Formalized Mathematics (DRAFT)

John Harrison
Åbo Akademi University, Department of Computer Science
Lemminkäisenkatu 14a, 20520 Turku, Finland
jharriso@abo.fi
http://www.abo.fi/~jharriso/

22nd February 1996

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 (Anonymous 1994) however, we do not offer polemics in support of such a project. We merely try to place the formalization of mathematics in its historical perspective, as well as looking at existing praxis and identifying what we regard as the most interesting issues, theoretical and practical.

1 History and Philosophy

Mathematics is generally regarded as the exact subject *par excellence*. But the language commonly used by mathematicians (in papers, monographs, and even textbooks) can be remarkably vague; perhaps not when compared with everyday speech but certainly when compared with the language used by practitioners of other intellectual disciplines such as the natural sciences or philosophy. Trybulec and Święczkowska (1992) point out that in a way this is natural: since the underlying semantics of mathematics is generally clearer than that of other disciplines, the mind is naturally trammelled into a precise mode of thinking, and terminological exactness is less important. But a knowledgeable reader is sometimes needed to separate rhetorical flourish from real content, and to appreciate the context in which results are asserted. Some everyday phrases are imbued with a particular significance; observe for example the crucial distinction between a 'mapping into X' and a 'mapping onto X', or the precise (albeit context-dependent) meaning of wooly-sounding expressions like 'for almost all x'. And on the other hand many issues which the author feels are obvious or unimportant may be glossed over. Bourbaki (1968) stresses the importance of '*abuses of language*, without which any mathematical text runs the risk of pedantry, not to say unreadability', but many of the conventions go beyond mere abuse of language. Consider the following examples taken 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.

Trybulec and Święczkowska (1992) remark that the language of mathematical texts isn't incontrovertibly a natural language at all; but it invariably contains a substantial admixture of natural language, and that has all the usual potential for ambiguity and imprecision. This might not be a problem 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 eventually pointed out in print by Heawood (1890), and it is only with the work of Appel and Haken (1976) that the theorem has finally come to be accepted.[1] The errors need not be deep mathematical ones, as shown by the following from Littlewood (1986):

> Professor Offord and I recently committed ourselves to an odd mistake (Annals of Mathematics (2) 49, 923, 1.5). In formulating a proof a plus sign got omitted, becoming in effect a multiplication sign. The resulting false formula got accepted as a basis for the ensuing fallacious argument. (In defence, the final result was known to be true.)

A book by Lecat (1935) gave 130 pages of errors made by major mathematicians up to 1900. With the abundance of theorems being published today, often emanating from writers who are not trained mathematicians, one fears that a project like Lecat's would be practically impossible, or at least would demand a journal to itself!

We have separated the concerns of imprecision and incorrectness, but they are linked. It's clear that if one mathematician uses another's result without fully appreciating the definitions underlying it or the conditions attached to it, this can lead to errors. Moreover, as famously shown by Lakatos (1976), incorrectness and imprecision can shade imperceptibly close to one another. Lakatos discusses how Euler's theorem about polyhedra has changed over time. It was shown to be incorrect, but rather than being overthrown, it merely evolved, incorporating additional explicit conditions which formerly were either left unsaid or not appreciated at all. In the final incarnation its validity stands almost as a definition of a certain kind of polyhedron! This is not such an uncommon fate for important theorems. For example, Stokes's theorem has been generalized so much, and been associated with the development of so much technical machinery, that the theorem itself is almost a triviality. Even if one does not subscribe in full to Lakatos's thesis that mathematics has a strong empirical component placing it almost on a par with physical sciences, such examples are worth reflecting on.

The formalization of mathematics addresses both these questions, precision and correctness. By *formalization* we mean expressing mathematics, both statements

[1] There are still qualms over the fact that the proof relied on computer checking of cases. We distinguish sharply between ad hoc checking programs like this and the kind of use we propose of computers to prove theorems *in a formal deductive calculus*.

and proofs, in a (usually small and simple) formal language with strict rules of grammar and unambiguous semantics. The latter point bears repeating: just writing something in a symbolic language is not sufficient for formality if the *semantics* of that language is ill-defined.[2] We can split the project of formalization 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. In practice, though, we are also interested in checking the correctness of proofs, and formalization of the statements and then the proofs is a prerequisite for doing this systematically. Without formalization, all the difficulties of interpretation and understanding of precedent, which are so prominent in jurisprudence, begin to invade mathematics. But given formalization, checking does turn out to be possible (with certain provisos), at least in principle.

1.1 Rigour and the axiomatic method

There are all sorts of arguments used in technical disciplines to establish the truth of assertions. Sometimes people become convinced of the truth of arithmetical conjectures on the basis of a preponderance of numerical evidence. But these methods can lead to errors, and since at least the time of the Greeks it has been generally accepted that a true mathematical proof must be *deductive*. That is, it must proceed from accepted, or at least explicitly stated, assumptions via a series of small incontrovertibly correct steps. Sometimes these steps themselves may be based on spatial or even mechanical intuition — Upsenskii (1961) gives some examples of the latter. But such intuition too has fallen into disrepute in recent times, following the discovery of such pathologies as Peano's space-filling curve and Bolzano's and Weierstrass's everywhere continuous nowhere differentiable functions.

Now it doesn't seem fair to dismiss all geometric reasoning as non-rigorous. Since the time of Descartes it has been known how to reduce geometry to arithmetic (understood in a wide sense); sometimes this makes things easier, but at times it can obscure the immediacy and simplicity of geometric arguments. Arnol'd (1990) detects a vivid contrast already between the inventors of calculus: Newton's reasoning was strongly geometric whereas Leibniz stressed algebraic manipulation. More recently, Minkowski has shown how quite elementary geometrical reasoning can lead to deep and important results in number theory. Nevertheless, geometric reasoning nowadays tends to be seen as a convenient and illuminating shorthand for a purely symbolic proof, as Littlewood (1986) says:

> A heavy warning used to be given that pictures are not rigorous; this has never had its bluff called and has permanently frightened its victims into playing for safety. Some pictures, of course, are not rigorous, but I should say most are (and I use them whenever possible myself). [...] pictorial *arguments*, while not so purely conventional, can be quite legitimate [...] in the sense that in translating into symbols no step occurs that is not both unequivocal and trivial.

[2] For example, the Z specification language (Spivey 1988), which has been around for many years, is really only now becoming formal in our sense, as a standardization effort clears up ambiguities in its semantics.

(By contrast, some early proofs of the Fundamental Theorem of Algebra, for example, leave some doubt over whether the simple geometric observations they use do in fact fit easily into a symbolic framework.) But geometry is just one particularly important example of how everyday intuitions can lead us astray. There is always a danger of being misled by the superficial similarity of an abstract structure to some particular concrete one; for example one might accidentally assume that multiplication in a ring must be commutative. Sometimes the results can be serendipitous, e.g. the use of complex numbers and Euler's successful manipulations of infinite series. But they can also lead to serious errors, e.g. the purported proof of Fermat's Last Theorem based on the assumption of unique factorization in arbitrary algebraic number fields.

The pathologies of real analysis and the discovery of non-Euclidean geometries, among other innovations, stimulated a general determination among mathematicians to bring out unstated assumptions and either justify them or avoid them. Even the Euclidean canon was not immune from criticism: indeed long before Gauss had pointed out that the notion of 'betweenness' is often used in Euclid, but never defined. This in turn led to the rise of the modern refinement of the axiomatic method, where a set of axioms is established and all conclusions must proceed from the axioms by steps which are *purely logical* and make no additional mathematical assumptions. Apart from the question of reliability, this turns out to be very useful, since the same set of axioms often admits various different realizations (a point made long before by Boole). Consequently the same reasoning can be applied in different situations, giving benefits of elegance and economy. One of the early classics of the axiomatic method was a treatise on geometry by Hilbert (1899), which made all assumptions explicit and analyzed the interdependence of the axioms in detail. Reputedly he once stated that it should be possible to replace the words 'point', 'line' and 'plane' by 'table', 'chair' and 'beer-mug' without invalidating any of the theorems of geometry.

Whatever the level of formalization of the deductive proof, it may be rather unenlightening to read. In cases where the deductive proof is arrived at via geometric intuition, then that intuition is valuable for understanding the proof. In general, the whole process by which the theorem and its proof were discovered may be of paramount importance to grasp the theorem, the proof and their significance (e.g. applicability to physical science or relation to other part of mathematics). It's well-known that one doesn't 'understand' a proof merely by having checked the correctness of each inference it contains. Some mathematicians prefer the intellectual purity of the deductive part, probably following the examples of such worthies as Newton and Gauss (the latter famously referred to other considerations as the scaffolding used to construct a beautiful building, which should be taken down when the building is finished). But generally, especially in introductory textbooks, there is a lot of discussion other than the purely deductive part. Trybulec distinguishes the 'formal stratum' and the 'informal stratum', which are present in different proportions depending on the style of the author and the intended audience.

Also present in the Zeitgeist at about the turn of the century was a foundationalist tendency. Not satisfied merely with uncovering the hidden assumptions in mathematics, many tried to justify them on the basis of less dubitable principles. Various mathematicians showed how the reals could be understood as certain infinite sets of natural numbers, and Dedekind (1888) showed how all the properties of natural numbers could be deduced from a very small set of axioms or even constructed in pure set theory, given an infinite set; Frege proposed another definition which he claimed was based on pure logic.

But Cantor's set theory marked a bold introduction of the actual infinite into mathematics. This had long been viewed with scepticism, Gauss's interdiction being well-known. Cantor distinguished carefully between 'consistent' and 'inconsistent'

multiplicities, but many who used superficially similar methods of reasoning started to encounter worrying paradoxes. Some of these paradoxes, notably Russell's ('the set of all sets which are not members of themselves'), involved only very simple concepts like membership and predication. Foundational studies were plunged into a new crisis, even the most trivial intuitive ideas seeming to cause major problems.

In this atmosphere, it's not surprising that the complete formalization of mathematical proof should be attempted; in any case, it can be seen as the natural next step in the evolution of ever greater precision and rigour in mathematics. If basic concepts like 'set' and 'member' are problematical, how can natural language and intuitive proof ever be trusted? Moreover Zermelo, who independently discovered Russell's paradox at about the same time, had now isolated a few set construction principles which seemed enough for all the applications of set theory in mathematics, and apparently did not lead to contradiction. But Zermelo's *Aussonderungsaxiome*[3] relied on the notion of a '*definit*' property, and attempts by others to make this precise inevitably focused on restricting the *language* in which that property was stated.

