Proof, message and certificate

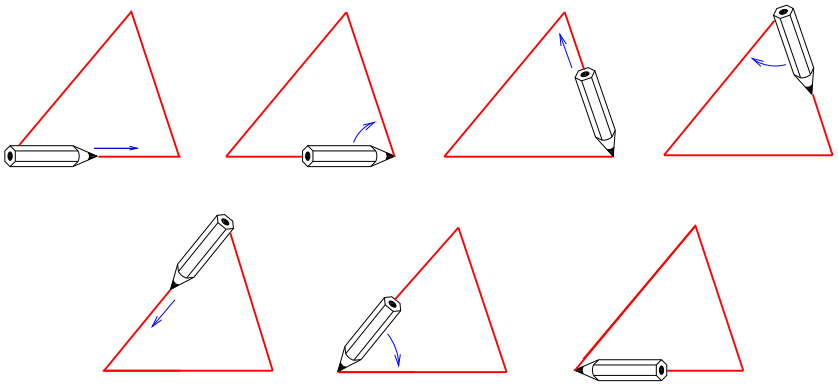
Andrea Asperti

Department of Computer Science
University of Bologna
asperti@cs.unibo.it

Abstract. The recent achievements obtained by means of Interactive Theorem Provers in the automatic verification of complex mathematical results have reopened an old and interesting debate about the essence and purpose of proofs, emphasizing the dichotomy between message and certificate. We claim that it is important to prevent the divorce between these two epistemological functions, discussing the implications for the field of mathematical knowledge management.

1 Introduction

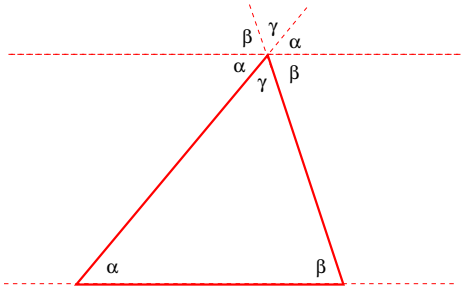
In December 2010, Aaron Sloman posted a message on the MKM mailing list that raised an interesting debate. His message was centered around the following "proof" of Euclid's Theorem, stating that the internal angles of a triangle add up to a straight line (the argument was attributed to Mary Pardoe, a former student of Aaron Sloman). The proof just involves rotating a pencil through each of the angles at the corners of the triangle in turn, which results with the pencil ending up in its initial location but pointing in the opposite direction.



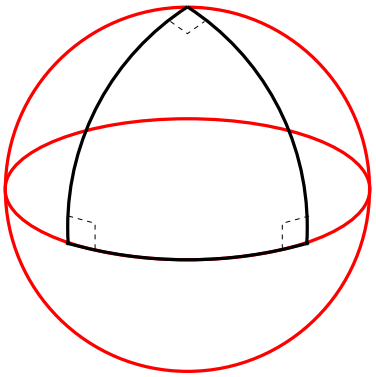
Sloman's point was to show the relevance of graphical methods, in contrast with a logicistic approach, in the spirit of the book *Proofs without words: Exercises in Visual Thinking* by R. B. Nelsen [37] (see also D. Scott introduction to [42], or Jamnik's book on diagrammatic proofs [28]). As Sloman expected, his post immediately raised a fierce debate in the community about the "validity"

of the above argument, with prestigious supporters on both sides. Dana Scott writes:

> *The proof is fine and really is the same as the classical proof. To see this, translate (by parallel translation) all the three angles of the triangle up to the line through the top vertex of the triangle parallel to the lower side. [...] Preservation of angles by parallel translation is justified by the Parallel Postulate.*



In fact, the delicate point of the "proof" is the connection between three rotations performed at different positions in the space, and their translation to a same point, in order to sum them. This becomes evident if, instead of working on the plane, we repeat the pencil experiment on a sphere:



Of course, the problems related to the fifth-postulate and non-Euclidean geometries were evident to all people involved in the discussion: in fact the discussion rapidly switched from the *validity* of the proof to its *pedagogical value*. The supporters of the "proof" appreciated its nature of "thought experiment" (in Lakatos sense) not eventually leading to the expected result.[1] The fact that it fails on the sphere is actually informative, and can be used to better explain the relevance of the parallel postulate, that could otherwise be misunderstood. On the other side, detractors of the proofs were more concerned with the risk

[1] In Lakatos' words [32], "after Columbus one should not be surprised if one does not solve the problem one has set out to solve."

to present to students as a "valid" proof, an argument that is actually flawed. Quoting Arnon Avron:

> *If this "proof" is taught to students as a full, valid proof, then I do not see how the teacher will be able to explain to those students where the hell Euclid's fifth postulate (or the parallels axiom) is used here, or even what is the connections between the theorem and parallel lines.*

2 Message and Certificate

It is usually acknowledged (see e.g. [36]) that proofs have a double epistemological function, playing both the role of *message* and *certification*. In the first incarnation, the emphasis is entirely on *communication*: not only the proof is supposed to explain – by providing intuitions – the reasons for believing in the validity of a given statement, but it should also convey information about the line of thought used to conjecture the result and the techniques used for approaching it. In the second incarnation, the proof is supposed to provide a precise line of reasoning that can be verified in an objective and essentially mechanical way: you can follow and check the validity of the argument even without having a clear understanding of its *meaning*.

