On Duplication in 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@inf.ug.edu.pl

Abstract. Building a repository of proof-checked mathematical knowledge is without any doubt a lot of work, and besides the actual formalization process there is also the task of maintaining the repository. Thus it seems obvious to keep a repository as small as possible, in particular each piece of mathematical knowledge should be formalized only once.

In this paper, however, we claim that it might be reasonable or even necessary to duplicate knowledge in a mathematical repository. We analyze different situations and reasons for doing so, provide a number of examples supporting our thesis and discuss some implications for building mathematical repositories.

1 Introduction

Mathematical knowledge management aims at providing tools and infrastructure supporting the organization, development, and teaching of mathematics using modern techniques provided by computers. Consequently, large repositories of mathematical knowledge are of major interest because they provide users with a data base of — verified — mathematical knowledge. We emphasize the fact that a repository should contain verified knowledge only together with the corresponding proofs. We believe that (machine-checked or -checkable) proofs necessarily belong to each theorem and therefore are an essential part of a repository.

However, mathematical repositories should be more than collections of theorems and corresponding proofs accomplished by a prover or proof checker. The overall goal here is not only stating and proving a theorem — though this remains an important and challenging part — but also presenting definitions and theorems so that the "natural" mathematical build-up remains visible. Theories and their interconnections should be available, so that further development of the repository can rely on existing formalizations. Being not trivial as such, this becomes even harder to assure for an open repository with a large number of authors.

In this paper we deal with yet another organizational aspect of building mathematical repositories: the duplication of knowledge, by which we mean that a repository includes redundant knowledge. At first glance this may look unacceptable or at least unnecessary. Why should one include — and hence formalize

S. Autexier et al. (Eds.): AISC/Calculemus/MKM 2010, LNAI 6167, pp. 300–314, 2010.

— the same thing more than once? A closer inspection, however, shows that mathematical redundancy may occur in different non-trivial facets: Different proofs of a theorem may exist or different versions of a theorem formulated in a different context. Sometimes we even have different representations of the same mathematical object serving for different purposes.

From the mathematical point of view this is not only harmless but also desirable; it is part of the mathematical progress that theorems and definitions change and evolve. In mathematical repositories, however, each duplication of knowledge causes an additional amount of work. In this paper we analyze miscellaneous situations and reasons why there could — and should — be at least some redundancy in mathematical repositories. These situations range from the above mentioned duplication of proofs, theorems and representations to the problem of generalizing knowledge. Even technical reasons due to the progress of a repository may lead to duplication of knowledge.

2 Different Proofs of a Theorem

In the following we present different proofs of the Chinese Remainder Theorem (CRT) and briefly sum up the discussion from [Sch09]. The "standard" version of the CRT reads as follows.

Theorem 1. Let $m_1, m_2, \ldots, m_r$ be positive integers such that m_i and m_j are relatively prime for $i \neq j$. Let $m = m_1 m_2 \cdots m_r$ and let $u_1, u_2, \ldots, u_r$ be integers. Then there exists exactly one integer u with

$$0 \leq u < m \quad \text{and} \quad u \equiv u_i \bmod m_i \quad \text{for all} \quad 1 \leq i \leq r. \qquad \diamond$$

We consider three different proofs of the theorem and discuss their relevance to be included in mathematical repositories. It is very easy to show, that there exists at most one such integer u; in the following proofs we therefore focus on the existence of u. The proofs are taken from [Knu97].

First proof: Suppose integer u runs through the m values $0 \leq u < m$. Then $(u \bmod m_1, \ldots, u \bmod m_r)$ also runs through m different values, because the system of congruences has at most one solution. Because there are exactly $m_1 m_2 \cdots m_r = m$ different tuples $(v_1, \ldots, v_r)$ with $0 \leq v_i < m_i$, every tuple occurs exactly once, and hence for one of those we have $(u \bmod m_1, \ldots, u \bmod m_r) = (u_1, \ldots, u_r)$. $\diamond$

This proof is pretty elegant and uses a variant of the pigeon hole principle: If we pack m items without repetition to m buckets, then we must have exactly one item in each bucket. It is therefore valuable to include this proof in a repository for didactic or aesthetic reasons. On the other hand the proof is non-constructive, so that it gives no hints to find the value of u — besides the rather valueless "Try and check all possibilities, one will fit". A constructive proof, however, can easily be given:

