Revisions as an Essential Tool to Maintain Mathematical Repositories

Adam Grabowski[1] and Christoph Schwarzweller[2]

[1] Institute of Mathematics, University of Białystok
ul. Akademicka 2, 15-267 Białystok, Poland
adam@math.uwb.edu.pl

[2] Department of Computer Science, University of Gdańsk
ul. Wita Stwosza 57, 80-952 Gdańsk, Poland
schwarzw@math.univ.gda.pl

Abstract. One major goal of Mathematical Knowledge Management is building extensive repositories, in which the mathematical knowledge has been verified. It appears, however, that maintaining such a repository is as hard as building it – especially for an open collection with a large number of contributors. In this paper we argue that even careful reviewing of contributions cannot cope with the task of keeping a mathematical repository efficient and clearly arranged in the long term. We discuss reasons for revisions of mathematical repositories accomplished by the "core implementors" and illustrate our experiences with revisions of MML, the Mizar Mathematical Library.

1 Introduction

Mathematical knowledge management aims at providing both tools and infrastructure supporting the organization, development, and teaching of mathematics using computers. Large repositories of mathematical knowledge are here of major concern since they provide users with a base of verified mathematical knowledge. We emphasize the fact that a repository should contain verified knowledge only: we believe that (machine-checkable) proofs necessarily belong to each theorem and therefore are an essential part of a repository. For repositories in the sense of Mathematical Knowledge Management community this implies even more: proofs should not only be understandable for the machine, but also – for human users of the repository.

From this follows that mathematical repositories are more than collections of theorems and proofs accomplished by a prover or proof checker. The overall goal is not proving a theorem – though this still is an important and challenging part – but presenting definitions and theorems so that the "natural" mathematical buildup remains visible. Theories and their interconnections must be available, so that further development of the repository can be based upon these. Being not trivial anyway, this becomes even harder to assure for an open repository with a large number of authors.

So, how to tackle this task? One possibility, of course, is reviewing submissions. Reviewing improves the quality of knowledge and proofs added to the repository, but we shall illustrate that in the long run reviewing cannot ensure that a mathematical repository meets the demands mentioned above. We therefore claim that revisions are an essential part of maintaining mathematical repositories: in order to keep it clean and attractive for users, from time to time a "core team" has to check and improve the organization, quality, and proofs of a mathematical repository.

In the following section we describe and discuss the goals and benefits of revisions compared to a straightforward reviewing process. Then, after a brief introduction to the Mizar system [11], we consider the reviewing process for submissions to MML in Section 3. We describe reviewing criteria and show which insufficiencies can be handled by reviewing. In contrast, Section 4, is devoted to revisions of MML illustrating on the one hand what kind of improvements reviewing cannot perform, on the other hand the role of revisions in maintaining MML. This is the process done mainly by human hand as of now, in the next section we discuss some issues concerned with this activity and describe some traps the developer could meet when enhancing the library. Then we conclude showing some related work and drawing some remarks for future.

2 The Need for Revisions

The goal of a revision is to improve the mathematical repository. In contrast to reviewing submissions, however, here the attention is turned to the repository as a whole, not to a single, new part of it. Consequently, motives for revisions can be for example:

- keeping the repository as small as possible,
- preserving a clear organization of the repository in order to attract authors,
- establishing "elegant" mathematics, that is e.g. using short definitions (without unnecessary properties) or better proofs.

Note that all these points characterize a qualitative repository and can hardly be achieved by reviewing single submissions. Of course there are different possibilities to achieve the points mentioned. Improving the prover e.g. can shorten proofs and hence – simplify the repository. (Re-)organizing a mathematical repository probably demands manipulating the whole file structure, not only the files themselves. Therefore we decided to classify revisions based on their occasion, that is on which kind of insufficiency we want to address. Based on our experiences with the Mizar Mathematical Library we distinguish four major occasions for revisions:

1. improving authors' contributions;
2. improving the underlying prover or proof checker;
3. reorganizing the repository;
4. changing representation of knowledge.

Improving an author's contribution is the classical task of reviewing and is of course to be recommended for mathematical repositories too: nomenclature can be polished up to fit to the yet existing one, definitions can be improved, that is e.g. generalized if appropriate. Proofs are also a matter of interest here, especially keeping them as short as possible, yet still understandable is of major concern. In a large, open repository however, authors sometimes may prove and submit theorems or lemmas not being aware that those are already part of the repository. Similarly, special versions of already included theorems can happen to be "resubmitted". It is doubtful, that this kind of flaws will be detected by ordinary reviewing.