The debate about the actual role of proofs in mathematics (see also [30] for a recent survey) essentially concerns the different relevance attributed to the role of message or certificate.

A very common position among mathematicians is to firmly negate any deductive validity to proofs; G. H. Hardy, who is traditionally credited with reforming British mathematics by bringing rigor into it, described the notion of mathematical proof *as we working mathematicians are familiar with* in the following terms [22]:

> *There is strictly speaking no such thing as a mathematical proof; we can, in the last analysis, do nothing but point; [. . .] proofs are what Littlewood and I call* gas*, rhetorical flourishes designed to affect psychology, pictures on the board in the lecture, devices to stimulate the imagination of pupils.*

The opposite position consists in negating any possibility of *communication* without a clear, objective and verifiable assessment of its actual content. The position is nicely summarized by the following words of de Bruijn[2] [16]

> *If you can't explain your mathematics to a machine it is an illusion to think you can explain it to a student.*

A simple example can probably help to understand the issue. Consider the problem of proving that the sum of the first n positive integers is equal to $\frac{n\cdot(n+1)}{2}$. A simple approach (anecdotically attributed to the precocious genius of Gauss[3]), is

[2] See [3] for a deeper discussion of de Bruijn's sentence.

[3] Brian Hayes, [25] collected several hundred accounts of the story of Gauss's boyhood discovery of the "trick" for summing an arithmetic progression, comprising the following:

to write the sum horizontally forwards and backwards, observe that the sum of each column amounts to $n+1$, and we have n of them, giving a total of $\frac{n\cdot(n+1)}{2}$.

– **Message**

$$\begin{array}{cccccc} 1 & 2 & \dots & n-1 & n \\ n & n-1 & \dots & 2 & 1 \\ \hline n+1 & n+1 & \dots & n+1 & n+1 \end{array}$$

A seemingly simple proof can be given by induction: the case "$n = 1$" is obvious, and the inductive case amounts to a trivial computation:

– **Certification**

$$\frac{(n-1)\cdot n}{2} + n = \frac{n\cdot(n+1)}{2}$$

The actual information provided by the two proofs is of course very different: Gauss' "trick" gives us a general *methodology* suitable to be used not only in the given situation but, *mutatis mutandi*, in a wide range of similar problems; in contrast, the inductive proof is quite sterile and uninformative: if we do not know in advance the closed form of the progression (in general, the property we are aiming to), induction provides no hint to guess it. The interesting part of Gauss' proof is its *message*, while the inductive proof is, in this case, a mere *certificate*. The importance of the message often transcends the actual relevance of the statement itself: the fact that the sum of the arithmetic progression is equal to $n(n+1)/2$ is of marginal interest, but Gauss' technique is a major source of inspiration. This is why we are interested in proofs: because they embody the techniques of mathematics and shape the actual organization of this discipline into a structured collection of interconnected notions and theories. What we expect to gain from a solution of the $P \stackrel{?}{=} NP$ problem is not quite the knowledge about the validity of this statement, but a new insight into notions and techniques that appear to lie beyond the current horizon of mathematics. And this is also the main reason why we are interested in *formal* proofs: because the process of formalization obliges to a deeper, philosophical and scientific reflection of the logical and linguistic mechanisms governing the deployment and the organization of mathematical knowledge [3].

It is interesting to observe that Gauss' argument is not so easy to formalize, requiring several small properties of finite summations.[4] In particular, it relies on the following facts: (perm) the sum is unchanged under permutation of the

When Gauss was 6, his schoolmaster, who wanted some peace and quiet, asked the class to add up the numbers 1 to 100. "Class," he said, coughing slightly, "I'm going to ask you to perform a prodigious feat of arithmetic. I'd like you all to add up all the numbers from 1 to 100, without making any errors." "You!" he shouted, pointing at little Gauss, "How would you estimate your chances of succeeding at this task?" "Fifty-fifty, sir," stammered little Gauss, "no more . . . "

[4] Since summation is defined by recursion, most proofs of its properties require recursion too.