Second proof: We can find integers M_i for $1 \leq i \leq r$ with

$$M_i \equiv 1 \bmod m_i \quad \text{and} \quad M_j \equiv 0 \bmod m_i \quad \text{for } j \neq i.$$

Because m_i and m/m_i are relatively prime, we can take for example

$$M_i = (m/m_i)^{\varphi(m_i)},$$

where φ denotes the Euler function. Now,

$$u = (u_1 M_1 + u_2 M_2 + \cdots + u_r M_r) \bmod m$$

has the desired properties. ◇

This proof uses far more evolved mathematical notations — namely Euler's function — and for that reason may also be considered more interesting than the first one. Formalization requires the use of Euler's function[1] which may cause some preliminary work. From a computer science point of view, however, the proof has two disadvantages. First, it is not easy to compute Euler's function; in general one has to decompose the moduli m_i into their prime factors. Second, the M_i being multiples of m/m_i are large numbers, so that a better method for computing u is highly desirable. Such a method has indeed been found by H. Garner.

Third proof: Because we have $\gcd(m_i, m_j) = 1$ for $i \neq j$ we can find integers c_{ij} for $1 \leq i < j \leq r$ with

$$c_{ij} m_i \equiv 1 \bmod m_j$$

by applying the extended Euclidean algorithm to m_i and m_j. Now taking

$$\begin{aligned}
v_1 &:= u_1 \bmod m_1 \\
v_2 &:= (u_2 - v_1)c_{12} \bmod m_2 \\
v_3 &:= ((u_3 - v_1)c_{13} - v_2)c_{23} \bmod m_3 \\
&\vdots \\
v_r &:= (\ldots((u_r - v_1)c_{1r} - v_2)c_{2r} - \cdots - v_{r-1})c_{(r-1)r} \bmod m_r
\end{aligned}$$

and then setting

$$u := v_r m_{r-1} \cdots m_2 m_1 + \cdots + v_3 m_2 m_1 + v_2 m_1 + v_1$$

we get the desired integer u. ◇

When constructing the v_i the application of the modulo operation in each step ensures that the occurring values remain small. The proof is far more technical than the others in constructing $\binom{r}{2} + r$ additional constants, the v_i in addition being recursively defined. On the other hand, however, this proof includes an efficient method to compute the integer u from Theorem 1.

[1] Actually a mild modification of the proof works without Euler's function.

3 Different Versions of Theorems

There are quite a number of reasons why different versions of the same theorem exist and may be included in mathematical repositories. Besides mathematical issues we also identified reasons justified by formalization issues or the development of repositories itself. For illustration we again use the CRT as an example.

3.1 Restricted Versions

Theorems are not always shown with a proof assistant to be included in a repository in the first place: Maybe the main goal is to illustrate or test a new implemented proof technique or just to show that this special kind of mathematics can be handled within a particular system. In this case it is often sufficient — or simply easier — to prove a weaker or restricted version of the original theorem from the literature.

In HOL Light [Har10], for example, we find the following theorem.

```
# INTEGER_RULE
  '!a b u v:int. coprime(a,b) ==>
                        ?x. (x == u) (mod a) /\ (x == v) (mod b)';
```

This is a version of the CRT stating that in case of two moduli `a` and `b` only there exists a simultaneous solution `x` of the congruences. Similar versions have been shown with `hol98` ([Hur03]), the Coq proof assistant ([Mén10]) or Rewrite Rule Laboratory ([ZH92]).

From the viewpoint of mathematical repositories it is of course desirable to have included the full version of the theorem also. Can we, however, in this case easily set the restricted version aside? Note that the above theorem in HOL Light also serves as a rule for proving divisibility properties of the integers. Erasing the restricted version then means that the full version has to be used instead. It is hardly foreseeable whether this will work for all proofs relying on the restricted version. So, probably both the restricted and the full version belong to the repository.