1.2 The History of Formal Logic

The idea of reducing reasoning to computation in some kind of formal calculus is an old dream, going back at least to Raymond Lull. Hobbes in his "Leviathan" coined the slogan 'Reasoning is but Reckoning'. A particularly well-known statement of the idea was Leibniz's. He envisaged a 'characteristica universalis' (universal language) and a 'calculus ratiocinator' (calculus of reasoning). His idea was that disputes of all kinds, not merely mathematical ones, could be settled if the parties translated their dispute into the characteristica and then simply calculated. Leibniz even made some steps towards realizing this lofty goal, but his work was largely forgotten. Meanwhile, mathematics continued to acquire new symbolisms, which as Whitehead (1919) says:

> [...] have invariably been introduced to make things easy. [...] by the aid of symbolism, we can make transitions in reasoning almost mechanically by the eye, which otherwise would call into play the higher faculties of the brain. [...] Civilisation advances by extending the number of important operations which can be performed without thinking about them.

It was to be expected that the use of symbolism would eventually extend beyond the subject matter of mathematics, to the reasoning used *in* mathematics. Boole (1848) developed the first really successful formal system for logical and set-theoretic reasoning. What's more, he was one of the first to emphasize the possibility of applying formal calculi to several different situations, and doing calculations according to formal rules without regard to the underlying interpretation. In this way he anticipated important parts of the modern axiomatic method. However Boole's logic was limited to propositional reasoning, and it was not until the much later development of quantifiers that formal logic was ready to be applied to general mathematical reasoning. The first systems adequate for this purpose were developed by Frege and Peano. Both were mathematicians, and to some extent they had a common motive of making mathematical reasoning more precise. But there were many contrasts between them.

Frege (1879) devised his 'Begriffsschrift' (concept-script) for formal logic and mathematics, with the particular aim of carrying through what later became known as the logicist programme. He wanted to prove that not only the reasoning used

[3] 'Separation', or perhaps better 'picking-out' axiom.

in mathematics, but the *underlying assumptions too*, and therefore the whole of mathematics, were just pure logic. This would show that mathematics is analytic, refuting Kant's dictum that it is synthetic a priori. In his work, Frege established such a standard of precision, and made so many important distinctions (e.g. between x and $\{x\}$ and between $\in$ and $\subseteq$), that he is commonly regarded as the founding father of modern logic. However his Begriffsschrift was cumbersome, quite unlike conventional mathematical notation, and a nightmare for printers.

Peano's teaching experience led him to be interested in stating mathematical results and arguments precisely. He developed a formal notation for expressing mathematical reasoning, which was rather closer to conventional symbolism than was Frege's concept-script. He was not so interested in a foundational reduction, but concentrated on rewriting mathematics in a formal framework. Together with his colleagues and assistants he published a substantial amount of formalized mathematics: his journal *Rivista di Matematica* was published from 1891 until 1906, and polished versions were collected in various editions of the *Formulaire de Mathématique*.

On meeting Peano at a mathematical congress, Russell was sufficiently impressed to adopt Peano's symbolism in his own foundational investigation of mathematics. He started a similar programme to Frege, at first independently, though on becoming aware of Frege's work, he was to some extent directly influenced. Though some people opine that even Russell's subsequent work was a step backwards from Frege's in terms of precision, Russell also found a logical inconsistency at the core of Frege's system (the Russell paradox which we have already mentioned). The story of how he informed Frege of this just as the second volume of Frege's book was in the press is well-known. Peano did not escape either: the co-discoverer of Russell's paradox, 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.

Russell developed his own logic, which was distinguished from others by its introduction of the notion of *type*. (We shall have more to say about this idea later.) This logic combined, to some extent, the notational convenience of Peano's system with the precision and foundational aspirations of Frege's work; moreover it wasn't obviously inconsistent. In their monumental work '*Principia Mathematica*', Whitehead and Russell (1910) began a formal development of mathematics from this basis. Without the flexibility which typelessness allowed, it was no longer possible to derive the existence of an infinite set from logic alone, and a separate Axiom of Infinity needed to be posited (as well as an Axiom of Reducibility, but that was an artifact of the unnecessary complication of ramified types). For this reason, *Principia* failed to establish the logicist thesis that mathematics is completely reducible to logic.

What *Principia* unmistakably succeeded in doing was showing that important parts of real mathematical reasoning could be written out in a completely formal logic, albeit starting from a few non-logical axioms. In a way this was a triumph which has never been surpassed. But the resulting text demands extremely close attention from the reader, and there can be few who made the effort. Indeed, Russell's own intellect, he reports, never quite recovered from the strain of writing it. Given this, the idea of a practical project to formalize mathematics completely, or even to use formal logic at all, seemed a dubious one. The development of *Principia* was never carried further, perhaps because of Russell's exhaustion, or simply because his primary interest was the logicist thesis rather than formal mathematics per se. Increasingly, formal logic came to be seen as a theoretical device, rather than as a practical tool. The aims of the pioneers of actually *using* logic were widely dismissed. 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.

1.3 Hilbert's Programme

But there was still considerable interest in logical systems *as themselves objects of study*. The crisis over the introduction of infinitistic methods into mathematics, which we have already mentioned, was still rumbling on, and Brouwer even rejected the general use of the law of the excluded middle ($P \vee \neg P$). Now on previous occasions, when mathematicians wished to provide a foundation for some new technique, they did it by an interpretation in terms of known mathematics. The classic example is the interpretation of complex numbers as pairs of real numbers. But for justifying infinite sets, there seemed to be no such way out: how can one find an interpretation for infinite sets in terms of finite ones? Hilbert's ingenious idea was to get round this by studying not the *objects*, but the mathematical *proofs*, which are always finite, even when they are about infinite sets.

Specifically, Hilbert proposed the following programme.[4] Formal systems for various mathematical theories should be developed; thanks to the line of work culminating in Russell's logic, combined with Hilbert's own flair for axiomatization, this was straightforward. Then these should be proven *consistent*, i.e. that it is not possible to deduce both a theorem and its negation; in symbols $\vdash P$ and $\vdash \neg P$. Now given this, any 'concrete' theorem of the form $\forall n \in \mathbb{N}.\, P[n]$ which is deducible in the system must be true. Otherwise there'd be some $k \in \mathbb{N}$ such that $\neg P[k]$, so any reasonable logical system will certainly be capable of proving $\vdash \neg P[k]$ (we assume that $P[k]$ is simply a matter of calculation for each specific k, in particular not itself involving unbounded quantifiers). But since $\vdash \forall n \in \mathbb{N}.\, P[n]$, we also have $\vdash P[k]$, which is impossible because the system was assumed consistent. This reasoning extends to arbitrary numbers of universal quantifiers; a good example of such an assertion is Fermat's Last Theorem: $\forall x, y, z, n \in \mathbb{N}.\, n > 2 \wedge x^n + y^n = z^n \Rightarrow x = 0 \vee y = 0$.

For Hilbert, therefore, the business of proving consistency of formal theories was central. An interesting subsidiary question was the *Entscheidungsproblem* (decision problem): is it possible to decide questions in these formal mathematical theories purely mechanically?[5] Hilbert and his assistants, as well as numerous other mathematicians and logicians, attacked these problems. Before long, there were negative conclusions.

First Gödel (1931) proved his famous First Incompleteness Theorem, which says roughly that no given (consistent) formal system can ever encompass even all the truths of elementary number theory: there will always be true statements which cannot be proved in the system.[6] Gödel's method was roughly as follows. It's easy to see that each sentence ϕ of the formal language can be allocated a unique 'Gödel number' $\ulcorner \phi \urcorner$, and likewise each proof – there are after all only countably many of each, and they have a quite simple structure. Moreover, for any sensible encoding, the question of whether p is a proof of ϕ can be encoded as a representable predicate[7] $Prov(p, \ulcorner \phi \urcorner)$, with the basic property that if $\vdash \phi$ then also $\vdash \exists p.\, Prov(p, \ulcorner \phi \urcorner)$. Now

[4] For a more detailed discussion of Hilbert's programme, we recommend the penetrating analysis by Kreisel (1958).

[5] This would amount to a realization of Leibniz's dream, at least if it is restricted to mathematics.

[6] True to the practices of informal mathematics that we spend so much energy berating, we shall not state precisely the conditions on a system for Gödel's theorem to apply, and shall sometimes take it for granted later that we are discussing such a formal system, and that it is consistent. This is a reasonable practical assumption.

[7] Roughly speaking, a predicate that can be calculated (by proof if need be) for each specific allocation of numeric arguments — this corresponds to the fact that a 'formal system' must allow an unambiguous algorithmic way of checking whether a purported proof is valid.

using a diagonalization argument, Gödel was able to exhibit a statement ϕ which (intuitively) asserts its own unprovability in the system:

$$\vdash \phi \equiv \neg\exists p.\, Prov(p, \ulcorner\phi\urcorner)$$

Now if the formal system is consistent, $\not\vdash \phi$, for if $\vdash \phi$ we have $\vdash \exists p.\, Prov(p, \ulcorner\phi\urcorner)$ and hence $\vdash \neg\phi$. However ϕ is certainly *true* because it states precisely that unprovability! On the additional assumption that the system is 1-*consistent*, i.e. all provable existential statements are true, then also $\not\vdash \neg\phi$. But note that since ϕ isn't provable, adding $\neg\phi$ as a new axiom retains consistency, so a formal system can be consistent yet *not* 1-consistent; thus the argument above that consistency implies the truth of provable *universal* formula cannot be sharpened. Now neither of these conclusions is fatal to Hilbert's programme:[8] he never identified mathematics with reasoning in a particular formal system, and apparently influenced by Brouwer's critique, was prepared to dismiss existential theorems without a construction as strictly meaningless anyway.[9] But soon after came Gödel's Second Incompleteness Theorem, which states roughly that any realistic mathematical theory, even elementary number theory, is unable to prove its own consistency. So any consistency proof of a formal system requires *greater* mathematical resources than available in the system, and so it certainly isn't possible to justify infinite sets finitistically. Despite the apparent promise of Hilbert's ingenious idea of using proofs, it turns out to be just as circular as using interpretations — it's as if there's some mysterious conservation principle at work like that forbidding perpetual motion! Moreover Church (1936) and Turing (1936) showed the Entscheidungsproblem to be unsolvable even for pure first order logic, Turing by introducing his famous machines and so giving the first really convincing definition of a 'formal system'. Finally Tarski (1936) showed that arithmetical truth is not even arithmetically definable, let alone recursively enumerable.

Despite the apparent failure of Hilbert's programme, the general interest in metalogical studies has yielded many benefits and thrown new light on parts of mathematics. (Though destructive to Hilbert's programme, some of the negative results mentioned above have a profound mathematical and philosophical significance.) First order logic seems particularly interesting from a metalogical point of view: it's strong enough to express some nontrivial mathematics, yet weak enough to allow interesting metatheorems (e.g. on the existence on nonstandard models of the first order theory of reals). There turn out to be some interesting first order mathematical theories which *are* decidable, and this often generalizes previous results. 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 such as Sturm's theorem.

1.4 The Bourbaki view

The book series in the name of Nicolas Bourbaki (a pen-name for a group of French mathematicians) 'takes up mathematics from the beginning, and gives complete proofs'. This beginning (Bourbaki 1968) is something like Zermelo set theory[10]

[8] Though it does refute the stronger form that all these theorems should be provable in a fixed finitistic system. Hilbert's precise agenda was never clear; at first he presented consistency as the objective *per se*.

