Marian Mrozek†
School of Mathematics,
Georgia Institute of Technology,
Atlanta, GA 30332-0160

Jacek Urbaniec
Computer Science Institute
Jagiellonian University
Kraków, Poland

Evolution of Mathematical Proof

1. Computer Calculations in Precise Arithmetic

One of the most interesting essays in the book *Mathematics Today: Twelve Informal Essays*[1] , which appeared in the United States in 1978, was the paper on the four-colour theorem by K. Appel and W. Haken. The authors presented in it the main ideas of their, much renowned by now, mathematical proof involving application of a computer.

Appel and Haken overcame mathematical intricacies of the four-color theorem by reducing it to simple reviewing a large but finite number of cases (naturally, performing this task effectively did involve an enormous amount of computer power; the investigators had to apply for allocation of as much as 1000 hours at a mainframe computer). Combinatorial computations in the graph theory performed by Appen, Haken and Koch require only precise arithmetic. In other words, all the mathematical steps needed in the computational part of the proof of the four-color theorem were executed in the exact, rather than approximate, manner, *i.e.* without round-off and termination errors.

Let's see it on a simple example. Within a fixed (finite) set of elements one can define, say, a two-argument operation in a number of ways. Only in few cases such an operation will have the specific property of being a group operation. Checking if the operation is associative (*i.e.* if for any a, b, c, $(a*b)*c = a*(b*c)$) can be cumbersome, but even a freshman in computer science can readily write a computer program to do it. The mathematical operations performed by the computer in this case will be equally strict and exact as done by a human mathematician with a ball-pen on a sheet of paper using his mental capabilities.[2] Another example of exact computer calculations can be addition and multiplication of natural numbers (provided the numbers involved are not excessively large).

2. Round-off Errors

† Supported by KBN, Grant No. 2 P03A 029 12

[1] L.A. Steen, *Mathematics Today: Twelve Informal Essays*, Springer Verlag, New York, 1978

[2] Of course, we do not provide here for human errors.

In many cases, however, computer calculations are of approximate character and involve intrinsic (usually tiny) errors. If you have an electronic calculator, you can readily see that if you divide 2 by 3, and multiply the result again by 3, you are not to obtain 2, but slightly less (say, 1.99999999**8**, as in our calculator). Since the approximated results provide input for subsequent calculations, the eventual error may further increase (in multiplying by 1000000, for instance) or decrease (when multiplied by 0.001).

What is the origin of errors in computer calculations? Computer memory can store a very large, but still finite, amount of information. In particular, when dealing with real numbers, only a certain subset of the set of all real numbers, called the set of representable numbers, may be represented. In theory there is no restriction what that subset can be if only it is sufficiently small (or the computer memory is sufficiently large). Practically, especially when the speed of computations is the primary concern, the set of representable numbers reduces to a certain set of binary fractions with a fixed number of binary digits (binary digits are convenient for computers but in what follows there is essentially no difference between binary and decimal digits). Whatever the set of representable numbers is, if a given real number is not representable, it is replaced by the nearest representable number. The process is known as rounding. For example, if the set of representable numbers consists of binary fractions with n digits, then the irrational number $\sqrt{2}$ as well as the non-binary fraction $1/3$ are not representable and consequently are rounded to n binary digits. This gives rise to input data errors. However, even if we were lucky to have all our input numbers representable, performing arithmetic operations on numbers containing n digits usually leads to numbers which require more than n digits. For instance, the product of two n-digit numbers (with $2n$ or $2n-1$ digits) before further processing has to be rounded off to n digits. This again leads to round-off errors. In textbooks on numerical methods one can find analysis of that kind of errors for simplest cases; for instance, the calculation of scalar product. Scientific calculations usually base on much more sophisticated algorithms, which makes a full analysis of round-off errors by a human virtually impossible, so as a rule they are ignored, all the more that one has to provide for another type of errors, truncation errors, which arise when a calculation procedure leading to some limiting value is terminated before reaching it.This is the case when *e.g.* we take into account only a finite number of terms in an infinite series, or approximate a non-linear function by a linear one. Truncation errors are present, among others, in all kinds of iteration methods.

