Reflections on Formalized Mathematics (DRAFT)

John Harrison
University of Cambridge Computer Laboratory
New Museums Site
Pembroke Street
Cambridge
CB2 3QG
England
jrh@cl.cam.ac.uk
http://www.cl.cam.ac.uk/users/jrh/

23rd August 1995

Abstract

It is generally accepted that in principle it's possible to formalize completely almost all of present-day mathematics. The practicability of *actually* doing so is widely doubted, as is the value of the result. But in the computer age we believe that such formalization is possible and desirable. In contrast to the QED Manifesto (Boyer 1994) however, we do not offer polemics in support of such a project, but merely look at existing praxis and identify what we regard as the most interesting issues.

1 What is formalized mathematics?

Mathematics is generally regarded as the exact subject *par excellence*. However, although its underlying meaning is more precise than in, say, the natural sciences or philosophy, the everyday language used by mathematicians is remarkably vague compared to that used by practitioners of some of these other disciplines. Perhaps this opposition is natural, since by its nature mathematics trammels the mind in quite precise modes of reasoning, so the vagueness seldom leads to serious ambiguities. But sometimes a knowledgeable reader is needed to separate rhetorical flourish from real content, and to appreciate the context in which the result is asserted. Consider the following examples from the early pages of Matsumura (1986):

> If $f : A \to B$ is a ring homomorphism and J is an ideal of B, then $f^{-1}(J)$ is an ideal of A and we denote this by $A \cap J$; if A is a subring of B and f is the inclusion map then this is the same as the usual set-theoretic notion of intersection. In general this is not true, but confusion does not arise.
>
> When we say that R has characteristic p, or write $char\ R = p$, we always mean that $p > 0$ is a prime number.
>
> In definitions and theorems about rings, it may sometimes happen that the condition $A \neq 0$ is omitted even when it is actually necessary.

Then on top of this is the problem of the essential ambiguity of natural languages. These might not be problems if mathematics were a small, unified subject easily graspable by a single practitioner. But on the contrary mathematics is heading increasingly in the direction of specialization, and it's not realistic to expect mathematical physicists, say, to appreciate deeply all the theoretical results in topology, differential geometry, numerical analysis and what not that they use.

There is also the question of the correctness of mathematical reasoning. Mathematical proofs are subjected to peer review before publication, but there are plenty of well-documented cases where published results turned out to be faulty. A notable example is the purported proof of the 4-colour theorem by Kempe (1879); the error in this proof was not pointed out in print till Heawood (1890). It is only with the work of Appel and Haken (1976) that the theorem has finally come to be accepted.

The formalization of mathematics addresses both these questions. We can split the project into two parts:

1. Formalizing the statements of theorems, and the implicit context (definitions etc.) on which they depend.

2. Formalizing the proofs of the results and subjecting them to precise checking.

These are usually considered together, but perhaps merely stating theorems formally without proving them has itself a valuable role. In the context of computer correctness, this corresponds to the value of formal specification even in the absence of verification, and its value is almost a commonplace in the design of critical systems. Lakatos (1976) shows vividly how theorems evolve and depend essentially on the precise definitions of the concepts concerned. Indeed, some theorems more or less transmute into definitions over the course of time.

In practice, though, we are also interested in checking the correctness of proofs. In this case the formalization of mathematics naturally becomes something a bit stronger: embedding mathematics in a foundational system of the kind invented by logicians in the early part of this century. It is a remarkable discovery, considering the parsimony of some of these systems, that this seems to be possible, and possible not merely in theory but in practice.

2 Reconstructive or normative?

There are two alternative approaches to the formalization of mathematics: reconstruct existing mathematics in a formal framework, or develop a new (better?) mathematics. It is clear that Zermelo had the former aim in mind when he introduced his set theory, but not all logicians and philosophers of mathematics take such a respectful view of existing practice. In particular, among many mathematical logicians there is a strong tendency to be:

1. Constructivist — although Brouwer had a visceral dislike of formal logic, constructivism is nowadays mainly the interest of logicians and computer scientists, not 'mainstream' mathematicians[1].

2. Minimalist — logicians are interested in discovering extremely simple systems in which a reasonable amount of mathematics can be done. For example, Takeuti (1978) has shown in detail how to reduce much of real and complex analysis and analytic number theory to a conservative extension of PA, and Fitch (1952) has claimed that his provably consistent set theory is sufficient for most applications of mathematics in physics.

[1]At least in the sense of constructive logic; classical reasoning about algorithms is not the same thing.

3. Sceptical — perhaps because of the difficulty of finding consistency proofs, many logicians are apt to be wary of strong systems, even ZF set theory. Actually there are those who do not accept as obvious the consistency of Peano arithmetic, and some ultra-finitists even find a belief in arbitrarily large numbers to be philosophically dubious.

Perhaps the formalization of mathematics provides an ideal opportunity to make a clean break, just as when one rewrites a program in a new language, it's a good opportunity to make some revisions. On the other hand, if we hope to interest many mathematicians, we need to accommodate existing mathematics. We are already trying to wreak one revolution by making the mathematicians formalize their work. Surely one revolution at a time is enough! Also, if we take a crude measure of productivity, it seems much easier to develop classical mathematics than constructive mathematics. Formalization is not a mechanical and trivial process, but a significant research problem in itself. If we add to that the new problems which arise, say, when trying to constructivize mathematics, then we are probably making life very hard for ourselves. Experience of Nuprl researchers indicates that formalization of constructive analysis and algebra is difficult, since the existing treatments leave a lot unsaid. And sometimes the textbook approaches do not scale up well in practice; for example, maintaining the standard Bishop $1/n$ bound on convergence of Cauchy sequences is very tedious when doing nontrivial things with constructive reals (like solving differential equations).

Of course to some extent classical and constructive mathematics can coexist. The approach of Bishop and Bridges (1985) to constructive analysis has the merit of being compatible with classical analysis[2] But it is more complicated: one classical theorem often splits into several constructive versions (classically equivalent). And certain constructive systems are actually in conflict with classical analysis and with each other — Bridges and Richman (1987) give an interesting comparative survey. So though it sounds very nice to say 'if a theorem can conveniently be proved constructively, then prove it constructively, and it can still be used classically', the real situation is more complicated. Those without constructive interests are unlikely to make the additional effort, anyway.

3 History

The idea of reducing reasoning to computation in some kind of formal calculus is an old dream, going back at least to Raymond Lull, and alluded to in Hobbes' "Leviathan". The first developed statement of the idea is due to Leibniz. He envisaged a 'characteristica universalis' (universal language) and a 'calculus ratiocinator' (calculus of reasoning). He even made some steps towards realizing this lofty goal, but his work was largely forgotten.

