The QED manifesto revisited

Freek Wiedijk

Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. We present an overview of the current state of formalization of mathematics, and argue what will be needed to make the vision from the QED manifesto come true.

This short and intentionally provocative paper is dedicated to Andrzej Trybulec. When I first wrote about the Mizar system, Andrzej wrote to me:

> *I have looked to pages that you have prepared.* [...] *I advertise them, even if they are a bit enemical* :-)

I hope that Andrzej will enjoy this paper, and will not consider it to be too 'enemical'.

1 Why the QED manifesto has not been a success (yet)

One of the most interesting documents about the formalization of mathematics is the *QED manifesto* [2]. This anonymous[1] pamphlet from 1994 paints a future in which most of mathematics will be put in the computer, even the proofs – *especially* the proofs – in such a way that the computer will be able to check it for correctness. The QED manifesto describes the development of a system, the QED system, that mathematicians will adopt for this purpose.

The future that the QED manifesto sketches has not happened. There is no such system as the QED system in regular use by mathematicians today.

There are two main reasons that the future from the QED manifesto currently is not much closer than it was in 1994:

– The most important reason is that *only very few people are working on formalization of mathematics.* If there had been a larger research community interested in that subject, a system like the QED system would have been realized long ago.
There are various reasons why not many people are working on the vision from the QED manifesto. For one thing formalization has no 'killer application'. If one writes a formalization of a mathematical result, then one has

[1] One of the main people behind the QED manifesto was Bob Boyer, but for idealistic reasons the authors of the QED manifesto were not identified in it.

to work quite hard, and then at the end one has a `tar.gz` file with several computer program-like files in it. However, unlike a computer program, those files have no immediate further use. The fact that they *fully* describe the mathematics has some aesthetic appeal, and it is nice that they make it completely certain that the mathematics is correct, but the unformalized version of the result already was beautiful and understandable, so not much is gained.

There are people who think that *education* is a good application for this kind of technology. The use of proof assistant would teach students what proof is, and it would allow them to work on proofs in a non-threatening, crystal clear environment. I strongly disagree with this point of view. The use of a proof assistant is a way to keep students busy, but it cannot compete with the more traditional way to do mathematics – which is much easier – to actually make them understand something.

An application that *might* become important at some point is the correctness of computer algebra. Computer algebra systems (the most important being, in order of the number of users: Mathematica [22], Maple [15], MuPAD [3] and Reduce [10]) are notorious for occasionally giving wrong answers. This can be very irritating. Therefore, building an integrated system that combines the automation of computer algebra with the correctness of proof assistants might be seen as a real step forward. If that happens, it could make people start using proof assistant technology without them even realizing it.

An aspect where 'QED-like systems' currently are notoriously bad at is *communication*. A formalization is completely useless for communicating the mathematics that is formalized in it. Here is the most to be gained for formal mathematics. In particular the visual side of mathematics is currently completely absent: there are no such things as graphs and diagrams in current proof assistants.[2]

– The other reason that there has not been much progress on the vision from the QED manifesto is that currently *formalized mathematics does not resemble real mathematics* at all. Formal proofs look like computer program source code. For people who *do* like reading program source code[3] that is nice, but most mathematicians, the target audience of the QED manifesto, do not fall in that class.

  To make things worse there are two choices made by large parts of the formalization community that scare mathematicians away even more. Often proof assistants are *constructive* instead of classical, and often they do proof in a *procedural* way instead of in the more recognizable declarative way [7]. If we want to make some progress of getting people actually to use formal mathematics, it has to be close to the way mathematics already is being

[2] Proof assistants sometimes show diagrams about the structure of the *proof process*, like for instance PVS does, but there never are diagrams about the *mathematics*.

[3] Books that appeal to people like that are for instance Lions' commentary on the Unix kernel source code [13], Donald Knuth's 'TEX: The Program' [12], and John Harrison's forthcoming book on theorem proving [9]. However, proof assistants should appeal to more than just those people.

done for centuries. Improving on tradition is good, but ignoring tradition is stupid. This means that the focus in formal mathematics should be on *classical* & *declarative* systems.[4] From this point of view, among the systems from Section 2 the Mizar and Isabelle/HOL systems – the two systems in that section that are both classical & declarative – are the most interesting.