3. Continuous Mathematics and Computers

Imagine that in order to prove some theorem a mathematician needs to check whether some function has a zero within an interval $[a, b]$ (*i.e.* whether there exists a point x within that interval for which $f(x) = 0$). This kind of problems is typical for continuous mathematics, since it involves a mathematical continuum: interval [a, b]. It may seem that computers are of no use in solving problems involving infinite (uncountable) sets, but that is not true. Let us assume that the given function is continuous (*e.g.* it is a polynomial). Such an assumption is usually easy to verify. Then the typical tool is the Darboux's theorem, stating that any continuous function takes all intermediate values, *i.e.* if $f(a) \leq f(b)$, then the range of the function contains the interval $[f(a), f(b)]$. Thus it is sufficient to determine if the product of $f(a)$ and $f(b)$ is negative, since if this is true, the function f is of opposite sign on the interval's ends, and consequently it has a zero within that interval. Thus a problem involving infinite sets has been reduced to a finite one: determining the product of two numbers $f(a)$ and $f(b)$. Is it the end of troubles for our mathematician? Not necessarily. It may happen that the values $f(a)$ and $f(b)$ could not be computed analytically as it is the case when the function f is given through a "malicious" infinite set. Then our theoretician can turn to an applied mathematician, who using his computer along with a couple of known algorithms can readily calculate $f(a)$ and $f(b)$, and their product is, say, -1.2.

Can we claim in such a case that the mathematician has proved his theorem, and his proof is a typical example of a computer-assisted proof? Certainly not - any careful reader will object - the calculation output is just approximate. Apart from estimating termination errors, one has to determine round-off errors, else one could not be sure if the eventual result of the numerical calculations does not involve an error greater than, say, 1.5 (then the product $f(a)f(b)$ could be positive and the Darboux's theorem would be no longer applicable). Let us notice that if in a computer-assisted proof all the steps are exact (performed in precise arithmetic), as in the proof by Appel and Haken, then the above difficulties do not arise.

Does it mean that there is no way to complete the proof by means of a computer? Determining rigorously the sign of the product seems to be easier then finding rigorously the zero of the function. All what we need is to be able to prove that the total error is less then 1.2. As we already mentioned, the error analysis by humans for all but very simple functions is just impossible. Fortunately a rigirous error analysis may be performed by the computer itself. The method, known as the interval arithmeitic, was developed already in '60s. If a number does not have a representation in a computer, instead of rounding, one can take an interval consisting of the lower and upper representation of that numebr and go on with all mathematical operations (as addition, multiplication ...) on intervals. This may be set up in such a way that the resulting interval always contains the exact result. Thus we do not know the exact result but we do have a rigorous lower and upper estimate of the exact result. If it happens that the interval containing $f(a)f(b)$ has its upper limit less than zero, then, whatever the exact value of $f(a)f(b)$ is, it is definitely less than zero! Therefore the proof of the existnce of a null of the given function via the Darboux's theorem is completed. If the outcome of the computations is an interval with its lower limit greater than zero, then it is clear that the Darboux's theorem cannot be applied, though obviously the function still may have a null in the given interval. Finally it is also possible to obtain an interval with its lower end less than zero and the upper end greater than zero. In such a situation we do not know anything and the only resort is to try to repeat the calculations with a better computer.

The above example should make it clear that computer assisted proofs in the mathematics of continua are possible. The only question is how many interesting problems may be reduced to the existence of a null of a real function or alternatively if the just presented scheme does have some multidimensional counterparts.

4. Computer assisted proofs in the mathematics of continua.

The argument in the previous section shows clearly that the attempts to apply approximate numerical methods to rigorous mathematical proofs are not totally hopeless. It turns out that not only it is possible to perform such proofs but several proofs of this kind are already available. To our best knowledge, the first proof in continuous mathematics and employing a computer is by O.E. Lanford[3] and concerns the Feigenbaum conjectures. The author reduces the problem to showing that a certain number is less than one and than uses interval arithmetic to prove this fact. The following years brought several more computer assisted proofs in analysis by various authors (see [4] for references). Recently K. Mischaikow and the first co-author of this paper 5 managed to solve by means of a computer assisted proof a long standing problem in the theory of differential equations: the genuineness of chaos in the Lorenz equations. These ordinary differential equations in three variables approximate some partial differential equations describing atmospheric convection. Approximate trajectories of the dynamical system generated by the Lorenz equations can be easily observed in a graphic form. The striking effect is that independently of the

[3] *Cf.* O.E. Lanford, A computer-assisted proof of the Feigenbaum conjectures, *Bull. AMS (N.S.)* **6**, 1982, pp. 427-434.