Boole and others developed formal calculi for limited parts of reasoning, but the first systems which were adequate for a substantial part of mathematics were developed by Frege (1879) and Peano. However their systems were inconsistent; the story of how Russell informed Frege of his paradox just as the second edition of Frege's book was in the press is well-known, and Zermelo (1908) waspishly rebutted Peano's criticism of the Axiom of Choice by remarking that since Peano's system was inconsistent, AC was deducible in it along with everything else.

Whitehead and Russell (1910) refined the systems of Frege and Peano with a theory of types, and proceeded to a laborious construction of some elementary

[2]Those unsympathetic to the distinction between 'all functions are continuous' and 'there are no functions which are not continuous' may find this compatibility artificial.

mathematics. This probably repelled more mathematicians than it attracted. Increasingly, logic came to be seen as a theoretical device, rather than as a practical tool. Indeed, many impressive results were obtained, showing the particular interest of first order logic from a metalogical point of view. For example, the quantifier elimination procedure for real closed fields given by Tarski (1954) is a natural generalization of well-known results on polynomial elimination. During this time, the aims of the pioneers of actually using logic were widely ignored. For example Rasiowa and Sikorski (1970) say:

> ... this mechanical method of deducing some mathematical theorems has no practical value because it is too complicated in practice.

However, the Bourbaki project amounts to a detailed derivation of all mathematics from set theory, albeit in an informal style[3]. In fact Bourbaki (1948) says:

> On these foundations, I state that I can build up the whole of mathematics of the present day; and if there is anything original in my procedure, it lies solely in the fact that, instead of being content with such a statement, I proceed to prove it in the same way as Diogenes proved the existence of motion; and my proof will become more and more complete as my treatise grows.

And it seems that natural deduction systems really do capture quite accurately the structure of ordinary mathematical proofs, as remarked by Gentzen (1935):

> It is remarkable that in the whole of existing mathematics only very few easily classifiable and constantly recurring forms of inference are used, so that an extension of these methods may be desirable in theory, but is insignificant in practice.

Both the fact that mathematics can be reduced to set theory and the fact that something like natural deduction can capture practically all existing mathematical proofs seems almost universally accepted, *in principle*. We may stop to query exactly where this intuition comes from. But rather than look for problems where none are believed to exist, we prefer to ask: why not in practice?

4 Enter the computer

We believe that because of computers, formal mathematics has become a practical possibility. The tedious, low-level details can be left to the machine, which can also provide an independent check on correctness. Several pioneering projects for the computer formalization of mathematics appeared in the 70s. Perhaps only by that time were computers and programming languages sufficiently powerful and flexible to perform such work, or sufficiently widespread to be devoted to such experimentation as distinct from 'serious' applications of numerical analysis and business data processing [4].

A notable example was the AUTOMATH project led by de Bruijn (1980). In this project, a language for formal mathematics was designed, together with a computer system for checking the correctness of formal texts. In the light of our above

[3]An interesting critical survey of the Bourbaki approach to foundations is given by Mathias (1991).

[4]Only a select few applications (maybe the World Wide Web is one) are sufficiently important to *drive* technology. Those interested in unusual applications usually need to scavenge for the leavings from their table.

remarks, it's interesting to note that AUTOMATH was ahead of its time technologically, always struggling to make the computer keep pace with the ideas. Significant parts of mathematics were proof-checked; for example van Bentham Jutting (1977) formalized the famous book on the construction of the real number field by Landau (1930). The history of the project and the lessons derived from it are detailed by Nederpelt, Geuvers, and de Vrijer (1994). Though the project was important and influential, the AUTOMATH system is hardly used today.

Perhaps one factor contributing to AUTOMATH's decline is that it used some rather original notational and foundational ideas (the banishment of named bound variables and the exploitation of the Curry-Howard correspondence [5], for example). The MIZAR project, which began a little later, is still quite vital today, and one of the reasons may be that, as its originator Trybulec (1978) explicitly stated, it attempted not to depart too radically from the usual accepted practices of mathematics. Though until recently it was not well known among the theorem proving community in the West, MIZAR has been used to proof-check a very large body of mathematics, spanning pure set theory, algebra, analysis, topology, category theory and various unusual applications like the analysis of Go, mathematical puzzles and computer models. Selected Mizar articles are automatically abstracted and printed in human-readable form in the journal 'Formalized Mathematics'[6].

At about the same time, other computer theorem-proving ideas were being developed. The development of fully automatic provers for first order logic, mostly based on resolution, is well-known. Some such systems, notably Otter, have achieved some impressive results, settling open problems in mathematics, or producing quite new theorems (Argonne National Laboratories 1995).

However general first order provers tend to be inefficient in restricted areas compared to specialized algorithms. Numerous special algorithms which tackle certain areas of mathematics more efficiently have been devised, for example for tautology checking (Bryant 1986), linear arithmetic (Shostak 1979), equational reasoning (Knuth and Bendix 1970) and deduction from universal assumptions in first order logic with equality (Nelson and Oppen 1980), not to mention the panoply of techniques used by computer algebra systems. The NQTHM prover, developed by Boyer and Moore (1979), is a general theorem prover for quantifier-free arithmetic; since the logic is quite simple, powerful proof automation (e.g. for induction proofs) is feasible. Though the logic is restricted, it is amazingly expressive in the hands of a skilled practitioner, and NQTHM has been used in a wide variety of applications.

There are reasons for dissatisfaction with very low-level proof checking à la AUTOMATH, simply because of the tedium involved and the unreadability of the resulting proofs. But fully automatic theorem provers, though they work well in some situations, are often hard to use in practice. Usually they need to be led to the required result via a carefully graded series of lemmas, each of which they can prove automatically. Choosing this series often requires intimate knowledge of the system if one is not to lead it up a blind alley. And if one is interested in the proof, a completely automated prover may not be what is wanted.

These disadvantages of both approaches led Milner to develop Edinburgh LCF (Gordon, Milner, and Wadsworth 1979). In LCF-like systems, the ML programming language is used to define data types representing logical entities such as types, terms and theorems. A number of ML functions are provided which produce theorems; these implement primitive inference rules of the logic. The use of an abstract type of theorems with these inference rules as its only constructors ensures that theorems can only be produced this way. However the user is free to write arbi-

[5] Independently discovered by de Bruijn.

