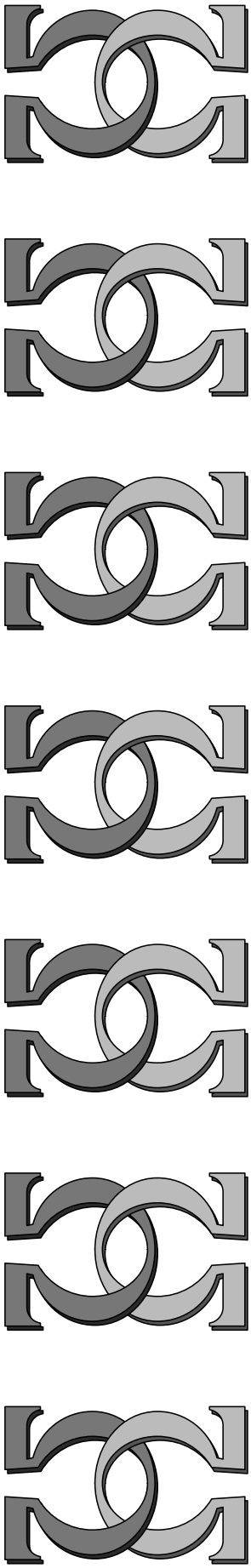

CDMTCS
Research
Report
Series

Proving and Programming

C. S. Calude, E. Calude, S. Marcus
University of Auckland, NZ
Massey University at Albany, NZ
Romanian Academy, Romania

CDMTCS-309
June 2007

Centre for Discrete Mathematics and
Theoretical Computer Science

Proving and Programming

Cristian S. Calude
Department of Computer Science, University of Auckland, New Zealand
cristian@cs.auckland.ac.nz

Elena Calude
IIMS, Massey University at Albany, New Zealand
e.calude@massey.ac.nz

Solomon Marcus
Romanian Academy, Mathematics, Bucharest, Romania
Solomon.Marcus@imar.ro

Abstract

There is a strong analogy between proving theorems in mathematics and writing programs in computer science. This paper is devoted to an analysis, from the perspective of this analogy, of *proof* in mathematics. We will argue that while the Hilbertian notion of proof has few chances to change, future proofs will be of various types, will play different roles, and their truth will be checked differently. Programming gives mathematics a new form of understanding. The computer is the driving force behind these changes.

1 Introduction

The current paper, a continuation of [12], is devoted to an analysis of *proof* in mathematics from the perspective of the analogy between proving theorems in mathematics and writing programs in computer science. We will argue that:

1. Theorems (in mathematics) correspond to algorithms and not programs (in computer science); algorithms are subject to mathematical proofs (for example for correctness).

2. The role of proof in mathematical modelling is very small: adequacy is the main issue.

3. Programs (in computer science) correspond to mathematical models. They are not subject to proofs, but to an adequacy and relevance analysis; in this type of analysis, some proofs may appear. Correctness proofs in computer science (if any) are not cost-effective.

4. Rigour in programming is superior to rigour in mathematical proofs.

5. Programming gives mathematics a new form of understanding.

6. Although the Hilbertian notion of proof has few chances to change, future proofs will be of various types and will play different roles, and their truth will be checked differently.

2 Proving vs. programming: today

Aristotle introduced the concept of proof as an epistemological tool, to establish *absolutely certain knowledge*. The argument follows a sequence of rigourously defined steps, starting from "first principles", which are claimed to be *self-evident truths*, using rules which are *truth-preserving*. The original intention was to derive *certain conclusions*, but this goal seems to be too ambitious.

In mathematics, the first principles are called axioms, and the rules are referred to as deduction/inference rules. A proof is a series of steps based on the (adopted) axioms and deduction rules which reaches a desired conclusion. Every step in a proof can be checked for correctness by examining it to ensure that it is logically sound.

According to Hilbert:

> The rules should be so clear, that if somebody gives you what they claim is a proof, there is a mechanical procedure that will check whether the proof is correct or not, whether it obeys the rules or not.

While ideally sound, this type of proof (called *Hilbertian* or *monolithic* [21]) cannot be found in mathematical articles or books (except for a few simple examples). However, most mathematicians believe that almost all "real" proofs, published in articles and books, can, with tedious work, be transformed into Hilbertian proofs. Why? Because real proofs look *convincing* for the mathematical community [21]. Going further, DeMillo, Lipton and Perlis argued that real proofs should be highly non-monolithic

because they aim to be heard, read, assimilated, discussed, used and generalised by mathematicians—they are part of a social process.

Deductive rules are truth-preserving, but although the conclusion, generically termed as theorem, yields *knowledge*[1], there is no claim that it yields *certain knowledge*. The reason is simple: nothing certifies the truth of the axioms. The epistemic status of axioms is an interesting and troubling question. Relativity appears in many instances, from the plurality of geometries (Euclidean and non-Euclidean), to Gödel's incompleteness theorem and various independence results (Continuum Hypothesis). Mathematically, there is no most preferred geometry neither the true set theory. *If there are no self-evident truths, then there is no certain knowledge.* For Thurston [54]

> ... the foundations of mathematics are much shakier than the mathematics that we do.