[4] *Cf.* M. Mrozek, Inheritable properties and computer assisted proofs in dynamics, in: Scientific Computing and validated numerics, G. Alefeld, A. Frommer and B. Lang editors, Akademie Verlag, Berlin, pp. 245-253.

5 *Cf.* K. Mischaikow, M. Mrozek, *Chaos in the Lorenz equation: a computer-assisted proof*, BAMS, 33 (1955), pp. 66-72 and *Chaos in the Lorenz equation: a computer-assisted proof. Part II: Details*, Mathematics of Computation, in print.

initial conditions chosen, almost immediately trajectories show a very strange behaviour: they rotate around two fixed points in a very irregular way. Marking consecutive rotations around the two fixed points respectively by 0 and 1, one obtains a sequence of 0's and 1's which seems to be completely random. Moreover, even a minimal change in the initial conditions leads to a completely different sequence. One is tempted to say that every sequence is possible. This is called chaos.

But the chaotic picture on the screen does not implicate that chaos is present in the Lorenz equations! It does not even implicate chaos in the the numerical scheme! First, plotting the trajectories by the computer did involve numerical methods and approximate operations. Second, we actually know that in case of the numerical approximations chaos is not possible: since the memory of the computer is finite, the number of all possible configurations of this memory is also finite and consequently the number of different sequences of 0's and 1's which may be obtained from the numerical scheme is possibly large but finite. This is not chaos! The large number of sequences may indicate that chaos is present in the Lorenz equations but definitely it does not prove this fact. In precisely the same way finding a large number of prime numbers is not sufficient to claim that the existence of an infinite number of primes has been proved.

Despite the attempts of many mathematicians and the abundance of papers dealing with the Lorenz equations for more than 30 years no mathematical proof of chaos in the Lorenz equations was known and the computer assisted proof was the first available.[6]. The proof is more sophisticeted than just applying the Darboux's theorem. Instead of this theorem it utilizes a criterion for chaos in terms of a topological invariant known as the Conley index. The computation of the Conley index for the Lorenz system was performed in several steps. First, the equations were integrated numerically but in the interval arithmetic for a set of approximately 700000 initial points. This led to a mulitvalued mapping called an enclosure (to each of the 700000 initial points a set of possible values resulting from the numerical integration in the interval arithmetic was assigned). Such a mapping is combinatorial in nature. In the next step the Conely index theory was carried over to such combinatorial multivalued maps and shown to be inheritable. This means that whenever the Conley index makes sense for a multivalued map then it makes sense for any single valued map enclosed by the mulitvalued map and both Conley indices are the same. Since the multivalued map is combinatorial, the computation of the Conley index for such a map is also purely combinatorial. Thus in the last step it was sufficient to find the Conley index

[6] Recently, the theorem on the presence of chaos in the Lorenz equations has been proved also in a conventional way but unlike the computer assisted proof that argument does not indicate explicitely for which parameter values the chaotic behaviour is proved.

for the multivalued map obtained from the Lorenz equations and check if such an index implies chaos.

If we look for any analogy with the previous example (finding nulls of a continuous function), the Conley index corresponds to the product $f(a)f(b)$ and its computation corresponds to finding an interval enclosing the product. However here the analogy stops, since the Conley index is a more sophisticated tool than just a number.

Summarizing, there are examples that numerical calculations can constitute an essential part of mathematical proof not only in the discrete mathematics but also in the mathematics of continua. Moreover, it seems inevitable that such proofs will be more and more common. On one hand todays proofs become more and more complex. A good example is the recent proof of the famous Fermat's theorem by A. Wiles, corrected by R. Taylor. On the other hand there is a little known theorem by Goedel which states that there are short theorems with arbitrarily long shortests proofs. This may mean that the todays mathematics has become sufficiently sophisticated to have to deal with that kind of theorems. If this is the case, computer assisted proofs may even become not only a convenient possibility but just a necessity: not a mathematical heresy, dwelling somewhere off mainstream mathematics, but a valid procedure to be used in proving theorems, commonly accepted by mathematical community.[7]

5. Computer-assisted Proofs vs. Automatic Theorem Proving

In recent years computer-assisted proofs got much publicity, but in most discussions, particularly involving non-mathematicians, the only example considered is the proof of the four-colour theorem provided by Appel and Haken. Various methodological aspects are considered, without making any difference at all between proofs using computer calculations in precise arithmetic and proofs involving approximations that give rise to round-off errors. Things get even worse when some authors can see no difference between automatic theorem proving and computer-assisted proofs.