Strenghtening the underlying prover or proof checker has also an impact on the repository. Proofs can be shortened or rewritten in a more clear fashion, both being fundamental properties of attractive mathematical repositories. Even more, theorems in such collection may now be superfluous, because the improved prover accepts and applies them automatically. A typical example here is the additional inclusion of decision procedures.

Reorganizing the repository deals with the fact that a repository is built up by a large number of contributors. For their development authors (should) use already existing theories as a basis. To establish their main results, however, they often have to prove additional theorems or lemmas just because the theory used does not provide them yet. So, these additional facts have to be put in the right place of the repository. Otherwise, it will be hard for other authors to detect them or at least searching the repository becomes less comfortable. In the same direction goes the building of monographs: a frequently used theory should be handled with extra care. Not only should all related theorems be collected in a distinguished place, but also still lacking theorems be complemented, in order to ease working for further authors. These tasks can only be accomplished when considering the repository as whole, that is by revisions.

The last point concerns the development of a repository in the long term. What if after while it turns out that another definition or representation of mathematical objects would serve our purposes better than the one chosen? Should it be changed? Note that a lot of authors already could have used these objects in their proofs, that is changing the definition or representation would imply changing all these proofs – and of course one cannot force authors to redo all their proofs. On the other hand, including both definitions or representations leads to an unbalance: the theory of the new prefered version is much less developed than the one of the old version, so authors hardly will base their developments on the new one. Again, the solution is a revision: In the best case definitions and representations are changed by a "core team", so that ordinary users can furthermore use all theorems without even noticing they changed.

In the following sections we will illustrate these considerations by examples taken from the Mizar Mathematical Library and in particular show how revisions maintain mathematical repositories.

3 Review of MML Submissions

Reviews of submissions to the MML – as reviews of ordinary submissions for conferences or journals – have the overall goal to check whether a submission should be accepted (for inclusion into the MML) and simultaneously improve the quality of a submission. For mathematical repositories, however, the criteria for acceptance and improvements are somewhat different.

Certainly the contents of a submission for a repository should likewise be original and interesting. Original here, of course, means that definitions and theorems presented are not part of the repository, yet. This is easy to check for the main theorem of a submission. For technical lemmas used to establish this main result, however, this task is much more difficult. So, for example, a reviewer will probably neither know nor be willing to check whether a theorem like

```
theorem
  for F,G being FinSequence, k being Nat st
    G = F|(Seg k) & len F = k + 1 holds F = G ^ <*F/.(k+1)*>;
```

is already included in a repository. Even if the textual search via grepping is no longer the only method to find such repetitions since the MML Query by Grzegorz Bancerek [3] is available, even after the volunteer will learn how to use this system, still there is no single automated bunch of tools which removes all repeated theorems effectively. Furthermore, the motivation to check these things in detail will be even decreased, because such a point will not decide between acceptance and rejection.

The question whether a submission is interesting should be handled more liberally. Of course, the usual issues, that is the quality of the main results, apply here also. There is, however, another kind of submissions to repositories: the one that deals with the further development of (basic) theories. This concerns collections of basically simple theorems providing necessary foundations so that more ambitous developments can be easier accomplished. Usually, these are theorems, that easily follow from the definitions, however are so often used, that repeating the proof over and over again is hardly acceptable. Examples here are the theories of complex numbers or polynomials, where among other things we can find the following theorems.

```
theorem
  for a,b,c,d being complex number st
    a + b = c - d holds a + d = c - b;

theorem
  for n being Ordinal,
      L being add-associative right_complementable add-left-cancelable
          right_zeroed left-distributive (non empty doubleLoopStr),
      p being Series of n,L holds
   0_(n,L) *' p = 0_(n,L);
```

Though hardly interesting from a mathematical point of view, such theorems are important for the development of a repository and should therefore be considered as interesting, too.

Improvements of a submission are a more difficult issue. Firstly, we can consider definitions and notations contained in the submission. Can they be arranged more sparsely, that is can the results be established based on fewer axioms? Is it possible or reasonable (in the actual repository) to generalize the definitions? This also applies to theorems. Note that the theorem from above, though applicable to polynomials, in fact is stated for power series. Again to address these issues a high knowledge of the repository by the reviewers is necessary.