[9] It seems bizarre to accept P as meaningful while dismissing $\neg P$ as meaningless. However there is some Popperian justification for this, since a universal statement is *refutable* by calculation. Anyway, Hilbert's views varied over time and were not always clear, so a detailed exegesis is difficult. Certainly it's necessary to accept universal formulas as meaningful since an assertion of consistency is itself of this form.

[10] Later editions included the Axiom of Replacement, giving something closer to ZFC. The author is not sure whether this axiom has ever been *used* in Bourbaki.

axiomatized in a system of first order logic, which includes an indefinite descriptor τ building in the Axiom of (Global) Choice.

> For whereas in the past it was thought that every branch of mathematics depended on its own particular intuitions which provided its concepts and prime truths, nowadays it is known to be possible, logically speaking, to derive practically the whole of known mathematics from a single source, the Theory of Sets.

Elsewhere 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.

The Bourbaki development is intellectually elegant, if rather austere. Stress is placed on developing properties of various key mathematical structures like topological spaces and groups. Some structures are instances of several others (e.g. topological groups are both topological spaces and groups), or are derived from others using certain canonical techniques such as completion. The relative sophistication of these techniques means that the set theory is used heavily even in the study of abstract structures; moreover, set theory is often necessary to provide existence proofs for many of them. So although Hilbert's axiomatic method was an important inspiration, deduction from the structure axioms uses much more than pure (first order) logic. Really, one is working all the time in axiomatic set theory. In the IMPS terminology (Farmer, Guttman, and Thayer 1990) the Bourbaki system is a 'big theory' rather than a mosaic of 'little theories'. A typical example of how concrete structures arise from consideration of abstract ones is the development of real numbers given by Bourbaki (1966). The reals are constructed as the completion of the uniform space (this structure is a Bourbaki innovation) induced by the topological group of rationals.

The stated purpose of the Bourbaki treatise is 'to provide a solid foundation for the whole body of modern mathematics'. They work, notionally anyway, with a fixed formal system axiomatizing set theory. Mathias (1991) expresses surprise and dismay that Gödel's theorems on the essential incompleteness of such systems seem not to perturb the Bourbachistes. Kreisel (1970) remarks:

> The failure of Hilbert's programme shows that formalism does not provide a theory of the actual process of mathematical reasoning; *it is not*, so to speak, *a fundamental theory*. (Of course, formalization is useful, either in limited areas or in combination with other considerations.)

But even Gödel's undecidable sentence for a simple number theory is really just a theoretical pathology. Paris and Harrington (1991) have come a bit closer to a realistic mathematical statement that cannot be proved in a simple first order arithmetic; even this relies on a rather artificial encoding of a combinatorial result in number theory. It seems we are nowhere near finding a mathematically interesting incompleteness in ZFC set theory, by which we mean a statement unprovable in ZFC but known for other reasons to be true, as distinct from statements like the Continuum Hypothesis which have no more been decided by non-formalistic methods. Certainly, it's hard to imagine a mainstream mathematical result which could not be proved in ZFC set theory; the Bourbaki project has developed extensive

parts of mathematics and failed so far to find any. So from a *practical* point of view, a disdainful attitude to the significance of Gödel's results seems justified. 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.

(This does however suggest the interesting research problem of systematically using non-formal reasoning to obtain new results; perhaps Gödel's theorems indicate that in our reliance on the axiomatic method we're failing to exercise the full potential of our mathematical faculties!) Although Bourbaki introduces a formal logical system in considerable detail, it is soon forgotten about for reasons of readability and practicality:

> If formalized mathematics were as simple as the game of chess, then once our chosen formalized language had been described there would remain only the task of writing out our proofs in this language, [...] But the matter is far from being as simple as that, and no great experience is necessary to perceive that such a project is absolutely unrealizable: the tiniest proof at the beginning of the Theory of Sets would already require several hundreds of signs for its complete formalization. [...] formalized mathematics cannot in practice be written down in full, [...] We shall therefore very quickly abandon formalized mathematics, [...]

Nevertheless, translatability *in principle* into the formal set theory is taken as the final arbiter of correctness. Bourbaki (1968) clearly says that 'the correctness of a mathematical text is verified by comparing it, more or less explicitly, with the rules of a formalized language'. The intention is that the whole Bourbaki series uses only layers of convention and abbreviation, and could in principle be written out completely formally. This view of formalizability as the criterion of correctness now seems to be widespread in the mathematical community. For example, Mac Lane (1986) says almost the same thing (p. 377):

> As to precision, we have now stated an absolute standard of rigor: A Mathematical proof is rigorous when it is (or could be) written out in the first-order predicate language $L(\in)$ as a sequence of inferences from the axioms ZFC, each inference made according to one of the stated rules. [...] When a proof is in doubt, its repair is usually just a partial approximation to the fully formal version.

That such a view is widespread really does seem to ignore Gödel's theorem. Simple reasoning shows that the Gödel sentence for ZFC set theory is not provable in ZFC, but, if we accept the consistency of ZFC (and surely if it is our criterion for rigour we must), it is true. Are we to dismiss this argument for the truth of the sentence as non-rigorous? In fact Lakatos (1980) identifies this, and other examples like the duality principle of projective geometry, as *post-formal* proofs, reached by analysis of a formal system rather than derived inside it, though of course the reasoning could in its turn be written out in another formal system. But once again, we accept that Gödel's theorem isn't likely to be relevant to any mathematical proofs we are interested in, and from a strictly practical point of view, this objective standard of rigour is a good one. Rather than discuss that, we want to focus on the following question: why do we only accept formalizability in principle, not formalization in practice?

1.5 Enter the computer

Thus far, formalized mathematics seems an unpromising idea. Bourbaki may be right that such formalization is simply impractical for people to do. In any case, as Naur (1994) points out, formalization sometimes leads to *more* errors, not fewer, as the underlying intuition which provides a check on correctness begins to get lost — so even one of the claimed merits of formalization is open to question. The problems of formalization have been well put by Constable, Knoblock, and Bates (1985):

> It is thus possible to translate the proof of any mathematical theorem into a completely formal proof. However, the prospect of actually doing this is quite daunting because an informal proof of modest length will expand to a formal one of prodigious size and will require in its production extreme care and detailed knowledge of the more or less arbitrary conventions of the particular formalism. These tedious details will in sheer number dominate the interesting mathematical ideas which are the *raison d'être* of the proof.

But like the authors just quoted, we believe that the arrival of the computer changes the situation completely, for several reasons.

1. As pointed out by Naur and others like van Gasteren (1990), as one relies less on intuition and more on rules of formal manipulation, accuracy in those manipulations becomes more important. This is tedious for people,[11] but checking conformance to formal rules is one of the things computers are very good at. Indeed, the Bourbaki claim that the transition to a completely formal text is routine, but too boring and tedious for people, seems almost an open invitation to give the task to computers.

2. The computer need not simply check the correctness of formal arguments, but can try to help in their construction. We will discuss in more detail just how this is to be done later, but at one extreme, it is sometimes possible for computers to find proofs of theorems completely automatically! Furthermore, the computer can provide interface assistance, helping to mitigate the austerity of formal deductive calculi and to organize the user's work.

3. Among modern computer scientists, formal logic is quite widely known and substantially used; many regard formal logic as the 'applied mathematics' underlying computer science, just as real analysis etc. underlies physics and engineering. Dijkstra (1985) has remarked that 'computing scientists may well have been the first to *use* the predicate calculus regularly [...] as far as the mathematical community is concerned George Boole has lived in vain'.

4. Computer correctness is itself a topic of concern, especially since computers are used in safety-critical systems like fly-by-wire aircraft, antilock braking systems, nuclear reactor controllers and radiation therapy machines. One means of verifying a design is to prove that a mathematical model of the system satisfies some formal properties, designed to correspond to real-world requirements. Such proofs are much more technically intricate, albeit perhaps shallower, than typical mathematics proofs, and of necessity involve very complicated formal manipulations.

[11] However mathematicians are expected to acquire facility at certain kinds of formal manipulations as part of their training, so perhaps the apparent greater difficulty presented by logical formalism is just a question of its familiarity.

It isn't very surprising that the computer should make such a dramatic difference; after all it has revolutionized work in many scientific disciplines. Previously unthinkable ideas like detailed simulations of weather systems have become possible – formalizing mathematics in a deductive calculus demands fairly modest computer power by comparison.

MacKenzie (1995) traces the history of computerized theorem proving. The early experiments were mainly concerned with completely automatic proofs of theorems, and work can broadly be divided into two tracks. Work in the Artificial Intelligence tradition, typified by Newell and Simon (1956), tried to emulate the way mathematicians actually think. Other more successful work, exemplified by Wang (1960), simply relied on the speed of the computer, using exhaustive search for a proof in certain formal systems for (usually) first order logic. Of course even here the search wasn't completely blind: at the very least the formal systems were chosen to limit the search space. For example, the pioneering work of Prawitz, Prawitz, and Voghera (1960) was based on Gentzen's cut-free sequent systems, in the guise of 'tableaux'.[12]

The next step forward, discussed by Kanger (1963), was to use free or metavariables, discovering the right instantiations gradually during search. Subsequent implementations of tableaux and similar calculi such as model elimination (Loveland 1968) invariably used some such technique. These methods are 'global', in that the instantiations need to be applied throughout the partial proof tree. It's therefore not surprising that the modern incarnations rely heavily on Prolog technology; see the work of Stickel (1988) and Beckert and Posegga (1994) for example.

Another stream of work approached the problem bottom-up rather than top-down. These 'local' methods sacrifice goal-directedness but make instantiation cheaper and, since all variables in intermediate results are effectively universal, they allow the garnering of lemmas which can then be re-used with different instantiations. The 'inverse method' of Maslov (1964) was presented in just this way, as a bottom-up approach to proof in cut-free sequent calculus. The most famous method though was resolution, invented by Robinson (1965). This took just a single rule of inference, with unification (introduced explicitly for the first time by Robinson), and its introduction caused a renaissance of interest in automated theorem proving. Otter, directly descended from this line of research, has recently achieved 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; Stålmarck 1994), linear arithmetic (Shostak 1979) and equational reasoning (Knuth and Bendix 1970), 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, notably 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.

Nevertheless the achievements of fully automatic provers soon began to reach a plateau, well below the ability to prove many substantial mathematical theorems

[12]For many proof procedures, the proof-theoretic details of Gentzen's results are not really relevant; all that is required is that the provability of $\exists x.\, P[x]$ is equivalent to the existence of some terms t_i with $P[t_1] \vee \ldots \vee P[t_n]$ provable. This result (which together with an explicit construction is essentially Herbrand's theorem) is an easy consequence of Gentzen's Hauptsatz, but it can be derived purely model-theoretically; see Kreisel and Krivine (1971) for a particularly elegant proof of what they pointedly call the 'uniformity theorem'.

unaided. The emphasis moved to the development of 'proof assistants', 'proof checkers' and 'interactive theorem provers', which are closer to our present interests. The idea is not to prove difficult theorems automatically, but simply to assist a human in constructing a formal proof, or at the very least check its correctness. Several pioneering projects for the computer formalization of mathematics appeared in the 70s. 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. 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.