Programming is the activity of solving problems with computers. It includes the following steps: a) developing the *algorithm* to solve the problem, b) writing the algorithm in a specific programming language (that is, coding the algorithm into a program), c) assembling or compiling the program to turn it into machine language, d) testing and debugging the program, e) preparing the necessary documentation.

Ideally, at d) one should have written: *proving the correctness of the algorithm*, testing and debugging the program. We said, "ideally", because correctness, although desired, is only practised in very few instances (for example, the programs involved in the proof of the Four-Color Theorem were not proved correct.)[2] In programming practice (in a commercial/industrial environment), correctness is seldom required and much less proved. Some reasons include: a) the fact that, in general, correctness is undecidable [11], so failure to prove correctness has little meaning, b) for most non-trivial cases, correctness is a monumental task which gives an added confidence at a disproportionate cost. There are many projects and products dedicated to automated testing of program correctness, for example, VIPER or the more recent TestEra (automated testing of Java programs). The current approach in software and hardware design involves a combination of empirical and formal methods aiming to get better and better programs.

3 Mathematical examples

We will analyse three mathematical problems which reflect the current evolution of mathematical proofs.

[1] We may called this *deductive knowledge*.

[2] See the proof [53, 10].

3.1 The twin prime conjecture

The twin prime conjecture states that there are infinitely many pairs of twin primes (i.e. pairs $(p, p+2)$, where p and $p+2$ are primes). To date, all attempts to solve the problem have failed. (Recently, Arenstorf's proof [2] had a serious error and the paper was retracted.) In 2005, Goldston, Pintz and Yildirim [29] proved that there are infinitely many primes for which the gap to the next prime is as small as possible when compared with the average gap between consecutive primes. This spectacular result comes very close to the twin prime conjecture, which asserts that, apart from the case of 2 and 3, the gap is the smallest possible, i.e. 2. The Goldston–Pintz–Yildirimir proof is explained in a recent paper by Soundararajan [52]. Here are the main questions discussed: Can we say anything about the statistical distribution of gaps between consecutive primes? Does every even number[3] occur infinitely many times as a gap between consecutive primes? How frequently do twin primes occur? Interesting for our discussion is the following comment [52] (p. 2):

> Number theorists believe they know the answers to all these questions but cannot always prove that the answers are correct.

Earlier, in 2003, Goldston and Yildirim announced that there are infinitely many primes such that the gap to the next prime is very small. The proof looked convincing till A. Granville and K. Soundararajan discovered a tiny flaw which looked fatal (see [23] for the story). The flaw was discovered not by carefully *checking* Goldston and Yildirim's proof, but by *extending* it to show that there are infinitely many primes such that the gap to the next prime is less than 12 (the gap-12 theorem), a result which was too close to the twin prime conjecture to be true: they didn't believe it! B. Conrey, the director of the American Institute of Mathematics, which was close to this work, is quoted by Devlin [23] by saying that, without the "unbelievable" Granville and Soundararajan gap-12 theorem,

> the Goldston-Yildirim proof would in all probability have been published and the mistake likely not found for some years.

How many proofs are wrong? Although many (most?) proofs are probably "incomplete or benignly wrong"—that is, they can be in principle fixed—it is almost impossible to make an educated guess about how many proofs are wrong. One reason is that many proofs are only superficially checked, because either they have limited interest or they never come to be used (or both).

[3] Every gap between primes is even except for 3 and 2.

3.2 The Kepler conjecture

In 1997, Thomas Hales [31] published a detailed plan—a mixture of mathematical proofs and extensive computer calculations (see also [55])—describing a new strategy for attacking the Kepler conjecture. Hales' full proof appears in a series of papers taking up more than 250 pages and gigabytes of computer programs and data [15], all posted on the Internet. The proof cannot be checked without running his programs. In the end, his paper [32] was published in Annals of Mathematics, in spite of the inconclusive verdict given by the panel of 12 referees (see more in [14]).

The semi-failure of the correctness-checking process motivated Hales to start the project Flyspeck [27] whose purpose is *to produce a formal proof of the Kepler Conjecture*. Hales estimates that the project will run for 20 years before reaching a final conclusion.

We are motivated to join T. Anderson [9] (p. 2389) in asking the question: *must we consign mathematics to the dustbin until computers have confirmed the validity of the theorems and proofs?*

3.3 The Poincaré conjecture

The Poincaré conjecture was stated in 1904 by Henri Poincaré [43]. Colloquially, it states that the three-dimensional sphere is the only type of bounded three-dimensional space possible that contains no holes. The first solution carries a $1 million prize, as one of the Millennium Problems of the Clay Mathematical Foundation [20].[4]

The most significant scientific achievement of 2006, the *Breakthrough of the Year*, was, according to *Science* magazine, the solution of the Poincaré conjecture by Grigory Perelman [35]:

> This year's Breakthrough salutes the work of a lone, publicity-shy Russian mathematician named Grigori Perelman, who was at the Steklov Institute of Mathematics of the Russian Academy of Sciences until 2005. The work is very technical but has received unusual public attention because Perelman appears to have proven the Poincaré Conjecture, a problem in topology whose solution will earn a $1 million prize from the Clay Mathematics Institute. That's only if Perelman survives what's left of a 2-year gauntlet of critical attack required by the Clay prize rules.[5]

[4] If you think that this is a lot of money, then refer to the BBC announcement—broadcast as we are writing this paper—confirming that "David Beckham will leave Real Madrid and join Major League Soccer side LA Galaxy at the end of the season"; he will be paid $1 million *per week* [4].

