondence to specification) of the design of computer hardware or software are called, are themselves normally conducted using computer programs; either automated theorem provers (software especially designed to produce proofs, albeit often with human guidance) or model checkers (which check whether a representation of a system is a ‘model’ for the logical formula expressing the system’s specification, in other words an interpretation of the logic in which the formula is true). Mathematicians themselves have also turned to the computer for assistance in proofs of great complication, the most famous such cases being the four-colour theorem and Kepler sphere-packing conjecture. Automated theorem provers are also of considerable interest and importance within artificial intelligence. They raise, for example, the question of the extent to which a computer can replicate the thought processes of human mathematicians.

One contribution of 13 to a Discussion Meeting Issue ‘The nature of mathematical proof’. 
The resultant issues are deep, and have played out in historical episodes that I have discussed elsewhere (MacKenzie 2001). In this paper, I focus on the most intriguing question this area throws up for the sociology of science: is it possible to develop a sociology of mathematical proof, in other words a sociological analysis of what kinds of procedures and arguments count for whom, under what circumstances, as proofs?

The question is of interest sociologically because of a certain imbalance in the sociological analysis of science, which has focused predominantly on the natural sciences. Relatively little work has been done on the ‘deductive’ sciences of mathematics and logic, and sociological discussion of deductive proof—which is at the core of those disciplines—is sparse (see MacKenzie 2001 for references to the main existing work).

Perhaps, the reader may suspect, there has been little ‘sociology of proof’ because such sociology is impossible? Perhaps mathematical proof is an absolute matter, not subject to the kind of variation that would make it susceptible to sociological analysis. The history of mathematics, however, reveals substantial variation in the kinds of argument that have been taken as constituting mathematical proof (see the survey by Kleiner 1991). Eighteenth century work in calculus, for example, often relied upon manipulating infinitesimally small quantities or infinite series in ways that became unacceptable in the nineteenth century. Early twentieth century mathematics was driven by dispute over the
acceptability, in proofs involving infinite sets, of the law of the excluded middle.
(The law is that for any proposition \( p \), ‘\( p \) or not-\( p \)’ must be true. Invocation of
excluded middle permits non-constructive existence proofs, which demonstrate
that a mathematical entity exists by showing that its non-existence would imply
a contradiction.)

The history of mathematics, therefore, suggests that mathematical proof is not
the straightforward, absolute matter than it is often taken to be. In 1987,
colleagues and I drew upon this evidence to make a prediction about the effort
(by then widely pursued) to apply mathematical proof to computer systems. We
noted that this effort involved moving mathematical proof into a commercial and
regulatory arena. We speculated that the pressures of that arena would force
potential variation in the meaning of proof out into the open, but that disputes
about proof would no longer simply be academic controversies. We suggested
that it might not be long before a court of law had to rule on what a
mathematical proof is (Pélaéz et al. 1987).

That prediction was nearly borne out in 1991, when litigation broke out in
Britain over the application of mathematical proof to a microprocessor chip
called VIPER (verifiable integrated processor for enhanced reliability), which
had been developed by computer scientists working for the Ministry of Defence’s
Royal Signals and Radar Establishment. At stake was whether the chain of
mathematical reasoning connecting the detailed design of VIPER to its
specification was strong enough and complete enough to be deemed a proof.
Some members of the computer-system verification community denied that it
was (Cohn 1989; Brock & Hunt 1990), and, largely for unconnected reasons, sales
of VIPER were disappointing. Charter Technologies Ltd, a firm which had
licensed aspects of VIPER technology from the Ministry of Defence, took legal
action against the Ministry, alleging, amongst other things, that VIPER’s design
had not been proven to be a correct implementation of its specification.

No ‘bug’ had been found in the VIPER chips; indeed, their design had been
subjected to an unprecedented amount of testing, simulation, checking and
mathematical analysis. At issue was whether or not this process, as it stood
immediately prior to the litigation (considerable subsequent work was done on
the VIPER verification), amounted to a mathematical proof. Matters of fact
about what had or had not been done were not central; the key questions that
had been raised by critics were about the status, adequacy and completeness,
from the viewpoint of mathematical proof, of particular kinds of argument. With
the Ministry of Defence vigorously contesting Charter’s allegations, the case
failed to come to court only because Charter became bankrupt before the High
Court heard it. Had it come to court, it is hard to see how the issue of what, in
this context, mathematical proof consists in could have been avoided.