[6] For more information about this journal, which is surprisingly inexpensive, contact: Fondation Philippe le Hodey, MIZAR Users Group, Av. F. Roosevelt 134 (Bte7), 1050 Brussels, Belgium, fax +32(2)6408968.

trarily complex proof procedures which ultimately decompose to these inferences. So in principle, any automatic proof procedure can be translated into an LCF program, while guaranteeing that even if the program has errors, no false 'theorems' will arise (the program may fail, of course). It seems in practice that this can be done reasonably efficiently; this issue is analyzed in some detail by Harrison (1995).

Many of the systems described here were not primarily developed in order to formalize textbook mathematics. For example, the HOL system was designed expressly for the purpose of verifying computer hardware. The verification of hardware and software is a promising new application area for theorem provers, and many, notably NQTHM and HOL, have been applied successfully to it. This is still mathematics of a sort, and often substantial pure mathematics is needed in a supporting role — witness the work of Harrison (1994) for example. But the theorems proved tend to be different; shallower but more technically complex. It may be that proof techniques which work well in pure mathematics are unsuitable for verification, and vice versa. The LCF style offers the most hope here, since its inherent programmability allows multiple proof styles to be based on the same logical core.

5 How much automation do we want?

Despite some *tours de force* by automated theorem provers, the emphasis in applications is moving increasingly towards interactive systems, where the user guides the prover. In mathematics this seems reasonable, since the proofs are important, and we don't usually want an *ex cathedra* assertion that something is true from a machine. On the other hand, we also don't want to have to direct proofs at very low levels, so some automation of 'obvious' steps is required.

As pointed out by John McCarthy, there is an important distinction between what is *mathematically* obvious and what is *logically* obvious. In mathematical proofs we want to skip over levels of reasoning which we can intuitively see to be trifling; but this intuition may be based on quite sophisticated domain-specific knowledge, and not be at all obvious in the sense of being technically simple logically. It seems that we need more work on automating what is mathematically obvious, whereas existing technology may already be capable of automating much larger logical steps than we can perceive as obvious (e.g. very large tautologies). An interesting discussion of just what is logically obvious is given by Rudnicki (1987).

However, there are certain steps which are completely routine for a machine to check, and baffling to a human being, which we probably *do* want to automate, simply because carrying out the proof in detail is no more illuminating to the human than knowing, based on confidence in the correctness of the theorem prover, that it is valid. A good example is the following identity:

$$\begin{array}{l}(x^2 + y^2 + z^2 + w^2)(x'^2 + y'^2 + z'^2 + w'^2) = \\ (xx' + yy' + zz' + ww')^2 + \\ (xy' - yx' + wz' - zw')^2 + \\ (xz' - zx' + yw' - wy')^2 + \\ (xw' - wx' + zy' - yz')^2\end{array}$$

It's used as a lemma in the proof that every integer is expressible as the sum of four squares. But proving it, at least simply by multiplying it out in a routine way, is no more illuminating than checking the result of a numerical calculation by hand. Of course a proof which exhibited a deep, simple reason for the above (e.g. something based on quaternions) is a different matter.

In any case, automating what is mathematically obvious is itself a research project in the artificial intelligence line. In the short term, it's probably better to accept a lower level of automation. As theorem-proving technology improves, it should be possible to simplify proofs automatically by excising intermediate steps which are now unnecessary for the machine. There is the important lesson that the theorem prover must admit extension with domain-specific reasoning without compromising soundness. We believe that this problem has already been solved by LCF-style systems.

6 Reduction to a foundational system

What do we gain by the reduction of mathematics to a simple foundational system like set theory? From a philosophical point of view, quite a lot; in particular the consistency of practically all of mathematics can be reduced to the problem of whether ZF set theory is consistent. In this way our foundational worries tend to come home to roost in set theory.

From a practical and methodological point of view the answer is more complicated. We should certainly avoid regarding the implementability of mathematical concepts as sets as constituting some kind of ontological discovery – this view is rightly parodied by Benacerraf (1965). Rather, it's an interesting and surprising *practical* discovery about the flexibility of set theory, to be compared with Gentzen's discoveries about the flexibility of a simple inferential apparatus. We can distinguish several levels of usefulness.

1. Kuratowski showed that ordered pairs could be defined by $(x, y) = \{\{x\}, \{x, y\}\}$ (a more complicated alternative was earlier produced by Wiener (1914)). However it seems that this tells us nothing new about ordered pairs, except for pathological theorems which we don't want anyway. Indeed in stratified set theories like Quine's NFU (Holmes 1995) and work in coinductive definitions (Paulson 1994), another definition is normally used.

2. von Neumann's definition of the ordinals is normally used to identify the natural numbers with the set ω of finite ordinals. This means that $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$ etc. Although this is also quite arcane and ad-hoc (do we really want $0 \in 1$?) it does have the benefit of streamlining and simplifying a lot of theorems about natural numbers and integrating them with their counterparts for ordinals and cardinals. Indeed Halmos (1963) finds the use of 2 as a canonical 2-element set sufficiently convenient to remark on it.

3. We identify sequences indexed by the natural numbers, as used for example in real analysis, with functions out of $\mathbb{N}$. It's quite conceivable that one could work with sequences without making explicitly the realization that the two concepts are identical. However it seems there are no reasons for forcing a distinction, so here the formalization is a unifying and simplifying force.

4. Variable binding constructs (extensional ones anyway) such as summation, integration and differentiation, can all be regarded as a higher order operator applied to a lambda-term. That is, $\Sigma_{i=1}^{10} i$ is 'really' $SUM(1, 10, \lambda i. i)$ and $\frac{d}{dx}x^2$ is really $DIFF(x, \lambda x.\, x^2)$. This really is a useful clarifying force, confronting us with awkward constructs where the everyday notation confuses free and bound variables or a function with its value. Consider for example the chain rule for differentiation, often written as:

$$\frac{d}{dx} f(g(x)) = f'(g(x))g'(x)$$

We shouldn't be too dogmatic about what is the 'natural' formalization of informal (or preformal) notions[7]. Furthermore, we do not claim that reductionism always reflects the most interesting aspects of mathematics; in fact as MacLane has remarked, it can give a rather one-sided view of mathematical activity. But this is no more a problem than the fact that formal grammar rules do not contribute to poetry.

7 The choice of a foundational system

As already indicated, the intellectual trend has in the past moved away from the idea of actually using formal systems towards using them at one remove. This is responsible for the almost exclusive concentration on first order logic, despite its obvious defects[8]. In general we should not be awed by existing foundational systems (as Kant was by Aristotelean logic) but should be prepared to invent our own new ideas. Even quite trivial syntactic changes, like making quantifiers bind more weakly than other connectives (most logic books adopt the opposite convention) can help with the usability of the system!