addends, (distr) $\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i = \sum_{i=1}^{n}(a_i + b_i)$ and (const) summing n constant elements c yields $n \cdot c$. The best approximation we can do of a "formal" version of Gauss' argument looks something like the following

$$\begin{aligned} 2 \cdot \textstyle\sum_{i=1}^{n} i &= \textstyle\sum_{i=1}^{n} i + \sum_{i=1}^{n} i \\ &= \textstyle\sum_{i=1}^{n} i + \sum_{i=1}^{n}(n - i + 1) \\ &= \textstyle\sum_{i=1}^{n}(i + n - i + 1) = \\ &= \textstyle\sum_{i=1}^{n}(n + 1) \\ &= n \cdot (n + 1) \end{aligned}$$

One could easily wonder if, in this process, the original *message* has not been entirely lost.[5] What is particularly annoying is that, in the formal proof, there isn't a *single* crucial step that embodies the essence of the proof: it is a clever combination of perm, distr and const that makes it work. In fact, the nice point of the graphical representation is to put them together in a single picture: adding rows is the same as adding columns, and each column sum up to the same value. But this raises another interesting issue: namely, if what makes Gauss' argument so appealing is not due to an intrinsic property of the proof but to the fact that it suits particularly well to the intellectual (and sensorial, synoptical) capacities of the human mind.

Most of the proofs in elementary arithmetic have a similar nature. For instance, this is a typical proof [27] of the main property of the Euler ϕ function, computing for any positive integer n the number of integers between 1 and n relative prime to n.

Proposition. $\sum_{d|n} \phi(d) = n$.
Proof. Consider the n rational numbers $\frac{1}{n}$, $\frac{2}{n}$, $\frac{3}{n}$, ..., $\frac{n-1}{n}$, $\frac{n}{n}$. Reduce each to lowest terms; i.e., express each number as quotient of relative prime integers. The denominators will all be divisors of n. If $d|n$, exactly $\phi(d)$ of our numbers will have d in the denominator after reducing to lowest terms. Thus $\sum_{d|n} \phi(d) = n$.

Again, a formal proof [10,2,6] requires a not trivial play with summations properties that is eventually going to hide the intuitive argument so readily communicated by the previous sketch (this is also why a good library of "big ops" [12] is *essential* for any formal development involving combinatorics).

In general, we should accept the fact that there will be many proofs that, once formalized, will loose that degree of *unexpectedness, combined with inevitability and economy* that according to Hardy [23] make the beauty of a mathematical proof. But, mathematics itself is entering a new era of results requiring *extraordinarily long and difficult megaproofs, sometimes relying heavily on computer calculations, and leaving a miasma of doubt behind them* [35]. Maybe, enumeration by cases, *one of the duller forms of mathematical argument* in Hardy's

[5] On the other side, one could also wonder if, after all, Gauss's argument doesn't hide too many details that are worth to be spelled out.

opinion [23], could turn out to be the only viable way to achieve a result, as in the case of the four color theorem [21].

According to Lakatos, *simplicity was the eighteenth-century idea of mathematical rigor* [32], and maybe as we already observed in [5] we should just learn to appreciate a different, and less archaic, kind of beauty.

3 A Social Process

Strictly intertwined with the dichotomy between message and certificate is the discussion about the actual nature of the process aimed to asses the validity of a mathematical argument:[6] a social process, or an objective, almost mechanical activity (see [5] for a recent survey on this topic). The two positions can be summarized by the following quotations:

social/subjective

> *We believe that, in the end, it is a social process that determines whether mathematicians feel confident about a theorem.*
>
> *– R. A. De Millo, R. J. Lipton, A. J. Perlis [17]*

decidable/objective

> *A theorem either can or cannot be derived from a set of axioms. I don't believe that the correctness of a theorem is to be decided by a general election.*
>
> *– L. Lamport [33]*

The main argument usually alleged by the paladins of the "social" perspective, is the practical impossibility of developing fully formal demonstrations, due to the "nearly inconceivable" length of a deduction from first principles.

> *Russell did succeed in showing that ordinary working proofs can be reduced to formal, symbolic deductions. But he failed, in three enormous, taxing volumes, to get beyond the elementary facts of arithmetic. He showed what can be done in principle and what cannot be done in practice.*
> *[. . .] A formal demonstration of one of Ramanujan's conjectures assuming set theory and elementary analysis would take about two thousand pages.*
>
> *– R. A. De Millo, R. J. Lipton, A. J. Perlis [5]*

Even Bourbaki,[7] who is traditionally enlisted in the ranks of the formalist school [34], labels the project of formalizing mathematics as *absolutely unrealizable*

[6] It is important to stress that the debate is not about the overall mathematical activity, that is indubitably a social process (at least, in the same complex, and often conflictual way, artistic creation is), but is really confined to correctness checking.