The VIPER controversy has been reported elsewhere (MacKenzie 1991), and a
single episode has inevitable idiosyncrasies. Let me turn, therefore, to wider
issues of the ‘sociology of proof’ raised by the domains listed in figure 1.

(a) Mechanized and non-mechanized proofs; formal and rigorous proofs

How might one characterize variations in the kinds of mathematical
procedures or arguments that are taken, in the fields listed in figure 1, as
constituting proofs? One dimension of that variation is very simple: it is whether

Phil. Trans. R. Soc. A (2005)
a procedure is conducted, or an argument generated, by a human being or by a machine. Controversy in respect to this dimension has focused above all on machine-performed procedures that are too extensive or too complicated for unaided human beings to check. The dependence of the proof of the four-colour theorem on such procedures led to much debate as to whether it was indeed a genuine proof (see MacKenzie 2001). However, the traditional mathematical preference for proofs performed by human beings (or at least surveyable by human beings) is contested when it comes to verification of the design of computer systems. The proponents of automated verification have argued that mechanization is preferable to reasoning by unaided human beings, prone as we are to lapses of concentration and wishful thinking. One strand of the criticism of the claim of proof for VIPER, for example, was that key steps in the chain of argument had not been subjected to mechanical checking.

A second dimension of variation is also familiar: it is whether an argument is a ‘formal proof’ or a ‘rigorous argument’. A formal proof is a finite sequence of ‘well-formed’ (that is, to put it loosely, syntactically correct) formulae leading to the theorem, in which each formula either is an axiom of the formal system being used or is derived from previous formulae by application of the system’s rules of logical inference. These rules will normally be syntactic in form, such as the famous modus ponens: if \( p \) and ‘\( p \) implies \( q \)’ are formulae in the sequence, then \( q \) can be added to the sequence. The steps in a formal proof are thus mechanical applications of inference rules, and their correctness can therefore be checked without understanding the meaning of the formulae involved.

Rigorous arguments, in contrast, are those arguments that are accepted by mathematicians (or other relevant specialists) as constituting mathematical proofs, but that are not formal proofs in the above sense. The proofs of ordinary Euclidean geometry, for example, are rigorous arguments, not formal proofs: even if they involve deducing a theorem from axioms (and some involve reasoning that it is not, at least directly, of this form), the steps in the deduction are typically not merely applications of rules of logical inference. This is not simply a reflection of the antiquity of Euclidean geometry: articles in modern mathematics journals, whatever their subjects, almost never contain formal proofs. A very simple sketch of a ‘rigorous argument’ proof is provided by the famous ‘mutilated chessboard’ puzzle; see figure 2. The argument in the caption leaves one in no doubt of the correctness of the conclusion, yet it is not a formal proof (and it would not become a formal proof, in the sense in which the term is used in this paper, even if the everyday terms used in the puzzle were replaced by more precise mathematical equivalents).

‘Rigorous arguments’ are often called ‘informal proofs’. I avoid the latter term, however, because informal proof is often assumed to be inferior to formal proof, while preferences between the two are amongst the issues that a sociology of proof needs to investigate. Instead, I draw the notion of ‘rigorous argument’, in the sense in which the phrase is used here, from a UK Ministry of Defence Procurement Executive Standard governing safety-critical software (Ministry of Defence 1991). What fascinates me as a sociologist of science about the formal verification of computer systems is that a document such as a defence procurement standard is driven onto the quintessentially philosophical terrain of having to define what ‘proof’ is!
2. Cultures of proving

The two dimensions of mechanized versus non-mechanized proofs, and formal proofs versus rigorous arguments, allow a simple map of ‘cultures of proving’ (a term I draw from Livingston 1999, though he discusses only non-mechanized rigorous arguments). In figure 3 a variety of disciplines or more specialized cultures are located according to the forms of proof they practice or value. Mainstream automated theorem proving (discussed in detail in MacKenzie 2001) values, above all, mechanized formal proofs. Some departures from formality are found in practice—many automated theorem-proving systems employ ‘decision procedures’ (algorithms that determine whether formulae in particular mathematical domains are theorems) that do not generate formal proofs—but such departures are regarded at best as pragmatic necessities and at worst as reasons for condemnation. One theorem-proving specialist interviewed for the research on which this paper is based said that using unverified decision procedures was ‘like selling your soul to the Devil—you get this enormous power, but what have you lost? You have lost proof, in some sense’. (For details of the interviews drawn on here, see MacKenzie 2001.)