3.2 Different Mathematical Versions

The most natural reason for different versions of theorems is that mathematicians often look at the same issue from different perspectives. The CRT presented in Section 2 deals with congruences over the integers: it states the existence of an integer solving a given system of congruences. Looking from a more algebraic point of view we see that the moduli m_i can be interpreted as describing the residue class rings $\mathcal{Z}_{m_i}$. The existence and uniqueness of the integer u from the CRT then gives rise to an isomorphism between rings [GG99]:

Theorem 2. Let $m_1, m_2, \ldots, m_r$ be positive integers such that m_i and m_j are relatively prime for $i \neq j$ and let $m = m_1\, m_2 \cdots m_r$. Then we have the ring isomorphism

$$\mathcal{Z}_m \;\cong\; \mathcal{Z}_{m_0} \times \cdots \times \mathcal{Z}_{m_r}. \qquad \diamond$$

This version of the CRT has been formalized in `hol98` [Hur03]. Here we find a two-moduli version that in addition is restricted to multiplicative groups. Technically, the theorem states that for relatively prime moduli p and q the function $\lambda x.(x \bmod p, x \bmod q)$ is a group isomorphism between $\mathcal{Z}_{pq}$ and $\mathcal{Z}_p \times \mathcal{Z}_q$.

$$\begin{aligned}\vdash\ & \forall p, q.\\ & \quad 1 < p \wedge 1 < q \wedge \mathbf{gcd}\ p\ q = 1 \Rightarrow\\ & \quad (\lambda x.(x \bmod p, x \bmod q)) \in\\ & \qquad \mathbf{group\_iso}\ (\mathbf{mult\_group}\ pq)\\ & \qquad (\mathbf{prod\_group}\ (\mathbf{mult\_group}\ p)\ (\mathbf{mult\_group}\ q))\end{aligned}$$

Note that, in contrast to Theorem 2, the isomorphism is part of the theorem itself and not hidden in the proof.

It is not easy to decide which version of the CRT may be better suited for inclusion in a mathematical repository. Theorem 2 looks more elegant and in some sense contains more information than Theorem 1: It does not state the existence of a special integer, but the equality of two mathematical structures. The proof of Theorem 2 uses the homomorphism theorem for rings and is therefore interesting for didactic reasons, too. On the other hand, Theorem 1 uses integers and congruences only, so that one needs less preliminaries to understand it. Theorem 1 and its proof also give more information than theorem 2 concerning computational issues[2] — at least if not the first proof only has been formalized.

3.3 Different Technical Versions

Another reason for different versions of a theorem may be originated in the mathematical repository itself. Here again open repositories play an important role: Different authors, hence different styles of formalizing and different kinds of mathematical understanding and preferences meet in one repository. So, it may happen that two authors formalize the same (mathematical) theorem, but choose a different formulation and/or a different proof. We call this technical versions.

Especially in evolving systems such versions may radically differ just because the system's language improved over the years. In the Mizar Mathematical Library, for example, we find the following CRT [Sch08]

```
theorem
for u being integer-yielding FinSequence,
    m being CR_Sequence st len u = len m
ex z being Integer
st 0 <= z & z < Product(m) & for i being natural number
   st i in dom u holds z,u.i are_congruent_mod m.i;
```

[2] To apply the homomorphism theorem in the proof of Theorem 2 one needs to show that the canonical homomorphism is a surjection with kernel (m). This sometimes is done by employing the extended Euclidean algorithm, so that this proof gives an algorithm, too.

Here, a `CR_Sequence` is a sequence of natural numbers, which are pair wise relatively prime. Note that this formulation of the theorem is very close to the textbook version of theorem 1.

In another Mizar article [Kon97], however, we find a different formulation of the CRT:

```
theorem :: WSIERP_1:44
len fp>=2 &
(for b,c st b in dom fp & c in dom fp & b<>c holds (fp.b gcd fp.c)=1)
implies for fr st len fr=len fp holds ex fr1 st (len fr1=len fp &
for b st b in dom fp holds (fp.b)*(fr1.b)+(fr.b)=(fp.1)*(fr1.1)+(fr.1));
```

In this version no attributes are used. The condition that the m_i are pair wise relatively prime is here stated explicitly using the `gcd` functor for natural numbers. Also the congruences are described arithmetically: $u \equiv u_i \bmod m_i$ means that there exists a x_i such that $u = u_i + x_i * m_i$, so the theorem basically states the existence of $x_1, \ldots, x_r$ instead of u.