2 The state of the art in formalization of mathematics

We define the *QED-like systems* to be the five highest ranking systems in an investigation of which theorems already have been formalized from an arbitrary list of 100 well-known theorems.[5] The counts as of March 2007 are:

HOL Light	63
Coq	38
ProofPower	37
Mizar	35
Isabelle/HOL	33
PVS	15
nqthm & ACL2	12
NuPRL & MetaPRL	8

(At that point in time, in all systems together 77 of the 100 theorems have been formalized.) Clearly there is some kind of gap after the fifth system, which is why we put the boundary of being a 'QED-like system' there.

This means that the current QED-like systems are:

- **Mizar** [16, 21]
- HOLs
  - **HOL Light** [8]
  - **Isabelle/HOL** [17, 19]
  - **ProofPower** [14]
- **Coq** [18]

We grouped the three systems from the HOL family together. They have (almost) the same logic and share a very similar architecture.[6]

Projects in the spirit of the QED manifesto have been proposed again and again. For instance, the Automath project from the seventies clearly already was one of

[4] Georges Gonthier, the author of the most impressive formalization in existence today – the formalization of the Four Color Theorem [6] –, disagrees with this opinion. He uses Coq, which is a constructive & procedural system. He claims that a primarily procedural approach is much more efficient than a declarative proof style, and that for that reason the declarative proof style cannot compete.

[5] The current state of this investigation can be found on the web page ⟨`http://www.cs.ru.nl/~freek/100/`⟩.

[6] In one respect they are not similar: unlike the other HOLs, Isabelle/HOL provides a declarative language called Isar, which is quite close in look and feel to Mizar.

these. Here is a list of projects that still are active and that might be considered to aim at implementing the vision from the QED manifesto. We indicate with each project which QED-like system it uses.

- **MML, the Mizar Mathematical Library** (Mizar)
- **John Harrison's work** on the formalization of mathematics (HOL Light)
- **Georges Gonthier's project** in the Microsoft/INRIA institute (Coq)
- **The Archive of Formal Proofs** (Isabelle/HOL)

The first and the fourth are just the 'collected works' of their respective provers. (The 'contribs' of Coq is another of those, but it does not have a fancy name, so I did not put it in the list.) The second and third are currently just the work of one person. However for the third there are plans to extend it into something that is more, in the joint Microsoft/INRIA institute in Paris.

I do not believe that these four projects are already implementing the vision from the QED manifesto. None of these projects have solved the difficult problem of how to integrate work by multiple people into a nice coherent whole. The first and the fourth project have contributions from multiple people, but they currently fail to build a coherent whole out of it.[7] The second and the third projects have a library that is a nice coherent whole, but these libraries do not yet contain contributions by a significant number of people.

3 A benchmark of four statements

Here are four mathematical statements that most mathematicians will consider to be totally non-problematic[8]:

1. $$\int_0^1 \int_0^1 \sum_{n=0}^{\infty} (xy)^n \, dx \, dy = \frac{\pi^2}{6}$$
2. *Every field has an algebraic closure.*
3. *Natural transformations are the morphisms of the functor category.*
4. *PFA (the proper forcing axiom) implies*
$$2^{\aleph_0} = \aleph_2$$

We claim that currently none of the QED-like systems can express all four statements in a good way. Either they do not have good syntax for it (statement 1 in Mizar), or their foundations do not give one enough power to define the notions in these statements in a good way (statement 2 in the HOLs, statement 3 in Mizar, and statement 4 in Coq.) Here is a table that shows which of the QED-like systems have the framework to express which of these four statements:

[7] Try for instance finding a lemma in the MML!

[8] Most of them will probably not know the statement of the proper forcing axiom, but they *will* know the meaning of $\aleph_0$ and $\aleph_2$, and will consider that meaning to be – again – non-problematic.

	Mizar	HOLs	Coq
1. Calculus	−	+	+
2. Abstract algebra	+	−	+
3. Category theory	−	−	+
4. Set theory	+	−	−

Of course one can easily extend the systems with axioms that allow one to write down these statements. However, that really amounts to 'changing the system'. It would mean that both the library and the automation of the system will not be very useful anymore. Classical & extensional reasoning in Coq or abstract algebra in the HOLs by *postulating* the necessary types and axioms will not be pleasant without re-engineering the system.