In contrast, in ‘ordinary’ mathematics (the mathematics conducted by most members of university mathematics departments, for example) proof is usually non-mechanized rigorous argument. Computers are, of course, playing an

---

Figure 2. The mutilated chessboard (from Black 1946). Two diagonally opposite corner squares are excised from a chessboard. Can the remaining 62 squares be covered entirely by 31 dominoes, each of which can cover two squares (and no more than two squares)? The answer is ‘no’. An unmutilated chessboard has an equal number of squares (32) of each colour. The two excised squares must be the same colour, so the mutilated chessboard has two squares more of one colour than of the other. Whenever we lay a domino, it covers one square of each colour. If we can cover 60 squares by laying 30 dominoes, the last two uncovered squares must be the same colour, and the 31st domino therefore cannot cover them.
ever-increasing role in mathematical research but, as noted above, there remains a pervasive sense that, in regard to proof, computerized procedures that human beings cannot check in detail are inferior to arguments that mathematicians can grasp in their entirety. Arguably, the key issue is not mechanization per se but unsurveyability: human-generated proofs that are too extensive realistically to be grasped in full are also seen as problematic.

Some computer-system verifications also fall within the quadrant of non-mechanized rigorous argument. Perhaps the key site of such verifications was IBM’s ‘Cleanroom’, an approach to the development of high-dependability software inspired by Harlan D. Mills (for details; see MacKenzie 2001). In the Cleanroom, proof was an explicitly human and intersubjective activity, a refinement of the familiar process of software review. A Cleanroom proof was an argument that convinced another human being: specifically, a designer’s or programmer’s argument that convinced his or her fellow team members or other reviewers of the correctness of his or her design or program, for example by convincing them that account had been taken of all possible cases and that the program or design would behave correctly in each case. The claim, ‘It is obvious’, counted as a proof, if to the reviewing team what was claimed was indeed self-evident. Even the use of mathematical notation was not an essential part of proof, except, in the words of Mills and colleagues, ‘in its effect on the person who is the experimental subject’ (Linger et al 1979). That Cleanroom proof explicitly aimed to produce ‘subjective conviction’ was no argument against it, said its proponents, because that was ‘the only type of reasoned conviction possible’ (Linger et al 1979). Mistakes were always possible, but systematic reasoning made human beings less error-prone, and systematic review by other people reduced mistakes even further.

Of course, human beings can perform formal proofs as well as rigorous-argument proofs. Prior to the digital computer, formal proofs had to be conducted by hand, as in the ‘logicist’ approach to the foundations of mathematics exemplified most famously by Whitehead & Russell (1910–1913).
What is more surprising is that modern computer science contains a subculture within which formal proof is preferred to rigorous argument but in which proofs are conducted by hand rather than by machine. This culture of ‘calculational proof’ was inspired by the great Dutch theoretical computer scientist Edsger W. Dijkstra and, like early logicism, it occupies the lower left quadrant of figure 3. Dijkstra believed that mathematics, including the mathematics of computer science, should be performed formally, that is, ‘by manipulating uninterpreted formulae accordingly to explicitly stated rules’ (Dijkstra & Scholten 1990). To do otherwise was, in Dijkstra’s view, to be ‘medieval’ (Dijkstra 1988). Yet Dijkstra was ‘not an enthusiast for the mechanization’ of proof, commenting: ‘Why delegate to a machine what is so much fun to do yourself?’ (MacKenzie 2001).

There is, indeed, a significant (and to the outsider, a surprising) current of ambivalence about mechanization in the culture of élite, theoretical computer science. The iconic representation of this ambivalence is in the matter of writing. Within this strand of computer science, the fountain pen, to others an archaic technology became something of an icon. Dijkstra’s beautifully handwritten lecture notes and correspondence have become famous. One of Dijkstra’s students even hand-wrote, and published in handwritten form, his PhD thesis (van de Snepscheut 1985), to which Dijkstra contributed a handwritten foreword.

The final, upper right quadrant in figure 3 is mechanized rigorous argument. Formal proof has been relatively easy to automate. The application of rules of inference to formulae considered simply as strings of symbols can be implemented on a digital computer using syntactic pattern matching. The automation of generic rigorous argument, on the other hand, has been a far more difficult problem. Some parts of what human mathematicians do, such as algebraic manipulation, can relatively readily be mechanized: there are now widely used commercial programs that automate symbol manipulation in fields like algebra and calculus. There are, however, as yet no ‘artificial mathematicians’, in the sense of automated systems capable of handling the full spectrum of rigorous arguments used in different fields of mathematics. Development in this direction is a hard problem in artificial intelligence, one that some commentators (such as Penrose 1989) deny will ever be solved fully: for progress in this field, see Bundy et al. (2005).