[5] Most mathematicians think he will.

Perelman's solution appears in a series of papers he circulated and posted on the Internet (not published yet?) in 2003 [40, 41, 42]. In 2006, his solution was officially recognised: Perelman was offered the Fields Medal. As is well-known (and publicised), Perelman declined the Fields Medal. Rumour has it that he has expressed little interest in the Clay prize too.

Beyond obvious journalistic aspects, the story is significant in at least two directions: a) it reinforces the social aspect of the process of doing mathematics, as the Clay prize rules stipulate that an acceptable solution has to resist the critical analysis of the mathematical community for no less than two years, and b) the community accepted a result which was only posted on the Internet. While a) is not surprising in the least, we may ask *whether b) is the starting of a new pattern in mathematical communication.*

4 Computer science examples

We continue with three examples from computer science.

4.1 Intel's bug

According to *Wikipedia*,

> A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result).

"Flaw Reported in New Intel Chip" made the headlines of the Technology/Cybertimes section of the *New York Times* on May 6, 1997:

> The Intel Corp. will not formally introduce its Pentium II microprocessor until Wednesday, but the World Wide Web is already buzzing with reports of a bug that causes the new chip to make errors in some complex mathematical calculations.

Only three years earlier, on December 20, 1994, Intel recalled its popular Pentium processor due to an "FDIV" bug discovered by Thomas Nicely, who was working on, guess what? He was calculating Brun's sum [38], the series formed with the reciprocal of twin primes:

$$\left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\left(\frac{1}{17}+\frac{1}{19}\right)+\cdots<\infty.$$

Nicely worked with five PC's using Intel's 80486 microprocessor and a Pentium [37]. Comparing the results obtained with the old machines and the new Pentium, he observed a discrepancy in the calculation of the reciprocals of the twin primes 824,633,702,441 and 824,633,702,443. Running various tests, he identified the source of error in the floating point hardware unit of the Pentium CPU. Twenty three other errors were found by Andreas Kaiser, while Tim Coe arrived at the simplest error instance: the division 4,195,835/3,145,727—which evaluates to $1.33382044\cdots$—appears on the Pentium to be $1.33373906\cdots$ Coe's ultra-simple example moved the whole story from the Internet to *New York Times*.

In contrast with errors found in mathematical proofs, which remain within the realm of mathematical experts, computer bugs attract the attention of a larger audience. For example, on January 17, 1995, Intel announced that it will spend $475 million to cover the recall of its Pentium chip to fix the problem discussed above, a problem that may affect only a few users.

Can bugs be avoided? More to the point of this article, *can the use of rigorous mathematical proofs guarantee that software and hardware perform as expected?*

4.2 From algorithms to programs

Bloch [6] identifies a bug in the Java implementation of a standard binary search[6]. Here is Bloch's code:

```
1:     public static int binarySearch(int[] a, int key) {
2:         int low = 0;
3:         int high = a.length - 1;
4:
5:         while (low <= high) {
6:             int mid = (low + high) / 2;
7:             int midVal = a[mid];
8:
9:             if (midVal < key)
10:                 low = mid + 1;
11:             else if (midVal > key)
12:                 high = mid - 1;
13:             else
14:                 return mid; // key found
15:         }
```

[6]Apparently was only recently reported to Sun, persisting for nine years.

```
16:         return -(low + 1);  // key not found.
17:     }
```

The bug is identified in line 6

```
 6:             int mid =(low + high) / 2;
```

with the explanation that the average value is truncated down to the nearest integer, a statement which is *true* for integers, but *false* for "bounded integers". If the sum of `low + high` is higher than $2^{31} - 1$, then the value overflows to a negative value and stays negative by division to 2. How frequently can this situation appear? For arrays longer in length than 2^{30}—not uncommon for Google applications—the bug appears. Bloch [6] offers some fixes and an implicit complaint: *how come that the bug persisted so long when he, as a PhD student, was taught a correctness proof [5] of the binary search algorithm?* Finally, he asks the crucial question: "and now we *know* the binary search is bug-free, right?"