Since the article has been written more than 10 years ago, a reason is hard to estimate. It may be that at the time of writing Mizar's attribute mechanism was not so far developed as today, i.e. the author reformulated the theorem in order to get it formalized at all. Another explanation for this version might be that the author when formalizing the CRT already had in mind a particular application and therefore chose a formulation better suited to prove the application.

In the Coq Proof Assistant [Coq10] the CRT has been proved for a bit vector representation of the integers [Mén10], though as a restricted version of Theorem 1 with two moduli `a` and `b`.

```
Theorem chinese_remaindering_theorem :
 forall a b x y : Z,
 gcdZ a b = 1%Z -> {z : Z | congruentZ z x a /\ congruentZ z y b}.
```

In fact this theorem and its proof are the result of rewriting a former proof of the CRT in Coq. So in Coq there exist two versions of the CRT — though the former one has been declared obsolete.

We see that in general the way authors use open systems to formalize theorems has a crucial impact on the formulation of a theorem, and may lead to different versions of the same theorem. Removing one — usually the older one — version is a dangerous task: In large repositories it is not clear whether all proofs relying on the deleted version can be easily changed to work with the other one. So often both versions remain in the repository.

4 Abstract and Concrete Mathematics

Practically every mathematical repository has a notion of groups, rings, fields and many more abstract structures. The advantage is obvious: A theorem shown in an abstract structure holds in every concretion of the structure also. This helps to keep a repository small: Even if concrete structures are defined there is no

need to repeat theorems following from the abstract structure. If necessary in a proof one can just apply the theorem proved for the abstract structure.

Nevertheless authors tend to prove theorems again for the concrete case. We can observe this phenomenon in the Mizar Mathematical Library (MML). There we find, for example, the following theorem about groups.

```
theorem
for V being Group
for v being Element of V holds v - v = 0.V;
```

For a number of concrete groups (rings or fields) this theorem, however, has been proved and stored in MML again, among them complex numbers and polynomials.

```
theorem
for a being complex number holds a - a = 0;

theorem
for L be add-associative right_zeroed right_complementable
        (non empty addLoopStr)
for p be Polynomial of L holds p - p = 0_.(L);
```

One reason might be that authors are not aware of the abstract theorems they can use and therefore believe that it is necessary to include these theorems in the concrete case. This might be especially true, if authors work on applications rather than on "core" mathematics. On the other hand it might just be more comfortable for authors to work solely in the concrete structure rather than to switch between concrete and abstract structures while proving theorems in a concrete structure.

Constructing new structures from already existent ones sometimes causes a similar problem: Shall we formalize a more concrete or a more abstract construction? Multivariate polynomials, for example, can be recursively constructed from univariate polynomials using $R[X,Y] \cong (R[X])[Y]$; or more concrete as functions from Terms in X and Y into the ring R. Which version is better suited for mathematical repositories? Hard to say, from a mathematical point of view the first version is the more interesting construction. The second one, however, seems more intuitive and may be more convenient to apply in other areas where polynomials are used. So, it might be reasonable to include both constructions in a repository. In this case, however, theorems about polynomials will duplicate also.

We close this section with another example: rational functions. Rational functions can be constructed as pairs of polynomials or as the completion $K(X)$ of the polynomial ring $K[X]$. As in the case of multivariate polynomials both constructions have its right in its own, so again both may be included in a repository. Note that this eventually might result in another (two) concrete version(s) of the theorem about groups from above, e.g.

```
theorem
for L being Field
for z being Rational_Function of L holds z - [0_.(L),1_.(L)] = z;
```

5 Representational Issues

In the majority of cases it does not play a major role how mathematical objects are represented in repositories. Whether the real numbers, for example, are introduced axiomatically or are constructed as the Dedekind-completion of the rational numbers, has actually no influence on later formalizations using real numbers. Another example are ordered pairs: Here we can apply Kuratowski's or Wiener's definition that is

$$(a, b) = \{\{a\},\ \{a, b\}\}$$

or

$$(a, b) = \{\{\{a\}, \emptyset\}, \{\{b\}\}\}$$

or even again the axiomatic approach

$$(a_1, b_1) = (a_2, b_2) \text{ if and only if } a_1 = a_2 \text{ and } b_1 = b_2.$$

Once there is one of the notions included in a repository formalizations relying on this notion can be carried out more or less the same.