The first of these statements[9] is from *Proofs from the Book* [1]. There we find the following 'calculation'[10]:

$$\begin{aligned} I &= \int_0^1 \int_0^1 \sum_{n\geq 0} (xy)^n dx\, dy = \sum_{n\geq 0} \int_0^1 \int_0^1 x^n y^n dx\, dy \\ &= \sum_{n\geq 0} \big(\int_0^1 x^n dx \big) \big(\int_0^1 y^n dy \big) = \sum_{n\geq 0} \frac{1}{n+1} \frac{1}{n+1} \\ &= \sum_{n\geq 0} \frac{1}{(n+1)^2} = \sum_{n\geq 1} \frac{1}{n^2} = \zeta(2) \end{aligned}$$

Now the Mizar system has support for 'iterated equalities' like this, but there is no hope of writing this calculation in Mizar in a way that resembles the way it is written in the book. Instead of being able to write

$$\int_0^1 \int_0^1 \sum_{n=0}^{\infty} (xy)^n dx\, dy$$

in Mizar one has to write

$$\begin{array}{lll} \int_0^1 f(y)dy & \textit{where} & f(y) = \int_0^1 g(x,y)dx \\ & \textit{where} & g(x,y) = \sum_{n=0}^{\infty} h(x,y,n) \\ & \textit{where} & h(x,y,n) = (xy)^n \end{array}$$

[9] Incidentally, computer algebra systems like Mathematica and Maple produce this result without trouble. Now why is there no proof assistant that can do the same? Also, note that the sum $\sum_{n=0}^{\infty}(xy)^n$ diverges when $xy \to 1$, so the inner integral is improper for $y = 1$.

[10] After this calculation it then is shown that $I = \frac{\pi^2}{6}$. The goal in *Proofs from the Book* is not to show the statement that we put in the benchmark (that is just an intermediate result there), but to show that $\zeta(2) = \frac{\pi^2}{6}$. However, for the benchmark we wanted an expression with many nested binders, which is the reason why we selected this intermediate equality.

Since Mizar has no user-definable '*binders*', one has to introduce a *name* for every sub-expression. (Another way of saying this is to say that Mizar is too much of a 'first order' system.) This makes doing calculations like the one from *Proofs from the Book* cumbersome in Mizar.

The second of the statements from this section is difficult to state in a nice way in the HOLs. The reason for this is that the HOL type system is too poor. It is not possible in HOL to define a type of *fields* in a uniform way. Therefore, instead of writing the formal equivalent of

For all fields ...

in HOL one has to write

For all fields of which the elements are from this given carrier type ...

The quantification of this carrier type will be implicit. For that reason only universal quantification over these carrier types is possible. Which means that in HOL it is *not* possible to write

There exists a carrier type together with a field with that carrier type ...

Instead one explicitly has to *construct* a carrier type from the already given types. This makes the whole statement clumsy.

Also, one gets a copy of a given field for many different carrier types, and it will need work to navigate these equivalences.

One would hope that with set theory – the foundation of Mizar – one does not have the kind of problem that the HOLs had with the second statement. Unfortunately, this is not true. Mizar has no problems with the statement about fields, but it encounters a similar problem in a different subject.

If one formalizes category theory in set theory, one steps out of the set theoretic universe, and therefore one gets exactly the same kind of problem there that the HOLs have with algebraic structures. Instead of being able to write

For all categories ...

one needs to write

For all small *categories ...*

or maybe

For all categories that are from this given universe ...

In both cases the categories that one talks about will not be proper classes, they will be *small* categories, and therefore the theorems that one proves will not apply to categories like the category of all groups (which is *not* a small category.) This is not acceptable. It means that one cannot talk categorically about the most common kind of algebraic structure.

When one talks about 'natural transformations', one is not only talking about natural transformations where only small categories are involved. Here is a statement that one would like to be able to formalize without making it more complicated than it informally is:

> *Consider the category* **Grp** *of all groups with group homomorphisms as morphisms. The identity functor* $\mathrm{id}_{\mathbf{Grp}} : \mathbf{Grp} \to \mathbf{Grp}$ *is naturally isomorphic to the opposite functor* $\cdot^{\mathrm{op}} : \mathbf{Grp} \to \mathbf{Grp}$.

Here the natural isomorphism that is being talked about is a natural transformation between functors that go from the category of groups to the category of groups. That is, between large categories.

The ideal QED system should allow statements like this without one having to talk (or even to *think*) about universes.