3. Conflicts over ‘proof’

Nearly all the time, the cultures of proving listed in figure 3 have coexisted peacefully alongside each other, often not interacting much. The institutional locations of the two most populous quadrants, mechanized formal proof and non-mechanized rigorous argument (in its ‘ordinary mathematics’ form) have differed, with the former based in university computer science departments, quasi-university research institutes (such as SRI International) and related industrial sectors. Only a small minority of academic mathematicians (that is, members of university mathematics departments) seem to use automated theorem provers, and almost none have contributed to the development of such systems (see MacKenzie 2001).
Certainly, the different forms of proving need not be taken to be in conflict. It is standard, for example, to take the view that a rigorous-argument proof is a sketch that can be translated into a formal proof. Nevertheless, a subtly different interpretation is possible, and was put forward by the mathematical logician Jon Barwise in a comment on the dispute begun by philosopher James Fetzer’s critique of program verification (Fetzer 1988). Barwise noted that Fetzer, as an ‘orthodox’ philosopher, and his critics, as proponents of automated theorem proving, took the canonical notion of proof to be formal proof. In this tacit agreement, said Barwise, both sides ‘stumble over a landmine left by the retreating formalists’; both believed that a ‘real proof’ was a formal one. ‘[A]t the risk of stepping on the toes of my fellow mathematical logicians’, Barwise argued that it was not. Formal proof was only a model of real proof, indeed a ‘severely impoverished’ model:

[T]here are many perfectly good proofs that are not modeled in any direct way by a formal proof in any current deductive system. For example, consider proofs where one establishes one of several cases and then observes that the others follow by symmetry considerations. This is a perfectly valid (and ubiquitous) form of mathematical reasoning, but I know of no system of formal deduction that admits of such a general rule. They can’t, because it is not, in general, something one can determine from local, syntactic features of a proof… [I]t could be that the best proofs (in the sense of being most enlightening or easiest to understand) of a program’s correctness will use methods, like symmetry considerations, that are not adequately modeled in the logician’s notion of formal proof, and so which would not be deemed correct by some automated proof checker designed around the formalist’s model (Barwise 1989, p. 849).

Differences of opinion over whether formal proof or rigorous argument constitutes ‘real proof’ indicates the potential for disagreement between cultures of proving. In practice, however, attacks from one culture of proving upon another are quite rare. They are interesting nevertheless, in what they reveal about the presuppositions of the culture from which the attack comes, rather than about the reality of the culture being attacked.

Statements of preference for rigorous-argument over formal proof, or vice versa, usually stop short of denying that the less favoured alternative is proof and of asserting that the corpus of knowledge verifiable only by its application is therefore defective and its contents are not theorems. Nevertheless, that denial can be found. In a famous attack upon program verification, DeMillo et al (1979) argued, in effect, that because program verifications were formal proofs, and because what mathematicians did was rigorous argument not formal proof, program verifications were therefore not proofs and their results not theorems. In 1957, the logician Peter Nidditch came close to the opposite claim, when he argued that what mathematicians do is not fully valid proof:

In the whole literature of mathematics there is not a single valid proof in the logical sense… The number of original books or papers on mathematics in the course of the last 300 years is of the order of $10^6$; in these, the number of even close approximations to really valid proofs is of the order of $10^1$… In the relatively few places where a mathematician has seriously tried to give a
valid proof, he has always overlooked at least some of the rules of inference and logical theorems of which he has made use and to which he has made no explicit reference... In addition, in these places, the mathematician has failed to pay sufficient critical attention to purely mathematical points of detail (Nidditch 1957, pp. v, 1, 6).

Nidditch’s comment was a critique of ordinary rigorous-argument mathematical proof from the viewpoint of logicism. Dijkstra offered a similar critique from the viewpoint of ‘calculational proof’. One of his famous, widely circulated, handwritten memoranda, dating from 1988, was caustically entitled ‘Real mathematicians don’t prove’. For Dijkstra, the struggle in computing between formalists and ‘real programmers’ (who, as Dijkstra put it, ‘don’t reason about their programs, for reasoning isn’t macho’) was part of a wider battle pervading ‘the rest of mathematics’ between formalists and ‘informalists’:

only they don’t call themselves by that negative name: presumably they present themselves as ‘the real mathematicians’—who constantly interpret their formulae and ‘reason’ in terms of the model underlying that interpretation.