Secondly, when it comes to proofs, there are hardly any guidelines, because proving in particular is a matter of style. We can hardly force an author to change his (finished) proof into another one using completely different proof techniques. What we can do, is to suggest improvements for the presented proof. We can, for example, propose a more accurate use of the proof language to get more elegant or better readable proofs. Or we can give pointers to other theorems in the repository that allow to shorten the proof.

Based on these considerations a reviewing process for Mizar articles, that is for submissions to the Mizar Mathematical Library, has been introduced. Using basically the commonly used scheme accept/revise/reject (and apart from its descriptive grade) the rating of a submission can be[3]

A. accept
 requires editorial changes only, which can be done by the editors
B. accept
 requires changes by the author to be approved by the editors
C. revise
 substantial author's revisions necessary, resubmission for another review
D. decision delayed
 revision of MML necessary
E. reject
 no hope of getting anything valuable

The most important issue here, of course, is the question whether an article should be included in MML. Note, that there are two grades (A and B) for acceptance. The reason is that accepted articles should be included in the MML as soon as possible to avoid duplication of results during the reviewing phase.[4] So, while submissions rated B or C need feedback from the authors, submissions rated A can be added to MML without further delays.

The most interesting point is D. Note that here already the problem of a revision of the whole repository is addressed. Reviewing can point out that – though the author has proven his main results – the way Mizar and MML support establishing the presented results is not optimal and should be improved.

As the most notable example here, we can cite the newly submitted definition of a kind of a norm for the elements of the real Euclidean plane, which are defined in the Mizar library just as finite sequences of real numbers.

[3] There has been and is still going on an email discussion about these options.

[4] There even has been the proposition of making public submissions before reviewing to avoid this problem, but we are not aware of a definite decision concerning this point.

```
definition let n be Nat, f be Element of TOP-REAL n;
  func |. f .| -> Real means
    ex g being FinSequence of REAL st
      g = f & it = |. g .|;
end;
```

where `|. g .|` is a usual Euclidean norm which was introduced in the MML before as

```
definition let f be FinSequence of REAL;
  func |. f .| -> Real equals
:: EUCLID:def 5
    sqrt Sum sqr f;
end;
```

After the change of the loci type (the submission obtained grade D, of course) from `FinSequence of REAL` into `real-yielding FinSequence` in the `EUCLID` article the earlier definition was no longer needed which helped to simplify the structure of notions in this new submission.

The decision is not a typical result of majority voting, because referees giving C grade point out possible improvements, so usually the lowest grade counts (luckily, in case of E marks, all three referees agreed).

To summarize the grades for 2006, let us look at Table 1.

Table 1. Number of submissions to the MML and their grades in 2006

	all	A	B	C	D	E
items	39	6	4	20	6	3
% of total	100	15.4	10.2	51.3	15.4	7.7

Basically, all ten submissions graded A and B were included into the MML, and among C and D candidate articles, which were returned to authors, other 15 were accepted; in total there were 25 Mizar articles accepted in 2006, the first year the reviewing procedure as described above was introduced.

All in all we have seen that reviewing MML submissions indeed addresses only the first point mentioned in Section 2. Of course a thorough reviewing process will improve the quality of MML articles and may even pilot authors into the direction of a good style of "Mizar writing". As we can conclude from Table 1, this is the case of the majority of submissions because the authors should enhance the articles according to the referees' suggestions. However there remain situations in which the MML as a whole should be improved; in the long term mere reviewing of submissions cannot avoid this. Here even carefully reviewing of Mizar articles – as already indicated by rate D above – can only help to detect the need for such revisions.

4 MML Enhancing

The Library Committee has been established on November 11, 1989. Its main aim is to collect Mizar articles and to organize them into a repository – the MML. Recently, from this agenda a new additional one was created – the Development Committee, which takes care of the quality of the library as a whole.

4.1 Types of Revisions

For the reasons we tried to point out before, the Mizar Mathematical Library is continuously revised. Roughly speaking, there are different kinds of revisions:

– an authored revision – consists of small changes in some articles in the library when somebody writing a new article notices a theorem or a definition in an old article that can be generalized. This is also the case of D grade as described above. To do this generalization, sometimes it is necessary to change (improve) some older articles that depend on the change. As a rule, a small part of the library is affected.
– an automatic revision – takes place frequently whenever either a new revision software is developed (e.g. software for checking equivalence of theorems, which enables to remove one or two equivalent theorems) or the Mizar verifier is strengthened and existing revision programs can use it to simplify articles.
– a reorganization of the library – although was very rare before, as of now it happens rather frequently. It consists in changing the order of processing articles when the Mizar data base is created. Its main steering force is the division of the MML into concrete and abstract parts.

