

The Chinese Remainder Theorem, its Proofs and its Generalizations in Mathematical Repositories

Christoph Schwarzweller

Department of Computer Science, University of Gdańsk
ul. Wita Stwosza 57, 80-952 Gdańsk, Poland
schwarz@inf.univ.gda.pl

Abstract. In the spirit of mathematical knowledge management theorems are proven with computer assistance to be included into mathematical repositories. In the mathematical literature one often finds not only different proofs for theorems, but also different versions or generalizations with a different background. In mathematical repositories, for obvious reasons, there is usually one version of a theorem with one proof only – the authors choose a version and a proof which can be formalized most easily. In this paper we argue that there are other issues to decide which proof of a theorem or which version of a theorem should be included in a repository. These basically depend on the intended further use of the theorem and the proof. We illustrate these issues in detail with the Chinese Remainder Theorem as an example.

1 Introduction

Over the years much effort has been spent proving more and more elaborated theorems with computer assistance. Some of the most prominent examples recently finished are the Four Colour Theorem using Coq by Georges Gonthier, the Prime Number Theorem using Isabelle/HOL by Jeremy Avigad, and the Jordan Curve Theorem using HOL Light by Tom Hales (shortly after also proven using Mizar by a large group of authors). In contrast, building up repositories of proven theorems is a topic much younger than proving them¹. Lately this topic has received more and more attention as the area of mathematical knowledge management evolved.

Mathematical knowledge management aims at providing both tools and infrastructure supporting the organization, development, and also teaching of mathematics using modern techniques provided by computers. Consequently, large repositories of mathematical knowledge are here 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.

¹ An exception is the Mizar Mathematical Library that goes back to 1989.

However, mathematical repositories should be more than collections of theorems and their 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 buildup remains visible. Theories and their interconnections should be available, so that the further development of the repository can be based upon these. 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 aspect in building mathematical repositories that is usually not – or only implicitly – taken into account: the evolution of theorems. By this we mean that in mathematics theorems and proofs do not remain stable. New, more elegant proofs for a theorem are found, employing maybe other (new) proof techniques or relying on different new lemmas. Connections between mathematical subfields often lead to a “reformulation” of a theorem in order to apply a different background in the proof. Generalization of theorems also is an issue. The discovery that a theorem holds in a more general case than the original formulation indicates, is a very natural one in mathematics. Hence, both theorems and their proofs very often come in more than one version.

From the mathematical point of view this is not only harmless but also desirable; it is part of the mathematical progress. In mathematical repositories, however, one usually finds only one version of a theorem with one proof only. This is of course reasonable, if formalizing – that is stating and proving the theorem in a system – is the major goal. At this point sometimes different versions of theorems or proofs are implicitly dealt with. Before starting the formalization alternatives are checked and the one (seemingly) being best suited for formalization is chosen.

The argument “Once we have the theorem included in a repository, we can use it for further development, no matter how it has been proven” however, excludes technical problems of building mathematical repositories. When extending the repository it might occur that to finish a proof relying on a special theorem, it would be much better to use another version of this theorem. And if repositories are used for teaching mathematics, different versions of theorems and proofs are of course of interest for didactic reasons.

Consequently, it may be reasonable to include different versions of theorems or proofs in mathematical repositories, though at first sight this seems not only to blow up the repository, but also causes additional work. Such decisions, therefore, will usually depend on special intentions of the theorem, e.g.: Will it be also used for didactic purposes? Can other versions be formalized with a reasonable amount of work? In the following we will illustrate the above considerations by using the Chinese Remainder Theorem [16,8] as a concrete example.

The plan of the paper is as follows. In the next section we present the Chinese Remainder Theorem (CRT) in its standard version together with three different proofs. To give the reader a feeling about formalizing such a theorem, we give a brief introduction to the Mizar system [21] in Section 3 and a glimpse on the Mizar proof of the CRT in Section 4. Section 5 then deals with other versions of the CRT. One version is rather technical – due to formalization issues in open mathematical repositories. The other one is a completely different formulation of the theorem

which, unlike the first one, relies more on abstract algebra. Section 6 finally is devoted to possible generalizations of the CRT and their relation to the original theorem.

