Towards the Structure of Mathematical Proof

Reinhard Kahle[⋆]

CENTRIA and DM, FCT, Universidade Nova de Lisboa
P-2829-516 Caparica, Portugal
kahle@mat.uc.pt

1 Introduction

In 2006, Wiedijk edited a book where seventeen theorem provers are presented how they *prove* the irrationality of $\sqrt{2}$.[1] This book gives an interesting insight into the state of the art of theorem proving.[2] In the introduction the editor presents a six line proof of the irrationality of $\sqrt{2}$, taken from the textbook of Hardy and Wright [HW60, p. 39f]:

> "The traditional proof ascribed to Pythagoras runs as follows. If $\sqrt{2}$ is rational, then the equation
>
> $$a^2 = 2b^2$$
>
> is soluble in integers a, b with $(a, b) = 1$. Hence a^2 is even, and therefore a is even. If $a = 2c$, then $4c^2 = 2b^2, 2c^2 = b^2$, and b is also even, contrary to the hypothesis that $(a, b) = 1$."

This proof should be understandable for everybody with basic mathematical knowledge. However, the editor draws attention to the fact, that a simple test of correctness of this proof is out of reach for the current computer provers [Wie06, p. 3]: "Ideally, a computer should be able to take this text as input and check it for its correctness. We clearly are not yet there."

Thus, he advances with the representation of proofs of this theorem given, or implemented, in the theorem provers presented in the book. The outcome is puzzling. Of course, all the theorem provers provide us with "correct proofs", but none of them would even come close to a human readable text, comparable with the proof given in the book of Hardy and Wright. This is addressed by Scott when he writes in the preface of the book, [Sco06, p. viiif]: "We can also

⋆ Research supported by the Portuguese Science Foundation, FCT, through the projects *Hilbert's Legacy in the Philosophy of Mathematics*, PTDC/FIL-FCI/109991/2009 and *The Notion of Mathematical Proof*, PTDC/MHC-FIL/5363/2012.

[1] The same example was chosen by Lamport in his note [Lam95] which, along with its "update" [Lam12], provides a good motivation for our work.

[2] Apparently, there took no major advance place in the last 8 years with respect to the question under consideration here, although, of course, many of the provers improved a lot.

see clearly from the examples in this collection that the *notations* for input and output have to be made more human readable."

Of course, there is a lot of effort put into the challenge to generate human readable output, receiving even more attraction in the mathematical community since the Fields medallist Tim Gowers started to contribute to this question, cf. [GG13].

But our question is not whether or how one can produce human readable output of *one* computer generated proof or proof system, but rather, how one can capture the *structure of the "abstract" mathematical proof* in a representation of a proof. With respect to the problem of checking automatically the proof of the irrationality of $\sqrt{2}$ given above, WIEDIJK writes [Wie06, p. 3]: "One of the reasons for this is that this version of the proof does not have enough detail." Of course, this "accusation" can easily be inverted by stating that computer generated proofs contain *too many details*, as it was expressed by by SCOTT [Sco06, p. ix f]: "[F]or verification (...) *checkable proofs* have to be generated and archived. Computers are so fast now that hundreds of pages of steps of simplifications can be recorded even for simple problems. Hence, we are faced with the questions, 'What really is a proof?' and 'How much detail is needed?' ".

Here, we would like to challenge the theorem prover community to provide *interfaces* which allow to transfer proofs performed in one theorem prover to another.[3] Of course, many systems differ significantly in the underlying logic, the internal representation of datatypes, automatic components, user defined extensions, etc. However, a proof—in the sense we look for—of the irrationality of $\sqrt{2}$ should not depend on any of these particularities. Thus, if there is something like an abstract proof of this theorem, the theorem provers should be able to give such one, and they should be able to exchange it among each other. We conjecture that a very lot of the disturbing details, which also depend on the specific implementation of a theorem prover, would be filtered out in the proof representation suitable for computer interaction.