4.2 MML Versions

Apart of the Mizar version numbering, the MML has also separate indexing scheme. As of the end of 2006, the latest official distribution of the MML has number 4.76.959.

As a rule, the last number, currently 959 shows how many articles are there in the library (this number can be sometimes different because 26 items were removed so far from the MML, but some additional items such as EMM articles, and "Addenda" which do not count as regular submissions, were added). The second number (76) changes if a bigger revision is finished and the version is made official. Although it is relatively small comparing with the age of the library, the changes are much more frequent.

4.3 Some Statistics

The policy of the head of Library Committee – to accept virtually all submissions from the developers and, if needed, enhance it by himself, was then very liberal. For these nearly twenty years there were only three persons taking a chair of a head of the Committee (Edmund Woronowicz, Czesław Byliński, and currently Adam Grabowski); their decisions were usually consulted with the other members of the committee, though.

Such an openness of the repository was justified: in the early years of the Mizar project the policy "to travel to Białystok and to get acquaintance with the system straight from its designer" resulted in the situation all authors knew each other personally, now the situation changed.

The MML evolved from the project, frankly speaking, considered rather an experiment of how to model mathematics to allow many users benefit from a kind of parallel development. Now, when the role of the library is to be much closer to the reality and the MML itself is just one among many mathematical repositories, the situation is significantly different.

Table 2. Submissions to the MML by year

Year	Add.	1989	90	91	92	93	94	95	96	97	98	99	2000	01	02	03	04	05	06
Articles	19	65	136	46	48	33	33	35	57	39	47	65	54	33	42	54	80	48	25

As it can be seen, the first two years were extremely fruitful. No doubt, the first one was most influential, when the fundamentals, such as basic properties of sets, relations, and functions, the arithmetics, and vector spaces were established – to enumerate among many these most important. Some articles from that time were more or less straight translation from those written in older dialect of the Mizar language (Mizar PC, Mizar-2 etc., see [9]). Especially the subsequent year – 1990, when many authors could benefit from introducing the basics, hence they were able to work on various topics in parallel, brought into the Mizar Mathematical Library the bigest number of submissions so far (136 by year), then the number stabilized.

4.4 Towards Concrete and Abstract Mathematics

As it was announced in 2001 [15], the MML will be gradually divided into two parts. As the library is based on the Tarski-Grothendieck set theory, the part devoted to the set theory (and related objects, as relations, functions, etc.) is indispensable. There is, however, huge amount of knowledge for which set theory is essential, but basing on the notion of structure by means of the Mizar language.

There are three parts of the Mizar Mathematical Library:

- concrete, which does not use the notion of structure (here of course comes standard set theory, relations, functions, arithmetic and so on);
- abstract, i.e. `STRUCT_0` and its descendants, operating on the level of Mizar structures; both parts are not completely independent – here the concrete part is also reused (abstract algebra, general topology including the proof of the Jordan Curve Theorem, etc.);

– SCM, the part Random Access Turing Machines are modelled, i.e. mathematical model of a computer is described.

This division is reflected in the file `mml.lar` in the distribution in which is the order of processing of articles when creating Mizar data base is given – the "concrete" articles go first, at the end are those devoted to the SCM series. The process of separating these three parts is very stimulating for the quality of the Mizar library – many lemmas are better clustered as a result of this activity.

As of the beginning of 2007, the division can be summarized in Table 3.

Table 3. Three parts of the MML

Part	Number of articles	% of total
concrete	266	27.70
abstract	640	66.67
SCM	54	5.63
Total	960	100.00

Note that apart of the revisions suggested by the referees when giving D-grades, any user can via TWiki mechanism suggest the change; he may of course also send improved version via email to the Library Committee; as an example, lemmas needed for the Gödel Completeness Theorem were reformulated to provide its better understanding as a result of the external call.

4.5 Library Management

As a first tool of collaborative work on the library we can enumerate Mizar TWiki (`wiki.mizar.org`) which gradually changes its profile from an experimental – and rarely used – forum into the place where suggestions/experiences with the MML can be described.