There are, however, mathematical objects having more than one relevant representation. The most prominent example are polynomials. Polynomials can be straightforwardly constructed as sequences (of coefficients) over a ring

$$p = (a_n, a_{n-1}, \ldots a_0)$$

or as functions from the natural numbers into a ring

$$p = f : \mathbb{N} \longrightarrow R \text{ where } |\{x | f(x) \neq 0\}| < \infty.$$

Note that both representations explicitly mention all zero coefficients of a polynomial, that is provide a dense representation.

There is an alternative seldom used in mathematical repositories: sparse polynomials. In this representation only coefficients not equal to 0 are taken into account — at the cost that exponents e_i have to be attached. We thus get a list of pairs:

$$p = ((e_1, a_1), (e_2, a_2), \ldots (e_m, a_m)).$$

Though more technical to deal with — that probably being the reason for usually choosing a dense representation for formalization — there exist a number of efficient algorithms based on a sparse representation, for example interpolation and computation of integer roots. Therefore it seems reasonable to formalize both representations in a repository, thus reflecting the mathematical treatment of polynomials.

Another example is the representation of matrices, also a rather basic mathematical structure. The point here is that there exist many interesting subclasses of matrices, for example block matrices for which a particular multiplication algorithm can be given or triangular matrices for which equations are much easier to solve. Hence it might be reasonable to include different representations of matrices, that is different (re-) definitions, in a repository to provide support for particular applications of matrices.

6 Generalization of Theorems

Generalization of theorems is everyday occurrence in mathematics. In the case of mathematical repositories generalization is a rather involved topic: It is not obvious whether the less general theorem can be eliminated. Proofs of other theorems using the original version might not work automatically with the more general theorem instead. The reason may be that a slightly different formulation or even a different version of the original theorem has been formalized. Then the question is: Should one rework all these proofs or keep both the original and the more general theorem in the repository? To illustrate that this decision is both not trivial and important for the organization of mathematical repositories we present in this section some generalizations of the CRT taken from [Sch09].

A rather uncomplicated generalization of Theorem 1 is based on the observation that the range in which the integer u lies, does not need to be fixed. It is sufficient that it has the width $m = m_1 m_2 \cdots m_r$. This easily follows from the properties of the congruence $\equiv$.

Theorem 3. Let $m_1, m_2, \ldots, m_r$ be positive integers such that m_i and m_j are relatively prime for $i \neq j$. Let $m = m_1\, m_2 \cdots m_r$ and let $a, u_1, u_2, \ldots, u_r$ be integers. Then there exists exactly one integer u with

$$a \leq u < a + m \text{ and } u \equiv u_i \bmod m_i$$

for all $1 \leq i \leq r$. ◇

It is trivial that for $a = 0$ we get the original Theorem 1. Old proofs can very easily be adapted to work with this generalization of the theorem. Maybe the system checking the repository even automatically infers that Theorem 3 with $a = 0$ substitutes the original theorem. If not, however, even the easy changing all the proofs to work with the generalization can be an extensive, unpleasant, and time-consuming task.

A second generalization of the CRT is concerned with the underlying algebraic structure. The integers are the prototype example for Euclidean domains. Taking into account that the residue class ring $\mathcal{Z}_n$ in fact is the factor ring of $\mathcal{Z}$ by the ideal $n\mathcal{Z}$, we get the following generalization.[3]

[3] Literally this is a generalization of Theorem 2, but of course Theorem 1 can be generalized analogously.

Theorem 4. Let R be a Euclidean domain. Let $m_1, m_2, \ldots, m_r$ be positive integers such that m_i and m_j are relatively prime for $i \neq j$ and let $m = m_1\, m_2 \cdots m_r$. Then we have the ring isomorphism

$$R/(m) \;\cong\; R/(m_0) \times \cdots \times R/(m_r). \qquad \diamond$$

This generalization may cause problems: In mathematical repositories it is an important difference whether one argues about the set of integers (with the usual operations) or the ring of integers: They have just different types. Technically, this means that in mathematical repositories we often have two different representations of the integers. In the mathematical setting theorems of course hold for both of them. However, proofs using one representation will probably not automatically work for the other one. Consequently, though Theorem 4 is more general, it might not work for proofs using integers instead of the ring of integers; for that a similar generalization of Theorem 1 could be necessary. So in this case in order to make all proofs work with a generalization, we need to provide generalizations of different versions of the original theorem — or just change the proofs with the "right" representation leading to an unbalanced organization of the repository.