Our own proposal is to develop an `xml` specification of mathematical proof which should be adequate to represent proofs as given by HARDY and WRIGHT; in a second step, one would have to write interpreters which would, first, translate proof generated by theorem provers in such `xml` scripts and, then, such scripts back into proofs of the different computer provers.

2 An `xml` specification for Mathematical Proof

In the following we like to give an *ad-hoc* example how a `xml` representation of HARDY and WRIGHT could look like. We are far from presenting here a

[3] There is some work going on in this direction: *Hybrid systems* try to combine components of different proof systems to work together; cf. for instance for HOL/Mizar [Har96], or for Elan/Coq [AN00]. These approaches, however, are usually *bilateral* and depend on the syntax of the specific systems which are combined. For OMDoc see below.

substantial proposal for an `xml` specification, but like to give the example to indicate some of the possible features such a specification should show.

```
 1  <theorem>
 2    <statement> $\sqrt{2} \notin \mathbb{Q}$ </statement>
 3    <proof>
 4      <assumption label="1"> $\sqrt{2} \in \mathbb{Q}$ </assumption>
 5      <hence>
 6        <exists> $a \in \mathbb{N}$ </exists>
 7        <exists> $b \in \mathbb{N}$ </exists>
 8        <line label="2"> $a^2 = 2b^2$ </line>
 9        <line label="3"> $(a, b) = 1$ </line>
10      <hence/>
11      <hence>
12        <line label="4"> $a^2$ is even </line>
13      <hence/>
14      <hence>
15        <line label="5"> $a$ is even </line>
16      </hence>
17      <hence>
18        <exists> $c \in \mathbb{N}$ </exists>
19        <line label="6"> $a = 2c$ </line>
20      </hence>
21      <hence>
22        <line label="7"> $4c^2 = 2b^2$ </line>
23      </hence>
24      <hence>
25        <line label="8"> $2c^2 = b^2$ </line>
26      </hence>
27      <hence>
28        <line label="9"> $b$ is even </line>
29      </hence>
30      <hence>
31        <contradiction>
32           <with>3</with>
33        </contradiction>
34      </hence>
35    </proof>
36  </theorem>
```

We hope that the reader agrees that this is a rather faithful representation of the proof given above. The only explicit addition is line 18, as c “falls from heaven” in the proof above, but we would like to avoid to have “free” variables hanging around in our representation.

Two remarks are in order. First, we gave the mathematical statements in the lines of the proof in the usual mathematical notation; of course, they could—

and probably should—also be specified in an `xml` representation. However, they are *secondary* for the *structure* of the proof. They will be, of course, absolutely essential for the *correctness* of the proof—but that's not our issue (at least, not at this stage).

Second, the main tag is obviously `<hence/>`, which should represent a "step" in the proof. We model it here "line-by-line" such that the reconstruction looks like "Hilbert-style"; if one would incorporate premises and conclusion in one tag which, in this case, would become nested, one would be closer to a "natural deduction-style" representation. But we think that such a nesting is not really present in the proof above, and that it would only be the result of a (first) meta analysis of the proof.

Two more minor remarks: first, the number labels of the `<line/>` tag are *ad-hoc*, just to have the possibility to refer to them; in the concrete case we use it line 32 for the contradiction. Already here one may notice that Hardy and Wright suppress a last step of the proof: it is not said explicitly that the contradiction obtained results in the negation of the assumption made at the beginning—and only this gives you the statement of the theorem. It is obvious that this last step is expected to be recognized by any mathematical reader.

Second, in the given example we are not sure whether our treatment of quantifiers is appropriate, in particular, because we do not indicate the scope of the quantifications (which, however, are also not indicated explicitly in the proof).

