

Thoughts on the process of producing mathematical documents

BY LIONEL ELIE MAMANE

EMAIL: `lionel@mamane.lu`

Toernooiveld 1
NL-6525 ED Nijmegen
The Netherlands

26th January 2004

Abstract

Mathematicians produce two things: Theories and documents about the theory. Computers should help them in both activities, but should both activities be covered by the same tool?

1 Introduction

I wish to have feedback on my views on how the mathematical document production “work flow” is organised, and what tools should cover what part of the work flow.

1.1 ... for mathematicians

Currently, a computer is more useful to a secretary, accountant, etc than to a mathematician. The former trades nowadays routinely use computer programs to help them in their tasks, and most (if not all) their tasks whose computerisation is a win are computerised. For mathematicians, the situation is less optimal: While for some tasks, like in computational algebra, automatisisation and computer help is available, effective and widely used, it is my opinion that computers can be much more useful to mathematicians than they are now. One of our goals is to change this, to ensure that, if they wish, mathematicians can make maximal use of computers and get maximal help from computers. For this, we need to understand what mathematicians need, and how the “ideal version” of programs they would use should be organised. This document is a call for opinions on a particular point of this subject.

2 Mathematical Documents

In the work flow of the mathematician I see two very different activities:

1. Developing a mathematical theory

2. Writing documents about the theory, e.g.

- Course
- Journal article
- Vulgarisation article to appear in magazine
- Book
- Presentation slides
- Tutorial

The result of the first is a complete, rigorous presentation of the theory that is read by colleagues. It is to be added to the “big library of all mathematics”. We will thus call it *theory library entry*. It has very strong requirements of correctness that are, in my opinion, ideally met by fully formalising it, and having the formalisation checked by a De Bruijn criterion compliant theorem checker. It would be “socially acceptable” to *require* such a theory to be fully formalised before it is generally accepted as valid¹; the mathematicians will accept to do this extra step, to spend this extra time (once the tools are mature and this represents an acceptable amount of effort and time). This will save us repeats of e.g. the story of Hilbert’s 16th problem (maximal number of limit cycles of a polynomial differential equation in dimension 2) throughout the century, as described in [Ghy01]². The “reference view” of the mathematical theory presentation can thus, and there are advantages to doing so, be derived more or less automatically from the formalisation.

The second has various audiences (students, colleagues, amateurs, ...). I explicitly make no statement here about the support of the document (cellulose-based paper, interactive or non-interactive electronic document, ...). The requirements of guarantees of total and formal correctness are sometimes/often looser. While a nice extra, I don’t see such guarantees becoming a real requirement; (some) mathematicians will refuse to spend the extra time. Weaker forms of checking / formality (like [Ned02] or [Wie]) may be applicable, either as a publication requirement, or as a help to the author to catch typing errors and slips, similarly to automatic spelling and grammar checks done in mainstream document preparation systems. These documents would obviously refer to and incorporate parts of the theory library entry.

1. This does not necessarily mean that it is a requirement for publication of an article. Publishing an article that says “Here is a proof sketch, the details are not fully worked out.” is still OK. A computer formalisation is simply a strong hint that if the claim “this proof is correct” is made, it is true. The social standards will evolve in the direction that if a claim of correctness is done, it is not believed unless backed up by a computer-verifiable formalisation.

2. In short, an upper bound was found and proved. The theorem was generally accepted as true, then an error in the proof is found. Another proof surfaces, is accepted, later an error is found. Iterate a few times, and you’ll get to the year 2004.

3 Tools for mathematical documents

The question this note aims to ask is “Do these two different activities warrant two different tools?”. Let’s call the tool used for writing library entries *LibPerfect* and the one used for writing documents about the theory *ImpressAudience*. It seems natural that *LibPerfect* would include everything useful to the user when he is building a theory or a result (proof checker, presentation of finished and unfinished proofs to send to colleagues, search facilities to find a lemma proven years earlier, computer algebra system, etc). *ImpressAudience* would include everything useful to write documents about mathematics (easy use of existing mathematical notations, easy definition of new notations, multiple support (paper, electronic, transparencs, ...), etc), ideally not requiring the user to retype what he has already typed in *LibPerfect*. The question now becomes the question of whether *LibPerfect* should be distinct from or equal to *ImpressAudience*.

3.1 Equality

- In *ImpressAudience*, automated checks should be optional and non-obtrusive. But this would be useful in *LibPerfect*, too, as long as one can ask the system the question “Does this document pass this check” easily and in $O(1)$ time³. Indeed, this would help the mathematician to organise ideas and intuitions that are not yet worked out in full details. For example, (s)he might want to turn off enforced logic correctness so that (s)he can explore usefulness of a (change of) definition, make bigger steps in reasoning than would be accepted by the theorem checker, ... Kind of “rapid prototyping” for mathematics.
- *ImpressAudience* needs to be able to refer to, and include parts of, *LibPerfect* documents. *LibPerfect*, too: Progress in mathematics is incremental.
- Less computer skills are necessary for an active mathematician to take full advantage of computer help; one program to learn instead of two.
- If two different activities a and b can cleanly be enclosed in one tool(set), then it should be done. This will produce a more general tool that will cover activities in a class E , whose broadness may not be known to the tool writer, but users will find new inhabitants of this class. This applies to our case, as exemplified by other points here.

3.2 Distinctiveness

- The two activities are too different in nature, one tool cannot cover both and do good for both. *LibPerfect* and *ImpressAudience* should be complementary, but distinct.
- The problem is between the chair and the keyboard: If using the same tool for both, mathematicians (in training) will tend to do one activity the way they should do the other, or be very confused. The tools should be separate because it should stay separate for users, and tools should mirror the mental model of the user.

3. This obviously doesn’t mean the answer will be output in $O(1)$ time.

- The equality approach will lead to a very tight integration between the theorem prover and the display engine parts of the system. Temptation will be very big to do a complete system that can work only with one theorem prover, e.g. Coq, as back-end. This will substantially stifle innovation in the theorem prover area, and remove free choice of theorem prover, thus of foundations of mathematics, from the user: People that want to use, e.g. Mizar, will be punished for it because they won't be able to use that magnificent unified tool to write documents, and people using the unified system won't be able to refer to their work. Mizar users will effectively be shunted from the community. This is politically unacceptable: we do not want unique thought to rule us. Science is all about having different people trying different perspectives.

The ideal system, thus, should be a galaxy of inter-operable components, with well-defined and universal interfaces between them. Users will pick their own mix from it: A theorem prover that matches their view of foundations mathematics, a display engine whose input language fits their preferences, ... The question whether this galaxy is possible is open, though. I feel it depends mostly on how much semantics are needed at what point of the chain. The more semantics are needed in a particular stratum, the more this stratum will depend on the exact implementation of the other stratus.

3.3 Discussion

The central, decisive, points seem to be

- whether a tool can be a good tool for both activities
- whether universal interoperability is possible

What's your opinion?

Bibliography

- [Ghy01] Etienne Ghys. Le rôle des erreurs dans le développement des mathématiques. Seminar <http://www.ens-lyon.fr/asso/groupe-seminaires/seminaires/eghys.php>, May 2001.
- [Ned02] Rob Nederpelt. Weak type theory: A formal language for mathematics. Computing Science Report 02-05, Eindhoven University of Technology, Department of Math. and Comp. Sc., Den Dolech 2, Postbus 513 - 5600 MB Eindhoven, May 2002.
- [Wie] Freek Wiedijk. Formal proof sketches. submitted to the TYPES 2003 proceedings, <http://www.cs.kun.nl/freek/notes/sketches2.tex>.