Let us remind that automatic theorem proving consists in finding a mechanical procedure, an algorithm enabling one to construct a proof for any true statement within a given theory. Such a goal is much more ambitious, so it should not be surprising that it is rarely achieved. Apart from the classic predicate calculus, automatic proofs are successful within the calculus of first-order predicates[8].

[7] We rejoiced at finding that the *Book of 1996* of *Encyclopaedia Britannica* lists the computer assisted proof of chaos in the Lorenz equations among the four most significant achievements in mathematics in 1995, alongside the proof of the Fermat's theorem.

[8] *Cf.* the *resolution rule* of J.A. Robinson in: Ch.-L. Chang, R. Lee, *Symbolic Logic and Me-*

It is worth mentioning one more application of computers in mathematics. Before formulating and proving any theorem a mathematician needs to review a sufficient number of instances, usually involving calculations. Biographies of C.F. Gauss, one of the greatest mathematicians in history, stress that aspect of his investigations. He meticulously filled subsequent notebooks with examples that underlay his later abstract theorems. Development of computers enabled a numerical mathematician to readily perform a large number of calculations which otherwise would not be possible. Even if the results are only approximate, they can give rise to relevant insights, in the same way as Gauss's notebooks. Using a computer this way can be described as heuristic, while in computer-assisted proofs, computers provide actual justification.

6. Methodological Implications

It is sometimes held that computer-assisted proofs cannot change much in our understanding of the notion of mathematical proof, since in proofs like that of chaos in the Lorenz equations, the computer performs just schematic, though cumbersome, operations and sufficiently large number of humans, given sufficiently long time, could achieve exactly the same result. So the difference would be that of appearance – a classical mathematical proof can be expressed on several, at most, a couple of dozens, pages, while putting down a computer assisted proof showing explicitly all the calculation steps would require a huge number of volumes.

Someone may maintain that in long, cumbersome calculations computers are much more reliable than humans – when it comes to performing, say, a hundred-page calculation, a computer is presumably much less likely to commit an error than a human mind.

Actually things are not that simple. Practically every computer-assisted proof requires using a programming language, for instance FORTRAN or C++. This means that the computations needed in the proof are not performed directly by the program prepaired by the mathematician but by a program in the internal code of the computer which is obtained via translating the original program by means of the so called compiler. This is another, highly sophisticated program, usually too complicated to undergo a formal analysis of its correctness in the mathematical sense.

A good illustration what may happen is the following story, recounted here in the way of an anecdote. When the first author came back to Poland after completing with K. Mischaikow in Atlanta the computer assisted proof of chaos in the Lorenz equations, he presneted the proof on a seminar in Krakó w (it is worth noticing that "presenting" has

chanical Theorem Proving, Harcourt, Boston, 1973; J.H. Gallier, *Logic for Computer Science: Foundations of Automatic Theorem Proving*, Harper, New York, 1986.

here a different meaning than with conventional proofs). After a few hours of attempts the computer outcome was far from satisfactory. Only then came the realization that the compiler in Krakó w was different and required some different settings. And indeed, when we repeated the procedure several days later, introducing some small modifications, the presentation was entirely successful.

The above anecdote could raise questions if computer assisted proofs make sense at all. However, one should realize that conventional mathematical proofs also are not faultless, mistakes and errors do happen and corrections are introduced. A good example is the fact that a sphere in an infinite dimensional space for some time was considered to be compact. Probably the most recent example of a gap which required a correction is the already mentioned proof of the Fermat's theorem. It seems that the situation with computer assisted proofs is similar. If several different compilers on different machines with different internal architectures give the same outcome, we can assume that the chances of a not noticed mistake are extremly small. This is even more convincing if one realizes that a random error (due to a fault of computer or compiler) usually results in the outcome of the program which is a total nonsense so that it is immediately visible that something is wrong.

The standards of proof procedures, and so the criteria of mathematical strictness seemed to remain unchanged through ages. After all, we still base much on the work of Euclid, which is 24 centuries old. However, if one takes to reading original works rather than present-day textbooks and monographs, it can be readily seen that the notion of proof in mathematics did evolve in time. Not so long ago, no proof was required for theorems closely related to physical reality (or rather theorems used within a physical theory consistent with predominant Newtonian mechanics). The computer-assisted proofs provide another example for the *evolution of mathematical proof.*