As the most important, and probably one of the better known MML tools, we can point out MML Query [3]. It has proven its feasibility when subsequent EMM items were created. Also researchers, when writing their Mizar articles, can find it useful. But usually, typical author does not care too much if his lemma which takes some ten lines of Mizar code is already present in the library. Actually, searching for such auxiliary fact can take much more time than just proving it. This results in many repetitions in the library MML Query cannot cope with. And although the author can feel uncomfortable with multiple hits of the same fact, annotating such situations and reporting it to the MML developers is usually out of his focus.

This is the area where another tool comes in handy. Potentially very useful for the enhancement of the MML as a whole, MoMM (Most of Mizar Matches) developed by Josef Urban was primarily developed to serve as assistant during authoring Mizar articles [18]. It is a fast tool for fetching matching theorems, hence existing duplications can be detected and deleted from the MML (according to [18], more than two percent of main Mizar theorems is subsumed by the others). The work with the elimination of these lemmas is still to be done – many of detected repetitions are useful special cases so their automatic removal is at least questionable.

Another popular software, MML CVS – the usual concurrent version system for the MML was active for quite some time, but then was postponed because the changes were too cryptic for the reader due to the lack of proper marking of items. Actually, one of the most general problems is that there are no absolute names for MML items and the changes are usually too massive to find out what really matters.

5 Traps for the Developer

Usually, the revision process via generalization of notions improves the MML. There are some dangerous issues, however; we address some of them in this section.

5.1 Mind the Gap!

Let us cite an example from the article `ABIAN` [14]:

```
definition let i be Integer;
  attr i is even means
:: ABIAN:def 1
    ex j being Integer st i = 2*j;
end;
```

These are usual definitions of odd and even integer numbers.

```
definition let i be Integer;
  attr i is odd means
    ex j being Integer st i = 2*j+1;
end;
```

Then, among the others, the theorem stating that all integer numbers are either odd or even, was proven; the proof was very simple, but there was something in it to do, at least both definitions were involved.

```
theorem LEM:
  for i being Integer holds i is odd or i is even;
```

However, after the revision (which was done by the author of `ABIAN`, after all), the second definition got simplified as follows:

```
notation let i be Integer;
  antonym i is odd for i is even;
end;
```

Still, it seems perfectly correct, introducing antonym we obtain the law of excluded middle automatically, in a sense, and the proof of the above lemma labeled `LEM` was no longer necessary.

But we made a step too far, as it seems. Because in the definition of even number, the `integer` was not needed (remember the type `Integer` is a shorthand for `integer number`), it was dropped in both – definition of an attribute and its antonym, and the latter got simplified into the form:

```
notation let i be number;
  antonym i is odd for i is even;
end;
```

Unfortunately, e.g. number `Pi` (the Mizar symbol for the usual constant π) can be proven to be odd which can be considered really odd. Observe that any automation of the process of dropping assumption about the types of used loci in the definition of attributes, however possible, could be dangerous.

5.2 Permissive Definitions

There are two unities for vector spaces defined in the MML – with symbol `1.` and `1_`, and the following definitions:

```
definition let FS be multLoopStr;
  func 1.FS -> Element of FS equals
:: VECTSP_1:def 9
    the unity of FS;
end;

definition let G be non empty HGrStr such that G is unital;
  func 1_G -> Element of G means
:: GROUP_1:def 5
    for h being Element of G holds h * it = h & it * h = h;
end;
```

where `multLoopStr` is `HGrStr` enriched by an additional selector, namely `unity` and the adjective `unital` in the permissive assumption (after `such that`) assures that the proper neutral element exists.

At first glance, the earlier approach is better – the less complicated a type in a locus, the less problems we have to assure the required type. In the second definition however, the underlying structure has only two selectors instead of three.

5.3 Meaningless Predicates

Suppose we have the following:

```
definition let a,b be natural number;
  assume a <> 0;
  pred a divides b means
    ex x being natural number st b = a * x;
end;
```

Well, we can freely delete the assumption, in Mizar predicate definitions do not require any correctness conditions proven. But, if we forget for a while that within the MML division by zero *is* defined, does it make any mathematical sense for *any* pair of natural numbers? According to the current policy of the Library Committee, we allow for any such underspecification.