2 The Chinese Remainder Theorem and Its Proofs

The CRT is a result about congruences over the integers. It states that an integer u can be completely described by the sequence of its remainders – if the number of remainders is big enough. The “standard” version of the theorem reads as follows.

Theorem 1. Let m_1, m_2, \dots, 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, \dots, u_r be integers. Then there exists exactly one integer u with

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

That this theorem should be part of a mathematical repository needs no further explanation. But what kinds of proofs exist and what reasons are there to include these proofs in mathematical repositories? In the following we present 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 proving the existence of u . The proofs are taken from [16].

First proof: Suppose integer u runs through the m values $0 \leq u < m$. Then $(u \pmod{m_1}, \dots, u \pmod{m_r})$ 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, \dots, v_r) with $0 \leq v_i < m_i$, every tuple occurs exactly once, and hence for one of those we have $(u \pmod{m_1}, \dots, u \pmod{m_r}) = (u_1, \dots, u_r)$. \diamond

This proof is pretty elegant and uses a rather obvious 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, formalization of the proof is not straightforward. This proof is one of those, that can technically really blow up being formalized. One has to argue about the number of different r -tuples and, more importantly, to show that there exists a bijection between the set of r -tuples and the non-negative integers smaller than m . Another disadvantage is that 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”. This is even more disturbing, because a constructive proof can easily be given:

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

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

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

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

Christoph Schwarzweller

where φ denotes the Euler function. Now,

$$u = (u_1M_1 + u_2M_2 + \cdots + u_rM_r) \bmod m$$

has the desired properties. \diamond

This proof constructs r constants M_i with which the sought-after u can easily be computed. It therefore, in some sense, contains more information than the first proof, that should be contained in the repository also. The 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 function² which may lead to a lot of preliminary work. From a computer science point of view 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 really big numbers, so that a better method for computing u is highly desirable. Such a method has indeed been found by H. Garner, which gives a third proof of Theorem 1:

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

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

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 . \diamond

The proof uses $\binom{r}{2}$ constants c_{ij} that can be computed with the extended Euclidean algorithm because we have $\gcd(m_i, m_j) = 1$ for $i \neq j$. When constructing the v_i the application of the modulo operation in each step ensures that the occurring values remain small. Note that the finally computed u actually is a radix number in m_1, m_2, \dots, m_r . 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. Therefore, a formalization of this proofs will be rather unpleasant. On the other hand, however, this proof includes an efficient method to compute the integer u from Theorem 1.

² Actually this is not completely true: a mild modification of the proof does without Euler’s function; compare Section 4.2.

We see that the question which proof of a theorem should be formalized, does not only depend on the hardness of the formalization in a given system. Both elegance and the amount of information are issues that can be taken into consideration – this may even result in formalizing more than one proof. This, however, is not the end of the line. In the literature we often find different versions or even generalizations of a theorem, and again the question is which one to choose. Before we discuss these issues in case of the CRT, we will briefly provide insight into how to formalize the theorem in the Mizar system [21].

3 The Mizar System

The Mizar language as well as the Mizar system have been described elsewhere (see e.g. [24,23,25,32]). Here we only give a brief overview necessary to follow the subsequent sections.

Mizar’s [24,21] logical basis is the classical first order logic extended with so-called schemes. Schemes allow for free second order variables, in this way enabling, for example, the definition of induction schemes. The current development of the Mizar Mathematical Library (MML) is based on Tarski-Grothendieck set theory (a variant of Zermelo-Fraenkel set theory using Tarski’s axiom on arbitrarily large, strongly inaccessible cardinals [28] which can be used to prove the axiom of choice), though in principle the Mizar language allows for other axiom systems also. Mizar proofs are written in the natural deduction style similar to the calculus of [13]. The rules of the calculus are connected with corresponding (English) natural language phrases, so that the Mizar language is close to the one used in mathematical textbooks. The Mizar proof checker verifies the individual proof steps using the notion of obvious inferences [5] to shorten the rather long proofs of pure natural deduction.