There are two major foundational schemes for mathematics: set theory and type theory. These can be traced back to Zermelo and Russell, respectively, and to their different reactions to the paradoxes of naive set theory[9]. Zermelo felt that the Russell set $\{x \mid x \notin x\}$ was paradoxical because (since so many sets *don't* have the property $x \in x$) it was simply *too big* to be a set. Indeed, Cantor had already distinguished between 'consistent' and 'inconsistent' multiplicities, more or less on the basis of size (whether they were equinumerous with the class of all ordinals). This is called the 'limitation of size' doctrine. Russell, on the other hand, felt that the problem was with the circular nature of the definition (considering whether a set is a member of itself), and proposed separating mathematical objects into layers, so that it only makes sense to ask whether an object at layer n is a member of an object of layer $n + 1$, so the Russell set is deprecated as meaningless on syntactic grounds.

In fact the distinction between set theory and type theory is not clear-cut. Fraenkel added the Axiom of Foundation to Zermelo's set theory, and this gave rise to an intuitive picture of the set-theoretic universe as built up in layers in a wellfounded way. (A picture which is nowadays *de rigeure* among set theorists, and even presented as the motivation for the ZF axioms!) This really amounts to an admixture of type theory, although the layers are cumulative (each layer includes the previous one) and continued transfinitely. What's more, Quine has proposed set theories which, while not having a global notion of type, have a kind of local type discipline based on stratified formulas in the comprehension schema. On the other hand, it seems that Martin-Löf, one of the standard-bearers of type theory, now prefers to refer to his systems as 'Martin-Löf set theory'.

We will concentrate on the *practicalities* of set theory and type theory as foundations, but it's worth pausing a while to examine the matter philosophically. It seems that there really is a notion of type at work in everyday mathematics; we do think of real numbers as something different from sets of complex numbers, and flattening the whole universe of mathematical discourse into a single undifferentiated realm doesn't seem natural. However the same can be said of any reductive foundationalist programme. In any case, whatever the intuitive appeal of types,

[7] In particular we should avoid dubious historical arguments designed to justify the presence or absence of types or sets.

[8] Note that the systems of the pioneers like Frege and Peano were clearly higher order. The basic Peano axioms in first order logic give only a trivial decidable theory — it's necessary to include addition and multiplication as primitive to get anything remotely interesting.

[9] In fact Russell's paradox was independently discovered by Zermelo at about the same time.

ZFC seems to indicate that they are not necessary to ensure consistency[10]. There are other special attributes that mathematical objects may have, and 'type' should perhaps just be one of these. Types as primitive can also present awkward choices over whether to formalize some set as a new type or as a subset of (or predicate on) an existing type. In any kind of collaborative programme, it's good policy to avoid giving people too many unnecessary choices (the C programming language and the Macintosh user interface both reflect this philosophy). While on the subject of programming, we might say that sticking to a type discipline is like structured programming, but nevertheless inside the machine, it's all implemented using the equivalent of `GOTO` statements.

The most direct descendant of Russell's type theory, as simplified by Ramsey (1926) and Church (1940), is what is now called 'simple type theory'. This is successfully implemented in the TPS prover (Andrews, Bishop, Issar, Nesmith, Pfenning, and Xi 1994) and, with the addition of a form of polymorphism, in HOL (Gordon and Melham 1993). True to its name, simple type theory is a simple system, formally and conceptually. In fact, only a higher-order equational calculus with λ-binding need be taken as basic; all the logical operations can be defined as higher order functions given some 2-element type of propositions[11] as shown by Henkin (1963). The resulting system is in many ways quite close to informal mathematics; the types are often helpful and natural, and functions are first-class objects. However it is rather inflexible in some ways; for example there are no dependent types (where a type depends on a term, e.g. a type of $n \times n$ matrices for variable n). So although convenient for some parts of mathematics (particularly concrete situations like number theory and real analysis), it becomes rather inflexible for other parts, e.g. abstract algebra.

There are many more advanced type theories which allow dependent types and other more sophisticated type constructors. Many of these are descended from Martin-Löf's constructive type theories, which are based on the Curry-Howard correspondence between propositions and types, so type constructors correspond to logical operators, e.g. the product type to conjunction. However there is no reason why these systems cannot be used classically. It seems that they are much more flexible than simple type theory, but they are also much more complicated. For example, Nuprl's type theory has over 200 inference rules. Quite how easy it is to formalize mathematics in these systems is still an open question. Many of the researchers are also constructivists, and it is not easy to separate the peculiarities of using constructive logic from the peculiarities of using type theory.

Set theory (e.g. ZFC) is probably the most obvious choice as a foundation. It isn't as simple as simple type theory, and as our earlier remarks indicate, is a rather ad-hoc solution to the paradoxes. However it is reasonably simple, and appears to be flexible enough for just about all present-day mathematics. Importantly, it is much better known among mathematicians than type theory[12]. There are still question marks over its suitability, even with additional axioms, for some parts of category theory (the category of all sets, etc.) Moreover, some theorems such as recursion on a wellfounded relation need to be stated in an artificial manner using locutions such as proper classes. There is no type theory to hinder us, but none to help us either. In practice it seems likely that some quite simple locutions using set constraints are sufficient to give most of the convenience of types, without

[10]The Axiom of Foundation was not added for this reason — of course one doesn't make a system consistent by *adding* an axiom!

[11]Even intuitionistically, implication and universal quantification are enough to define all the usual connectives, whereas in first order logic they are all independent except for negation — see Prawitz (1965).

[12]Although among computer scientists the opposite is probably true; only a few programming languages, e.g. BCPL, are untyped.

irrevocably tying the user to them.

The exact set theory to choose is open to debate. The 'standard' foundation for mathematics is ZFC based on first order logic. However basing the system on higher order logic brings some advantages. Allowing higher order bound variables in axioms like Replacement (Gordon 1994) extends the power of the set theory[13] but makes the axiomatization much neater, without artificial locutions. If the relevant axiom schemas are restricted appropriately, the system is conservative over first order ZFC, but may still offer some advantages (Corella 1990). There is also the possibility of retaining first order logic, but making classes first-class objects, as in NBG. Some descendants of Quine's set theories admit an embedding of ZF, while providing some convenient additional facilities like a set universe which is a Boolean algebra. Finally, it is possible to add extra axioms, e.g. the Tarski-Grothendieck axiom of universes, or take them away — for example Replacement is seldom used in mainstream mathematics[14].