3 `<hence/>`

As said, the `<hence/>` tag is the main ingredient of our `xml` example. It should represent a "step" as it can be identified in the proof above. One may note first, that we do not follow a literal translation which would represent the second sentence of the proof "If $\sqrt{2}$ is rational, then ..." as an implication; in fact, it is not treated as an implication in the proof, as the premise of it is never assumed separately. Thus, if the reader agrees, that this tag gives a faithful representation of Hardy and Wright's steps, we can ask how these steps are related to reasoning implemented in automatic or interactive theorem provers.

Wiedijk's question. Wiedijk has asked for an automatic verification of Hardy and Wright's proof, admitting that we are not yet there. Would it be possible to verify our "`xml` proof" by a computer assisted theorem prover? We think that this might be possible. Of course, at this stage the mathematical formulae would need to be expressed in a syntax understandable for the theorem prover, which should not be a big deal. The main question is whether a prover would/could be able to "fill" the missing *logical* arguments to justify the `hence` steps. This would require a "small" automatic theorem prover device which would have to search for a justification of the conclusion by browsing through the lines before. To go from the line with label 4 to the line with label 5, a small lemma would be needed saying that x^2 is even implies x is even. It is to expect that such statements are available in appropriate libraries. Of course, it would be easy to add a `<justification/>` tag within the `<hence/>` tag which could represent information like "By line x and lemma $y.z$". Even if these information are not given explicitly in the mathematical proof (as in the case of our proof above), one could ask for it, if the `xml` proof is generated *interactively.* Thus, for a *given* "`xml` proof" we would expect that it can be translated to (and then verified by) a theorem prover which has an appropriate automatic/interactive search tool for the mathematical and logical steps still hidden in a `<hence/>` step.

Our question. Our question is whether it might be possible to go the other way around: given a proof generated by a theorem prover, could we extract a `xml` proof which *hides* sufficiently many information such that this proof is digestible for a Mathematician? This is an open question. But to discuss it, we like point out that our `xml` specification should be kept sufficiently slim that the logical particularities of the different provers—as their logical framework, their internal representation of datatypes, their calculus, etc.—are not necessarily directly expressible. It is our aim to *abstract* from these particularities, and if we are able to do so, we might reach the mathematical core of a proof.

4 Discussion

We report here on work in progress for an `xml` specification to represent mathematical proofs. While the technical aspect of this work is still on its very initial stage, we have already identify a couple of important aspects which we would like to summarize here:

- It seems to be useful to separate the structure of the proof from its mathematical content (somehow as it is done in our `xml` proof by leaving the mathematical content in the usual mathematical notation). Also, we do not aim for a formal *logical* representation of the argument; to the contrary, the surface structure of a mathematical proof seems to need rather restricted logical reasoning.[4] This should be reflected in the specification.
- The example of Hardy and Wright's proof suggests that the mathematical argument is essentially independent of the underlying calculus one could choose to formalize it.[5] Thus, the *structural aspects of logical calculi* seem to be inessential from the mathematical point of view. We would like to put this in a positive way: the mathematical content of a formalized proof is probably invariant under the change of the calculus; in other words: we could probably get a good part of the mathematical content of a formalized proof

[4] Our example, in fact, seems to use just Modus Ponens steps, after some definitional rewriting, and one "contradiction"; of course, this is due to the fact that many mathematical steps are hidden in lemmata (as the one mentioned: x^2 is even implies x is even). What is clearly missing in our example are instances of *case distinction* and *induction.*

[5] This should come, of course, to no surprise; Mathematicians were doing proofs for millenniums without use any logical calculus in the modern sense.

by *abstracting* from the structural aspects fundamental in any formalized version.