4.3 Bugs everywhere and Hoare's question

Computer bugs are, literally, everywhere and they may affect many users. Most important software companies maintain bug databases: `bugs.sun.com/bugdatabase/index.jsp`, `bugs.kde.org`, `MySQLBugs`, `bugzilla.mozilla.org`, `bugs.apache.org`, etc. Here is a model of how to report bugs at Sun:

> If we don't know about your problem, we can't fix it. If you've isolated a problem that you think we're causing, and you can't find it here, submit a bug! Make sure you include all the relevant information including all the details needed to reproduce the problem. Submissions will be verified and prioritized. (Please note that *bug fixes are not guaranteed.*)[7]

Bugs can be of different types, hence producing varying levels of inconvenience to the user of the program. The costs of some bugs may be almost incalculable. A bug in the code controlling the Therac-25 radiation therapy machine led to at least six deaths between 1985 and 1987 [60]. The European Space Agency's $1 billion prototype Ariane 5's first test flight on June 4, 1996, failed, with the rocket self-destructing 37 seconds after launch [3]; the reason was a software bug, arguably one of the most expensive bugs in history. More recently, a security flaw in PayPal was exploited by fraudsters to steal credit card numbers and other personal information belonging to PayPal users (June

[7] Our Italics.

16, 2006); and the Y2K7 bug (January 3, 2007) affected Microsoft's preview version of Expression Design. Finally, a bug discovered by M. Schwartz [50] found no interest in the community, so he had to write a small script showing how to use it to get all the email addresses from members subscribing to a Google group; Google fixed the problem on January 5, 2007. Improperly coded software seems to have caused the Mars Global Surveyor failure in November 2006; in January 2007, NASA launched an investigation [62].

The list can easily be continued. *Wired* magazine maintains a history of the worst software bugs [28].

In spite of all the examples discussed above, bugs and faulty software have killed remarkably few people. They caused more embarrassment, nuisance, inconvenience, but many fewer catastrophes. Early in January 2007, a 6.7 earthquake in Taiwan produced serious interruptions in the internet in Asia [58]; this showed that the internet is far from shockproof, but consequences were, again, not catastrophic. Finally, one more example: Boeing 777, one of the most automatic fly-by-wire air-planes, has flown since 1995 without any crashes or serious problems. So, we can ask with Hoare [34] the question: *how did software get so reliable without proof?*

5 Proving vs. programming: tomorrow

5.1 Theorems and programs

The practice of programming, by and large, produces "discoursive knowledge", a knowledge resulting from computing. "Deductive knowledge", complementary to discoursive knowledge, can be obtained by the mathematical analysis of the program (in some given context). These notions of knowledge correspond to Dijkstra's approaches (see [24]) to programs: *postulational* and *operational*. Under the postulational approach, the program text is considered a mathematical object. The semantic equivalence of two programs means that they meet the same specification. According to the operational approach, reasoning about programs means building a computational model with respect to which the program text is interpreted as executable code.

According to Dijkstra:

> The tragedy of today's world of programming is that, to the extent that it reasons about programs at all, it does so almost exclusively operationally.

The argument? "With growing size or sophistication of the program, the operational

argument quickly becomes impossible to carry through, and the general adherence to operational reasoning has to be considered one of the main causes of the persistence of the software crisis". Dijkstra believes that, ultimately, real programmers don't reason about their programs, "they rather get their substitute for intellectual satisfaction from not quite understanding what they are doing in their daring irresponsibility and from the subsequent excitement of choosing the bugs they should not have introduced in the first place".

The last quotation reminds us what Bertrand Russell said about mathematics and what Martin Heidegger wrote about science: "Wissenschaft denkt nicht" (Science does not think). For Dijkstra, the analoguous situation in mathematics is the distinction between formal and informal; perhaps, he had in mind situations such as the distinction between axiomatic and naive set theory.

It makes sense to prove the correctness of an algorithm, but *not* the correctness of a program as various authors have argued [21, 26]. Programs are analogues of mathematical models; they may be more or less adequate to code algorithms. Adequacy is a property which depends on many factors, from pure formal/coding ones to physical and engineering ones. One can even argue that a "correctness proof" for a program, if one could imagine such a thing, adds very little to one's confidence in the program. In Knuth's [36] words:

> Beware of bugs in the above code: I have only proved it correct, not tried it.

The computer science analogy of the operational–postulational distinction corresponds to the difference—considered already at the beginning of the 19th century—between mathematics understood as calculation and mathematics as qualitative conceptual reasoning. In the analogy between proving and programming, *theorems correspond to algorithms not programs; programs correspond to mathematical models.*

5.2 Mathematics = proof?

The role of proof in mathematical modelling is very small: *adequacy is the main issue!* As mathematical modelling is closer to coding algorithms into programs, selecting algorithms to code, designing specifications to implement, one can re-phrase the arguments against the idea of proof of correctness of programs [21, 26] as arguments against the idea of proof of correctness of mathematical models. Models evolve and become more and more adequate to the reality they model: however, they are never *true*. Here is an illuminating description by Schwartz [49]:

> . . . it may come as a shock to the mathematician to learn that the Schrödinger

> equation for the hydrogen atom ... is not a literally correct description of this atom, but only an approximation to a somewhat more correct equation taking account of spin, magnetic dipole, and relativistic effects; that this corrected equation is itself only an ill-understood approximation to an infinite set of quantum field-theoretical equations; and finally that the quantum filed theory besides diverging, neglects a myriad of strange-particle interactions whose strength and form are largely unknown. ... The physicist, looking at the original Schrödinger equation, learns to sense it ... and this sense inspires ...