We can judge a foundational system by several criteria: aesthetic and philosophical appeal, flexibility, simplicity and closeness to ordinary mathematical language. These are partly subjective of course. And unfortunately many of them, especially the last two, tend to be in conflict. It seems that at present set theory is the winner, but with more research on different foundational systems, this may change. We do not believe the last word has been said.

8 Definitions and locutions

In practice, even in quite rich foundational systems, let alone in pure set theory, representations of mathematical objects becomes unwieldy. For example, Bourbaki (1968) estimate[15] that even the number 1 if fully expanded would occupy thousands of symbols in their system. So some kind of definitional mechanism is needed, at least for human consumption, and possibly to stop overflow even for a machine! The fundamental property of definitions is that it should be possible in principle to do without them, though we need to say carefully precisely what this means.

The simplest kinds of definitions are simply shorthands for more complicated expressions, for example 2 rather than $\{0,1\}$, (x,y) rather than $\{\{x\},\{x,y\}\}$ and $\exists! x. P[x]$ rather than $\exists x. P[x] \wedge \forall y. P[y] \Rightarrow y = x$. We can understand such definitions in two ways, either as metalogical abbreviations used to make formal texts more palatable to the reader, or as a method of sanctioning the addition (in the object logic) of new equational axioms. Generally, the latter is more efficient, since it keeps the underlying terms small, but the former is more flexible: the defined notion need not correspond to any kind of logical object, nor respect any underlying type system. If fact the Nuprl system allows boths kinds of definition.

Assuming that the object language has facilities for variable abstraction (something like lambda-calculus), then any definitional constructs of this kind can be reduced to new abbreviations or axioms of the form $c = t$ where c is a new constant name any t any closed term. For example, instead of writing $f(x) = E[x]$, we use $f = \lambda x.\, E[x]$. In any reasonable logical system the addition of such axioms makes no difference: in principle we could do without them (replace c by its expansion everywhere), and we do not derive any new facts not mentioning c. They thus fulfil all the criteria we could demand of definitions.

There is a more general means of extending the logic with new constant symbols:

[13] Perhaps to something like Morse-Kelley? The author doesn't know enough set theory to be sure.

[14] Though it streamlines some parts of pure set theory: every woset is isomorphic to a von Neumann ordinal, recursion is admissible on a wellfounded class-relation etc.

[15] See E III.25.

using names for things which have been proved to exist. If we can prove $\vdash \exists x.\, P[x]$, then adding a new constant c and an axiom $\vdash P[c]$ is a conservative extension of most logical systems, in the sense that anything not involving c could equally well be deduced without using c or its axiom. (Simply because something like the natural deduction $\exists$-elim rule is usually valid.) It's not quite conservative in the sense that it can be written away from formulas, though any formula $\vdash Q[c]$ can be read as $(\exists x.\, P[x]) \wedge \forall x.\, P[x] \Rightarrow Q[x]$, or something like that. Given a theorem $\forall x.\, \exists! y.\, P[x, y]$, it is usually conservative to add a new function symbol f and an axiom $\vdash \forall x.\, P[x, f(x)]$. However if unique existence is weakened to existence, this can smuggle in the Axiom of Choice. The situation for constructive logics is even more delicate.

A useful feature in a practical logic is a *descriptor*. For example, we may allow a new term constructor $\iota x.\ P[x]$, read 'the (unique) x such that $P[x]$', together with an axiom $\vdash (\exists! x.\ P[x]) \Rightarrow P[\iota x.\ P[x]]$, or $\varepsilon x.\ P[x]$, read 'some x such that $P[x]$', together with an axiom $\vdash (\exists x.\, P[x]) \Rightarrow P[\varepsilon x.\, P[x]]$. These are called definite and indefinite descriptors, respectively. Note that the latter is something like the Axiom of Global Choice if we allow $P[x]$ to contain other free variables, and even the former is weakly nonconstructive because it effectively allows us to introduce a total function even though $\exists x.\, P[x]$ may be conditional on some other property. A descriptor is very useful for rendering informal notions of 'the x such that ...' (it's much used by Bourbaki), and can even be used to implement lambda-binding, set abstraction and so on, e.g. define $\lambda x.\, E[x]$ to be $\varepsilon f.\, \forall x.\, f(x) = E[x]$.

If we can prove $\vdash \exists x.\ P[x]$ and we have an indefinite descriptor, then defining $c = \varepsilon x.\ P[x]$ gives us $\vdash P[c]$. Conversely, for any term t we can prove $\exists x.\ x = t$, so the existential kind of definition always sanctions the equational kind. There is thus a close relationship between the two kinds of definition.

There are more complex definitional mechanisms which cannot be expanded away in such a simpleminded manner, but can nevertheless be understood as inessential abbreviations which could in principle be done without. These are usually referred to as *contextual definitions*. A good example is the use of ZF proper classes, where statements about 'set' membership are really an encoding of a logical fact. Another example: cardinal numbers can in principle be done without, since all statements about arithmetic and equality of cardinals may be regarded as statements about equinumerosity etc. of sets. For example instead of $|X| = |Y|^2|Z|$ we can write $X \sim (Y \times Y) \times Z$. The most sophisticated example of all is using category-theoretic language like 'the category of all sets' purely as a syntactic sugar on top of quite different statements. One of the interesting questions in formalized mathematics is the extent to which this is possible.

In any case, we shouldn't lay too much stress on the theoretical eliminability of definitions. It's mainly a way of assuring ourselves that we aren't 'really' extending the logic. Life without definitions would be unbearably complicated, and anyway the whole significance of a theorem may subsist in the definitions it depends on.

There are other forms of definition, for example definition by recursion and definition of inductive sets or relations. However given a mathematical theorem justifying the existence of such a function, it can then be defined using one of the mechanisms given above. In programmable systems, the tedious matter of deriving the required case of this theorem can be automated completely.

9 Partial functions and undefined terms

In informal mathematics, our minds filter out — even subconsciously perhaps — troublesome degenerate cases. However in a formal treatment we must address these matters. One of the most interesting questions is: what does it mean to apply a function f to a value x outside its domain? For example, what is 0^{-1} in the reals?