One of Automath's difficulties was that it was ahead of its time technologically, always struggling to make the computer keep pace with the ideas. Perhaps another factor contributing to its decline was that it used some rather original notational and foundational ideas (the banishment of named bound variables and the exploitation of the Curry-Howard correspondence,[13] 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 mathematical puzzles and computer models. Selected Mizar articles are automatically abstracted and printed in human-readable form in the journal 'Formalized Mathematics'.[14]

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 arbitrarily 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 or produce a theorem other than the one intended, of course, but the latter is easy to guard against and both are rare). It seems in practice that this can be done reasonably efficiently; we discuss this issue in more detail later. HOL (Gordon and Melham 1993), which uses classical higher order logic rather than the LCF logic, takes this a step further by insisting

[13] Independently discovered by de Bruijn.

[14] 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. Also available on the Web from '`http://math.uw.bialystok.pl/~Form.Math/`'.

all theories be built up using special definitional extension mechanisms. These give a guarantee that consistency is preserved. Such an approach is consonant with the LCF philosophy, since it entails pushing back the burden of consistency proofs or whatever to the beginning, once and for all, such that all extensions, whether of the theory hierarchy or proof mechanisms, are correct per construction.

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. As we said above, this 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 quite a bit of pure mathematics is needed in a supporting role — see 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.

2 Formalizing mathematics

In formalizing mathematics, we must use our judgement to define informal constructs in terms of formal ones, which is merely an extreme case of defining non-rigorous concepts rigorously. This isn't always easy, but there can be surprising successes — for example, before Turing it would have seemed implausible that a simple and rigorous formalization of 'computable' could be given. For a good historical precedent, consider how Bolzano and Weierstrass arrived at rigorous '$\epsilon - \delta$' definitions of notions like limits and continuity. We may have our own intuitive ideas of what a continuous function is: perhaps one whose graph we can draw without taking our pencil from the graph paper, or which passes through all intermediate values. But the definition of continuity which is now standard is neither of these; we say a function $f : \mathbb{R} \to \mathbb{R}$ is continuous on $\mathbb{R}$ iff:

$$\forall x \in \mathbb{R}.\ \forall \epsilon > 0.\ \exists \delta > 0.\ \forall x'.\ |x - x'| < \delta \Rightarrow |f(x) - f(x')| < \epsilon$$

In what sense is this or any other formal definition 'right'? Why not take the intermediate value property (called 'Darboux continuity') instead?

$$\forall x, x', z \in \mathbb{R}.\ x < x' \wedge f(x) < z < f(x') \Rightarrow \exists w.\ x < w < x' \wedge f(w) = z$$

Such worries can be allayed when the various plausible-looking formalizations are all provably equivalent. For example, the various definitions of computability in terms of Turing machines, Markov algorithms, general recursive functions, untyped λ-calculus, production systems and so forth all turned out so. But here this is not the case: continuity implies Darboux continuity but not conversely.[15] The usual definition of continuity has probably won out because it is intuitively the most satisfying, leads to the tidiest theory or (most likely) admits the most attractive generalization to other metric and topological structures. But note that it also led to the counterintuitive pathologies of real analysis which, as we have already remarked, caused such disquiet once. This example well illustrates how formalization can be a difficult and creative task, and that the formal version of a concept can take on an unexpected life of its own. However, since present-day mathematics is mostly

[15] For example, the derivative of $f(x) = x^2 sin(1/x^2)$ (with $f(0) = 0$) is Darboux continuous, as are all derivatives, but it is not continuous at $x = 0$, nor in fact even Lebesgue integrable over any interval including 0.

quite rigorous, the final step to complete formalization shouldn't usually be so problematical.

2.1 Formalization

We shouldn't be too dogmatic about what is the 'natural' formalization of informal (or preformal) notions. We do not claim that reductionism always reflects the most interesting aspects of mathematics; in fact as Mac Lane 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. And we should certainly avoid regarding the implementability of mathematical concepts as sets, say, 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. (At the same time the reduction is not without philosophical significance: in particular the consistency of practically all of mathematics can be reduced to the problem of whether ZF set theory is consistent, so our foundational concerns tend to come home to roost in set theory.)

With these disclaimers in mind, let's look now at a few examples of how the basic concepts of formal systems can be used to define other notions. We'll arrange these in order of the amount of light they throw on the informal concept (in the opinion of the present author).

1. Kuratowski showed that ordered pairs could be defined by $(x, y) = \{\{x\}, \{x, y\}\}$ (a more complicated alternative was earlier proposed 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 often 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. In fact, this observation seems first to have appeared with Peano's '*Formulaire*', so it provides a concrete example of a mathematical development being stimulated by a programme of formalization.

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' $SUM(1, 10, \lambda i.\ i)$ while the derivative notation $\frac{d}{dx} x^2$ 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. For example, in the example above of Leibniz notation for differentiation, x has

both free and bound occurrences.[16] The breakdown to a lambda-term has helped to make this explicit. Though the Leibniz notation is familiar, it can be awkward and is often abandoned in advanced or multivariate work. For example, try writing the Chain Rule for differentiation using just the Leibniz notation! Other notations like $f'(x)$ seem better for such purposes, but can still lead to confusion; the author has witnessed a heated debate on `sci.math` among good mathematicians over whether $f'(g(x))$ denotes the derivative of f evaluated at $g(x)$ (this view seems most popular) or the derivative of $f \circ g$ evaluated at x ('the prime notation (f') is a shorthand notation for derivative of a univariate function with respect to the free variable.')

2.2 Criticism and reconstruction

The process of formalization must also incorporate a critical aspect when the need arises. Certain concepts which do not admit a satisfactory formalization may have to be jettisoned, modified, or relegated to the status of heuristic aids. It is hoped that this procedure leads only to superficial notational rigorization of mathematics, rather than the mutilation of important ideas and principles.

In a computer-based system especially, it's vital to avoid all notational ambiguities. Generally, the more flexible grammatical rules are, the more complicated they are too, and the more scope they leave for misunderstanding. A balance needs to be struck. For example, most people would agree that different precedences for binary operators are a good idea, but sometimes the precedences chosen can confute people's expectation. (For example in C the bitwise AND operator '`&`' binds less tightly than the equality relation '`==`', a historical accident that must have caused innumerable programming errors.) But many people do not accept the arbitrary solution of the 'dangling else' problem,[17] and many languages strive to avoid it in different ways. Another controversial question is the overloading of operators, e.g. the use of + for addition of both integers and floating point values.

A good example of abuse of notation in mathematics books is the use of f^{-1} to denote both the inverse function of f, i.e. $f^{-1}(f(x)) = x$, and the preimage of a subset of f's range, i.e. $f^{-1}(S) = \{x \mid f(x) \in S\}$. Books usually glibly observe that the notation is not confusing because the former meaning is only used if f is bijective, and then the image of S under f^{-1} is the same as the preimage of S under f. That's true, but there's quite a sophisticated use of context to determine whether f is bijective, and at the very least we need some kind of 'typechecking' to resolve whether f^{-1} is being applied to an element of $ran f$ or a subset. Another notation that is usually clear intuitively but hard to describe formally is indexed set abstraction $\{f(x) \mid P[x]\}$, for example $\{(m, n) \mid m+n = p\}$. In general, it's not easy to determine whether variables are supposed to be bound by the construct or not (for example, the above example is probably intended to bind both m and n, but if m means something outside, that might no longer be so.) Mathematics contains other confusions between free and bound variables and between constants and variables. There is a popular disclaimer attached to the general quartic equation: '$ax^4 + bx^3 + cx^2 + dx + e$ where e is not necessarily the base of natural logarithms'. And we have already discussed the problems of the Leibniz derivative notation. Another popular abuse of language is to use X to refer to the poset $(X, \leq)$, leaving the ordering implicit. Bourbaki (1968) has already shown a good way out here: we can take the ordering relation as primitive, replacing the reflexivity property in the usual definition of partial order by $x \leq y \Rightarrow x \leq x \wedge y \leq y$, recovering X if desired as

[16] It's precisely analogous to $\int_0^x 2x\, dx$.

[17] The parsing of '`if E then if E' then C else C''` as '`if E then (if E' then C else C')`' rather than '`if E then (if E' then C) else C''`.

$\{x \mid x \leq x\}$.[18]

It is quite possible that the critical faculties of the formalizer may go well beyond cleaning up sloppy notation. The formalizer may merely want to reconstruct existing mathematics in a formal framework, but may prefer to develop a new (better?) mathematics. It is clear that Zermelo had the former aim in mind when he introduced his set theory,[19] 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.[20]

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.

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! And, 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 going to find the process particularly difficult. 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). But on the positive side, this can be seen as an interesting motivation for new research!

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.[21] But it is more complicated: one classical theorem often splits into several constructive versions (classically equivalent);

[18] This approach was independently arrived at by the present author and used in the standard library of wellorderings in the HOL system.

[19] Zermelo didn't want to do truly formal reasoning, merely to make set-theoretic assumptions explicit.

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

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

the classicist may regard this as meaningless clutter. 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.

2.3 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.[22] 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. 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 layer by layer 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 can still perhaps distinguish types in that they render ill-typed phrases *meaningless*, whereas ZF set theory, say, merely makes them *false*. Moreover, typing judgements have traditionally been syntactic, or more precisely, decidable. This is easily true for simple type theory, even with polymorphism (the Hindley-Milner algorithm generates a most general type for any well-typed expression). It also holds for systems motivated by the Curry-Howard correspondence, since after all the corresponding *proof* judgements (whether p is a proof of ϕ) are meant to be decidable. But though true in principle this often isn't true in practice (many

[22] Note that the systems of the pioneers like Frege and Peano were clearly higher order. For example, the basic Peano axioms for natural number arithmetic give, in first order logic, only a trivial decidable theory — it's necessary to include addition and multiplication as primitive to get anything really interesting.

proofs cannot realistically be normalized), and in some other systems of dependent types, such as PVS's, typechecking is not decidable even in principle.

Types also arose, perhaps at first largely independently of the foundational schemes originating with Russell, in programming languages. Even FORTRAN distinguished integers and floating point values. One reason was clearly efficiency: the machine can generate more efficient code by knowing more about a variable. For example the implementation of C's pointer arithmetic varies according to the size of the referenced objects. If p is a pointer to objects occupying 4 bytes, then what C programmers write as $p + 1$ becomes $p + 4$ inside the (byte-addressed) machine. C's ancestor BCPL was untyped, and was therefore unable to distinguish pointers from integers; to support the same style of pointer arithmetic it was necessary to apply a scaling at every indirection, a significant penalty. Apart from efficiency, as time went by, types also began to be appreciated more and more for their value in organizing programs and providing limited checks on the their correctness. It doesn't seem entirely unreasonable to ascribe some of the same advantages to types in mathematics.