Mizar objects are typed, the types forming a hierarchy with the fundamental type `set` [1]. New types are constructed using type constructors called modes. Modes can be decorated with adjectives – given by so-called attribute definitions – in this way extending the type hierarchy. For example, given the mode `Ring` and an attribute `commutative` a new mode `commutative Ring` can be constructed, which obeys all the properties given by the mode `Ring` plus the ones stated by the attribute `commutative`. Furthermore, a variable of type `commutative Ring` is also of type `Ring`, which implies that all notions defined for `Ring` are available for `commutative Ring`. In addition all theorems proved for type `Ring` are applicable for objects of type `commutative Ring`; indeed the Mizar checker itself infers subtype relations in order to check whether theorems are applicable for a given type.

4 A Mizar Formalization of the Chinese Remainder Theorem

We believe that in mathematical repositories both the formalization of the theorem itself – that is its representation and in particular its readability – and the formalization of the proof are equally important. Therefore, first we discuss our

representation of the theorem and then describe how we formalized the second proof of Section 2.

4.1 The Theorem

We represent the given integers u_1, \dots, u_r and m_1, \dots, m_r as finite sequences over the integers. For the m_i we need the additional condition that they are pairwise relatively prime. Using the already defined predicate `are_relative_prime` [18] this property is introduced in a Mizar attribute definition:

```
definition
  let f be integer-yielding FinSequence;
  attr f is Chinese_Remainder means
    for i,j being natural number
      st i in dom f & j in dom f & i <> j holds f.i, f.j are_relative_prime;
end;
```

Then a finite sequence of integers m_i fulfilling the assumption of Theorem 1 – a Chinese remainder sequence – can be described by the following mode definition. Note that the number r of moduli is given simply by the length of the sequence.

```
definition
  mode CR_Sequence is non empty positive-yielding Chinese_Remainder
    (integer-yielding FinSequence);
end;
```

As a consequence each element of type `CR_sequence` describes a set of moduli m_i fulfilling the assumptions of Theorem 1. Note also that due to Mizar's type widening mechanism all theorems proven for elements of type `FinSequence` and `integer-yielding FinSequence` can also be automatically applied to elements of type `CR_Sequence`.

Using the predicate `are_congruent_mod` [30] and the functor `Product` [2] giving the product of the elements of a finite sequence it is now easy to state the CRT in Mizar in a very readable fashion. Here we present only the theorem on the existence of the integer u . A second theorem describing the uniqueness of u is straightforward.

```
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;
```

4.2 A Mizar Formalization of the Second Proof

The proof starts with the definition of the constants M_i . Instead of Euler's function we use a variant that analogously to Garner's proof finds the constants by employing the extended Euclidean algorithm: If m is the product of the m_i we

have $\gcd(m/m_i, m_i) = 1$ for $i = 1, \dots, r$. We can therefore find integers s_i with $s_i * (m/m_i) \equiv 1 \pmod{m_i}$ and thus constants M_i with the desired properties. In Mizar we defined a finite sequence holding the values of M_i for $i = 1, \dots, r$:

```

definition
  let m be CR_Sequence;
  mode CR_coefficients of m -> FinSequence means
    len it = len m &
    for i being natural number st i in dom it holds
      ex s being Integer, mm being Integer
      st mm = Product(m)/m.i & s*mm,1 are_congruent_mod m.i &
      it.i = s * (Product(m)/m.i);
end;

registration
  let m be CR_Sequence;
  cluster -> integer-valued CR_coefficients of m;
end;

```

To be more precise, we actually do not fix the constants s_i , we just state (and prove) their existence. Therefore every finite sequence using appropriate values for the s_i is of type `CR_coefficients`, and not only the one with the value for the s_i computed by the extended Euclidean algorithm. As a consequence all that follows holds for arbitrary choices of the s_i .

The next step is to construct the integer $u = (u_1M_1 + \dots + u_rM_r) \pmod{m}$ from the constants M_i . This again is straightforward using the Mizar functors (#) for componentwise multiplication of sequences [22] and Sum for summing up the elements of a sequence [2]. We defined the Mizar functor `to_int` as follows.

```

definition
  let u be integer-yielding FinSequence,
      m be CR_Sequence such that len m = len u;
  func to_int(u,m) -> Integer means :Def_to_int:
    for c being CR_coefficients of m holds it = Sum(u(#)c) mod Product(m);
end;

```

Note that the term $\text{Sum}(u(\#)c) \pmod{\text{Product}(m)}$ denotes the same integer for an arbitrary choice of the coefficients in c ; or in other words $(u_1 * M_1 + \dots + u_r M_r) \pmod{m}$ always gives the same integer u no matter what concrete values s_i have been used in the construction of the `CR_sequence` c . This in fact is the reason that we can state `to_int(u,m)` in this way as a Mizar functor.