[7] Bourbaki did not particularly like logic, that he considered as a mere tool: "logic, as far as we mathematicians are concerned, is no more and no less than the grammar of the language which we use, a language which had to exist before the grammar could be constructed" [14].

> *The tiniest proof at the beginning of the Theory of Sets would already require several hundreds of signs for its complete formalization.*
>
> – *Bourbaki [13]*

As we observed in [5], the argument is reminiscent of the general disbelief about the possibilities of writing long programs at the beginning of the fifties: in fact, they were reasoning in terms of assembly languages, and the mistake was due to the inability to conceive a process of automatic translation from a high level programming language to a machine-understandable code. The same is true for formal proofs: a modern interactive prover is precisely a tool interpreting a high level mathematical language (we shall discuss them in the next section) into a set of low level logical instructions automatically checked for correctness by the machine.

> *The arrival of the computer changes the situation dramatically. [...] checking conformance to formal rules is one of the things computers are very good at. [...] the Bourbaki claim that the transition to a completely formal text is routine seems almost an open invitation to give the task to computers.*
>
> – *J. Harrison[24]*

It is important to observe that the high level "proof" can be *arbitrarily* compact: it only depends on how much time you are ready to pay for the translation. From this point of view, a "proof" is *any information* sufficient to make decidable the correctness of a statement (that, in principle, fixed the formal system, is only semidecidable). For instance, an upper bound to its dimension (or any function of it) is a perfectly formal (and decidedly compact) proof. So, something like "10 lines" should be accepted as a perfectly formal argument (that we shall henceforth call "à la Fermat", paying homage to to his actual inventor): we have just to generate and check all proof-terms of the formal language within the given bound: if one of them proves the statement the argument is correct, and otherwise it is wrong.[8]

This sheds new light on the dichotomy between message and certificate: in fact, what kind of message can be found in a proof *à la Fermat*? In other words, it is true that formal proofs can be arbitrarily compact, but it is equally true that the certificate can be *arbitrarily distant* from *any* message.

The divorce between the notions of *message* and *certification* induced by computer proof assistants has been already remarked by Dana Mackenzie in a

[8] In a recent invited talk at CICM 2011, Trybulec suggested to use time complexity to make a distinction between *proofs* and *traces*, reserving the title of proofs only for those certificates that can be checked with a "reasonable" (say, polynomial) complexity. For Automatic Interactive Provers it is of course important to be able to perform proof checking in a reasonable amount of time (hence, certificates are usually proofs in Trybulec sense). At the current state of the art, the dimension of formal certificates is sensibly more verbose (2 to 10 times larger) than the corresponding high level mathematical proof (see [4] for a discussion).

recent article that appeared on Science, where he apparently attributes a positive value to such a separation:

> *Ever since Euclid, mathematical proofs have served a dual purpose: certifying that a statement is true, and explaining why it is true. Now those two epistemological functions may be divorced. In the future, the computer assistant may take care of the certification and leave the mathematician to look for an explanation that humans can understand.*
>
> *– D. Mackenzie [35]*

Mackenzie's argument, however, is pretty weak: if the certificate is divorced from the message, it is enough (up to the adequacy of the encoding) to certify the correctness of the statement, but it says nothing about the correctness of its supposed "explanation". Often, what is doubtful is not the validity of statements, but of their proofs: so, if message and certificate are distant, we are essentially back to the original situation: we are not sure if the "message" the author is trying to communicate to the reader is correct.

The usual objections raised by mathematicians to the issue of automatic verification of statements concern either the correctness of the proof checker itself or the correctness of the encoding of the problem (*adequacy*). For the first point, proof checkers are single applications, often open source, in competition with each other and subject to a severe experimental verification (see also [19] for a list of properties that could strengthen our conviction that a proof checker is reliable). For the second point, sometimes underestimated in the proof assistant community, we should observe that the adequacy of the formalization only depends on the formulation of definitions and statements, and checking that they reflect their intended meaning is a much easier task than checking the correctness of the entire proof. For instance, Gonthier emphasizes that the formal statement of the 4-color theorem [21], including "all definitions" required to understand it, fits on one A4 page; while we believe that this is somewhat an overstatement and more pages are actually required, the point is that, if you trust the proof assistant, you just need to understand and verify a small amount of information.

A more substantial objection concerns the real added value provided by having a formal proof *in case we are not able to convey a human readable message out of it*. We would be in the odd situation to know the existence of a proof, but to have a pretty vague idea of how it concretely works.[9]

[9] With have a similar situation with *proofs by reflection* [15]. The basic idea of this technique is that checking a proof involves running some certified decision procedure. For instance, in order to compare two regular expressions, we can build the corresponding automata and run a suitable bisimulation algorithm. In this case too, we have no direct grasp of the specific proof, but the big difference is that we have a clear understanding of the reasons why the proof is correct, i.e. of the metatheory underlying the approach; if the algorithm is implemented correctly (and this can be mechanically verified), then we can be confident in the proof.