Finally, I have never seen the infinities from set theory – of which $\aleph_0$ and $\aleph_2$ are the first and the third – been formalized in a proof assistant that is based on type theory. For some reason the Coq system has not really been designed to talk about set theoretical notions like the cardinal numbers.

Even worse, also more in general it seems that classical & extensional mathematics is much more difficult in Coq than one would like it to be.[11]

Intermezzo: on constructive proof assistants

A system that implements the QED manifesto should be usable to people who are not aware of the existence of constructive mathematics. The possibility to do constructive mathematics with the system, if at all present[12], should be hidden to people who are not interested in it.

Constructive mathematics seems to be all about being able to point your finger at other people and say 'Oh! They are bad!' It is a very moralistic kind of mathematics. The Coq people point at the NuPRL people and say 'Oh! They are extensional!' ('Type checking of proof terms is not even decidable!') The Agda people point at the Coq people and say 'Oh! They are impredicative!' (Apparently impredicativity is not to be trusted for philosophical reasons.) The other people point at the Agda people and say 'Oh! Their system does not even check termination!' Or maybe someone will say about someone else 'Oh! They use countable choice!' And so on, and so forth.

Instead of trying to prove *as many* true statements as possible, constructive mathematics is about making it *difficult* to prove something. (Of course, *if* you then prove it, the proof contains a bit more information.) Constructivism also takes a strange position with respect to a question like 'does this mathematical statement hold?' The answer often is: 'It depends on how you interpret the statement'; or maybe: 'It depends on what principles you allow yourself.' I guess that this would surprise many classical mathematicians.

Of course for logicians and other philosophers this is all very interesting, but classical mathematicians should not be bothered by these issues. This kind of fine structure of the axiomatics probably does not interest them.

[11] Of course I would not mind being shown wrong here.

[12] Henk Barendregt talks about a *dial* on a proof assistant that allows people to put it in more or less constructive modes. This intermezzo describes some settings on such a dial.

There are various ways in which you can be restrictive:

- First of all you can allow yourself the use of the law of the excluded middle[13] or not. This is the distinction between being *classical* or *constructive*.
- Second, you can be *extensional* or *intensional*. If you are extensional there is *one* equality, that behaves like you would expect equality to behave. If you are not extensional you get into a morass of many different kinds of equality: syntactic identity, convertibility (also called $\beta\delta\iota$-equality), Leibniz equality, John-Major equality[14], setoid equality, etc. etc.
- Third, there is *predicativity*. Predicativity means that you are not allowed to define something by referring to a whole that contains the thing being defined. Although this seems rather clear, logicians do not agree on which systems are predicative and which are not. Some of them claim that the natural formalization of predicativity is a system called *Feferman-Schütte* [11]. Others think that 'inductively generated structures' are also predicative. Here we agree with the second – less strict – opinion, because else no system in our list below would turn out to be predicative.
- Finally there is the *axiom of choice*. Even most classical mathematicians are aware of the fact that one might do mathematics without choice.[15] Now there are various levels of having choice:
  - The weakest version of choice is to take the statement of the axiom of choice
    $$\forall R \left([\forall x \, \exists y \, R(x,y)] \to [\exists f \, \forall x \; R(x, f(x))] \right)$$
    only to give existence of a function f that does *not* have to respect equality. This is called *intensional choice*, and one gets it for free in type theory because of the Curry-Howard-de Bruijn interpretation. (In Coq this is slightly subtle: there this only works for the 'informative' existential quantifier.) In the table below we call this 'weak choice'.
  - Then there is the axiom of choice from set theory. Here one knows that there is a choice function that respects equality, but one cannot necessarily define one explicitly. This axiom is called *extensional choice*. Constructivists think that this axiom is a confused version of their intensional choice axiom.
  - Finally there is the existence of a *Hilbert choice operator* ϵ, an operator that returns a specific element for any given non-empty set.
    In the table below we call both of these last two properties 'strong choice'.[16]

[13] $p \vee \neg p$.

[14] Also called 'heterogeneous equality'. This is a typed equality in which one allows the terms that are being compared to have different types. Heterogeneous equality generally is a good idea. The equality of the HOLs is homogeneous, but the equality of Mizar is heterogeneous.

[15] If you believe that the union of countably many countable sets is countable, then you do believe in choice.