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, ZFC seems to indicate that they are not necessary to ensure consistency.[23] 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 Chwistek (1922), 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; as shown by Henkin (1963), all the logical operations can be defined as higher order functions given some 2-element type of propositions.[24] 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. However note that some objections often made to type theory are rather specious. For example, it's true

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

[24] 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).

that $\mathbb{R}$ and $\mathbb{N}$ are completely different types in simple type theory, but there are subtyping schemes which can make $\mathbb{N} \subseteq \mathbb{R}$; in any case this inclusion also fails in the standard set-theoretic development of the reals.

There are plenty of 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.[25] 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 are no types to hinder us, but none to help us either. In practice it seems likely that some quite simple abbreviations for set constraints are sufficient to give most of the convenience of types, without 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[26] 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 such as uniformity of definitional mechanisms (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.[27] Mizar settled on Tarski-Grothendieck set theory, after a few experiments with Morse-Kelley. This latter theory, by the way, was the basis for an unusually formal but very elegant book by Morse (1965); the present author been told that Morse used this formalism in his teaching and research in analysis.

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.

[25] Although among computer scientists the opposite is probably true.

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

[27] 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.

2.4 Definitions and locutions

In practice, even in quite rich foundational systems, let alone in pure set theory, representations of mathematical objects become unwieldy. For example, Bourbaki (1968) estimates[28] 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 just 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: 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 $Q[c]$ can be read as $(\exists x.\ P[x]) \wedge \forall x.\ P[x] \Rightarrow Q[x]$, or something like that. Given a theorem $\vdash \forall x.\ \exists!y.\ P[x, y]$, 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, given a descriptor.

[28] Section III.§3.1, 'The cardinal of a set', footnote.

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.

2.5 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; this approach can be exploited by choosing a particularly convenient value. For example, if we decide $0^{-1} = 0$, we have lots of nice unconditional theorems, like $\forall x \in \mathbb{R}.\, (x^{-1})^{-1} = x$, $\forall x \in \mathbb{R}.\, -x^{-1} = (-x)^{-1}$, $\forall x, y \in \mathbb{R}.\, (xy)^{-1} = x^{-1}y^{-1}$ and $\forall x \in \mathbb{R}.\, x^{-1} \geq 0 \Leftrightarrow x \geq 0$. 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 is not syntactically well formed. This avoids the problems above, but it might mean that in certain situations in analysis, the types become 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), and 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 antecedant 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.[29] 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 $x = y \Leftrightarrow x - y = 0$ 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). Actually, Freyd and Scedrov (1990) uses a special asymmetric 'Venturi tube' equality meaning 'if the left is defined then so is the right and 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 seems surprising when equality is considered such a basic thing, but the conclusion is hard to avoid when one sees phrases like 'if y is the unique value with $(x, y) \in f$, we write $f(x) = y$'. This isn't specific to a foundational discussion of function application; many analysis texts do similar things for the limiting operation, for example. In fact it's rather common to see mathematics books make 'conditional' definitions — 'if x is ..., then we define $f(x) = E[x]$' rather than just 'we define $f(x) = E[x]$' — again, there is probably a wish to have certain contextual information carried around with the definition. 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.

3 Practical issues

We now move onto issues of system engineering. These largely cut across the divisions which spring up over set theory versus type theory, classical versus constructive logic, and so on. Generally, the same kinds of problems need to be solved in each case, though to be sure, they differ in detail and in severity.

3.1 Feasibility

We have stated that the computer can handle the low-level details of proof. This isn't just an airy idea; on the contrary we have already described the LCF methodol-

[29] Which is what the original LCF (Logic of Computable Functions) logic did.

ogy for allowing the user to create higher-level proof structures which decompose to simple primitives inside the computer. But what grounds do we have for believing that this is practical even for a computer? (Accepting that some kind of definitional principle will be used as detailed above.) It's well-known that computers are amazingly fast, and getting faster, but it's equally well-known that their speed cannot defeat problems which have an inherent combinatorial explosion. The fact that fully automatic theorem provers have *not* displaced human mathematicians, or that the World Chess Champion is not (quite yet) a computer is evidence of this. For example, if the length of a fully formal proof were an exponential function of the length of an informal proof, then the proposed approach would probably be doomed.

We've already seen that for any formal system of the kind we need to consider, there are true statements which aren't provable in the system (Gödel's first theorem). At the same time we have argued that such statements are pathologies and we're unlikely to hit them in practice. We will make an exactly parallel claim about the *feasibility*, rather than *possibility in principle*, of proofs.

The measure of feasibility will be the *size* of a proof, which is meant to denote something like the number of symbols in the proof based on a finite alphabet — this seems the most reasonable practical measure. Certainly, from a theoretical perspective, there must be sentences out there whose proofs are possible but not feasible. Indeed, given any (total) recursive function f (e.g. a large constant function or a huge tower of exponentials), there are provable sentences of length n which have no proofs of size $\leq f(n)$. For otherwise, provability would be decidable (just check all possible proofs up to length $f(n)$), contradicting the results of Church and Turing that the *Entscheidungsproblem* is unsolvable. More explicitly, we can modify Gödel's diagonal argument and exhibit a sentence which says not 'I am unprovable', but rather 'all proofs of me are unacceptably big':[30]

$$\vdash \phi \Leftrightarrow \forall p.\, Prov(p, \ulcorner \phi \urcorner) \Rightarrow size(p) > f(\ulcorner \phi \urcorner)$$

Now, whether or not there are proofs of ϕ whose length is bounded by $f(\ulcorner \phi \urcorner)$ is decidable, and so ϕ is provable iff it is true. But it cannot be false, since that would mean there exists a proof of it (indeed, one within the stated length bounds). Therefore ϕ must be true, and have no proof within the stated bounds; on the other hand it does have some proof.

But now we claim once again that all such theorems are likely to be theoretical pathologies, never to arise in 'normal' mathematics. Of course such a claim is rather bold, and always open to refutation, but we can at least point to some agreeable empirical evidence. First, looking at a modern, rigorous mathematical textbook such as Bourbaki's, it's hard to see how an unbridgeable gap between informal and formal proofs could arise. And in practice some significant pieces of mathematics have been formalized, notably in Mizar, Automath and HOL, without any exponential dependency appearing. On the contrary, all the empirical evidence so far is that the length of a formal proof is a fairly modest *linear* function of the length of a textbook proof, as explicitly stated by de Bruijn (1970):

> A very important thing that can be concluded from all writing experiments is the *constancy of the loss factor*. The loss factor expresses what we lose in shortness when translating very meticulous 'ordinary' mathematics into AUTOMATH. This factor may be very big, something like 10 or 20 (or 50), but it is constant; it does not increase if we go further in the book. It would not be too hard to push the constant factor down by efficient abbreviations.

[30] In a reasonable logic f, being recursive, will be representable, though in general only by a predicate rather than a function.

3.2 Extensibility and LCF

As de Bruijn says, we want to use abbreviations for common patterns of inference. The patterns of inference which are common may, of course, depend on the particular piece of mathematics being formalized, so it's desirable to allow ordinary users to extend the inference rules system, while at the same time being confident that only correct inferences are made. LCF realizes this desire exactly. Arbitrary programs can be written to prove a fact, perhaps from several premisses; the only restriction being that although terms and types may be built up, and theorems decomposed, quite arbitrarily, theorems can only be *created* by the primitive inference rules. It is not necessary that the program should prove some fact in a uniform way; it might, depending on the form of the fact to be proved, invoke one of several quite different algorithms. This is not always appreciated by outsiders, as one can see from the following assertion by Davis and Schwartz (1979): 'an LCF tactical is limited to a fixed combination of existing rules of inference'. It is *not*, and therein lies the value of the LCF approach as compared with simple macro languages.[31]

In principle, whenever there is a class of proofs of the form $A_1, \ldots, A_n \vdash B$, then a corresponding inference rule is *derivable*: we can write a HOL derived rule which implements that class of derivations. If nothing else, we can blindly search all possible proofs till we arrive at this theorem. But of course that's not practical, so really one is interested in cases where there is a natural, feasible algorithm for finding the derivation. For example, consider the class of theorems $\vdash \phi$ where ϕ is a propositional tautology. We can start from any given way of deciding that a formula is a tautology (e.g. the use of truth tables), and attempt to translate this algorithm into a version which performs proof by primitive inference.

Just how practical is this? At first sight, when one considers the enormous number of special-purpose algorithms and the likely cost of generating formal proofs, it looks completely impractical; Armando, Cimatti, and Viganò (1993) opine that it 'turns out to be both unnatural and ineffective' (though apparently on general grounds rather than on the evidence of having tried). And indeed, not all LCF systems have stuck with the rigorous policy of decomposing all inference to the primitives. For example, Nuprl users have shown no particular compunction about throwing in extra proof procedures for linear arithmetic or tautologies when the need arises. However these human factors do not provide hard evidence about feasibility.[32] The users of the HOL system at least have ploughed their own furrow, implementing a variety of quite sophisticated proof techniques, all decomposing to primitive inferences. We are now able to step back and survey the lessons. There are two questions: whether it is a reasonably tractable task for the programmer, and whether the resulting program is acceptably efficient (the latter question is closely related to our previous discussion of feasibility). It has emerged that many HOL derived rules are feasible and practical. We can identify two clear reasons why the apparent difficulties are often not really serious.

First, complex patterns of inference can be encoded as single theorems, proved once and for all. These can then be instantiated for the case in hand rather cheaply. This idea has long been used by HOL experts; for an early example see Melham (1989). A more recent, and complex, example is the following (Harrison 1993)

[31] There is also a misunderstanding of terminology outside the LCF community. A *tactic*, as described by Gordon, Milner, and Wadsworth (1979), is a specific way of supporting backward proof in terms of forward proof; a *tactical* is a higher order function for combining tactics. In some quarters today the distinguishing feature of a 'tactic' is considered to be the fact that it decomposes to primitives à la LCF. But this decomposition, which we focus on here, is a separate issue; rather than obliterate the original meaning of 'tactics', we will refer to 'derived rules' without greatly concerning ourselves whether they work forward or backward.