4 Declarative vs. Procedural

It is usually believed that declarative languages are in a better position than procedural ones to preserve the relation between message and certificate in the realm of formal mathematics.

Before entering in this discussion, we would like to attempt a clarification of the two classes of languages. To this end, we make a simple analogy with chess. A chess game can be described in essentially two ways: as a sequence of *moves* or as a sequence of *positions* (see Figure 1). In the first case, *positions are implicit*: they can be reconstructed by executing a subsequence of the moves; in the second case, *moves* are implicit: they can be deduced by the difference between consecutive positions. Moves and positions are simple examples of, respectively, a procedural and a declarative language.

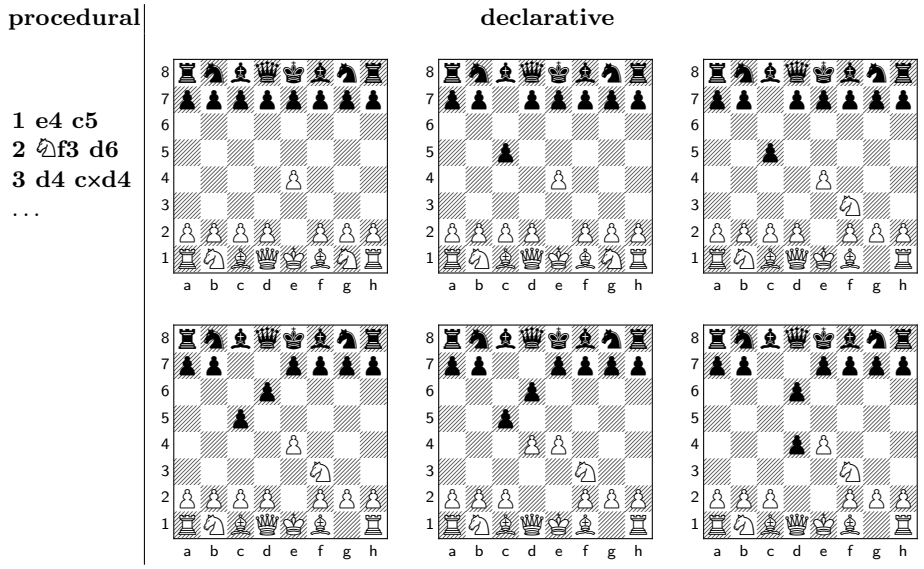


Fig. 1. Moves and Positions

In the case of logic, you have a similar situation. A proof of P_n, in Hilbert's acception, is a sequence of formulae $P_1, \ldots, P_n$ where each P_i is either an axiom or is obtained from formulae preceding P_i in the sequence by means of suitable (fixed) logical rules. This description is actually redundant: you may just give the sequence of rules (procedural description) leaving implicit the sequence of formulae, or you may give the whole sequence of intermediate results $P_1, \ldots, P_n$ leaving to the reader the (easy) task to understand the logical rule required for each inference (declarative description).

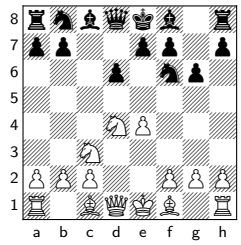
The relative merits of the two representations should be clear. A procedural description is *very compact* but quite unreadable: the point is that each move

refers to a state implicitly defined by the previous steps. To understand the proof, you should be able to figure out in your mind the state of the system after each move. In the case of chess, this is still possible for a trained human, since the board is a relatively simple structure, moves are elementary operations, and games are not too long. However, it becomes practically impossible as soon as you deal with more complex situations, such as symbolic logic.[10]

On the other side, declarative descriptions provide, at each instant, a full description of the current state: since the evolution does not depend on the past (the game is history free), you do not need to remember or rebuild any information and may entirely focus on the given state. Declarative descriptions are hence immediately readable,[11] but also (as it is evident in the case of chess), much more *verbose*.

The gap between procedural and declarative languages is not so large as it may appear at first sight: in fact, they *complement* each other and *integrate together* very well. For instance, when discussing a chess game in the procedural style it is customary to explicitly draw the state of the board at particularly interesting or instructive places (see Figure 2 - Fisher vs Larsen, Portoroz 1958, Sicilian Defense, Yugoslavian Attack at the Dragon Variation)

1 e4 c5
2 ♘f3 d6
3 d4 c×d4
4 ♘×d4 ♘f6
5 ♘c3 g6



6 ♗e3 ♗g7
7 f3 O-O
8 ♕d2 ♘c6
9 ♗c4

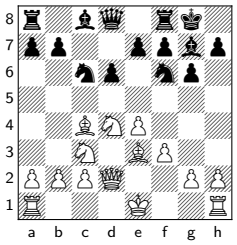


Fig. 2. Cuts

In the terminology of logic, this operation is called a *cut*, since it divides a complex description (a game, a proof) into smaller components, each one with an independent interest (not every position of the board is worth a draw, in the same way as not every intermediate logical step is worth a cut). The tendency, among interactive provers adopting a procedural style, to promote the use of cuts and a more structured description of the proof (and hence to implicitly

[10] The criticism that procedural languages lack readability is a bit unfair: the point is that they *are not meant* to be read, but to be interactively re-executed.