By rejecting formalism, with its clear distinction between ‘provability’ and ‘the fuzzy metaphysical notion of “truth”’, mathematics remained according to Dijkstra ‘still a discipline with a sizeable pre-scientific component, in which the spirit of the Middle Ages is allowed to linger on’. Amongst its ‘medieval characteristics’ was that ‘how to do mathematics is not taught explicitly but only by osmosis, as in the tradition of the guilds’ (Dijkstra 1988).

Dijkstra’s ‘calculational’ proofs were themselves attacked by Panagiotis Manolios and J Strother Moore, a leading member of the automated theorem proving community, on the grounds that such proofs were not ‘formal’, but ‘rigorous arguments in a strict format... where the notion of proof is “convincing enough for your fellow mathematicians”’ (Manolios & Moore 2001). Two points about this critique are of interest. First, proof as conducted in ordinary mathematics is here taken as inferior to formal proof, and not as establishing what real proof is (as DeMillo, Lipton and Perlis had argued). Second, Manolios and Moore’s argument can be taken as suggesting that the non-mechanized, formal quadrant of figure 3 is unstable: that in the absence of the ‘discipline’ imposed by mechanized systems, human beings cannot realistically be expected to produce extensive, entirely formal proofs. Elements of higher-level, rigorous-argument reasoning will inevitably creep in.

Whether or not this suggestion is valid, it can be noted that mechanized systems have enabled a considerable expansion of the domain of formal proof. In the first half of the twentieth century, the notion of formal proof was largely a tool of metamathematics: a way of modelling ‘proof’ so as to permit precise reasoning about it. It was not a viable practical alternative to rigorous-argument proof in anything other than very limited mathematical domains. In the late twentieth century, however, the advent of automated theorem provers and proof-checkers permitted the formal proof of significant parts of mathematics and mathematical logic, including quite difficult theorems like Gödel’s incompleteness theorem (Shankar 1994). At least two notions of proof—non-mechanized rigorous argument, and mechanized formal proofs—are now practised on a
relatively large scale, and are available to be counterposed by those who wish for whatever reason to do so.

4. Disciplines and applications

One factor underpinning the cultures of proving listed in figure 3 is the structure of academic disciplines, in particular the separation between mathematics and philosophy and the resultant sometimes uneasy interstitial situation of logic. While most mathematicians are in practice committed to rigorous-argument proof, formal proof has become the canonical notion of ‘proof’ in modern philosophy, and Barwise is almost certainly correct in suggesting that most logicians also adhere to this view.

Computer science appears heterogeneous in this respect. It is the disciplinary home of mechanized formal proof, and many logicians have found posts in computer science easier to obtain than in their parent discipline, with at least some of them bringing with them a preference for formal proof. However, computer science also generated the celebrated attack on mechanized formal proof by DeMillo et al (1979). Furthermore, commitment to artificial intelligence can create a preference for rigorous-argument proof. If one’s goal is automated replication of human reasoning, then the mechanization of rigorous argument can be seen as a more appropriate objective than the mechanization of formal proof, for all the latter’s greater technological tractability.

However, overall factors of this kind should not be overemphasized; sometimes very specific factors condition the appropriateness of particular notions of proof. For example, because at the time of the VIPER lawsuit the formal, mechanized proof of the correctness of its design was incomplete, defence of the claim for proof for VIPER was in effect forced implicitly to defend rigorous-argument proof. One such defence was mounted by a leading figure in the UK software industry, Martyn Thomas, founder of the software house Praxis. ‘We must beware’, he wrote, ‘of having the term “proof” restricted to one, extremely formal, approach to verification. If proof can only mean axiomatic verification with theorem provers, most of mathematics is unproven and unprovable. The “social” processes of proof are good enough for engineers in other disciplines, good enough for mathematicians, and good enough for me.... If we reserve the word “proof” for the activities of the followers of Hilbert [David Hilbert, leader of “formalism” within mathematics], we waste a useful word, and we are in danger of overselling the results of their activities’ (MacKenzie 2001). Although Thomas was in a sense defending proof as conducted by mathematicians, he was not himself a mathematician: he trained as a biochemist before entering the computer industry.