For example, engineers use theorems by "plugging in" values and relying on some (physical) interpretations of the conclusion. This is what makes planes fly and bridges stand.

The modelling component of mathematics appears not only in applications, but also in the way mathematics develops new concepts. Many important notions in mathematics reached an accepted definition only after a long process of modelling, from an intuitive, pre-mathematical notion to a more precisely defined, formal one. In the end, the accepted definition is adopted as a thesis claiming its adequacy [51]. For example, "Weierstrass's thesis" is the statement that the intuitive notion of continuity is extensionally equivalent to the notions yielded by the now standard definitions of "continuous function". Other examples include: the "function thesis" (identification of a function with a set of ordered pairs), "Tarski's thesis" (identification of Tarski's definition of truth in a formalised language with the intuitive notion of truth), "the Church-Turing thesis", etc. None of these theses can be *proved*, but various analyses can conclude their degrees of plausibility/adequacy/applicability. Mathematics in both its practice and development is an "open-texture" [51].

5.3 Checking proofs

There are many new types of proofs: probabilistic, experimental or hybrid proofs [7, 8] (computation plus theoretical arguments). Reflecting somehow Jean-François Lyotard's "No truth without money", Zeilberger [56] has argued in favour of the transition from rigorous proofs to an "age of *semi-rigorous* mathematics, in which identities (and perhaps other kinds of theorems) will carry price tags" measured in the computer and human resources necessary to prove them with a certain degree of confidence.

Zeilberger [57] sees the evolution of mathematics as follows:

> The real work of us mathematicians, from now until, roughly, fifty years from now, when computers won't need us anymore, is to make the transition

> from human-centric math to machine-centric math as smooth and efficient as possible.

Let's do the following mental experiment: apply literally to mathematical practice Hilbert's requirement for proof stated in Section 2 (in logical terms, *the proofs of a theory form a computable set*). Then Anderson's question, posed at the end of Subsection 3.2, is not only not surprising, but should be answered in an affirmative way. This could be a reasonable motivation for the project Flyspeck.

Probabilistically checkable proofs are mathematical arguments that can be checked probabilistically by reading just a few of their bits. In the early 1990's it was proved that every proof can be effectively transformed into a probabilistically checkable proof with only a modest increase in the original proof length. However, the transformation itself was complex. Recently, a very simple procedure was discovered by Dinur; see the presentation [48].

Now, feeling a loss of certitude, we should remember that Thales was the first to stimulate his disciples to criticise his assertions. This tradition was later lost, but recovered with Galilei. With Thales and Galilei we learned that human knowledge is essentially conjectural (see also [45]). Should mathematics and computer science accept being guided by this slogan, or is it adequate only for the natural and social sciences?

5.4 Communication and understanding

Of course, no theorem is validated before it is communicated to the mathematical community (orally and, eventually, in writing). Manin states it clearly:

> Proof is not just an argument convincing an imaginary opponent. Not at all. Proof is the way we communicate mathematical truth.

However, as Rota [46] pointed out:

> One must guard, however, against confusing the *presentation* of mathematics with the *content* of mathematics.

Proofs have to be written on paper, which means proofs are *physical*. From this perspective, proofs depend upon the physical universe (see more in [13]). We already have to cope with existing, extremely long proofs. What about proofs that are too long to be written down? They may exist in principle, but they are impossible to read.[8]

[8] An exponentially long quantum proof cannot be written down, since that would require an exponential amount of "classical" paper, but a quantum mind could directly perceive the proof, [13].

In order to be able to amplify human intelligence and prove more complicated theorems than we can now imagine, we may be forced to accept incomprehensible or only partially comprehensible proofs. We may be forced to accept the help of machines for mental, as well as physical, tasks.

If mathematics depends on physics and mathematics is the main tool to understand physics, don't we have some kind of circularity? One explanation is that, in Lakatos's words, "mathematics is quasi-empirical", as has been extensively discussed by Chaitin (see [16, 17]).

Does physical interference combined with the use of computers destroy the understanding of mathematical facts? One could know that a theorem is true, but not really understand it! For Chaitin [18] (p. xiii) this is not the case:

> To me, you understand something only if you can program it. (You, not someone else!)

Why? Because [19] (p. 30):

> ... programming something forces you to understand it better, it forces you to really understand it, since you are explaining it **to a machine**.

5.5 Rigour: operational vs. conceptual

Standards of rigour have changed throughout the history of mathematics, and not necessarily from less rigour to more rigour. For Bertrand Russell, certainty has to match in power religious faith: "I wanted certainty in the kind of way in which people want religious faith".