5.4 Apparent Generalizations

There are sometimes cases the price for the revision is too high comparing to gains or the enhancing is apparent. If we remind that pathwise connectedness of the topological space T denotes the existence of a function from the unit interval into T which has the values a and b in 0 and 1, respectively, for any pair of points a, b of T, *a path* between two points is just an underlying mapping, if it exists. It is enough however to have an assumption about the existence of appropriate function for just the pair of points currently under considerations.

```
definition let T be TopStruct; let a, b be Point of T;
  assume a, b are_connected;
  mode Path of a, b -> Function of I[01], T means
:: BORSUK_2:def 2
    it is continuous & it.0 = a & it.1 = b;
end;
```

With no doubt, the assumption of the existence of a path not for arbitrary, but just for these two fixed points is more general, the gains from stating every now and then that considered two points can be connected by a path, are at least doubtful. Similarly, we often write `Abelian add-associative right_zeroed right_complementable RealLinearSpace-like (non empty RLSStruct)` dropping an attribute or two to have slightly more general setting instead of using the mode `RealLinearSpace` which is equivalent to that complicated string above.

6 Related Work

Contemporary standards of the publishing process open some new possibilities – there are many journals online, Springer also announces his books/proceedings at their webpage. Paper-printed editions have some obvious limitations, vanishing for electronically stored and managed repositories of knowledge. We can notice, as an example, new functionalities of [10] in comparison to (even online) version of Abramowitz and Stegun [1].

Of course, what is published on paper, is fixed. We can mind some real-life situations – rough sets as an example of obtaining new results via a kind of revision process (originally considered to be classes of abstraction with respect to some equivalnce relation, then some of its attributes were dropped to generalize the notion – see [6], [7]); also Robbins algebras and related axiomatizations of algebras are a good example, when a classical problem could be rewritten and reused when solved. In the aforementioned examples these were subjects for another papers, within the computerized repository the enhancement (the generalization of results) can be obtained via revision process.

As a rule, building an extensive encyclopedia of knowledge needs some investment; on the one hand, it can be considered by purely financial means as "information wants to be free, people want to be paid" [2]. That is the way Wolfram MathWorld [19] has been raised, as a collection closed in style, in fact authored by one person, Eric Weisstein.

But right after this service has been closed due to the court injunction, it soon appeared that the need to bridge this gap is that strong – many volunteers were working to develop a concurrent service to that of Wolfram's, but of the more open type, based on the mechanism similar to Wiki.

The effort of PlanetMath is now a kind of Wikipedia for mathematics (in fact they even cooperate closely); with its content somewhat questionable because virtually anyone can contribute, but frankly speaking, also nobody really asks about the verification of other, even commercial resources. Although we believe the use of proof checkers could enable the automatic verification of the proofs; still the correctness of the definitions, i.e. how the encoded version reflect real mathematical objects, is under question only human can answer to a full extent.

But even in the projects of GNU type, people want to get their payment in another form: at least the added annotation such as "This article is owned by...", as in PlanetMath, which can also be considered a kind of motivation to keep higher standards of the encyclopedia since the authorship is not fully anonymous.

In the MML the authorship is somewhat fixed, there were however, especially recently, cases when the parts of submissions moved between them so that the authorship actually exchanged (as for example, with the formalization of the Zorn Lemma, originally created by Grzegorz Bancerek, and now, after the changes concerned with the move of this article to the concrete part, attributed to Wojciech Trybulec). In a sense, the Mizar library is much closer to PlanetMath, but the official distributions are created by the Library Committee which decides about the acceptance of revisions.

7 Conclusions

To meet the expectations of researchers being potential users of repositories of mathematical knowledge, such collections cannot be frozen. The availability of the contemporary electronic media open new directions of the development of the new encyclopedias yet unavailable for their paper counterparts. The need of the enhancement stems not only of the fact there may be some obvious mistakes in the source; the reasons are far more complex.

In the paper, we tried to point out some of the issues connected with the mechanism of revisions performed on the Mizar Mathematical Library, large repository of computer-verified mathematical knowledge. The dependencies between its items and the environment declaration (notation and especially, constructors) are as of now too complex to freely move a single definition or a theorem between separate articles.

In our opinion, the current itemization of the MML into articles does not fit the needs we expect from the feasible repository of mathematical facts; if we try to keep authors' rights unchanged, there is an emerging need to have some other, smaller items which guarantee the developer's authorship rights, a kind of ownership similar to that used in the PlanetMath project.

Also the better automation of the MML revision process is strongly desirable. However possible, at least to some extent, but due to some difficulties which can be met as we pointed out, the human supervision of such automatic changes will probably always be needed.