Nor are the cultures of proving depicted in figure 3 homogeneous. Our symposium revealed differences between mathematicians in views of ‘proof’, and automated theorem proving and formal verification likewise have their divides. One such divide, alluded to above, concerns the use in proof of decision procedures that have not themselves been subject to formal verification. This is one manifestation of a more general set of divides concerning the attitude to be taken to the fact that automated theorem provers are themselves quite complicated computer programs which may contain faults in design or
implementation. In interviews for this research, designers of automated theorem provers often reported experience of ‘bugs’ in their systems that would have allowed ‘theorems’ that they knew to be false nevertheless to be proven. Such bugs were not large in number, they were corrected whenever they were discovered, and I know of only one case in which a theorem-proving bug caused a false result whose falsity was not detected immediately (it led to a claim of an automated proof of the Robbins conjecture—that Robbins algebras are Boolean—that later had to be withdrawn; see MacKenzie 2001). But no designer seemed able to give an unequivocal guarantee that no such bugs remained.

Reactions to the issue varied amongst those interviewed. One interviewee (Alan Robinson, developer of the fundamental theorem-proving procedure of ‘resolution’) suggested that the possibility of unsoundness in the design of theorem provers indicates that the overall enterprise of formal verification is flawed, because of what he has come to believe to be the impoverishment of the formal notion of proof:

> You’ve got to prove the theorem-proving program correct. You’re in a regression, aren’t you?...That’s what people don’t seem to realize when they get into verification. They have a hairy great thing they’re in doubt about, so they produce another hairy great thing which is the proof that this one’s OK. Now what about this one which you’ve just [used to perform the proof]?...I say that serves them jolly well right.

That is not the response of program and hardware verification ‘insiders’. While paying considerable attention to soundness, they feel that theorem-prover bugs are not important practical worries compared to ensuring that the specification of a system expresses what, intuitively, is intended:

> If you...ask where the risks are, and what are the magnitudes of the risks, the soundness of the logic is a tiny bit, a really tiny bit, and the correctness of the proof tool implementing the logic is slightly larger [but] actually...quite a small risk.

As that last quotation reminds us, automated theorem provers are developed not just as an important intellectual exercise in its own right, but to support verification in contexts in which hardware or software design faults can be fatal, can compromise national security, or can be very expensive in their consequences. For many years, the most important single source of funding for the development of automated theorem provers was the national security community, in particular the US Defense Advanced Research Projects Agency and National Security Agency. The demands of national security—in particular the desire for a theorem-proving system that prevents a hostile agent in a development team from constructing a spurious ‘proof’ of security—have influenced how at least some theorem-provers have been developed (see MacKenzie 2001).

A more recent influence is more generic. A major barrier to the practical use of theorem provers in computer-system verification is that their use requires large amounts of highly skilled human input. The theorem provers used in verification are automated, not automatic. Human beings guide them, for example by breaking up a desired proof into a structure of lemmas that are within the prover’s capacity, a task that requires a grasp both of what needs to be proved...
and of how the prover goes about constructing proofs. It is often a slow, painstaking and expensive process. In contrast, model checkers and other systems that implement decision procedures are automatic. Human skill may still be needed in order to represent the design of a system and its specification in such a way that model checking is feasible, but once that is done a model checker is effectively a ‘push button’ device. The attractiveness in an industrial context of such an approach is obvious. Accordingly, in recent years, the goal of automatic rather than human-guided operation has transformed research efforts in the field of automated verification. Model checking and decision procedures have become an increasingly dominant focus of attention (Alan Bundy, personal communication). The practical demands of successful industrial application have thus reshaped research in automated verification.

5. Conclusion

In a world increasingly dependent upon computer systems, the diverse domains summarized in figure 1 are of obvious practical importance. This article has argued that they also provide fruitful material for the development of a sociology of proof. Alongside the cultures of proving that constitute human-performed mathematics have grown up the other approaches listed in figure 3, as well as approaches based upon decision procedures (which do not fit within figure 3’s grid because they involve mechanized procedures that neither produce formal proofs nor resemble rigorous arguments).

The health of the domains of research listed in figure 1 is thus not simply of practical importance. These domains also constitute a set of experiments in the meaning of deductive ‘proof’. They already have a rich history that demands far more attention from historians of science than it has received. Their future practical fortunes, and the controversies they will surely spark, should also offer fruitful terrain for sociologists of science in the years to come.