Serre was quoted [47] saying that mathematics is the only producer of "totally reliable and verifiable" truths. This opinion seems to be contradicted by both Knuth [36]:

> ... programming demands a significantly higher standard of accuracy. Things don't simply have to make sense to another human being, they must make sense to a computer.

and Thurston [54]:

> The standard of correctness and completeness necessary to get a program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.
>
> When one considers how hard it is to write a computer program even approaching the intellectual scope of a good mathematical paper, and how much greater time and effort have to be put into it to make it "almost" formally correct, it is preposterous to claim that mathematics as we practice it is anywhere near formally correct.

But we can go further into the past. Old Greek mathematics, with Pythagoras, Plato and Euclid, was essentially conceptual and this is the reason why they were able to invent what we call today a mathematical proof. Babylonian mathematics was exclusively operational. The move from an operational to a conceptual attitude in computer programming is similar to the evolution from Babylonian to Greek mathematics.

Coming to more recent periods in the history of mathematics, we observe the strong operational aspect of calculus in the 18th century, in contrast with the move to the predominantly conceptual aspect of mathematical analysis in the 19th century. Euler is a king of operational mathematics; Riemann and Weierstrass express *per excellence* the conceptual attitude. The transition from the former to the latter is represented by giants such as Abel and Cauchy. When Cauchy believed that he had proved the continuity of the limit of a convergent sequence of continuous functions, Abel, with no ironical intention, wrote: "Il semble que le théorème de Monsieur Cauchy admet des exceptions."[9] But, at that moment, neither of them was able to invent the notion of uniform convergence and, as a matter of fact, neither convergence nor continuity was effectively clarified. Only the second half of the 19th century brought their full understanding, together with the idea of uniformity, either with respect to convergence or with respect to continuity. We see here all characteristic features of a transition period, the transition from the operational to the conceptual attitude.

To stress the two facets of Cauchy's mathematics, one belonging to the intuitive-operational, the other to the rigourous-conceptual attitude, let us recall that, despite the fact that Cauchy is undoubtedly the founder of the exact differential calculus in the modern sense, he is also the mathematician who was convinced that the continuity of a function implies its differentiability and hence that any continuous function can be geometrically represented. We had to wait for Weierstrass and Riemann to understand the gap existing between some mathematical concepts and their intuitive representation.

However, this evolution does not concern only calculus and analysis. It can be observed in all fields of mathematics, although the periods in which the transition took place may

[9] It seems that the theorem of Mr Cauchy admits exceptions.

be different for various fields. For instance, in algebraic geometry it took place only in the 20th century, with the work of Oscar Zarisky. In fact, any conceptual period reaches its maturity under the form of an operational one, which, in its turn, is looking for a new level of conceptual attitude. The whole treatise of Bourbaki is a conceptual reaction to an operational approach. Dirichlet's slogan asking to replace calculations with ideas should be supplemented with another, complementary slogan, requiring us to detect an algorithmic level of concepts.

Can we expect a similar alternation of attitudes in respect to programming? Perhaps it is too early to answer, taking into account that the whole field is still too young. The question is not only academic, as the project Flyspeck reminds us.

5.6 Is it meaningful to speak about the truth of axioms?

In Section 2 we argued that mathematical proofs do not produce certain knowledge; they produce *rational belief*. The epistemological value of a proof reside in the degree of belief of its axioms. What is then the value of proof? Is it meaningful to speak about the truth of axioms?

First, a few more words should be said about axioms and primitive terms. Euclid avoids reference to primitive terms, but they exist in his *Elements*, hidden by pseudo-definitions such as "We call point what has no parts". Only modern axiomatic systems make explicit reference to primitive terms. Obviously, programs too could not be conceived in the absence of some primitive terms. A similar remark is in order about axioms. To what extent is it meaningful to raise the question about the truth of some axioms? Semantics is a matter of interpretation of a formal system, which, in its turn, has some primitive terms and some axioms among its bricks. Circularity is obvious. Gödel's (true) statements that cannot be proved could not be conceived in the absence of the respective formal system, which in its turn has among its bricks some primitive terms and some axioms. Maybe we can refer to another way to understand meaning, a way avoiding Hilbert's itinerary? For instance, the way it is understood in C. S. Peirce's semiotics or in modern linguistics. But do they have the rigour we expect from a mathematical approach?

The whole idea of a formal proof is strictly dependent of the idea of a formal system. Is it meaningful to raise the question whether the Zermelo–Fraenkel axioms for set theory are or not true? We adopt a convention and it is proved by Gödel that if these axioms are consistent, then they remain consistent by adding the choice axiom or the continuum hypothesis.

It is interesting to observe that some authors have supplemented the Zermelo–Fraenkel

axioms with the foundation axiom (aiming to interdict Russell's sets which are elements of themselves), while more recently Aczel and Barwise [1] have replaced the foundation axiom with the anti-foundation axiom, where hypersets are allowed. An object A is a hyperset if there exists an infinite sequence $A(n)$ such that $A(1) = A$ and for each n, $A(n+1)$ is an element belonging to $A(n)$. In the particular case when $A(n) = A$ for each n, we get the Russell sets. So, an axiom was replaced by its negation and the resulting theory proved to be very interesting. It has applications in mathematics (non-standard analysis), computer science (data bases, logical modelling of non-terminating computational processes), linguistics and natural language semantics (situation theory), and philosophy (the Liar Paradox). The well-known scenario of non-Euclidean geometries proves to be once more valid.