There are numerous different approaches. The simplest is to regard $f(x)$ for $x \notin dom(f)$ as something arbitrary; in effect we are expanding the domain of f but saying nothing about the values it takes there. Alternatively, we can specify an explicit value for values outside the domain; e.g. we might decide to define $0^{-1} = 0$. This latter approach can be exploited by choosing a particularly convenient value, e.g. $0^{-1} = 0$. Then we have lots of nice unconditional equations, like $\forall x \in \mathbb{R}.\, (x^{-1})^{-1} = x$ and $\forall x \in \mathbb{R}.\, -x^{-1} = (-x)^{-1}$ and $\forall x, y \in \mathbb{R}.\, (xy)^{-1} = x^{-1}y^{-1}$. However some may regard these theorems as pathological, obscene or simply untrue. Actually even if an arbitrary value is chosen, these freak theorems show up occasionally. If we define, as one normally would, $x/y = xy^{-1}$, then $0/0 = 0$, because whatever 0^{-1} might be, it's some real number, and multiplying it by zero gives zero! In an untyped system this problem happens less (in the example given we wouldn't even know that $0^{-1} \in \mathbb{R}$), but does not disappear completely.

There is a more serious disadvantage of this scheme. (In constructive systems there is yet another: membership of the domain may be undecidable, so the above solution simply isn't available.) In some mathematical contexts, writing down $f(x) = y$ is taken to include an implicit assertion that $x \in dom(f)$. For example, when we write:

$$\forall x.\, \frac{d}{dx} sin(x) = cos(x)$$

we take this to include an assertion that sin is in fact differentiable everywhere. But if the differentiation operator is total, it will yield a value, regardless of whether the function is actually differentiable at the relevant point. It might accidentally happen that the above equation were true even if the function weren't differentiable! This situation becomes even more serious when constructs like this are nested, e.g. in differential equations. They need to be accompanied by a long string of differentiability assumptions, which in informal usage are understood implicitly.

What are the alternatives? If we have a typed logic, then we can simply make $f(x)$ a typing error when $x \notin dom(f)$; that is, the term in not syntactically well formed. This avoids the problems above, but it might mean that in complicated situations in analysis, the types become outrageously complicated, since they need to excise all the singularities. It also makes it difficult to use constructs which are permissive of point singularities (for example, it often makes sense to integrate such functions). It also means that theorems like $\forall x \in \mathbb{R}.\, tan(x) = 0 \Rightarrow \exists n \in \mathbb{N}.\, x = n\pi$ don't hold, because the left is a type error if x is one of tan's singularities.

The most sophisticated alternative of all is to have a special logic which allows certain terms to be undefined, as in the IMPS system (Farmer, Guttman, and Thayer 1990). This is more or less the same (there are some differences in detail) as taking an extra 'undefined' element on top of a conventional logic, so the result of a function application may be this special element. The undefined value propagates up through terms, so a term with an undefined subterm is itself undefined. In IMPS, predicates involving an undefined argument become *false*. For example, $a = b$ means 'a and b are both defined and are equal'. One might question these choices, but since a definedness operator is provided, one can invent one's own bespoke notion of equality. In particular, IMPS supports 'quasi-equality', where $a = b$ means 'either a and b are both undefined, or are both defined and equal'. In some parts of mathematics, this is probably the usual convention, but it has some surprising consequences, e.g. the logical equivalence $x = y \equiv x - y = 0$ is false for quasi-equality. One could even invent circumstances in which $\forall x \in \mathbb{R}.\, (x^2 - 1)/(x - 1) = x + 1$ was desired behaviour (wherever the two sides are both defined, they are equal).

We spoke above of the informal convention in mathematics. It seems that many authors are actually doing something like defining the equality predicate contextually. This is a far cry from what one would expect, but it seems an inescapable conclusion from phrases like 'if y is the unique value with $(x, y) \in f$, we write $f(x) = y$'. Set theory texts are occasionally more explicit, e.g. they may say '$f(x)$ is the (unique) y such that $(x, y) \in f$'. One formalization of this is the descriptor term $\varepsilon y.\ (x, y) \in f$, but, depending on the precise semantics of the descriptor and the logic, this can mean different things.

10 User interaction

We want to avoid making systems which are difficult for mathematicians to use. This means we should try to keep the interface close to the familiar style of mathematics books. Perhaps even small changes in notation (e.g. '&' instead of '$\wedge$') could help a lot. Alternative styles of expression should be permitted ('for real x', 'for any $x \in \mathbb{R}$' or 'for any real number x') and perhaps even a few redundant 'noise words'. For example, Mizar allows the word 'that' (e.g. 'suppose X' or 'suppose that X') but essentially ignores it. However we must avoid the error of making the interface *too* flexible and intelligent, for two reasons:

1. If there is an additional complicated layer of interpretation between the user and the underlying logic engine, then we are bringing back all the potential for vagueness and ambiguity that we are trying to banish. It will require substantial work for a new user to understand the relationship between what is seen at the user level and what exists inside the machine.

2. Making an intelligent flexible interface is probably just too hard; we're getting embroiled in a difficult AI project. The success of Unix partly because it is formal, simple and makes no attempt to be clever. Anyway, opinions and fashions about what is best change quickly — remember the light pens in 60s visions of future computers?

Even though the system itself may still be reasonably formal, there is plenty of potential for translating automatically or semi-automatically into something readable by any mathematician. Mizar's journal 'Formalized Mathematics' points the way here. The translation is by no means sophisticated, but works quite well (admittedly it is only the statements of theorems, not their proofs, at present). In line with the above remarks on alternative modes of expression, sometimes different alternatives are chosen using a random number generator, to give something like the 'elegant variation' discussed by Fowler (1965). ('Now we prove', 'The following hold', 'The following propositions are true', ...)

It's much easier to translate from a formal language to (possibly stilted and artificial but correct) natural language than vice versa. So mathematics expressed in a formal way is a valuable resource: it could be automatically translated into many different languages, avoiding some of the difficulties of communication among the world's mathematicians (most of whom are now forced to use English; a few decades ago German or French). It may even be that the formal language would itself become an accepted interlingua for mathematical texts; though it would be rather spartan and colourless, the success of international chess publications like Informator, which are languageless, is worth noting.

11 Experience of formalized mathematics

There has already been considerable experience of formalized mathematics in widely differing theorem proving systems. It's important to absorb the lessons which this teaches about the process in general and the strengths and weaknesses of particular formal systems and proof methodologies. Most of the following is based on personal experience, but I know that much of it will be echoed by other practitioners.

Formalizing mathematics is sometimes quite difficult. It's not uncommon to make a definition, set about proving some consequences of it, fail, and then realize that the definition was faulty. Usually this is because some trivial degenerate case was ignored, sometimes because of a more substantive misunderstanding. This can be tedious, but at other times it is clarifying, and forces one to think precisely.