– The idea to abstract from the mathematical content of a proof was even taken further by Baaz, Krajíček, and Pudlák who removed the content completely, obtaining what they called the *skeleton of a proof*,[6] [KP88,BP93,Baa99].
– With respect to the missing details in mathematical proofs, about which Wiedijk complained, it is obvious that mathematician fill them with their *background knowledge*. What we expect, might be called *intelligent proof reading*.[7] And, it doesn't seem to be impossible that the missing information taken from background knowledge should be recoverable by theorem provers, at least with some interactive help.
– There is, of course, a lot of existing work related to our approach. As examples let us mention the `xml`-representation of `Mizar` proof [Urb06]; the `OMDoc` initiative (`http://www.omdoc.org/`) headed by Michael Kohlhase; the *MathLang* project of Fairouz Kamareddine and J. B. Wells; and Ganesalingham's work backed by Gowers.

  • In the case of `Mizar`, of course, the internal representation of `Mizar` proofs make part of the `xml` specification, while we, explicitly, want to abstract from these kind of representation.
  • We share the objective of `OMDoc` to provide a specification which makes it possible to exchange proofs between different theorem provers. But while `OMDoc` starts from an in-depth analysis of all kinds of mathematical texts, we restrict ourselves to the analysis of (the structure of) mathematical proof.
  • In this respect, *MathLang* [KW08] is probably the approach which comes closest to ours; it has, however, a broader aim, including the linguistic analysis of mathematical texts. It might be appropriate to characterize our proposal as a "subtask" which could be found in *MathLang*, i.e., the task which uncovers the mathematical structure of a proof.
  • Ganesalingham's work [Gan13], also together with Gowers [GG13], is also concerned with linguistic aspects which are not in our focus; but it seems to rather mature with respect to the conceptional analysis of the mathematical concepts used in a proof and is, therefore, highly relevant for our project.

Our work will definitely profit from all these experiences. But let us note, that we approach the question of proof representation from a different perspective, which is, in general, more *conceptional* rather than *technical* (see also [Kah1x]).

[6] As S.S. Wainer commented [Baaz personal communication]: the skeleton of a proof is what remains when a proof is dead.
[7] This term was coined by Jesse Alama.

References

[AN00] Cuihtlauac Alvarado and Quang-Huy Nguyen. ELAN for equational reasoning in Coq. In J. Despeyroux, editor, *2nd Workshop on Logical Frameworks and Metalanguage - LFM'00, Santa Barbara, USA*. INRIA, 2000.

[Baa99] Matthias Baaz. Note on the generalization of calculations. *Theoretical Computer Science*, 224(1–2):3–11, 1999.

[BP93] Matthias Baaz and Pavel Pudlák. Kreisel's conjecture for $l\exists_1$. In P. Clote and J. Krajíček, editors, *Arithmetic Proof Theory and Computational Complexity*, pages 30–49. Oxford University Press, 1993.

[Gan13] Mohan Ganesalingam. *The Language of Mathematics*, volume 7805 of *Lecture Notes in Computer Science*. Springer, 2013.

[GG13] M. Ganesalingam and W. T. Gowers. A fully automatic problem solver with human-style output. *CoRR*, abs/1309.4501, 2013.

[Har96] John Harrison. A mizar mode for HOL. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *TPHOLs'96*, volume 1125 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 1996.

[HW60] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford, 4th edition, 1960.

[Kah1x] Reinhard Kahle. What is a proof? 201x. Submitted.

[KP88] Jan Krajíček and Pavel Pudlák. The number of proof lines and the size of proofs in first order logic. *Archive for Mathematical Logic*, 27:69–84, 1988.

[KW08] Fairouz Kamareddine and J. B. Wells. Computerizing Mathematical Text with MathLang. *Electronic Notes in Theoretical Computer Science*, 205:5–30, 2008.

[Lam95] Leslie Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600–608, 1995.

[Lam12] Leslie Lamport. How to write a 21$^{\text{st}}$ century proof. *Journal of Fixed Point Theory and Applications*, 11:43–63, 2012.

[Sco06] Dana Scott. Foreword. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*, pages vii–xii. Springer, 2006.

[Urb06] Josef Urban. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In Michael Kohlhase, editor, *Mathematical Knowledge Management*, volume 3863 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2006.

[Wie06] Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.