We close this section with a generalization of the CRT that abstracts away even from algebraic structures. The following theorem [Lün93] deals with sets and equivalence relations only and presents a condition whether the "canonical" function σ is onto.

Theorem 5. Let α and β be equivalence relations on a given set M. Let $\sigma : M \longrightarrow M/\alpha \times M/\beta$ be defined by $\sigma(x) := (\alpha(x), \beta(x))$. Then we have $\ker(\sigma) = \alpha \cap \beta$ and σ is onto if and only if $\alpha \circ \beta = M \times M$. $\diamond$

Here almost all of the familiar CRT gets lost. There are no congruences, no algebraic operations, only the factoring (of sets) remains. Therefore, it seems hardly possible to adapt proofs using any of the preceding CRTs to work with this generalization. Any application will rely on much more concrete structures, so that too much effort has to be spent to adapt a proof. Theorem 5 in some sense is too general to reasonably work with. However, even though hardly applicable, the theorem stays interesting from a didactic point of view.[4] It illustrates how far — or even too far — we can generalize and may provide the starting point of a discussion whether this is — aside from mathematical aesthetics — expedient.

7 Which Way To Go?

Having seen that it's more or less unavoidable to duplicate knowledge in mathematical repositories, the question is how to deal with such situations. In the following we will mention some issues hoping to start a discussion towards guidelines for building mathematical repositories.

[4] In fact the proof of Theorem 5 has been an exercise in lectures on linear algebra.

Different mathematical versions of theorems (as discussed in section 3.2) describe the same mathematical issue from different points of view. So it would be a natural approach to actually prove that such theorems are equivalent. Though troublesome to accomplish, this also would make explicit that more than one version is present in a repository. The mathematical context in which a particular version is formulated and proven then would be clear from the proof, thus would be visible to users for further applications. Much harder to deal with is the question of which version of a theorem to use. Here, not only deep mathematical understanding and knowledge, but also fondness of particular mathematical views and techniques may play important roles.

In the case of different representations (see section 5) one idea is to provide theorems describing the connection between these. For example, we can prove that sparse and dense polynomials are isomorphic (as rings). This does not only provide information between these representation, but also gives the possibility to translate from one representation into another. Consequently, though a bit tedious, theorems for one representation can be used in the other one. One future goal could be to automate such translations.

Focusing on individual operations and not on structures as a whole, the process of translation between representations has been (partially) automated in Mizar. In the Mizar Mathematical Library we find definitions of both the integers as (just) numbers and the ring of integers. So, integers and elements of the ring of integers are different objects with different operations realizing addition, multiplication, and so on. Using a special registration `identify` the user, however, can identify terms and operations from different definitions ([Kor09]), in our example integers with elements of the ring of integers:

```
registration
  let a,b be integer number;
  let x,y be Element of INT.Ring;
  identify x+y with a+b when x=a, y=b;
  identify x*y with a*b when x=a, y=b;
end;
```

After this registration, theorems proved for integers can be applied to elements of the ring of integers without any translation issues.

We believe that the duplication of theorems in more special cases (compare section 4) can be avoided by providing more support for searching in mathematical repositories. Here, of course, we do not speak of ordinary searching: having a general theorem for an algebraic structure such as group or ring, we have to "search" for concrete structures in which this theorem holds. Or, putting it the other way round, when working in a concrete structure we'd like to find theorems true for this structure, though proved in a more general case. One possibility here, is to provide software that computes the theory of a given structure taking into account that part of this theory have to be generated from more general theories. Such a theory generator, in some sense, would transfer knowledge from a mathematical repository itself into the supporting software.

When it comes to generalizations of theorems, we have seen in section 6 that from a technical point of view they are hard to deal with: Deleting a theorem for which a more general version exists, can imply major changes of the repository. One can, however, think of a software detecting more general versions of a theorem. This would at least give the possibility to automatically identify generalizations. If more than one version is present in a repository such a software in addition can support users in their decision which version of a theorem to use best to construct their proofs.