[11] By "readability" we mean here the mere possibility to follow more easily the chain of reasoning in a proof; of course this does not imply a real grasp of the information it is supposed to convey. For instance, it is difficult to learn chess by just studying past games, no matter how they are represented, without auxiliary expertise.

move towards a more declarative and readable style of proofs) is clearly testified by the most recent applications such as Ssreflect [20] or Matita [38,8].

On the other hand, we can make the declarative description less verbose by simply augmenting the granularity of steps. For instance, without any loss of information, we can decide to represent the board at each player-opponent move instead of considering single half moves (see Figure 3). But we could arbitrarily

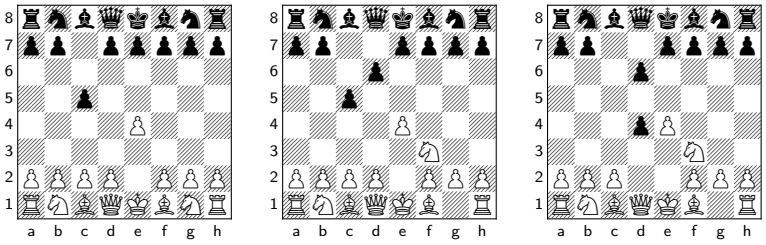


Fig. 3. Augmenting the granularity

decide to represent the board every, say, pair of moves, *relying on the reader's ability to fill in the missing information*. The granularity could of course depend on the state of the game, and the knowledge we expect from the reader. For instance, chess *openings* have been deeply investigated, and each good chess player immediately recognizes a particular opening from the state of the board in which it ends up: so, a procedural part of several moves can be left implicit and we can immediately jump in the middle of the game. In the case of declarative languages for interactive provers, the intended reader is the machine, that is, a device with limited intelligence: as a consequence the granularity cannot be too coarse-grained.[12] However, we can further reduce it by providing suitable proof hints to the machine, or explicit procedural fragments.

Having clarified that procedural and declarative languages are not mutually exclusive, and may easily integrate (see also [11,41]), there is however an important observations that must be made.

In declarative languages, the level of granularity is somehow *imposed by the intelligence of the machine*, and there is no reason to believe this corresponds with the requirement of a good exposition in human terms. As we already observed, granularity can be modified by adding suitable proof hints, but again

[12] This use of the machine, meant to relieve the user from the burden to fill in relatively trivial steps by automatically completing the missing gaps (the underlying "logical glue" of the mathematical language) is called *small scale automation* in [9], in contrast with *large scale automation* referring to the more creative and complex activity to help the user in the process of figuring out the actual proof. The two kinds of automation seem to have different requirements and can possibly deserve different approaches and solutions.

these proof-hints, similarly to procedural fragments, must be machine understandable, and they may hardly improve the readability of the text (they are essentially meant to reduce verbosity).

On the other side, a procedural proof can be more or less arbitrarily split in smaller fragments, for expository purposes. Moreover, the kind of enrichment we can do on the text can be entirely aimed for humans: if we ask an automatic chess player to print the state of the board at a given instant we do not necessarily expect the player to be able to reparse this representation, purely meant for human convenience (even if we may agree that it could be a desirable property). In other terms, the kind of enrichment corresponding with a "cut" does not necessarily need to be a *formal statement*: it could be a comment, a picture, a diagram, an animation, or whatever. The same is essentially true with a declarative language, but in this case any additional comment or explanation is eventually going to interfere with the original vernacular, adding a confusing level of "explanation" to a language that was already supposed to be self-explanatory.

5 A Complex Problem

The complexity of the problem of preserving the relation between message and certificate can be understood by an analogy[13] with software, where we have essentially the same situation: writing a program requires understanding and solving a problem, but it is extremely difficult to extract such a knowledge (the *message*) from the final code (playing the role of *certification*). The major investment, in programming as well as in formalization, is not the actual writing up of the program, but the preliminary phase of analysis, planning and design; it is a real pity that this information gets essentially lost in the resulting encoding. In spite of the evident relevance of the problem, we have assisted during the last decades to the substantial failure of many interesting projects aimed to improve writing and readability of programs. A relevant example is *literate programming* [31], that in the intention of Knuth was not just a way to produce high-quality formatted documentation, but a methodology aimed to improve the quality of the software itself, forcing programmers to explicitly state the relevant concepts behind their code. Literate programming was meant to represent what Knuth called the "web of abstract concepts" hiding behind the design of software; in particular, it supported the possibility to change the order of the source code from a machine-imposed sequence to one more convenient to the human mind.