The proof of the CRT now consists of showing that `to_int(u,m)` has the desired properties. The key property here is of course that the constructed sum $\text{Sum}(u(\#)c)$ is congruent with u_i modulo m_i for all $i = 1, \dots, r$ as stated in the following theorem.

```

theorem congsum:
  for u being integer-valued FinSequence,
      m being CR_Sequence st len m = len u

```

Christoph Schwarzweller

```

for c being CR_coefficients of m,
  i being natural number st i in dom m
  holds Sum(u(#)c),u.i are_congruent_mod m.i;

```

The proof is pretty technical and shows by induction on the length l of the sequence $u(\#)s$ that its sum is congruent to 0 if $l < i$ and congruent to m_i if $l \geq i$. Together with the following result from [30]

```

theorem :: INT_1:41
  for i1,i2,i3,i4,i5 being Integer
  holds i4 * i5 = i3
  implies (i1,i2 are_congruent_mod i3 implies i1,i2 are_congruent_mod i4);

```

the rest of the proof basically consists of applying properties of integer congruences, that is applying theorems already present in the Mizar Mathematical Library. The obvious fact that $0 \leq \text{to_int}(u,m) < \text{Product}(m)$ has been shown in lemma2. To give the reader an impression of how Mizar proofs are written, we include the proof:

```

proof
let u be integer-yielding FinSequence, m be CR_Sequence;
assume AS: len u = len m;
take z = to_int(u,m);
now let i be natural number;
  assume A: i in dom u;
  consider c being CR_coefficients of m;
  set s = Sum(u(#)c) mod Product(m);
  B: dom m = Seg(len u) by AS,FINSEQ_1:def 3
  . = dom u by FINSEQ_1:def 3;
  then Sum(u(#)c),u.i are_congruent_mod m.i by A,AS,congsum;
  then C: Sum(u(#)c) mod m.i = u.i mod m.i by INT_3:12;
  dom m = Seg(len u) by AS,FINSEQ_1:def 3
  . = dom u by FINSEQ_1:def 3;
  then m.i in rng m by A,FUNCT_1:12;
  then D: m.i > 0 by PARTFUN3:def 1;
  consider y being Integer such that E: y * m.i = Product(m) by A,B,thm;
  s mod Product(m) = Sum(u(#)c) mod Product(m) by INT_3:13;
  then s,Sum(u(#)c) are_congruent_mod Product(m) by INT_3:12;
  then s,Sum(u(#)c) are_congruent_mod m.i by E,INT_1:41;
  then s mod m.i = Sum(u(#)c) mod m.i by INT_3:12;
  then s,u.i are_congruent_mod m.i by D,C,INT_3:12;
  hence z,u.i are_congruent_mod m.i by Def_to_int,AS;
end;
hence thesis by AS,lemma2;
end;

```

We would like to add that using the functor `to_int` one can easily show that modular integer arithmetic based on the CRT is correct [26]. Modular integer arithmetic performs integer arithmetic by transforming integers into sequences of their remainders, that is an integer u is transformed into an r -tuple of the form $(u \bmod m_1, \dots, u \bmod m_r)$ for given moduli m_1, m_2, \dots, m_r . After performing the

arithmetic operation componentwise on the tuples the CRT (via `to_int`) allows to transform the result back to the ordinary integers without loss of information – if $m = m_1 m_2 \cdots m_r$ is big enough.