The writing of this paper was supported by DIRC, the Interdisciplinary Research Collaboration on the Dependability of Computer-Based Systems (UK Engineering and Physical Sciences Research Council grant GR/N13999). The original interviewing was supported by the UK Economic and Social Research Council under the Program in Information and Communication Technologies (A35250006) and research grants R000234031 and R00029008; also by the Engineering and Physical Sciences Research Council under grants GR/J58619, GR/H74452 and GR/L37953. I owe the example of the mutilated chessboard to Alan Robinson.

References


Dijkstra, E. W. 1988 Real mathematicians don’t prove. Handwritten memo, EWD1012 (Austin, Texas, January 24).


Discussion

D. B. A. Epstein (Department of Mathematics, University of Warwick, UK). Is it feasible (in some technical sense) to formalize a typical mathematical proof? Can one estimate the complexity of the process of formalization? Can one prove for example that it is NP-hard (to formalize)? This requires formalization of the question itself.

D. MacKenzie. It does appear to be feasible in practice to formalize many typical mathematical proofs, at least the simpler such proofs. However, the process generally has to be guided by human beings: today’s automatic systems are usually quite unable to handle any other than relatively simple cases. The computational complexity of theorem proving is undoubtedly part of the reason. The underlying formal results are well beyond my competence as a sociologist, but I believe that amongst complexity-theory results in mathematical domains relevant to automated theorem-proving are (a) that the problem of checking whether formulae in propositional logic are tautologies is NP-complete, and (b) that the complexity of the decision problem in Presburger arithmetic is worse than exponential. Both results suggest constraints on the usefulness of

Phil. Trans. R. Soc. A (2005)
‘brute force’ searches for proofs. However, since complexity-theory constraints do not stop human beings proving ‘hard’ theorems, it remains possible that progress in automated reasoning techniques may lead to systems with far greater capacities than those of the present.

S. COLTON (Department of Computing, Imperial College London, UK). Does mainstream mathematicians’ seemingly genuine difficulty (or inability) to introspect on the processes they use to generate theorems and proofs add to the reason why automating rigorous argument is difficult for artificial intelligence?

D. MACKENZIE. Again, I should emphasize that I am not a technical specialist in this area, but I suspect the answer to Colton’s question is ‘yes’. One way of designing automated theorem provers to circumvent the ‘combinatorial explosion’ generated by ‘brute force’ searches would be to guide the search for proofs by incorporating the ‘heuristics’ used by human mathematicians. Attempts to do this do not, however, seem to have been consistently successful.

A. V. BOROVIK (School of Mathematics, University of Manchester, UK). The culture of (traditional) mathematics is that of openness; proofs of theorem are supposed to be open to everyone to check. Can we trust proofs produced by a commercial company on behalf of another company and kept out of the public domain?

D. MACKENZIE. In brief, no! However, the practical consequences of this may be less than might be imagined. Many of the benefits of subjecting designs to ‘formal verification’ come from the very effort to do so, for example in the way it typically brings design faults to light. The finished product of formal verification—the ‘proof object’—may thus be less important than the process of constructing it. It might also be naïve to think that many such proof objects will be subject to detailed scrutiny by people not directly involved, even if the proof objects are publicly available. Many proof objects are far from ‘readable’ in the way in which a traditional mathematical proof is readable. Nevertheless, proofs of safety-critical systems clearly should be in the public domain, and it may not be utopian to hope that eventually it will be common for them to be checked by automated proof-checking systems different from those on which they were produced.

J. G. HENDERSON (Program Notes Ltd, Pinner, Middlesex, UK). Is mathematical proof becoming an ‘act of faith’ within our culture? e.g. whilst I ‘believe’ the four-colour theorem to be true—as a software engineer, I have yet to see the computer program’s design and code for it!

D. MACKENZIE. Any complex society, such as ours, involves an extensive division of labour in which much has to be taken on trust. Few of us pause to check the soundness of the aerodynamic design of an airliner before we board it or verify the correctness of the implementation of the cryptographic protocols before submitting our credit card number to a website! There is a sense in which we normally have no practical alternative but to trust that such things have been thoroughly checked by appropriate experts. Cutting-edge research mathematics is often going to be comprehensible to only a limited number of specialists: there are many proofs (for example, the proof of Fermat’s last theorem) that even
many professional mathematicians outside the relevant specialist area will struggle to understand. Increased specialization within mathematics almost certainly means that the number of such cases is far greater now than a century or two centuries ago, and in those cases non-specialist mathematicians may indeed have no practical alternative but to trust that the scrutiny of putative proofs by their specialist colleagues has been thorough. I do not see it as a ‘problem’: to my mind, it is an inevitable consequence of specialization.