The reasons why literate programming failed to have a significant impact on software development are not so evident. We could probably admit that in modern programming languages (both object oriented and functional), the kind of abstraction mechanisms provided by the language are already sufficient to

[13] The analogy we are making here, comparing programs with proofs, is essentially the so-called Curry-Howard analogy [26]. The fact of finding in the two realms the same dichotomy message/certification proves once again the deep philosophical implications of this correspondance that largely transcend the technical aspects of proof theory.

prevent that "dictatorship" of the machine that Knuth seemed to suffer in a particular way.[14] This is probably enough to explain why, at present, a simple documentation generator like say, Doxygen,[15] is largely more popular than the more sophisticated and ambitious literate programming approach.

It is likely to expect a similar situation in the realm of interactive provers. Nowadays, procedural languages for interactive provers permit to adhere with sufficient precision to the natural "flows of thoughts", and techniques and methodologies like small scale reflection [20], mathematical components [18] or small scale automation [9] are meant to further improve on this situation. So, a simple documentation generators is likely to be more rapidly adopted by users of interactive provers than a more sophisticated authoring interface.[16]

Many available proof assistant already provide functionalities for enriched HTML presentation of their libraries, like the Coqdoc tool for the Coq System. These tools allow to produce interesting ebooks, like for instance the "Software Foundations" course at `http://www.cis.upenn.edu/~bcpierce/sf/`. The next natural step would consist in improving on line interactivity with the document, especially for editing and execution. An additional layer (called Proviola) is currently being developed [39] on top of coqdoc, with the goal of providing more dynamic presentations of the Coq formalizations. Several system are developing web-interfaces for their application (see [29,7]), many of them inspired by wikis [40,1]. The wiki-approach looks particularly promising: a large repository of mathematical knowledge may be only conceived as a collaborative working space. This is especially true in a formal setting where, in the long term, re-use of mathematical components will be the crucial aspect underpinning the success of interactive theorem provers.

Acknowledgments
This paper is partially based on two talks given by the author, one at the Tata Institute of Technology, Mumbai (India) in 2009 and another at the Summer School of Logic of the Italian Association for Logic and Applications, held in Gargnano, Italy, in August 2011. I would like to thank, respectively, Raja Natarajan and Silvio Ghilardi for offering me these opportunities. I would also like to thank the anonymous reviewers for their detailed and stimulating comments.

References

1. Jesse Alama, Kasper Brink, Lionel Mamane, and Josef Urban. Large formal wikis: Issues and solutions. In *Proceedings of Intelligent Computer Mathematics (CICM 2011), Bertinoro, Italy*, volume 6824 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2011.

[14] In any case, it is surely more interesting, and probably more useful, to improve the programming language, than solving the problem at a meta level, via a generative approach.

[15] `http:www.doxygen.org`

[16] This does not imply that one is better than the other; in the long run, a highly sophisticated authoring environment can still be the better solution.