[32] We are uncomfortably aware that this resembles the argument that various political systems are good in principle, but merely badly implemented in practice!

justifying a step in a quantifier-elimination procedure for the reals; it states that a finite set (list) of polynomials are all strictly positive throughout an interval if and only if they are all positive at the middle of the interval and are nonzero everywhere in it.

$$\begin{array}{lll} \vdash \forall l, a, b. & a < b \wedge & \\ & (\forall x.\ a < x \wedge x < b \Rightarrow \text{FORALL (POS } x) \ l) & \\ & = \quad a < b \wedge & \\ & \quad \text{FORALL (POS } (\frac{a+b}{2})) \ l \wedge & \\ & \quad \neg \exists x.\ a < x \wedge x < b \wedge \text{EXISTS (ZERO } x) \ l & \end{array}$$

Many special-purpose algorithms can be understood as a natural process of transformation. It is not difficult to store theorems which justify the individual steps; the algorithm can often be implemented almost as one would do without the primitive inference steps, and generally with a modest linear slowdown (there are a few delicate points over the efficiency of equality testing, but that's more or less true). This applies, for example, to HOL's implementation of rewriting and definition of inductively defined relations, as well as numerous special-purpose rules that users write in the course of their work, e.g. the conversion to differentiate algebraic expressions by proof described by Harrison (1994).

Second, many other proof procedures rely heavily on *search*. For example, automated theorem provers for first order logic often perform thousands, even millions, of steps which are purely devoted to *finding* a proof. Once the proof is found, it is usually short, and can be translated into HOL inferences very easily, and proportionately the burden of translating to primitive inferences is practically nil; performance can match any other system, written in the same language. There are now several tools for first order automation in HOL, the earliest being due to Kumar, Kropf, and Schneider (1991); they all work in this way. Actually, it's not even necessary to perform proof search in the same language or the same machine; Harrison and Théry (1993) describe the use of the Maple computer algebra system as an oracle, to find solutions to integration and factorization problems which can then be checked fairly efficiently inside HOL.

The combination of these two factors is enough to make it likely that any inference patterns likely to occur in normal mathematics can be done acceptably efficiently as LCF-style derived rules. Even for the special algorithms often used in system verification, it's not inconceivable that an LCF implementation would be fast enough in practice; for example Boulton (1993) describes an implementation of a linear arithmetic decision procedure. A particularly difficult case is the BDD algorithm, to which neither of the above factors applies with great strength; however a HOL implementation has been produced by Harrison (1995) which exhibits only a linear slowdown, albeit a large one (around 50).

There is another potential difficulty to discuss. As pointed out by Pollack (1995), there are other valid inference rules, which he calls *admissible* to distinguish them from the *derivable* rules above. These are ones where the process of passing from $A_1, \ldots, A_n$ to B cannot be done in a natural way based only on the statements of the A_i, but requires more information, in particular about how the A_i were *proved*.[33] Two examples which come to mind are the Deduction theorem for Hilbert-style proof systems, which allows passage from $\Gamma \cup \{p\} \vdash q$ to $\Gamma \vdash p \Rightarrow q$, and 'Skolemization' in certain systems of intuitionistic logic, allowing a step from $\forall x.\ \exists y.\ P[x, y]$ to $\exists f.\ \forall x.\ P[x, f(x)]$. Both these can be justified by a transformation on the proofs of the hypothesis, but apparently not easily just from the form of the hypothesis. Our

[33] Of course our statement 'in a natural way' is vague; the point is that admissible rules are trivially derivable if we consider blind proof search an option. This distinction could no doubt be made rigorous.

answer to this question is: there seem to be no relevant instances of the phenomenon for 'reasonable' logics (such as HOL), and if this ever does become a problem, it is possible to store proofs too as concrete objects with the theorems. Various constructive systems are already capable of doing that, since extracting programs from proofs is one of the key ideas underlying such systems. Of course there is a question mark over whether such manipulations on proofs are computationally practical.

3.3 Metatheory and reflection

If our speculation above is wrong, what is to be done when we come across an inference pattern which can't be implemented efficiently as an LCF derived rule? One alternative is to throw in whatever additional proof procedures might be deemed necessary. But for a serious project of formalization, we need a high level of assurance that the rule is correct. If it doesn't actually perform a formal proof, how can this be done?

One idea is to use metatheory. Suppose we formalize the object logic under consideration, its syntax and its notion of provability, in an additional layer of logic, the metalogic. The idea is that metatheorems (i.e. theorems in the metalogic) can then be proved which justify formally the *possibility* of proving a result in the object logic, without actually producing the proof in complete detail. Though it's certainly possible to use a completely separate layer of logic for that purpose, as discussed by Pollack (1995) for example, it's also possible to use the *same* logic; this is often referred to as *reflection*. From Gödel's work we already know how a system can formalize its own provability. The only extra ingredient we need is a rule acting as a bridge between the internal formalization and the logic itself:[34]

$$\frac{\vdash \exists p.\ Prov(p, \ulcorner \phi \urcorner)}{\vdash \phi}$$

It is clear that if the system is 1-consistent, this rule does not introduce any new theorems, since all provable existential statements are true. Note however that the absence of hypotheses in the above theorems was crucial. The subtly different rule with a nonempty set of hypotheses, or equivalently additional axioms of the form $\vdash (\exists p.\ Prov(p, \ulcorner \phi \urcorner)) \Rightarrow p$ *do* make the logic stronger. In the special case where p is $\bot$, it amounts to a statement of consistency, which Gödel's second theorem rules out, and in fact Löb (1955) proved that such a theorem is provable in the original system only in the trivial case where $\vdash p$. Since these assertions amount to an expression of confidence in the formal system, as seen by reflecting on the system from outside (the system is consistent, everything provable is true)[35], they were dubbed 'reflection principles' by Feferman (1962), who, following on from work by Turing (1939), investigated their iterated addition as a principled way of making a formal system stronger. Our reflection rules are intended rather to be a principled way of making proofs in a system easier.

The value of using metalogic or reflection in this way depends, of course, on the fact that the metalogical proof of provability is easier (in some practically meaningful sense) than doing a full proof in the object logic. For example, this definitely seems to be the case for the Deduction Theorem and Skolemization steps discussed above: the metatheorem can be established once and for all in a general form by induction over the structure of object logic proofs, and thereafter instantiated for the

[34] Note that the formalized provability is for the *original* system, without the reflection rule. Care is needed if instances of the reflection rule are to be nested; this has been studied in a number of papers by Constable and others.

[35] By Tarski's theorem we can't actually have a predicate representing truth, but the above axioms say something intuitively close.

cases in hand very cheaply. But we've already expressed scepticism over whether these examples are realistic for logics that one would actually use in practice.

The most popular examples cited are simple algebraic manipulations. Suppose one wants to justify $a_1 + \ldots + a_n = b_1 + \ldots + b_n$ (for real numbers, say). An object-level proof would need to rewrite delicately with the associative and commutative laws; this is in fact what HOL's function `AC_CONV` does automatically. When implemented in a simple-minded way this takes $O(n^2)$ time; even when done carefully, $O(n\ log(n))$. However we can prove a metatheorem stating that if the multisets $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ are equal, then the equation holds, without any further proof. Davis and Schwartz (1979) say confidently that 'the introduction of an appropriate "algebra" rule of inference shortens to 1 the difficulty of a sentence which asserts an algebraic identity'. But assigning a difficulty of 1 is of no practical significance. What of the complexity of checking multiset equality? In fact, it is $O(n^2)$ in the worst case, according to Knuth (1973). Even if the elements are pairwise orderable somehow, we can't do better than $O(n\ log(n))$. So all this additional layer of complexity, and the tedium of proving a precise metatheorem, has achieved nothing except perhaps shaving something off the constant factor.

The constant factor is not even likely to change much. However it would change a lot more if the multiset equality test didn't have to be performed in some formal system, but could be compared directly by a program. This suggests another popular variant of the reflection idea: computational reflection. The idea is to verify that a program will correctly implement a rule of inference (e.g. correctly test for multiset equality and in the case of success, add the appropriate equation to the stock of theorems), then add that program to the implementation of the theorem prover, so thereafter it can be executed directly. This does indeed offer much more efficiency, and in certain special situations like arithmetical calculation, the speedup could be considerable. It would also be a systematic way of gaining confidence in the many theorems which are nowadays proved with the aid of computer checking. The proof of the 4-colour theorem has already been mentioned; more prosaic examples include simply assertions that large numbers are prime. Journals in some fields already have to confront difficult questions about what constitutes a proof in this context; the issue is discussed by Lam (1990) for example.

However there are a number of difficulties with the scheme. In particular, correctly formalizing the semantics of real programming languages and then proving nontrivial things about them is difficult. The formal system becomes dramatically more complicated, since its rules effectively include the full semantics of the implementation language, instead of being separate and abstract. (This has a negative effect on the possibility of *checking* proofs using simple proof-checking programs, an oft-suggested idea to increase confidence in the correctness of machine-checked proofs.) And finally, reflection, though extensively studied, has not often been done in practice, the work of Boyer and Moore (1981) being probably the most significant example to date.[36]

We should add that the idea of proving program correctness doesn't necessarily demand some kind of logical reflection principle to support it. Boyer and Moore, for example, need no such thing. Typical decision procedures only involve a rather restricted part of the logic (e.g. tautologies or arithmetic inequalities), and the semantics of this fragment can perfectly easily be internalized, even though by Tarski's theorem this cannot be done for the whole logic. (Nuprl's type theory, stratified by universe level, allows one to come very close to the ideal: the semantics of one layer can be formalized in the next layer up.) Ostensibly 'syntactic' manipulations can often be done directly in ordinary logic or set theory, defining some appropriate

[36]The author has been told that an attempt was made to verify in Nuprl the correctness of a Presburger arithmetic implementation, but the researcher involved eventually became discouraged by the difficulty of the proofs.

semantics; following Howe (1988), this is often called 'partial reflection'. We should also point out the following. It is often stated that, issues of efficiency apart, some of the theorems in mathematics books *are* metatheorems, or can only be proved using metatheory — 'most of' them according to Aiello, Cecchi, and Sartini (1986). Matthews (1994) is one of the few who cites an example:

> For instance in a book on algebra one might read:
>
> > 'If A is an abelian group, then, for all a, b in A, the equivalence
> >
> > $$\overbrace{(a \circ b) \circ \ldots \circ (a \circ b)}^{n\ times} = \overbrace{(a \circ \ldots \circ a)}^{n\ times} \circ \overbrace{(b \circ \ldots \circ b)}^{n\ times}$$
> >
> > holds'
>
> ... On the other hand, instead of a book, imagine a proof development system for algebra; there the theorem cannot be stated, since it is not a theorem of abelian group theory, it is, rather, a meta-theorem, a theorem *about* abelian group theory.

But all this is from the completely irrelevant perspective of an axiomatization of group theory in first order logic. As we have already stated, any serious development of group theory is going to use higher-order or set-theoretic devices, as Bourbaki does for example. With these mathematical resources, iterated operations are trivially definable and their basic properties easily provable — this appears right at the very beginning of the well-known algebra textbook of Lang (1994) for example! The stress on direct first order axiomatizations in the AI community probably results from a laudable scepticism in some quarters towards the fancy new logics being promulgated by others, but it completely fails to reflect real mathematical practice. What *is* true is that one often sees statements like 'the other cases are similar' or 'by symmetry we may assume that $m \leq n$'. These have the flavour of metatheorems in that they relate similar *proofs* rather than *theorems*. But bear in mind that when presenting proofs informally, this is just one of many devices used to abbreviate them. Certainly, a way of formalizing statements like this is as metatheorems connecting proofs. But another is simply to do (or program the computer to do!) all the cases explicitly, or to use an object-level theorem directly as justification, e.g.

$$(\forall m, n.\ P(m, n) \Leftrightarrow P(n, m)) \Rightarrow ((\forall m, n.\ P(m, n)) \Leftrightarrow (\forall m, n.\ m \leq n \Rightarrow P(m, n))$$

Performing similar proofs in different contexts may anyway be a hint that one should be searching for a suitable mathematical generalization to include all the cases.