It's an interesting question whether modest changes to normal mathematical style can render it completely formal, or whether it will always be an unbearably tedious load (some say that 'rigour means rigor mortis'). It certainly doesn't seem too much trouble always to explicitly say that p is a prime, that n is a natural number, that when we talk about the poset X we are talking about the partial order $\leq_X$, or that we are considering a function not its value. But it may be that the accumulation of such details renders mathematics too inelegant. Only experience can tell.

Unsurprisingly, formal proofs are long and tedious, and tend to look quite unreadable compared to their textbook models. However we believe several existing systems such as Mizar show that one can still provide fairly readable proof texts, and the main difficulty in practice is automating just the right parts of the reasoning. In principle, there seems no reason why they cannot be brought down to something close to the length of textbook proofs; even less perhaps.

In any case (though this comment might seem to smack of desperation) there are advantages in the fact that proofs are more difficult. One tends to think more carefully beforehand about just what is the best and most elegant proof. One sees common patterns that can be exploited. All this is valuable. Compare the greater care one would take when writing a book or essay in the days before word processors!

In practice formalized mathematics is quite addictive (at least, for a certain kind of personality). This may be similar to the usual forms of computer addiction, but we believe there is another important factor at work. Since one can usually see one's way intuitively to the end of the proof, one has a continual mirage before one's eyes, an impression of being almost there! This must account for the times people stay up till the small hours of the morning trying to finish 'just one last proof'.

Many of the problems are not connected with the difficulty of formalizing and proving theorems, but with organizing the resulting database of theorems. Since one will often need to refer back to previous theorems, possibly proved by other people, it's useful to have a good lookup technique. In HOL, theorems are named (rather than numbered as in Mizar) which makes things a bit easier; but some theories (notably natural number arithmetic) do not have a consistent naming convention. It's very useful to be able to find a theorem by its structure, e.g. find a theorem that will match a given term, or contains a certain combination of constants. HOL and IMPS have such facilities.

Then there are the social problems: when different people work at formalizing mathematics, they will choose slightly different conventions, and perhaps reinvent the same concepts with different names. For example, one may take $<$ as the basic notion of ordering, one may take $\leq$. Perhaps there needs to be some kind of central authority (like the Mizar 'library committee') which establishes certain canons, but this is rife with potential for disagreement and petty politicking. We should reiterate our earlier remarks here: there are real advantages in only allowing users one way to do things!

And in practice, a given arrangement is not set in stone; people may want to add more theorems to existing libraries, perhaps slightly modify the theorems on which other libraries depend. It can be very difficult to write machine proofs which are insensitive even to trifling changes in the assumptions. This problem has been studied by Curzon (1995) with respect to large hardware verification proofs. How much worse if we want to change the foundational system more radically (e.g. constructists might like to excise the use of nonconstructive principles in proofs). It seems one should try to write proofs which avoid too many unnecessary parochial assumptions (e.g. the assumption that there are von Neumann ordinals floating around). This is comparable to avoiding machine dependencies in programming.

Only with more practical experience, ENOD[16] in the words of Kreisel (1990), can we really decide the best way to go. Perhaps nobody is farsighted enough to anticipate all the problems that will arise, and we will only discover them by colliding heavily with them. For example, there are still many questions over the formalization of category theory, the role of types, the use of partial functions, and many other things. We might find that some problem domains, e.g. those demanding a lot of geometric intuition, are very hard to formalize, present new problems and require new ideas.

Acknowledgements

Thanks to Andrzej Trybulec, Czeslaw Bylinski, Grzegorz Bancerek and Roman Matuszewski for many interesting discussions and for looking after me during my visit to Białystok, where most of this paper was written. I haven also benefited from conversations with Mike Gordon and with other people at the QED conference.

References

Andrews, P. B., Bishop, M., Issar, S., Nesmith, D., Pfenning, F., and Xi, H. (1994) TPS: A theorem proving system for classical type theory. Research report 94-166, Department of Mathematics, Carnegie-Mellon University.

Appel, K. and Haken, W. (1976) Every planar map is four colorable. *Bulletin of the American Mathematical Society*, **82**, 711–712.

Argonne National Laboratories (1995) New results with ANL's ATP software. Unpublished; available on the Web as `http://www.mcs.anl.gov/home/mccune/ar/new_results/index.html`.

Benacerraf, P. (1965) What numbers could not be. *Philosophical Review*, **74**, 47–73. Reprinted in Benacerraf and Putnam (1983), pp. 272–294.

Benacerraf, P. and Putnam, H. (1983) *Philosophy of mathematics: selected readings* (2nd ed.). Cambridge University Press.

van Bentham Jutting, L. S. (1977) *Checking Landau's "Grundlagen" in the AUTOMATH System.* Ph. D. thesis, Eindhoven University of Technology. Useful summary in Nederpelt, Geuvers, and de Vrijer (1994), pp. 701–732.

Biggs, N. L., Lloyd, E. K., and Wilson, R. J. (1976) *Graph Theory 1736–1936.* Clarendon Press.

Bishop, E. and Bridges, D. (1985) *Constructive analysis*, Volume 279 of *Grundlehren der mathematischen Wissenschaften.* Springer-Verlag.

[16] Experience Not Only Doctrine.

Bourbaki, N. (1948) Foundations of mathematics for the working mathematician. *Journal of Symbolic Logic*, **14**, 1–14.

Bourbaki, N. (1968) *Theory of sets.* Elements of mathematics. Addison-Wesley. Translated from French 'Théorie des ensembles' in the series 'Eléments de mathématique, originally published by Hermann in 1968.

Boyer, R. S. (1994) A mechanically proof-checked encycopedia of mathematics: Should we build one? Can we? In Bundy, A. (ed.), *12th International Conference on Automated Deduction*, Volume 814 of *Lecture Notes in Computer Science*, Nancy, France, pp. 237–251. Springer-Verlag.

Boyer, R. S. and Moore, J S. (1979) *A Computational Logic.* ACM Monograph Series. Academic Press.

Bridges, D. and Richman, F. (1987) *Varieties of Constructive Mathematics*, Volume 97 of *London Mathematical Society Lecture Note Series.* Cambridge University Press.

de Bruijn, N. G. (1980) A survey of the project AUTOMATH. In Seldin, J. P. and Hindley, J. R. (eds.), *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus, and Formalism*, pp. 589–606. Academic Press.

Bryant, R. E. (1986) Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, **C-35**, 677–691.

Church, A. (1940) A formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, **5**, 56–68.

Corella, F. (1990) Mechanizing set theory. Technical Report 232, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK. Author's PhD thesis.

Curzon, P. (1995) Tracking design changes with formal machine-checked proof. *The Computer Journal*, **38**, 91–100.