[16] There are no systems in the table that have extensional choice but do not have a Hilbert choice operator.

Here is the table that shows which systems satisfy which restrictions:

	Mizar	HOLs	Coq CIC	NuPRL	Agda ITT	IZF	CZF
Classical	+	+	−	−	−	−	−
Impredicative	+	+	+	−	−	+	−
Extensional	+	+	−	+	−	+	+
Weak choice	+	+	+	+	+	−	−
Strong choice	+	+	−	−	−	−	−

Surprisingly there is no 'bottom' proof assistant that has minuses *everywhere*, one that both does not support extensionality nor any form of the axiom of choice.[17]

4 A criticism of current systems

At one of the meetings of the TYPES project, Georges Gonthier was mentioning a paper from the Mizar community to me in which Mizar was described as a 'state of the art proof assistant'. This seemed to amuse him. As Mizar is certainly in the same ball park as the proof assistant that he uses himself – Coq – I asked him which of the proof assistants in his opinion *are* 'state of the art'. His answer was that 'there is no state of the art proof assistant yet.'

I agree that the QED-like systems that exist today are not good enough to start developing a library as is described in the QED manifesto. I will for each of the three classes of the QED-like systems indicate what in my opinion are their most important weaknesses:

Why Mizar is not acceptable as the QED system yet.

There are three major reasons why Mizar is not yet attractive enough to be taken to be the QED system:

- In the previous section I already addressed the problem that Mizar has no way to define binders. This means that common operations from calculus, like $\sum_{x=a}^{b} f(x)$, $\lim_{x\to c} f(x)$, $\frac{d}{dx} f(x)$, $\int_a^b f(x)\,dx$, etc., can not be properly written down without naming sub-expressions.
- An aesthetic flaw of Mizar that is not very important but that *is* irritating, is that Mizar does not support empty types.[18] For this reason the Mizar library MML has many lemmas that are restricted to non empty sets, while there really is no good mathematical reason for that.
  Also it means that many natural mathematical notions cannot be expressed as a Mizar type. For instance it is not possible to talk about 'the

[17] The 'logic-enriched type theories' of Peter Aczel [5] are designed to remove the Curry-Howard-de Bruijn interpretation from intensional type theory. However, currently this seems primarily to be used as a logical framework, and not yet as a foundational system for mathematics.

[18] For a type theorist this restriction to non empty types is *very* strange.

normal form of a term in a given term rewriting system' as a Mizar type (because some terms might not *have* a normal form.)
- Finally, and I do not know how important this is but I expect it to be very important, Mizar does not support 'user automation'. If you, as a user of the system, know a way to solve a certain class of mathematical problems algorithmically, it is not possible to 'teach' this to the system. The only people who can do that kind of automation are the developers of the Mizar system itself, who can add so-called '*requirements*' to the system. This 'closed' architecture is too restrictive.

Why the HOLs are not acceptable as the QED system yet.

There is one important reason why the HOLs are not yet attractive enough to be taken to be the QED system:
- The HOL type system is too poor. As we already argued in the previous section, it is too weak to properly do abstract algebra.
  But it is worse than that. In the HOL type system there are no *dependent types*, nor is there any form of *subtyping*. (Mizar and Coq both have dependent types and some form of subtyping. In Mizar the subtyping is built into the type system. In Coq a similar effect is accomplished through the use of automatic *coercions*.)
  For formal mathematics it is essential to both have dependent types and some form of subtyping.

Why Coq is not acceptable as the QED system yet.

There are two important reasons why Coq is not yet attractive enough to be taken to be the QED system:
- The foundations of Coq are too complicated. They are baroque to say the least. Maybe they are even *beyond* baroque. They might even be called *rococo*.
  There is no paper in which the foundations of Coq are spelled out in full mathematical precision. Bob Solovay told me that this was one of the reasons for him to lose interest in Coq as a system.
  Also, apparently giving a set theoretical interpretation to the foundations of Coq in all its details – to show Coq's consistency relative to some set theory – might not be as easy as it sounds.
  Also, the foundations of Coq are sufficiently complicated that they are tinkered with, and therefore change between versions of the system.
- As already noted in the previous section, Coq is not designed for classical mathematics, which means that doing classical & extensional mathematics in it is not as easy as one would like it to be.
  At the very least one would need to add some axioms for that. For instance axioms that would be needed are:
  - excluded middle
  - the K axiom[19]

[19] The K axiom states that if two dependent pairs $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$ are equal , then y_1 has to be equal to y_2. In the Coq logic this is not provable. (It *is* provable that x_1 has to be equal to x_2.) 'Dependent pairs' means that the types of the y_i depend on the x_i.