To prove this we can easily define a functor `mod` that transforms an integer `u` into the sequence of `u`'s remainders. The moduli m_i are here given by a finite sequence `m`.

```

definition
  let u be Integer,
      m be integer-yielding FinSequence;
  func mod(u,m) -> FinSequence means
    len it = len m &
    for i being natural number st i in dom it holds it.i = u mod m.i;
end;

```

The componentwise arithmetic operations are then just the operations for sequences given in [22]. So, for example, `mod(u,m) (#) mod(v,m)` describes the componentwise multiplication of remainders for the integers `u` and `v`. Translating this finite sequence to an integer `z` using the functor `to_int` from above results in $z = u * v$, if $u * v < \text{Product}(m)$. We thus get the following

```

theorem
  for u,v being Integer,
      m being CR_Sequence st 0 <= u * v & u * v < Product(m)
  holds to_int(mod(u,m) (#) mod(v,m), m) = u * v;

```

Analogous theorems for addition and subtraction of integers can be easily stated and proven (see [26]).

5 Other Versions of the Chinese Remainder Theorem

In this section we consider other versions of the CRT, that is, generally speaking, other theorems stating the same fact as the original theorem. We try to analyze where these different versions come from and, more importantly, their impact on mathematical repositories.

5.1 Another Mathematical Version

The most natural reason for different versions of theorems is of course 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, that is concentrating on the structures being involved aside from the integers, 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 these rings. So the reformulated CRT looks as follows [8].

Theorem 2. Let m_1, m_2, \dots, 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

$$\mathcal{Z}_m \cong \mathcal{Z}_{m_1} \times \cdots \times \mathcal{Z}_{m_r}. \diamond$$

It is not easy to decide which version of the CRT is 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.³ 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 issues⁴ – at least if not the first proof only has been formalized.

5.2 Another Technical Version

Another reason for additional versions of a theorem may be based in the mathematical repository itself. Here again especially open repositories play an important role. 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 or representational versions.

In the Mizar Mathematical Library in [17], for example, we find another version of the CRT from Section 2:

```
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));
```

This theorem is not immediately understandable, also because the variables' types are not explicitly stated but given by a so-called reservation – that in the article of course occurs before the theorem.

```
reserve b,c for Nat,
        fp for FinSequence of NAT,
        fr,fr1 for FinSequence of INT;
```

³ Of course this equality easily follows from Theorem 1, but is not explicitly stated there.

⁴ 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.

In this version no attributes are used. The condition that the m_i are pairwise relatively prime is here stated explicitly using the `gcd` functor for natural numbers. Also the congruences are described arithmetically: $u \equiv u_i \pmod{m_i}$ 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, \dots, x_r instead of u .

Since the article has been written more than 10 years ago, a reason for this technical formulation is hard to find. 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 second technical 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.

We see that in general the way authors use open systems to formalize theorems has a crucial impact on the formulation, that is on the technical version of a theorem – and may lead to different versions of the same theorem.

6 Generalizations of the Chinese Remainder Theorem

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 (mathematical or technical) 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.

6.1 Mathematical Generalizations

A rather mild 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, \dots, 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, \dots, u_r be integers. Then there exists exactly one integer u with

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

for all $1 \leq i \leq r$. \diamond

It is trivial that for $a = 0$ we get the original Theorem 1. The 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 $n\mathcal{Z}$, it is rather obvious that the following generalization⁵ holds.

Theorem 4. Let R be a Euclidean domain. Let m_1, m_2, \dots, 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

$$R/(m) \cong R/(m_1) \times \cdots \times R/(m_r). \quad \diamond$$

This generalization may cause some problems: In mathematical repositories it is an immense difference whether one argues about the set of integers (with the usual operations) or the ring of integers: They have just different types.⁶ 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 not work for the other one. Consequently, though Theorem 4 is more general, it will not work for proofs using integers instead of the ring of integers; for that a similar generalization of Theorem 1 is 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 subsection with a generalization of the CRT that abstracts even from algebraic structures. The following theorem [19] 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 $\sigma : M \rightarrow M/\alpha \times M/\beta$ 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

In this generalization almost all of the familiar CRTs 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 in a reasonable amount of time. 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 if not applicable, the theorem stays interesting from a didactic point of view.⁷ It illustrates how far we sometimes can generalize and may provide the

⁵ Literally this is a generalization of Theorem 2, but of course Theorem 1 can be analogously generalized to Euclidean domains.

⁶ Though often the ring of integers is constructed using the (set of) integers.