7.1 MML — The State of the Art

In this subsection we collect some solutions of the considered issues based on the policy of the Library Committee of the Association of Mizar Users.

Ordinary repetitions. Although direct explicit repetitions of definitions or simple theorems with standard proofs are not desirable in large repositories of mathematical knowledge — they make more noise to search within — one can potentially find the pros of such approach; even if the fact is available from some other articles, we obtain complete source of all properties in a single file. In MML such freedom is not allowed and usually leads to the rejection of the submitted article (or, at least to the removal of such straight duplications).

Proof variants. In the above case the author usually lacks knowledge about formalized facts, or is unable to formulate a proper query (e.g. when using MML Query searching tool). Sometimes the situation is much more complex — submitted proofs can be better, shorter or just the original ones. As of now, the author is supposed either to delete such fact from his/her current submission or to revise the one already available in the repository. This policy met with the general criticism; potentially MML lost some valuable submissions; but the origins of such policy stemed when the software automatically removing identical theorems was quite frequently used.

Simple consequences. By a simple consequence we understand the theorem which is justified only by single theorem already present in the MML. Such consequences were automatically removed but as of now it is postponed.

Generalizations. Such activity is twofold: On the one hand after more complex revisions of the library, unused assumptions are automatically removed; hence facts are generalized by the software. On the other hand, MML is gradually divided into concrete and abstract parts (those articles which don't use the notion of the structure and the rest). The articles belonging to the concrete part are put earlier on the list of processing of the MML and are usually more general than those remaining. As an example, we can consider functions with values in the ring of complex, real or integer numbers. Here the generalization which comes to mind quite naturally is just considering complex functions with associated operations — and hence all these are special cases. The danger is here that if we go too far, all the usefulness can get lost (e.g. quaternions or two possible extensions of real numbers available in the same time: complex and extended real numbers).

Important facts. "Important" theorems are usually exception of the above rule — they remain in their original place. Although it is generally hard to measure such importance, one of the criteria is, e.g. the presence of the fact in Freek Wiedijk's Top 100 mathematical theorems list [Wie06]. Sometimes well-known concrete instantiations are proven earlier as simpler ones — as many of the lemmas proven for the Jordan Curve Theorem. To keep the library as compact as possible we should at least to hide these items (not to delete them completely as they might still be used by the main theorem), i.e. not to export them into the Mizar database; final decision was not to do this (however "obsolete" pragma could be useful). However, the proof of the irrationality of the square root of 2 is not present in the MML — but the irrationality of any prime number obviously is.

Automatic discovery. Detecting and removing identical theorems was always important problem for the MML, but when checker was strengthened, especially via attributes mechanism, such activity was less intensive. Here the work of Josef Urban (MoMM) [Urb06] are worth noticing; many connections between proven facts are "up-to-environment description" — obvious for software, but not for a human (to give a trivial but explaining example — it is automatically derived via the cluster registration mechanism that any singleton is non-empty, so all theorems true for non-empty sets can be applied to singletons; if someone does not include proper identifiers into his environment declaration, it will not be the case).

Tighter net of notions. Many of the notions within the Mizar Mathematical Library are stated in terms of adjectives, plaing a role of axioms for theories. If this structure will be more precise, the possibility of the automatic detection of the interconnections between various notions will be higher. That's what the MML referees have especially in mind.

Parallel developments. There are some exceptions of the above rules which can be easily justified. E.g., we have two approaches to lattices (based on the ordering relation and equationally defined), two alternative views for the category theory, and even four distinct approaches to Petri nets (although one of them is currently being eliminated). There is however no clear view of how to merge them, or, as in the case of lattices, the expressive power of the Mizar language does not allow for such elimination.

8 Conclusions

When building a mathematical repository it seems plausible to not duplicate knowledge in order to avoid an unnecessary blow-up of the repository. This is similar to — and may be inspired by — mathematical definitions, in which the number of axioms is kept as small as possible.

In this paper we have argued that this, however, is not true in general. We have analyzed miscellaneous situations in which it might be reasonable or even necessary to duplicate knowledge in a repository. The reasons for that are manifold: Different proofs may be interesting for didactic reasons or different representations of the same knowledge may better support different groups of users. Even

improvements of a repository may lead to duplication of knowledge because e.g. a theorem, that has been generalized, cannot always be trivially erased without reworking lots of proofs.