- axioms for extensionality/quotient types
- choice

and it is not very clear when enough axioms like that will have been added.

5 What has to be done?

To realize the vision from the QED manifesto, three things need to happen:

– First of all, *a project should be started that is not based on the expectation that other people will do the work.*
Currently people often just build a system and then hope that other people will start using it for formalization of mathematics. The QED manifesto is an even more extreme version of this: it just presents a *vision*, and then hopes that other people will *both* make a system and start using it for formalization of mathematics. That approach does not work. If you want something accomplished, you should work on it yourself.
An important aspect of this project should be that it should involve a *small* number of people,[20] and have a clear focus. Only that way there will be enough coherence to get a good result.
It seems important to focus on formalization of actual mathematical practice, and *not* also try to 'improve' on the way that people currently do mathematics.
– In this project, *a good system for formalization of mathematics should be developed.*
As I argued in the previous section, the current crop of systems is not yet good enough to be the 'QED system' in which to build a library of all of mathematics.
Ideally, the system should be set up in such a way, that the mathematics that is formalized in it can survive a 'redesign of the foundations'. Suppose that one builds a QED-like library on top of Coq, and then changes one's mind and switches to a HOL-like system. Basically, this will mean that one will have to start from scratch.[21] It seems very arrogant to think that we already know what the best foundations for our formal library should be, and one would therefore prefer not to be 'locked in' to a specific version of the foundations.
This is one more argument for having a declarative proof style over a procedural proof style. Proofs in a declarative style are much more 'robust' with

[20] The original Unix system was created by only two people (who shared an office at the time), Ken Thompson and Dennis Ritchie. Starting something very good does not need a large group of people.

[21] Even if one does not mind having to start from scratch like that (taking that as the way that science progresses), it still seems sensible to work with foundations that are not *too* idiosyncratic.

respect to changes in foundation than proofs in a procedural style. An argument for this is the close similarity between the Mizar and Isar[22] proof languages, despite the wildly different foundations of their underlying systems.

I do *not* believe that the QED system will consist of many different systems living peacefully together. One of the systems – hopefully the best one – will kill the others. That is how evolution works. It is this system that should be built.[23]

– Finally, *a multi-contributor library of formal mathematics should be created that is well-organized.* As I claimed in Section 2, the sociological puzzle has not been solved yet of how to set up a library in such a way that it both admits many contributors, but also stays well-organized. Current attempts that have not yet been very successful in this respect are:
  - *multi-contributor, but not really well-organized:*
    * MML
    * Coq contribs
    * AFP
  - *well-organized, but not really multi-contributor:*
    * HOL Light library
    * C-CoRN [4]

  A good first goal for this library will be to formalize 'all of undergraduate mathematics'. This will take more than a hundred man-years [20], which means that it will need the involvement of a non-trivial number of contributors. Also, with something like that it will be clear whether it is well-organized or not.

  The success of Wikipedia suggests that it might be possible to solve the puzzle of having *both* many contributors and still also have a well-organized whole. However, aiming for a 'Wikipedia for formalized mathematics' is for me too much like hoping that someone else will do the work. I will believe that such an approach can be successful when I see it happening.

6 What will happen?

Henk Barendregt always asks people when they expect the future from the QED manifesto to arrive. He tells me that the more experience people have with proof formalization, the more pessimistic they are about this. (The answers seem to range from 'it already is here!' to 'in about fifty years.')

I myself certainly believe that the QED system will come. If we do not blow up the world to a state that mathematics will not matter much anymore, then at some point in the future people will formalize most of their proofs routinely in the computer. And I expect that it will happen earlier than we now expect.

So I *do* believe that in a reasonable time

[22] Isar is the proof language of the Isabelle/HOL system [19].

[23] From that point of view, spending time on having different systems be able to communicate their mathematics is lost energy.

- a proper system will be created
- a proper basic library for this system is made
- a proper infrastructure is set up for keeping this library well-organized despite it having many contributors

I expect that if the first and third of these items arrive, then the second of these items – the QED library – will flow from that in a natural way.