G. WHITE (Computer Science, Queen Mary, University of London, UK). What constitutes ‘formal’? There are calculations which prove theorems, but which can only be proved to be rigorous by informal argument. For example, the two-dimensional notations for traced monoidal categories (Joyal et al. 1996). And one is surely only tempted to think that there is a unitary concept of ‘formal’ if one tacitly identifies the formal with the foundational—an extremely problematic assumption.

D. MACKENZIE: The word ‘formal’ is a contested one, and it can certainly be used in senses other than that in my paper. I believe that it is also the case that when mathematicians and logicians seek to prove theorems about matters such as the soundness, completeness and consistency of formal systems, the proofs they produce are, in the terminology of the paper, often (indeed usually) ‘rigorous argument’ proofs, not formal proofs.

R. CHAPMAN (SPARK Team, Praxis, Bath, UK). When the presenter revealed the source of his definitions of ‘formal proof’ and ‘rigorous argument’ to be the MoD Interim Def-Stan 00–55, why did people laugh?

D. MACKENZIE. I suspect the reason for laughter was the way in which a fundamental ‘philosophical’ issue (the nature of ‘proof’) appeared in an apparently bureaucratic document such as a defence procurement standard. It is, of course, part of the fascination of this area that a document such as a procurement standard has to venture into the quintessentially philosophical terrain of having to say what ‘proof’ is.

R. POLLACK (School of Informatics, University of Edinburgh, UK). Donald MacKenzie showed a $2 \times 2$ matrix of different ‘proof cultures’. I pointed out that MacKenzie’s cultures had many subcultures. In particular I objected to MacKenzie’s use of the phrase ‘automated proof’ for the culture of formal proofs using computers, as a major group does not use automated search for proofs as its main approach, but machine checking of proofs developed by human users interactively with machine support.

D. MACKENZIE. I plead guilty: Pollack is correct in regard to subcultures. In a brief paper, it is impossible to do them justice. I hope I have done a little better in D. MacKenzie, Mechanizing proof: computing, risk, and trust (Cambridge, MA: MIT Press, 2001). In regard to terminology, I distinguish between ‘automated’ proof (for example, ‘automated theorem prover’) and fully ‘automatic’ proof. As I use the word, ‘automated’ proof includes the large and important category identified by Pollack: proofs which are developed on computerized systems, and checked by such systems, but in which proof construction is guided, often in detail, by human beings. Perhaps ‘semi-automated’ would thus be a better term, but the field uses the term ‘automated theorem prover’, not ‘semi-automated’.

Phil. Trans. R. Soc. A (2005)
and I followed the field’s usage. As I noted in response to Epstein’s question, the capacities of ‘semi-automated’, human-guided systems are currently much greater than those of fully automatic systems.

M. Atiyah (Department of Mathematics and Statistics, University of Edinburgh, UK). In both mathematics and physical science there is a hierarchical structure, involving a fundamental level and a higher level. In physics or chemistry we have quantum mechanics with the Schrödinger equation, but for most of chemistry or solid state physics one cannot in practice deduce everything from the foundations. In mathematics we have formal proof, but usual mathematical proof (‘rigorous reasoning’) cannot be reduced to formal proof. In both physics and mathematics we have to live with these imperfections.

D. MacKenzie: Atiyah is right, and his analogy is elegant and apposite. I would note, however, that the situation he describes is changing and will continue to change. The growing capacities of ‘semi-automated’ proving systems mean that although many areas, especially of current research mathematics, remain intractable, vastly larger numbers of rigorous argument proofs have now been reduced to formal proofs than was the case 30 years ago. Interestingly, though, the outcome of this effort supports the defence of rigorous-argument proof that I think underpins Atiyah’s comment. The semi-automated formalization of rigorous-argument proofs only very seldom uncovers serious mistakes in ‘usual mathematical proof’, at least in the context of well-trodden mathematical terrain. Such proof thus seems quite robust, and it need not be conceded that reliance upon it is an ‘imperfection’, at least in the sense of having deleterious practical consequences. The practical benefits of automatic or semi-automated proof-checking may therefore lie not primarily in mathematics, but in computer systems development, where conventional practices are far from robust in the sense of being capable of reliably producing systems free from serious flaws.

Additional reference