2. Andrea Asperti and Cristian Armentano. A page in number theory. *Journal of Formalized Reasoning*, 1:1–23, 2008.
3. Andrea Asperti and Jeremy Avigad. Zen and the art of formalization. *Mathematical Structures in Computer Science*, 21(4):679–682, 2011.
4. Andrea Asperti and Claudio Sacerdoti Coen. Some considerations on the usability of interactive provers. In *Intelligent Computer Mathematics, 10th International Conference, Paris, France, July 5-10, 2010*, volume 6167 of *Lecture Notes in Computer Science*, pages 147–156. Springer, 2010.
5. Andrea Asperti, Herman Geuvers, and Raja Natarajan. Social processes, program verification and all that. *Mathematical Structures in Computer Science*, 19(5):877–896, 2009.
6. Andrea Asperti and Wilmer Ricciotti. About the formalization of some results by Chebyshev in number theory. In *Proc. of TYPES'08*, volume 5497 of *LNCS*, pages 19–31. Springer-Verlag, 2009.
7. Andrea Asperti and Wilmer Ricciotti. A web interface for matita. In *Proceedings of Intelligent Computer Mathematics (CICM 2012), Bremen, Germany*, Lecture Notes in Artificial Intelligence. Springer, 2012. This volume.
8. Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi. The Matita interactive theorem prover. In *Proceedings of the 23rd International Conference on Automated Deduction (CADE-2011), Wroclaw, Poland*, volume 6803 of *LNCS*, 2011.
9. Andrea Asperti and Enrico Tassi. Superposition as a logical glue. *EPTCS*, 53:1–15, 2011.
10. Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. A formally verified proof of the prime number theorem. *ACM Trans. Comput. Log.*, 9(1), 2007.
11. Henk Barendregt. Towards an interactive mathematical proof language. In Fairouz Kamareddine, editor, *Thirty Five Years of Automath*, pages 25–36. Kluwer, 2003.
12. Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In *TPHOLs*, pages 86–101, 2008.
13. Nicolas Bourbaki. The architecture of mathematics. *Monthly*, 57:221–232, 1950.
14. Nicolas Bourbaki. *Theory of Sets*. Elements of mathematics. Addison Wesley, 1968.
15. Samuel Boutin. Using reflection to build efficient and certified decision procedures. In Martin Abadi and Takahashi Ito editors, editors, *Theoretical Aspect of Computer Software TACS'97, Lecture Notes in Computer Science*, volume 1281, pages 515–529. Springer-Verlag, 1997.
16. N.G.De Bruijn. Memories of the automath project. Invited Lecture at the Mathematics Knowledge Management Symposium, 25-29 November 2003, Heriot-Watt University, Edinburgh, Scotland.
17. Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis. Social processes and proofs of theorems and programs. *Commun. ACM*, 22(5):271–280, 1979.
18. François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging mathematical structures. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, volume 5674 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2009.
19. Herman Geuvers. Proof Assistants: history, ideas and future. *Sadhana*, 34(1):3–25, 2009.
20. Georges Gonthier and Assia Mahboubi. An introduction to small scale reflection in coq. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.

21. Geroges Gonthier. Formal proof – the four color theorem. *Notices of the American Mathematical Society*, 55:1382–1394, 2008.
22. Godfrey H. Hardy. Mathematical proof. *Mind*, 38:1–25, 1928.
23. Godfrey H. Hardy. *A Mathematician's Apology.* Cambridge University Press, London, UK, 1940.
24. John Harrison. Formal proof – theory and practice. *Notices of the American Mathematical Society*, 55:1395–1406, 2008.
25. Brian Hayes. Gauss's day of reckoning. *American Scientist*, 4(3):200–207, 2006.
26. William A. Howard. The formulae-as-types notion of construction. In J.P.Seldin and J.R.Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, Boston MA, 1980.
27. K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory.* Springer Verlag, 2006.
28. Mateja Jamnik. *Mathematical Reasoning with Diagrams: from intuition to automation.* CSLI Press, Stanford, California, USA, 2001.
29. Cezary Kaliszyk. Web interfaces for proof assistants. *Electr. Notes Theor. Comput. Sci.*, 174(2):49–61, 2007.
30. Manfred Kerber. Proofs, proofs, proofs, and proofs. In *Proceedings of Intelligent Computer Mathematics, 10th International Conference, Paris, France*, volume 6167 of *Lecture Notes in Computer Science*, pages 345–354. Springer, 2010.
31. Donald E. Knuth. *Literate Programming.* Center for the Study of Language and Information, 1992.
32. Imre Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery.* Cambridge University Press, 1976.
33. Leslie Lamport. Letter to the editor. *Communications of the ACM*, 22:624, 1979.
34. Joong Kwoen Lee. Philosophical perspectives on proof in mathematics education. *Philosophy of Mathematics Education Journal*, 16, 2002.
35. Dana MacKenzie. What in the name of Euclid is going on here? *Science*, 207(5714):1402–1403, 2005.
36. Donald Mackenzie. *Mechanizing Proof.* MIT Press, 2001.
37. Roger B. Nelsen. *Proofs without Words: Exercises in Visual Thinking.* The Mathematical Association of America, 1997.
38. Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. Tinycals: step by step tacticals. In *Proceedings of User Interface for Theorem Provers 2006*, volume 174 of *Electronic Notes in Theoretical Computer Science*, pages 125–142. Elsevier Science, 2006.
39. C. Tankink, H. Geuvers, J. McKinna, and F. Wiedijk. Proviola: A tool for proof re-animation. In *Proceedings of AISC 2010, Heidelberg*, volume 6167 of *Lecture Notes in Computer Science*, pages 440–454. Springer, 2010.
40. J. Urban, J. Alama, P. Rudnicki, and H. Geuvers. A wiki for mizar: Motivation, considerations, and initial prototype. In *Proceedings of AISC 2010, Heidelberg*, volume 6167 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2010.
41. Freek Wiedijk. Formal proof sketches. In Wan Fokkink and Jaco van de Pol, editors, *7th Dutch Proof Tools Day, Program + Proceedings*, 2003. CWI, Amsterdam.
42. Freek Wiedijk. The seventeen provers of the world. *LNAI*, 3600, 2006.