⁷ In fact the proof of Theorem 5 has been an exercise in lectures on linear algebra.

starting point of a discussion whether this is – aside from mathematical aesthetics – expedient; a topic that is also of great interest for the organization of mathematical repositories.

6.2 Generalization towards Concepts

Another kind of possible generalization does not stem from mathematics, but from the intention of bringing more structure into the organization of mathematical repositories and can be compared with the ideas of [27,7] on theory interpretation. Modular arithmetic from Section 4.2 using the CRT is a typical example for the concept of computing with homomorphic images: transforming data into another algebraic structure, performing there algebraic operations and then reconstructing the result in the original structure; or, to say it a little bit shorter, performing the algebraic operations in another representation.

A second example is the multiplication of polynomials using the discrete Fourier transformation [8]. A discrete Fourier transformation DFT_ω changes the representation of a polynomial p with $\deg(p) < n$ to an n -tuple by evaluating p for n values. A special choice of these n values – $\omega^0, \omega^1, \dots, \omega^{n-1}$ for an n -th primitive root ω – ensures that no information is lost. After carrying out a componentwise multiplication on the n -tuples another discrete Fourier transformation – in fact $(\text{DFT}_\omega)^{-1}$ – constructs the result polynomial.

Though different in nature, both the CRT and the discrete Fourier transformation serve to prove that a special instantiation of the concept computing with homomorphic images is correct. Putting it the other way round, both of them can in some sense be generalized to the concept computing with homomorphic images. This adjacency should be reflected in mathematical repositories. This gives rise to a theorem describing in essence the correctness of the concept.⁸

Theorem 6. Let R and S be \langle structures \rangle , and let $f : R \rightarrow S$ be a function. Let u and v be elements of r . If f, u and v fulfill \langle a condition \rangle , then we have $u \text{ op}_R v = f^{-1}(f(u) \text{ op}_S f(v))$ for all operations op_R in R . \diamond

Note that the usual homomorphic condition $f(u \text{ op}_R v) = f(u) \text{ op}_S f(v)$ does not allow to transform the result back into the structure R . If f is an isomorphism, then of course the theorem becomes trivial. It therefore would be interesting to find weaker realizations of \langle a condition \rangle .

Theorem 6 states (in an abstract way) that to perform operations in R the change of the representation from R to S using f is admissible if \langle a condition \rangle holds. Having such a theorem in a mathematical repository would allow to handle the concept of computing with homomorphic images by just showing that a special case fulfills \langle a condition \rangle . Modular integer arithmetic, for example, according to Theorem 2 sets R to \mathcal{Z}_m and S to $\mathcal{Z}_{m_0} \times \dots \times \mathcal{Z}_{m_r}$. It then takes the `mod` functor as f and the `to_int` functor as f^{-1} . Another variant according to Theorem 1 is setting R to \mathcal{Z} instead of \mathcal{Z}_m . In this case however we get the additional condition

⁸ The following “theorem” holds for arbitrary (algebraic) structures and a yet to find condition, which we indicated by using Backus-Naur-like brackets \langle and \rangle .

that $u \text{ op}_R v < m$ holds. In the case of multiplication of polynomials p and q we get that $f = \text{DFT}$ and $f^{-1} = \text{DFT}^{-1}$ with the additional condition that $\deg(p * q) < n$.

7 Related Work

Formalizations of the CRT can be found in other systems besides Mizar. The first computer proof of the CRT dates back to 1992 using Rewrite Rule Laboratory [15]. Because the rewriting approach cannot handle quantifiers (all variables are assumed to be universally quantified), existential quantifiers are eliminated by introducing Skolem functions. So in [33] we find the following CRT.

$$\begin{aligned} &(\text{allpositive}(Y) \wedge \text{allprime}(Y)) \Rightarrow \\ &(\text{allcongruent}(\text{soln}(Y), Y) \wedge \\ &((\text{allcongruent}(x_1, Y) \wedge \text{allcongruent}(x_2, Y)) \Rightarrow \\ &(\text{rem}(x_1 - x_2, \text{products}(Y)) = 0))) \end{aligned}$$

where soln is the above mentioned Skolem function. The goal was to experiment with the cover set induction principle implemented in Rewrite Rule Laboratory, that is the challenging point was proving the theorem – in whatever formulation.