Acknowledgments

We are grateful to Andrzej Trybulec and Artur Korniłowicz for their continuous cooperation on the enhancement of the MML. The first author acknowledges the support of the EU FP6 IST grant TYPES (Types for Proofs and Programs) No. 510996. We acknowledge also anonymous referees for their useful suggestions, unfortunately due to space constraints not all of them were reflected in this final version.

References

1. M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*; National Bureau of Standards, Applied Mathematics Series No. 55, U.S. Government Printing Office, Washington, DC, 1964, see also `http://www.convertit.com/Go/ConvertIt/Reference/AMS55.ASP`.
2. A.A. Adams and J.H. Davenport, *Copyright issues for MKM*, in: A. Asperti, G. Bancerek, and A. Trybulec (eds.), Proc. of MKM 2004, Lecture Notes in Computer Science 3119, pp. 1–16, 2004.
3. G. Bancerek, *Information retrieval and rendering with MML Query*; in: J.M. Borwein and W.M. Farmer (eds.), Proc. of MKM 2006, Lecture Notes in Artificial Intelligence 4108, pp. 65–80, 2006.
4. N.G. de Bruijn, *The Mathematical Vernacular, A Language for Mathematics with typed sets*; in: P. Dybjer et al. (eds.), Proc. of the Workshop on Programming Languages, Marstrand, Sweden, 1987.
5. J. H. Davenport, *MKM from book to computer: A case study*; in: A. Asperti, B. Buchberger, and J. Davenport (eds.), Proc. of MKM 2003, Lecture Notes in Computer Science 2594, pp. 17–29, 2003.

6. A. Grabowski, *On the computer-assisted reasoning about rough sets*; in: B. Dunin-Kęplicz et al. (eds.), Monitoring, Security, and Rescue Techniques in Multiagent Systems, Advances in Soft Computing, Springer, pp. 215–226, 2005.
7. A. Grabowski and Ch. Schwarzweller, *Rough Concept Analysis – theory development in the Mizar system*; in: A. Asperti, G. Bancerek, and A. Trybulec (eds.), Proc. of MKM 2004, Lecture Notes in Computer Science 3119, pp. 130–144, 2004.
8. F. Kamareddine and R. Nederpelt, *A Refinement of de Bruijn's Formal Language of Mathematics*; Journal of Logic, Language and Information, 13(3), pp. 287–340, 2004.
9. R. Matuszewski and P. Rudnicki, MIZAR*: the first 30 years*; Mechanized Mathematics and Its Applications, 4(**1**), pp. 3–24, 2005.
10. B.R. Miller and A. Youssef, *Technical aspects of the Digital Library of Mathematical Functions*; Annals of Mathematics and Artificial Intelligence, 38, pp. 121–136, 2003.
11. The Mizar Homepage; `http://www.mizar.org/`.
12. A. Naumowicz and Cz. Byliński, *Improving Mizar texts with properties and requirements*; in: A. Asperti, G. Bancerek, and A. Trybulec (eds.), Proc. of MKM 2004, Lecture Notes in Computer Science 3119, pp. 190–301, 2004.
13. PlanetMath web page; `http://planetmath.org/`.
14. P. Rudnicki and A. Trybulec, *Abian's fixed point theorem*, Formalized Mathematics, 6(**3**), pp. 335–338, 1997.
15. P. Rudnicki and A. Trybulec, *Mathematical Knowledge Management in Mizar*; in: B. Buchberger and O. Caprotti (eds.), Proc. of MKM 2001, Linz, Austria, 2001.
16. P. Rudnicki and A. Trybulec, *On the integrity of a repository of formalized mathematics*; in: A. Asperti, B. Buchberger, and J. Davenport (eds.), Proc. of MKM 2003, Lecture Notes in Computer Science 2594, pp. 162–174, 2003.
17. C. Sacerdoti Coen, *From proof-asistants to distributed knowledge repositories: tips and pitfalls*; in: A. Asperti, B. Buchberger, and J. Davenport (eds.), Proc. of MKM 2003, Lecture Notes in Computer Science 2594, pp. 30–44, 2003.
18. J. Urban, *MoMM – fast interreduction and retrieval in large libraries of formalized mathematics,* International Journal on Artificial Intelligence Tools, 15(1), pp. 109–130, 2006.
19. Wolfram Mathworld web page; `http://mathworld.wolfram.com/`.