To summarize: we have yet to find the LCF approach inadequate, and yet to see really useful implementations of reflection. Whatever exotic principles are added, the provable sentences will form a recursively enumerable set (we are after all considering a computer implementation), and once again there will be unprovable statements and infeasible ones. Now since we haven't come across any interesting infeasible statements yet it seems extremely conjectural that they won't still be infeasible in the expanded system. Boyer reports that the metafunction facility of NQTHM is not widely used in practice. And Constable, Knoblock, and Bates (1985), who are no enemies of the idea of reflection as indicated by their extensive writings on the subject, say: 'Of all these methods [of providing extensibility] we have found the LCF idea easiest to use and the most powerful.'

3.4 How much automation do we want?

We have that automation is *possible* even while producing an explicit proof, but how much do we actually 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 the step may not be at all obvious in the sense of having a formal proof which is easy to find using one of the common techniques for automated proof search. An interesting discussion of just what is logically obvious is given by Rudnicki (1987). 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 or even subtle predicate calculus reasoning like the following theorem due to Łoś:

$$\begin{array}{l}(\forall x\ y\ z.\ P(x,y) \wedge P(y,z) \Rightarrow P(x,z)) \wedge \\ (\forall x\ y\ z.\ Q(x,y) \wedge Q(y,z) \Rightarrow Q(x,z)) \wedge \\ (\forall x\ y.\ Q(x,y) \Rightarrow Q(y,x)) \wedge \\ (\forall x\ y.\ P(x,y) \vee Q(x,y)) \\ \Rightarrow (\forall x\ y.\ P(x,y)) \vee (\forall x\ y.\ Q(x,y))\end{array}$$

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:

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

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.

3.5 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. It should be possible to provide a degree of tailoring to suit individual tastes.[37] 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 the Unix environment is partly because it is formal, simple, flexible, 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.

It is not necessary to use a full-blown logical symbolism; one can quite well have stylized constructs of ordinary English. Mizar, for example, does not use special symbols for quantifiers and connectives. This is largely a matter of taste. Some writers like Halmos oppose the use of logical symbolism, but as far as we can see this is just a matter of personal taste, and no more in need of justification than the use of symbols for arithmetic operations. Current systems certainly leave a lot to be desired in the matter of user interaction. HOL tactic scripts are an arcane mix of higher order functions; some systems like IMPS, NQTHM and ONTIC use LISP syntax, which is even further from conventional notation. Mizar's proof scripts are undoubtedly the closest to familiar mathematical notation. This was after all one of the goals of the Mizar project; perhaps the developers were also less corrupted by the tendency to ready acceptance of machine-oriented syntax which tends to arise among computer scientists.

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

[37] It has been remarked that computer scientists would rather use each other's toothbrushes than each other's notation.

itself become an accepted interlingua[38] for mathematical texts; though it would be rather spartan and colourless, the success of international chess publications like Šakovski Informator, which are languageless, is worth noting.[39]

Rendering proofs in HTML or using another hypertext format, to allow users to examine proofs at different levels of details, is a topic which has already attracted some attention. Conversely, one could imagine, in education, having different levels of automated theorem proving. Just as now, students are supposed to master arithmetic before being given calculators, it might be thought desirable for them to master constructing formal proofs before being given computer assistance. (To some extent, this depends on whether one is trying to teach logic or mathematics.) Several systems have already been used in education, notably Mizar, with interesting results as recounted by Szczerba (1989):

> There occurred a substantial change in my role as a teacher. Earlier, when assigning homework tests, I was treated as an enemy who has to be forced to accept the solution, sometimes in not an exactly honest way. Now the enemy to be defeated was the computer and I was turned into an ally helping to fight this horrible device. This small fact has seriously influenced my contacts with the students. They were much more eager to approach me with their problems, to report on their difficulties, and ask for help.

3.6 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 to distinguish between a function and 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. Of course, for a few exceptionally explicit and precise textbooks, this may already be the case.

[38] The artificial language Interlingua was, by the way, invented by Peano, whose interest in new languages was not restricted to the mathematical field.

[39] And apropos of Halmos's sceptical view of logical symbolism, the author has heard chess players use the Informator abbreviations in conversation, e.g. 'triangle' instead of 'with the idea'.

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 profound and interesting difficulties of formalizing and proving theorems, but with the prosaic matter of 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 mathematics texts, readers can only remember a few 'big names' like the Heine-Borel theorem, the Nullstellensatz, and the rather ironic relic of 'Hilbert's Theorem 90' which in retrospect needed a name!. For run of the mill theorems, it's better to number them: although the numeration scheme may be without significance, it does at least facilitate relatively easy lookup (though some books unhelpfully adopt distinct numbering streams for theorems, lemmas, corollaries and definitions). Mizar numbers its theorems, though inside textual units which are named. But this policy seems to us less suitable for a computer theorem prover because it's more common to want to pass from a (fancied) theorem to its identifier, rather than vice versa. Admittedly in the long term we want people to read formal proofs, but at present the main emphasis on constructing them.

With a computer system, textual lookup is at least as easy as numeric lookup, and names can have a mnemonic value, so using names throughout seems a better choice. It also makes it easier to insert or rearrange theorems without destroying any logic in the sequence of numbers. HOL names all theorems, but some theories do not have a consistent naming convention, notably natural number arithmetic which used to include the aptly-named `TOTALLY_AD_HOC_LEMMA`. A computer can offer the new possibility of looking up a theorem according to 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. (In a computer system, there is a tendency to have the writer's own convenience at heart rather than that of a later reader or user. Perhaps therein lies the mistake.)

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 $\leq$ (or $\geq$!). 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.

3.7 The Future

Only with more practical experience, ENOD[40] 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. Accepting that geometrical insight, or even geometric proof techniques (even algebraic subjects often include commuting diagrams!) likely to be useful, the problem arises of fitting them into existing formal calculi, or developing better calculi. The difficulty has been noted by Yamamoto, Nishizaha, Hagiya, and Toda (1995):

> Among many fields of mathematics and computer science, discrete mathematics is one of the most difficult to formalize because we prove theorems using intuitive inferences that have not been rigorously formalized yet.

Recently a proposal to draw together the world's theorem proving communities in a 'QED Project' (Anonymous 1994) to formalize all mathematics has been proposed. Such an undertaking would demand a huge collaborative venture, perhaps comparable to the Human Genome Project. We haven't spent so much of our time here on advocacy, though some of the potential benefits of formalized mathematics have been mentioned. Whatever one's opinion of these grandiose dreams, we share the optimism of the QED proponents that formalization of mathematics is both useful and practicable. What's more, given the disparate research community and the wealth of experience collected piecemeal, its greater development, for good or bad, seems to be inevitable.

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 this paper was started. For making this and other meetings of researchers possible, Bob Boyer deserves every credit; I am also grateful for his encouragement to write this paper. I have also benefited from conversations with Mike Gordon and with other people at the QED conference, and with members of the Programming Methodology group at Åbo Akademi. My visit to Poland was funded by the U.S. Office of Naval Research, and my later work was supported by the U.K. Physical Sciences and Engineering Research Council, the Isaac Newton Trust, and more recently by the European Commission under the Human Capital and Mobility programme. Piotr Rudnicki gave me some helpful comments on an early draft, and discussions with Randy Pollack about the issues of feasibility and extensibility were very stimulating.

Glossary

Here we list some of the logical symbols we use in the paper, for the benefit of readers who aren't used to symbolism at all, or are used to different symbols. Binary

[40] Experience Not Only Doctrine.

connectives bind according to their order in the table below, $\neg$ being the strongest. The scope of a quantifier, lambda-abstraction or descriptor is as far to the right of the dot as possible. For example $\forall x.\ P[x] \wedge Q[x] \Rightarrow R[x]$ is parsed as $\forall x.\ ((P[x] \wedge Q[x]) \Rightarrow R[x])$. We make a fuss about this last point because some books adopt the opposite convention. The table below gives the symbolic expressions we will use.

Symbol	Other notations	English reading
$\bot$	F, 0	false
$\top$	T, 1	true
$\neg P$	$\sim P$, $-P$	not P
$P \wedge Q$	P & Q, P . Q, PQ	P and Q
$P \vee Q$	$P \mid Q$, P *or* Q, $P + Q$	P or Q
$P \Rightarrow Q$	$P \rightarrow Q$, $P \supset Q$	P implies Q
$P \Leftrightarrow Q$	$P \sim Q$, $P \equiv Q$	P if and only if Q
$\forall x.\ P$	$(\forall x)\ P$, $\forall x\ P$, $(x)\ P$, $\bigwedge x\ P$	for all x, P
$\exists x.\ P$	$(\exists x)\ P$, $\exists x\ P$, $(Ex)\ P$, $\bigvee x\ P$	there exists an x such that P
$\varepsilon x.\ P$		some x such that P
$\iota x.\ P$	$\tau x.\ P$	the unique x such that P
$\lambda x.\ t$	$x \mapsto t$, $[x]t$	the function of x which yields t
$\vdash P$		P is deducible (in a formal system)

References

Aiello, L., Cecchi, C., and Sartini, D. (1986) Representation and use of metaknowledge. *Proceedings of the IEEE*, **74**, 1304–1321.

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.

Anonymous (1994) The QED Manifesto. See Bundy (1994), pp. 238–251.

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`.

Armando, A., Cimatti, A., and Viganò, L. (1993) Building and executing proof strategies in a formal metatheory. In Torasso, P. (ed.), *Advances in Artifical Intelligence: Proceedings of the Third Congress of the Italian Association for Artificial Intelligence, IA*AI'93*, Volume 728 of *Lecture Notes in Computer Science*, Torino, Italy, pp. 11–22. Springer-Verlag.

Arnol'd, V. I. (1990) *Huygens and Barrow, Newton and Hooke: pioneers in mathematical analysis and catastrophe theory from evolents to quasicrystals*. Birkhauser. Translated from the Russian by Eric J.F. Primrose.

Beckert, B. and Posegga, J. (1994) leanTAP: lean, tableau-based theorem proving. See Bundy (1994), pp. 793–797. Extended version available on the Web from `http://i12www.ira.uka.de/~posegga/LeanTaP.ps.Z`.

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.

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.

Boole, G. (1848) The calculus of logic. *The Cambridge and Dublin Mathematical Journal*, **3**, 183–198.