In more current proof systems the CRT also have been formalized. Interestingly, we found only two-number versions, that is CRTs where the number of moduli is restricted to two. In HOL Light [11], for example, we find such a version of Theorem 1 stating that in case of two moduli \mathbf{a} and \mathbf{b} there exists a simultaneous solution \mathbf{x} of the congruences.

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

INTEGER_RULE is a rule for proving divisibility properties of the integers. The rule is partly heuristic and most of the statements automatically proven with it are universally quantified. Again the main purpose of the CRT is to illustrate the power of a proof technique.

The CRT has been formalized in `ho198`, too [12]. Here we find a two-number version of Theorem 2 that in addition is restricted to multiplicative groups. Technically, the theorem states that for moduli p and q the function $\lambda x.(x \bmod p, x \bmod q)$ is a group isomorphism between \mathcal{Z}_{pq} and $\mathcal{Z}_p \times \mathcal{Z}_q$.

$$\begin{aligned} &\vdash \forall p, q. \\ &1 < p \wedge 1 < q \wedge \mathbf{gcd} \ p \ q = 1 \Rightarrow \\ &(\lambda x.(x \bmod p, x \bmod q)) \in \\ &\quad \mathbf{group_iso} \ (\mathbf{mult_group} \ pq) \\ &\quad (\mathbf{prod_group} \ (\mathbf{mult_group} \ p) \ (\mathbf{mult_group} \ q)) \end{aligned}$$

Note that, in contrast to Theorem 2, the isomorphism is part of the theorem itself and not hidden in the proof. The main goal of [12] was the verification of the Miller-Rabin probabilistic primality test. Therefore the restriction to multiplicative groups is reasonable, because this version of the CRT is sufficient for the verification.

In the Coq Proof Assistant [3] the CRT has been proved for a bit vector representation of the integers [20]. We see that this again is a version of Theorem 1 restricted to 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.

8 Conclusions

In order to discuss the question which proof and which version of a theorem is best suited for inclusion in mathematical repositories, we have presented various proofs, versions and generalizations of the Chinese Remainder Theorem.

It is probably no surprise that each version or generalization comes with its pros and cons, so that it seems not possible in general to decide which one is best suited to be included in a repository. On the contrary it may even be reasonable to include more than one proof or version convenient for different purposes. One maybe is better suited for further development (of applications) of the repository, whereas another one better for didactic reasons.

It is not foreseeable whether it is possible to develop criteria for deciding which proof, version or generalization of a theorem to include in a mathematical repository. However, it might be that the attempt to do so – like hopefully the considerations in this paper – is a step towards the development of schemata how to organize mathematical repositories.

Acknowledgment: I would like to thank the reviewers for their detailed comments which have greatly improved the presentation of the paper.

References

1. Bancerek, G.: *On the Structure of Mizar Types*; in: H. Geuvers and F. Kamareddine (eds.), Proceedings of MLC 2003, ENTCS 85(7), 2003.
2. Byliński, C.: *The Sum and Product of Finite Sequence of Real Numbers*; in: Formalized Mathematics, 1(4), pp. 661–668, 1990.
3. *The Coq Proof Assistant*; available at <http://coq.inria.fr>, 2008.
4. Davenport, J.H.: *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.
5. Davies, M.: *Obvious Logical Inferences*; in: Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 530–531, 1981.
6. de Bruijn, N.G.: *The Mathematical Vernacular, a language for mathematics with typed sets*; in: P. Dybjer et al. (eds.), Proceedings of the Workshop on Programming Languages, Marstrand, Sweden, 1987.