6 Acknowledgements

We thank Douglas Bridges, Andreea Calude, Greg Chaitin, and Nick Hay for many discussions and suggestions that improved our paper.

References

[1] P.Aczel, J. Barwise. Non-well-founded sets, *Journal of Symbolic Logic* 54, 3 (1989), 1111–1112.

[2] R. F. Arenstorf. There Are Infinitely Many Prime Twins, `http://arxiv.org/abs/math.NT/0405509` 26 May 2004.

[3] Ariane Flight 501, `http://spaceflightnow.com/cluster2/000714featur e/ariane501_qt.html`.

[4] Beckham agrees to LA Galaxy move, `http://news.bbc.co.uk/sport2/hi/football/6248835.stm`, January 11, 2007.

[5] J. Bentley. *Programming Pearls*, Addison-Wesley, New York, 1986; 2nd ed. 2000 (Chapter 5).

[6] J. Bloch. Nearly All Binary Searches and Mergesorts are Broken, *Google Research Blog*, February 2, 2006, `http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html`.

[7] J. M. Borwein, D. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*, A.K. Peters, Natick, MA, 2003.

[8] J. M. Borwein, D. Bailey, R. Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*, A.K. Peters, Natick, MA, 2004.

[9] A. Bundy, M. Jamnik, A. Fugard. What is a proof, *Phil. Trans. R. Soc.* A 363 (2005), 2377–2391.

[10] A. S. Calude. The journey of the four colour theorem through time, *The NZ Math. Magazine* 38, 3 (2001), 27–35.

[11] C. S. Calude. *Theories of Computational Complexity*, North-Holland, Amsterdam, 1988.

[12] C. S. Calude, Elena Calude, S. Marcus. Passages of proof, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 84 (2004), 167–188.

[13] C. S. Calude, G. J. Chaitin. A dialogue on mathematics & physics, *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 90 (2006), 31–39.

[14] C. S. Calude, S. Marcus. Mathematical proofs at a crossroad? in J. Karhumäki, H. Maurer, G. Păun, G. Rozenberg (eds.). *Theory Is Forever*, Lectures Notes in Comput. Sci. 3113, Springer-Verlag, Berlin, 2004, 15–28.

[15] B. Cipra. Gaps in a sphere packing proof? *Science* 259 (1993), 895.

[16] G. J. Chaitin. Two philosophical applications of algorithmic information theory, in C. S. Calude, M. J. Dinneen, V. Vajnovszki (eds.). *Proceedings DMTCS'03*, Springer Lecture Notes in Computer Science, vol. 2731, 2003, 1–10.

[17] G. J. Chaitin. *From Philosophy to Program Size*, Tallinn Institute of Cybernetics, 2003.

[18] G. J. Chaitin. *Meta Math!*, Pantheon, 2005.

[19] G. J. Chaitin. Epistemology as information theory: from Leibniz to Omega, *Collapse* 1 (2006), 27–51.

[20] Clay Mathematics Institute: Millennium Prize Problems, 2000, `http://www.claymath.org/millennium/Poincare_Conjecture/`.

[21] R. A. De Millo, R. J. Lipton, A. J. Perlis. Social processes and proofs of theorems and programs, *Comm. ACM* 22, 5 (1979), 271–280.

[22] K. Devlin. *The Millennium Problems*, Basic Books, Cambridge, MA, 2002.

[23] K. Devlin. When is a proof, *Devlin's Angle*, `http://www.maa.org/devlin/devlin_06_03.html`.

[24] Edsger W. Dijkstra. Real mathematicians don't prove, *EWD1012*, 24 January 1988, `http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1012.html`.

[25] L. F. Fass. Approximations, anomalies and "Ther Proof of Correctness Wars", *ACM SIGSOFT* 29, 2 (2004), 1–4.

[26] J. H. Fetzer. Program verification: the very idea, *Comm. ACM* 31 (1988), 1048–1063.

[27] Flyspeck, http://www.math.pitt.edu/~thales/flyspeck/ (consulted 28 March 2007)

[28] S. Garfinkel. History's worst software bugs, *Wired*, August 11, 2005, http://www.wired.com/news/technology/bugs/0,2924,69355,00.html?tw=wn_tophead_1.

[29] D. Goldston, J. Pintz, C. Yildirim. Primes in tuples, http://arxiv.org/math.NT/0508185.

[30] R. K. Guy. *Unsolved Problems in Number Theory*, Springer-Verlag, New York, 1994, 2nd ed. (pp. 19–23).

[31] T. C. Hales. Sphere packings. I, *Disc. Comput. Geom.* 17 (1997), 10–51.

[32] T. C. Hales. A proof of the Kepler conjecture, *Ann. Math.* 162 (2005), 1065–1185.