Acknowledgments. Thanks to Jacques Carette for interesting discussions about the relation between formal mathematics and computer algebra. Thanks to Bas Spitters for explaining to me that there is no consensus on what the word 'predicative' means, by pointing out that there are people who claim that one can predicatively prove Kruskal's theorem. Finally, thanks to Michael Beeson, John Harrison and Bob Solovay for helpful comments on a draft of this paper.

References

1. M. Aigner and G.M. Ziegler. *Proofs from the Book*. Springer-Verlag, second edition, 2001.
2. R. Boyer et al. The QED Manifesto. In A. Bundy, editor, *Automated Deduction – CADE 12*, volume 814 of *LNAI*, pages 238–251. Springer-Verlag, 1994. ⟨`http://www.cs.ru.nl/~freek/qed/qed.ps.gz`⟩.
3. C. Creutzig and W. Oevel. *MuPAD Tutorial*. Springer-Verlag, second edition, 2004.
4. L. Cruz-Filipe, H. Geuvers, and F. Wiedijk. C-CoRN: the Constructive Coq Repository at Nijmegen. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Mathematical Knowledge Management, Proceedings of MKM 2004, Białowieza, Poland*, volume 3119 of *LNCS*, pages 88–103. Springer-Verlag, 2004.
5. N. Gambino and P. Aczel. The generalised type-theoretic interpretation of constructive set theory. *Journal of Symbolic Logic*, 71(1):67–103, 2006.
6. G. Gonthier. A computer-checked proof of the Four Colour Theorem. ⟨`http://research.microsoft.com/~gonthier/4colproof.pdf`⟩, 2006.
7. J.R. Harrison. Proof style. In Eduardo Giménez and Christine Paulin-Möhring, editors, *Types for Proofs and Programs: International Workshop TYPES'96*, volume 1512 of *LNCS*, pages 154–172, Aussois, France, 1996. Springer-Verlag.
8. J.R. Harrison. *The HOL Light manual (1.1)*, 2000. ⟨`http://www.cl.cam.ac.uk/users/jrh/hol-light/manual-1.1.ps.gz`⟩.
9. J.R. Harrison. *Introduction to Logic and Automated Theorem Proving*. To be published by Cambridge University Press, 2007.
10. A.C. Hearn. *Reduce, User's Manual Version 3.6*. Santa Monica, CA, 1995.
11. K. Schütte. *Proof theory*. Springer-Verlag, 1977.
12. D.E. Knuth. *TEX: The Program*. Addison Wesley, 1986.
13. J. Lions. *Lions' Commentary on UNIX 6th Edition*. Peer-to-Peer Communications, San Jose, CA, 1977.
14. Lemma 1 Ltd. *ProofPower – Description*. Lemma 1 Ltd., 2000. ⟨`http://www.lemma-one.com/ProofPower/doc/doc.html`⟩.
15. M. Monagan, K. Geddes, K. Heal, G. Labahn, and S. Vorkoetter. *Maple V Programming Guide for Release 5*. Springer-Verlag, Berlin/Heidelberg, 1997.

16. Michał Muzalewski. *An Outline of PC Mizar*. Fondation Philippe le Hodey, Brussels, 1993. ⟨http://www.cs.ru.nl/~freek/mizar/mizarmanual.ps.gz⟩.
17. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. ⟨http://www.cl.cam.ac.uk/Research/HVG/Isabelle/dist/Isabelle2004/doc/tutorial.pdf⟩.
18. The Coq Development Team. *The Coq Proof Assistant Reference Manual*, 2006. ⟨http://pauillac.inria.fr/coq/doc/main.html⟩.
19. M. Wenzel. *The Isabelle/Isar Reference Manual*. TU München, 2002. ⟨http://isabelle.in.tum.de/doc/isar-ref.pdf⟩.
20. F. Wiedijk. Estimating the Cost of a Standard Library for a Mathematical Proof Checker. ⟨http://www.cs.ru.nl/~freek/notes/holl2coq.ps.gz⟩, 2002.
21. F. Wiedijk. Writing a Mizar article in nine easy steps. ⟨http://www.cs.ru.nl/~freek/mizar/mizman.ps.gz⟩, 2007.
22. S. Wolfram. *The Mathematica book*. Cambridge University Press, Cambridge, 1996.