7. Farmer, W., Guttman, J. and Thayer, F.: *IMPS: An Interactive Mathematical Proof System*; in: Journal of Automated Reasoning 11, 213–248, 1993.
8. von zur Gathen, J. and Gerhard, J.: *Modern Computer Algebra*; Cambridge University Press, 1999.
9. Grabowski, A. and Schwarzweller, C.: *Translating Mathematical Vernacular into Knowledge Repositories*; in: M. Kohlhase (ed.), Proceedings of the 4th International Conference on Mathematical Knowledge Management, Lecture Notes in Artificial Intelligence 3863, pp. 49–64, 2006.
10. Graham, R.E., Knuth, D.E. and Patashnik, O.: *Concrete Mathematics*; Addison-Wesley, 1994.
11. Harrison, J.: *The HOL Light System Reference*; available at http://www.cl.cam.ac.uk/~jrh13/hol-light/reference_220.pdf, 2008.
12. Hurd, J.: *Verification of the Miller-Rabin Probabilistic Primality Test*; in: Journal of Logic and Algebraic Programming, 50(1-2), pp. 3–21, 2003.
13. Jaśkowski, S.: *On the Rules of Supposition in Formal Logic*; in: Studia Logica, vol. 1, 1934.
14. Kamareddine, F. and Nederpelt, R.: *A Refinement of de Bruijn’s Formal Language of Mathematics*; in: Journal of Logic, Language and Information, 13(3), pp. 287–340, 2004.
15. Kapur, D. and Zhang, H.: *An Overview of Rewrite Rule Laboratory (RRL)*; in: N. Dershowitz (ed.), Proceedings of the 3rd International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 355, pp. 559–563, 1989.
16. Knuth, D.: *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*; 3rd edition, Addison-Wesley, 1997.
17. Kondracki, A.: *The Chinese Remainder Theorem*; in: Formalized Mathematics, 6(4), pp. 573–577, 1997.
18. Kwiatek, R. and Zwara, G.: *The Divisibility of Integers and Integer Relative Primes*; in: Formalized Mathematics, 1(5), pp. 829–832, 1990.
19. Lüneburg, H.: *Vorlesungen über Lineare Algebra* (in German), BI Wissenschaftsverlag, 1993.
20. Ménessier-Morain, V.: *A Proof of the Chinese Remainder Lemma*; available at <http://logical.saclay.inria.fr/coq/distrib/current/contribs/ZChinese.html>, 2008.
21. The Mizar Home Page, <http://mizar.org>, 2009.
22. The Mizar Mathematical Library Committee, *Properties of Number-Valued Functions*, 2007.
23. Naumowicz, A. and Byliński, C.: *Improving Mizar Texts with Properties and Requirements*, in: A. Asperti, G. Bancerek, and A. Trybulec (eds.), Proceedings of the 3rd International Conference on Mathematical Knowledge Management, Lecture Notes in Computer Science 3119, pp. 190–301, 2004.
24. Rudnicki, P. and Trybulec, A.: *Mathematical Knowledge Management in Mizar*; in: B. Buchberger, O. Caprotti (eds.), Proceedings of the 1st International Conference on Mathematical Knowledge Management, Linz, Austria, 2001.
25. Schwarzweller, C.: *Mizar Attributes: A Technique to Encode Mathematical Knowledge into Type Systems*; in: Studies in Logic, Grammar and Rhetoric, vol. 10(23), pp. 387–400, 2007.
26. Schwarzweller, C.: *Modular Integer Arithmetic*; in: Formalized Mathematics, 16(3), pp. 247–252, 2008.
27. Shoenfield, J.: *Mathematical Logic*; Addison-Wesley, 1967.

28. Tarski, A.: *On Well-Ordered Subsets of Any Set*; in: *Fundamenta Mathematicae*, vol. 32, pp. 176–183, 1939.
29. Treyderowski, K. and Schwarzweller, C.: *Multiplication of Polynomials using Discrete Fourier Transformation*; in: *Formalized Mathematics*, 14(4), pp. 121–128, 2006.
30. Trybulec, M.: *Integers*; in: *Formalized Mathematics*, 1(3), pp. 501–505, 1990.
31. Wiedijk, F.: *On the Usefulness of Formal Methods*; *Nieuwsbrief van de NVTI*, pp. 14–23, 2006.
32. Wiedijk, F.: *Writing a Mizar Article in Nine Easy Steps*; available at <http://www.mizar.org/project/bibliography.html>, 2008.
33. Zhang, H. and Hua, X.: *Proving the Chinese Remainder Theorem by the Cover Set Induction*; in: D. Kapur (ed.), *Automated Deduction – CADE-11, Proceedings of the 11th International Conference on Automated Deduction, Lecture Notes in Computer Science 607*, pp. 431–455, 1992.