[33] D. Hilbert. Mathematical problems (Lecture delivered before the International Congress of Mathematicians at Paris in 1900), in F. E. Browder (ed.). *Proceedings of Symposia in Pure Mathematics of the AMS*, AMS, 28, 1976, p. 25.

[34] C. A. R. Hoare. How did software get so reliable without proof?, in *Proceedings of the Third International Symposium of Formal Methods Europe on Industrial Benefit and Advances in Formal Methods*, Lect. Notes Comput. Sci., Vol. 1051, Springer-Verlag London, 1996, 1 –17.

[35] D. Kennedy. Breakthrough of the year, December 22, 2006, http://www.sciencemag.org/cgi/reprint/314/5807/1841.pdf.

[36] D. E. Knuth. Theory and practice, *EATCS Bull.* 27 (1985), 14–21.

[37] D. Mackenzie. *Mechanizing Proof*, MIT Press, Cambridge, Mass. 2001.

[38] T. R. Nicely. Enumeration to 10^{14} of the twin primes and Brun's constant, *Virginia Journal of Science* 46, 3 (Fall, 1995), 195–204.

[39] G. Odifreddi. *The Mathematical Century*, Princeton University Press, Princeton, 2004.

[40] G. Perelman. Ricci Flow and Geometrization of Three-Manifolds, Massachusetts Institute of Technology, Department of Mathematics Simons Lecture Series, September 23, 2004 http://www-math.mit.edu/conferences/simons.

[41] G. Perelman. The Entropy Formula for the Ricci Flow and Its Geometric Application, November 11, 2002, http://www.arxiv.org/abs/math.DG/0211159.

[42] G. Perelman. Ricci Flow with Surgery on Three-Manifolds, March 10, 2003, http://www.arxiv.org/abs/math.DG/0303109.

[43] H. Poincaré. *Oeuvres de Henri Poincaré*, tome VI. Gauthier-Villars, Paris, 1953, pp. 486 and 498.

[44] K. R. Popper. Indeterminism in quantum physics and in classical physics. Part II, *The British Journal for the Philosophy of Science* 1, 3 (1950), 173–195.

[45] K. R. Popper. Part 1 in David Miller (ed.). *Popper Selections*, Princeton University Press, 1985.

[46] G.-C. Rota. *Indiscrete Thoughts*, Birkhäuser, Boston, 1997, p. 96.

[47] J.-P. Serre. Interview, *Liberation* May 23 (2003).

[48] J. Radhkrishnan, M. Sudan. On Dinur's proof of the PCP theorem, *Bull. AMS* 44, 1 (2007), 19–61.

[49] J. Schwartz. The pernicious influence of mathematics on science, in M. Kac, G.-C. Rota, J. Schwartz (eds.). *Discrete Thoughts*, Birkhäuser, Boston, 2nd ed.,1992, 19–26.f

[50] M. Schwartz. Google Groups XSS Bug [Part 2], January 3, 2007, `http://weblogs.asp.net/mschwarz/`.

[51] S. Shapiro. Computability, proof, and open-texture, in Adam Olszewski, Jan Woleński, Robert Janusz (eds.). *Churchs Thesis After 70 Years*, Ontos Verlag, Berlin, 2006, 420–455.

[52] K. Soundararajan. Small gaps between prime numbers: the work of Goldston-Pintz-Yildirim, *Bull. AMS* 44, 1 (2007), 1–18.

[53] R. Thomas. The Four Color Theorem, `http://www.math.gatech.edu/~thomas/FC/fourcolor.html`.

[54] W. P. Thurston. On proof and progress in mathematics, *Bull. AMS* 30, 2 (1994), 161–177.

[55] Eric W. Weisstein. Kepler Conjecture. From *MathWorld–A Wolfram Web Resource*, `http://mathworld.wolfram.com/KeplerConjecture.html`.

[56] D. Zeilberger. Theorems for a price: tomorrow's semi-rigorous mathematical culture, *Notices AMS* 40(1993), 978–981.

[57] D. Zeilberger. Don't Ask: What Can The Computer do for ME?, But Rather: What CAN I do for the COMPUTER?, March 5, 1999, `http://www.math.rutgers.edu/~zeilberg/Opinion36.html`.

[58] Asia's Net: Hanging by a Thread, January 4, 2007, `http://www.time.com/time/magazine/article/0,9171,501070115-1573933,00.html`.

[59] PayPal Security Flaw Allows Identity Theft, `http://news.netcraft.com/archives/2006/06/16/ paypal_security_flaw_allows_identity_theft.html`.

[60] Therac25: Computerized Radiation Therapy, `http://neptune.netcomp.monash.edu.au/ cpe9001/assets/readings/www_uguelph_ca_~tgallagh_~tgallagh.html`.

[61] New Year Bug Cripples Microsoft Expression Preview, January 3, 2007, `http://www.ddj.com/dept/debug/196800800`.

[62] Bad Software May Have Doomed Mars orbiter, http://www.msnbc.msn.com/id/16580498.