Farmer, W., Guttman, J., and Thayer, J. (1990) IMPS: an interactive mathematical proof system. In Stickel, M. E. (ed.), *10th International Conference on Automated Deduction*, Volume 449 of *Lecture Notes in Computer Science*, Kaiserslautern, Federal Republic of Germany, pp. 653–654. Springer-Verlag.

Fitch, F. B. (1952) *Symbolic Logic: an introduction.* The Ronald Press Company, New York.

Fowler, H. W. (1965) *A Dictionary of Modern English Usage* (2nd, revised ed.). Oxford University Press. Revised by Sir Ernest Gowers.

Frege, G. (1879) *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Halle. English translation, 'Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought' in van Heijenoort (1967), pp. 1–82.

Gentzen, G. (1935) Über das logische Schliessen. *Mathematische Zeitschrift*, **39**, 176–210. This was Gentzen's Inaugural Dissertation at Göttingen. English translation, 'Investigations into Logical Deduction', in Szabo (1969), p. 68–131.

Gordon, M. (1994) Merging HOL with set theory: preliminary experiments. Technical Report 353, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK.

Gordon, M. J. C. and Melham, T. F. (1993) *Introduction to HOL: a theorem proving environment for higher order logic.* Cambridge University Press.

Gordon, M. J. C., Milner, R., and Wadsworth, C. P. (1979) *Edinburgh LCF: A Mechanised Logic of Computation*, Volume 78 of *Lecture Notes in Computer Science.* Springer-Verlag.

Halmos, P. R. (1963) *Lectures on Boolean algebras*, Volume 1 of *Van Nostrand mathematical studies.* Van Nostrand.

Harrison, J. (1994) Constructing the real numbers in HOL. *Formal Methods in System Design*, **5**, 35–59.

Harrison, J. (1995) Metatheory and reflection in theorem proving: A survey and critique. Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, UK. Available on the Web as `http://www.cl.cam.ac.uk/users/jrh/papers/reflect.dvi.gz`.

Heawood, P. J. (1890) Map-colour theorem. *Quarterly Journal of Pure and Applied Mathematics*, **24**, 332–338. Reprinted in Biggs, Lloyd, and Wilson (1976).

van Heijenoort, J. (ed.) (1967) *From Frege to Gödel: A Source Book in Mathematical Logic 1879–1931.* Harvard University Press.

Henkin, L. (1963) A theory of propositional types. *Fundamenta Mathematicae*, **52**, 323–344.

Holmes, R. M. (1995) Naive set theory with a universal set. Unpublished; available on the Web as `http://math.idbsu.edu/faculty/holmes/naive1.ps`.

Kempe, A. B. (1879) On the geographical problem of the four colours. *American Journal of Mathematics*, **2**, 193–200. Reprinted in Biggs, Lloyd, and Wilson (1976).

Knuth, D. and Bendix, P. (1970) Simple word problems in universal algebras. In Leech, J. (ed.), *Computational Problems in Abstract Algebra.* Pergamon Press.

Kreisel, G. (1990) Logical aspects of computation: Contributions and distractions. In *Logic and Computer Science*, Volume 31 of *APIC Studies in Data Processing*, pp. 205–278. Academic Press.

Lakatos, I. (1976) *Proofs and Refutations: the Logic of Mathematical Discovery.* Cambridge University Press. Edited by John Worrall and Elie Zahar. Derived from Lakatos's Cambridge PhD thesis; an earlier version was published in the *British Journal for the Philosophy of Science* vol. 14.

Landau, E. (1930) *Grundlagen der Analysis.* Leipzig. English translation by F. Steinhardt: 'Foundations of analysis: the arithmetic of whole, rational, irrational, and complex numbers. A supplement to textbooks on the differential and integral calculus', published by Chelsea; 3rd edition 1966.

Mathias, A. R. D. (1991) The ignorance of Bourbaki. *Physis; rivista internazionale di storia della scienza (nuova serie)*, **28**, 887–904.

Matsumura, H. (1986) *Commutative Ring Theory*, Volume 8 of *Cambridge Studies in Advanced Mathematics.* Cambridge University Press. Translated from Japanese 'Kakan kan ron' by Miles Reid.

Nederpelt, R. P., Geuvers, J. H., and de Vrijer, R. C. (eds.) (1994) *Selected Papers on Automath*, Volume 133 of *Studies in Logic and the Foundations of Mathematics.* North-Holland.

Nelson, G. and Oppen, D. (1980) Fast decision procedures based on congruence closure. *Journal of the ACM*, **27**, 356–364.

Paulson, L. C. (1994) A concrete final coalgebra theorem for ZF set theory. Technical Report 334, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK.

Prawitz, D. (1965) *Natural deduction; a proof-theoretical study*, Volume 3 of *Stockholm Studies in Philosophy.* Almqvist and Wiksells.

Ramsey, F. P. (1926) The foundations of mathematics. *Proceedings of the London Mathematical Society (2)*, **25**, 338–384.

Rasiowa, H. and Sikorski, R. (1970) *The Mathematics of Metamathematics* (3rd ed.), Volume 41 of *Monografie Matematyczne, Instytut Matematyczny Polskiej Akademii Nauk.* Polish Scientific Publishers.

Rudnicki, P. (1987) Obvious inferences. *Journal of Automated Reasoning*, **3**, 383–393.

Shostak, R. (1979) A practical decision procedure for arithmetic with function symbols. *Journal of the ACM*, **26**, 351–360.

Szabo, M. E. (ed.) (1969) *The collected papers of Gerhard Gentzen*, Studies in Logic and the Foundations of Mathematics. North-Holland.

Takeuti, G. (1978) *Two applications of logic to mathematics.* Number 13 in Publications of the Mathematical Society of Japan. Iwanami Shoten, Tokyo. Number 3 in Kanô memorial lectures.

Tarski, A. (1954) *A Decision Method for Elementary Algebra and Geometry.* University of California Press.

Trybulec, A. (1978) The Mizar-QC/6000 logic information language. *ALCC Bulletin (Association for Literary and Linguistic Computing)*, **6**, 136–140.

Whitehead, A. N. and Russell, B. (1910) *Principia mathematica (3 vols).* Cambridge University Press.

Wiener, N. (1914) A simplification of the logic of relations. *Proceedings of the Cambridge Philosophical Society*, **17**, 387–390.

Zermelo, E. (1908) Neuer Beweis für die Möglichkeit einer wohlordnung. *Mathematische Annalen*, **65**, 107–128. English translation, 'A new proof of the possibility of a wellordering' in van Heijenoort (1967), pp. 183–198.