Boulton, R. J. (1993) Efficiency in a fully-expansive theorem prover. Technical Report 337, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK. Author's PhD thesis.

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

Bourbaki, N. (1966) *General topology*, Volume 1 of *Elements of mathematics*. Addison-Wesley. Translated from French 'Topologie Genérale' in the series 'Eléments de mathématique', originally published by Hermann in 1966.

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

Boyer, R. S. and Moore, J S. (1981) Metafunctions: proving them correct and using them efficiently as new proof procedures. In Boyer, R. S. and Moore, J S. (eds.), *The Correctness Problem in Computer Science*, pp. 103–184. 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. (1970) The mathematical language AUTOMATH, its usage and some of its extensions. See Laudet, Lacombe, Nolin, and Schützenberger (1970), pp. 29–61.

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.

Bundy, A. (ed.) (1994) *12th International Conference on Automated Deduction*, Volume 814 of *Lecture Notes in Computer Science*, Nancy, France. Springer-Verlag.

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

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

Chwistek, L. (1922) Über die Antinomien de Principien der Mathematik. *Mathematische Annalen*, **14**, 236–243.

Constable, R. L., Knoblock, T. B., and Bates, J. L. (1985) Writing programs that construct proofs. *Journal of Automated Reasoning*, **1**, 285–326.

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.

Davis, M. (ed.) (1965) *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Raven Press, NY.

Davis, M. and Schwartz, J. T. (1979) Metatheoretic extensibility for theorem verifiers and proof-checkers. *Computers and Mathematics with Applications*, **5**, 217–230.

Dedekind, R. (1888) *Was sind und was sollen die Zahlen*. Vieweg, Braunschweig. English translation in Dedekind (1901).

Dedekind, R. (ed.) (1901) *Essays on the theory of numbers*. Open Court, NY. Reprinted by Dover, 1963.

Dijkstra, E. W. (1985) Invariance and non-determinacy. In Hoare, C. A. R. and Shepherdson, J. C. (eds.), *Mathematical Logic and Programming Languages*, Prentice-Hall International Series in Computer Science, pp. 157–165. Prentice-Hall. The papers in this volume were first published in the Philosophical Transactions of the Royal Society, Series A, vol. 312, 1984.

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.

Feferman, S. (1962) Transfinite recursive progressions of axiomatic theories. *Journal of Symbolic Logic*, **27**, 259–316.

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.

Freyd, P. J. and Scedrov, A. (1990) *Categories, allegories*. North-Holland.

van Gasteren, A. J. M. (1990) *On the shape of mathematical arguments*, Volume 445 of *Lecture Notes in Computer Science*. Springer-Verlag. Foreword by E. W. Dijkstra.

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.

Gödel, K. (1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte für Mathematik und Physik*, **38**, 173–198. English translation, 'On Formally Undecidable Propositions of Principia Mathematica and Related Systems, I', in van Heijenoort (1967), pp. 592–618 or Davis (1965), pp. 4–38.

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. (1993) A HOL decision procedure for elementary real algebra. See Joyce and Seger (1993), pp. 426–436.

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

Harrison, J. (1995) Binary decision diagrams as a HOL derived rule. *The Computer Journal*, **38**, 162–170.

Harrison, J. and Théry, L. (1993) Extending the HOL theorem prover with a computer algebra system to reason about the reals. See Joyce and Seger (1993), pp. 174–184.

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.

Hilbert, D. (1899) *Grundlagen der Geometrie*. Teubner. English translation 'Foundations of Geometry' published in 1902 by Open Court, Chicago.

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`.

Howe, D. J. (1988) Computational metatheory in Nuprl. In Lusk, E. and Overbeek, R. (eds.), *9th International Conference on Automated Deduction*, Volume 310 of *Lecture Notes in Computer Science*, Argonne, Illinois, USA, pp. 238–257. Springer-Verlag.

Joyce, J. J. and Seger, C. (eds.) (1993) *Proceedings of the 1993 International Workshop on the HOL theorem proving system and its applications*, Volume 780 of *Lecture Notes in Computer Science*, UBC, Vancouver, Canada. Springer-Verlag.

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.

Kanger, S. (1963) A simplified proof method for elementary logic. In Braffort, P. and Hirschberg, D. (eds.), *Computer programming and formal systems*, pp. 89–95. North-Holland.

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.

Knuth, D. E. (1973) *The Art of Computer Programming; Volume 3: Sorting and Searching*. Addison-Wesley Series in Computer Science and Information processing. Addison-Wesley.

Kreisel, G. (1958) Hilbert's programme. *Dialectica*, **12**, 346–372. Revised version in Benacerraf and Putnam (1983).

Kreisel, G. (1970) Hilbert's programme and the search for automatic proof procedures. See Laudet, Lacombe, Nolin, and Schützenberger (1970), pp. 128–146.

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.

Kreisel, G. and Krivine, J.-L. (1971) *Elements of mathematical logic: model theory* (Revised second ed.). Studies in Logic and the Foundations of Mathematics. North-Holland. First edition 1967. Translation of the French 'Eléments de logique mathématique, théorie des modeles' published by Dunod, Paris in 1964.

Kumar, R., Kropf, T., and Schneider, K. (1991) Integrating a first-order automatic prover in the HOL environment. In Archer, M., Joyce, J. J., Levitt, K. N., and Windley, P. J. (eds.), *Proceedings of the 1991 International Workshop on the HOL theorem proving system and its Applications*, University of California at Davis, Davis CA, USA, pp. 170–176. IEEE Computer Society 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.

Lakatos, I. (1980) What does a mathematical proof prove? In Worrall, J. and Currie, G. (eds.), *Mathematics, science and epistemology. Imre Lakatos: Philosophical papers vol. 2*, pp. 61–69. Cambridge University Press.

Lam, C. W. H. (1990) How reliable is a computer-based proof? *The Mathematical Intelligencer*, **12**, 8–12.

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.

Lang, S. (1994) *Algebra* (3rd ed.). Addison-Wesley.

Laudet, M., Lacombe, D., Nolin, L., and Schützenberger, M. (eds.) (1970) *Symposium on Automatic Demonstration*, Volume 125 of *Lecture Notes in Mathematics*. Springer-Verlag.

Lecat, M. (1935) *Erreurs de Mathématiciens*. Brussels.

Littlewood, J. E. (1986) *Littlewood's Miscellany*. Cambridge University Press. Edited by Bela Bollobas.

Löb, M. H. (1955) Solution of a problem of Leon Henkin. *Journal of Symbolic Logic*, **20**, 115–118.

Loveland, D. W. (1968) Mechanical theorem-proving by model elimination. *Journal of the ACM*, **15**, 236–251.

Mac Lane, S. (1986) *Mathematics: Form and Function*. Springer-Verlag.

MacKenzie, D. (1995) The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, **17**(3), 7–29.

Maslov, S. J. (1964) An inverse method of establishing deducibility in classical predicate calculus. *Doklady Akademii Nauk*, **159**, 17–20.

Mathias, A. R. D. (1991) The ignorance of Bourbaki. *Physis; rivista internazionale di storia della scienza (nuova serie)*, **28**, 887–904. Also in 'The Mathematical Intelligencer' vol. 14, nr. 3, pp. 4–13, 1992.

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.

Matthews, S. (1994) A theory and its metatheory in FS_0. Publication unknown.

Melham, T. F. (1989) Automating recursive type definitions in higher order logic. In Birtwistle, G. and Subrahmanyam, P. A. (eds.), *Current Trends in Hardware Verification and Automated Theorem Proving*, pp. 341–386. Springer-Verlag.

Morse, A. P. (1965) *A theory of sets*. New York.

Naur, P. (1994) Proof versus formalization. *BIT*, **34**, 148–164.

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.

Newell, A. and Simon, H. A. (1956) The logic theory machine. *IRE Transactions on Information Theory*, **2**, 61–79.

Paris, J. and Harrington, L. (1991) A mathematical incompleteness in Peano Arithmetic. In Barwise, J. and Keisler, H. (eds.), *Handbook of mathematical logic*, Volume 90 of *Studies in Logic and the Foundations of Mathematics*, pp. 1133–1142. North-Holland.

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.

Pollack, R. (1995) On extensibility of proof checkers. In Dybjer, P., Nordström, B., and Smith, J. (eds.), *Types for Proofs and Programs: selected papers from TYPES'94*, Volume 996 of *Lecture Notes in Computer Science*, Båstad, pp. 140–161. Springer-Verlag.

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

Prawitz, D., Prawitz, H., and Voghera, N. (1960) A mechanical proof procedure and its realization in an electronic computer. *Journal of the ACM*, **7**, 102–128.

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.

Robinson, J. A. (1965) A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**, 23–41.

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.

Spivey, J. M. (1988) *Understanding Z: a specification language and its formal semantics*, Volume 3 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Stålmarck, G. (1994) System for determining propositional logic theorems by applying values and rules to triplets that are generated from Boolean formula. United States Patent number 5,276,897.

Stickel, M. E. (1988) A Prolog Technology Theorem Prover: Implementation by an extended Prolog compiler. *Journal of Automated Reasoning*, **4**, 353–380.

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

Szczerba, L. W. (1989) The use of Mizar MSE in a course in foundations of geometry. In Srzednicki, J. (ed.), *Initiatives in logic*, Volume 2 of *Reason and argument series*. M. Nijhoff.

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. (1936) Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica*, **1**, 261–405. English translation, 'The Concept of Truth in Formalized Languages', in Tarski (1956), pp. 152–278.

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

Tarski, A. (ed.) (1956) *Logic, Semantics and Metamathematics*. Clarendon Press.

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

Trybulec, Z. and Święczkowska, H. (1991-1992) The language of mathematical texts. *Studies in Logic, Grammar and Rhetoric*, **10/11**, 103–124.

Turing, A. M. (1936) On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (2)*, **42**, 230–265.

Turing, A. M. (1939) Systems of logic based on ordinals. *Proceedings of the London Mathematical Society (2)*, **45**, 161–228. Reprinted in Davis (1965), pp. 154–222.

Upsenskii, V. A. (1961) *Some Applications of Mechanics to Mathematics*, Volume 3 of *Popular lectures in mathematics*. Pergamon. Translated from the Russian by Halina Moss; translation editor Ian N. Sneddon.

Whitehead, A. N. (1919) *An Introduction to Mathematics*. Williams and Norgate.

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.

Wang, H. (1960) Toward mechanical mathematics. *IBM Journal of research and development*, **4**, 2–22.

Yamamoto, M., Nishizaha, S.-y., Hagiya, M., and Toda, Y. (1995) Formalization of planar graphs. In Windley, P. J., Schubert, T., and Alves-Foss, J. (eds.), *Higher Order Logic Theorem Proving and Its Applications: Proceedings of the 8th International Workshop*, Volume 971 of *Lecture Notes in Computer Science*, Aspen Grove, Utah, pp. 369–384. Springer-Verlag.

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.