Based on our analysis we have also outlined some ideas how to cope with or maybe avoid duplication of knowledge giving rise to further research. In general, it is hardly foreseeable in which cases which kind of knowledge should be duplicated. This strongly depends on different kind of users the repository should attract.

References

[Coq10] The Coq Proof Assistant, http://coq.inria.fr

[Dav03] Davenport, J.H.: MKM from Book to Computer: A Case Study. In: Asperti, A., Buchberger, B., Davenport, J.H. (eds.) MKM 2003. LNCS, vol. 2594, pp. 17–29. Springer, Heidelberg (2003)

[DeB87] de Bruijn, N.G.: The Mathematical Vernacular, a language for mathematics with typed sets. In: Dybjer, P., et al. (eds.) Proceedings of the Workshop on Programming Languages, Marstrand, Sweden (1987)

[GG99] von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (1999)

[GS06] Grabowski, A., Schwarzweller, C.: Translating Mathematical Vernacular into Knowledge Repositories. In: Kohlhase, M. (ed.) MKM 2005. LNCS (LNAI), vol. 3863, pp. 49–64. Springer, Heidelberg (2006)

[Har10] Harrison, J.: The HOL Light Theorem Prover, http://www.cl.cam.ac.uk/~jrh13/hol-light

[Hur03] Hurd, J.: Verification of the Miller-Rabin Probabilistic Primality Test. Journal of Logic and Algebraic Programming 50(1-2), 3–21 (2003)

[KZ89] Kapur, D., Zhang, H.: An Overview of Rewrite Rule Laboratory (RRL). In: Dershowitz, N. (ed.) RTA 1989. LNCS, vol. 355, pp. 559–563. Springer, Heidelberg (1989)

[KN04] Kamareddine, F., Nederpelt, R.: A Refinement of de Bruijn's Formal Language of Mathematics. Journal of Logic, Language and Information 13(3), 287–340 (2004)

[Knu97] Knuth, D.: The Art of Computer Programming. In: Seminumerical Algorithms, 3rd edn., vol. 2, Addison-Wesley, Reading (1997)

[Kon97] Kondracki, A.: The Chinese Remainder Theorem. Formalized Mathematics 6(4), 573–577 (1997)

[Kor09] Korniłowicz, A.: How to Define Terms in Mizar Effectively. Studies in Logic, Grammar and Rhetoric 18(31), 67–77 (2009)

[Lün93] Lüneburg, H.: Vorlesungen über Lineare Algebra, BI Wissenschaftsverlag (1993) (in German)

[Mén10] Ménissier-Morain, V.: A Proof of the Chinese Remainder Lemma, http://logical.saclay.inria.fr/coq/distrib/current/contribs/ZChinese.html

[Miz10] The Mizar Home Page, http://mizar.org

[NB04] Naumowicz, A., Byliński, C.: Improving Mizar Texts with Properties and Requirements. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 290–301. Springer, Heidelberg (2004)

[PSK04] Pollet, M., Sorge, V., Kerber, M.: Intuitive and Formal Representations: The Case of Matrices. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 317–331. Springer, Heidelberg (2004)

[RT01] Rudnicki, P., Trybulec, A.: Mathematical Knowledge Management in Mizar. In: Buchberger, B., Caprotti, O. (eds.) Proceedings of the 1st International Conference on Mathematical Knowledge Management, Linz, Austria (2001)

[Sch08] Schwarzweller, C.: Modular Integer Arithmetic. Formalized Mathematics 16(3), 247–252 (2008)

[Sch09] Schwarzweller, C.: The Chinese Remainder Theorem, its Proofs and its Generalizations in Mathematical Repositories. Studies in Logic, Grammar and Rhetoric 18(31), 103–119 (2009)

[Urb06] Urban, J.: MoMM — Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. International Journal on Artificial Intelligence Tools 15(1), 109–130 (2006)

[Wie06] Wiedijk, F.: On the Usefulness of Formal Methods. In: Nieuwsbrief van de NVTI, pp. 14–23 (2006)

[ZH92] Zhang, H., Hua, X.: Proving the Chinese Remainder Theorem by the Cover Set Induction. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607, pp. 431–455. Springer, Heidelberg (1992)