Content Development for Distance Education in Advanced University Mathematics Using Mizar

Takaya IDO[1], **Hiroyuki OKAZAKI**[1], **Hiroshi YAMAZAKI**[1], **Pauline Naomi KAWAMOTO**[1], **Katsumi WASAKI**[1], **and Yasunari SHIDAMA**[1]
[1]Shinshu University, 4-17-1 Wakasato Nagano-city, Nagano 380-8553, Japan

Abstract— *The Mizar project focuses on the formalization of mathematical theorems and their proofs using a formal descriptive language. It is an international joint project concerning the construction of a library system that will make automatic verification possible by a computer. The purpose of this study is to develops contents for distance education programs in advanced university mathematics using Mizar, and here we report on the current situation of this study.*

Keywords: Formal Verification, Mizar, Euclidean Algorithm

1. Introduction

When teaching mathematical content to students who have varying levels of knowledge and understanding, making the material perfectly understandable without lapses in logic can be exceedingly difficult. This is not simply limited to the topic of logic algorithms covered in the present study. Even if an extensive textbook were to be used, a student would still need to make an effort to “read between the lines," and a teacher would need to teach in accordance with the student’s level of understanding. In addition, to confirm and consolidate that understanding, in all likelihood, it would also be necessary for the students solve some part2.1 of a proof problem and completely describe the logical steps taken. Is it truly possible for teachers to explain the material in the text to each student individually, correct and edit solutions submitted by all students, and follow up according to instructions until the student’s answers are complete? As a solution to these issues, we have been researching and developing a system of teaching materials that uses the formal mathematical descriptive language of the Mizar processing system to support teachers.[1][1] Mizar is an international joint project in which current work includes using formal mathematical descriptive language to provide formal descriptions of present mathematical theorems and their proofs, thereby forming a system of automatic checking via a computer and creating a library.[2] The authors are participants in this project, and the system of teaching materials introduced in this study is being developed using the achievements of Mizar as a resource. We developed the formalized library on the Euclidean algorithm;[3] it is also recorded in the Mizar Project Library and is publically available at

http://mizar.uwb.edu.pl/version /current/html/ntalgo_1.html

These libraries store complete documentation of every single result and proof and link all necessary theorems to the mathematical system of axioms from which they are derived. They are presented as self-complete libraries that use formal mathematical descriptive language and have no need for outside referencing. Upon clicking the “proof" button on the webpage, every line of the proof is displayed. Furthermore, links for the definitions of all theorems and terminology used in each library are included, so that by continually linking back to definitions, one can return back to the axiom system of set theory. The Mizar system is based on Tarski-Grothendieck (TG) set theory and first-order predicate logic. Using a formal descriptive language based on this, Mizar documents formal proofs of theorems and automatically detects mistakes in a proof by using a processing system known as the Mizar Proof Checker. Our current research aims to set up a system of teaching materials that will integrate both the libraries and proof checker with a general-use CMS system.(CMS system + Mizar)[4][5][6] A group of students used this system. We present the data and discuss the lessons learned.

2. MANUSCRIPT PREPARATION

In the process of developing the teaching materials, it is necessary to have a text where all contents of the teaching materials are formally described, their validity is expressed as mathematical propositions, and descriptions of complete proofs are provided. The full formal developed in this study are available at http://mizar.uwb.edu.pl/version/current/html/ntalgo_1.html . Because of presentational constraints, from this point on, please refer to the above URL for details whenever “reference webpage" is mentioned. As mentioned in the Introduction, when clicking on the “proof" button on the webpage, one can read the complete proof. Furthermore, links are included for all terminology and current theorems used in the library. In this section, we provide an overview of how this works.[7][8] The Euclidean algorithm is a well-known algorithm where, given two arbitrary integers a and

b, the greatest common divisor is obtained through repeated calculation. An example implementation in Python is shown, as Code 2-1. There are many ways to express this algorithm

Table 1: Code 2-1

```
def gcd(a, b):
  Return the greatest common
  divisor of 2 integers a and b
  a, b = abs(a), abs(b)
  while b:
  a, b = b, a % b
  return a
```

in formal mathematical descriptive language. A simple way is to use the given integers a, b- to be exact, their absolute values abs (a) and abs (b),respectively- as initial values and then express them in terms of two successive constructive procedures, shown below, in the form of progressions of natural numbers.(Mizar Code 2-2) The equal to sign (=)

Table 2: Mizar Code 2-2

```
A . 0 = abs a
B . 0 = abs b
A . (i + 1) = B . i
B . (i + 1) = (A . i) mod (B . i)
( for all natural number i )
```

is a symbol literally expressing that the terms on both sides of the sign are equal to each other, unlike in the C programming language, where the value of the variable on the right replaces that of the variable on the left. Thus, in the process of finding the greatest common divisor of a, b, we took the operating variables temporarily used to keep the results in memory and expressed them as progressions with subscripts that indicate the number of times the calculations are successively repeated. Furthermore, to express the stopping conditions for the aforementioned algorithm, as well as express what equates to the return value of the function (e.g., in a programming language) obtained when the stopping conditions are satisfied, we introduced the set of all subscripts “i" of the number progression B satisfying the stopping conditions, in other words, B. i = 0 (Mizar Code 2-3) and minimum natural number belonging to this subset of

Table 3: Mizar Code 2-3

```
{ i where i is natural number
such that B.i = 0}
```

natural numbers (Mizar Code 2-4) giving it the value “A.i0."

Table 4: Mizar Code 2-4

```
i0= min { i where i is natural
number such that B.i = 0}
```

Here i0 is the first subscript- the natural number indicating the number of times the algorithm is repeated- to satisfy the stopping conditions. Below is the formal expression using Mizar language, which expresses the process for obtaining the greatest common divisor of a, b using the method mentioned above, by defining it as the functional application ALGO_GCD (a, b). Reference Mizar Code 2-5. INT and NAT, respectively, are expressions for all integers

Table 5: Mizar Code 2-5

```
definition
let a, b be Element of INT ;
func ALGO_GCD (a,b)
-> Element of NAT means
ex A, B being sequence of NAT st
( A . 0 = abs a \& B . 0 = abs b
\& ( for i being Element of NAT holds
( A . (i + 1) = B . i \& B . (i + 1)
= (A . i) mod (B . i) ) )
\& it = A . (min* { i
where i is Element of
    NAT : B . i = 0 }));
existence (WEB page reference)
uniqueness (WEB page reference)
end;
```

and natural numbers. “st" is an abbreviation for “such that." The English pronoun “it" is a natural number, and represents ALGO_GCD (a, b) itself (in other words, what equates to the return value of the function in a programming language). For the above definition to be mathematically valid, the natural number “it" satisfying the description above needs to exist and also needs to be uniquely identified in relation to a, b; Mizar calls these conditions “existence" and “uniqueness," respectively. In the formally described definition including the proofs which show that both can be established is necessary. This enables us to deductively prove that the computation algorithm is feasible for any given integers a, b, which always stops and gives some value as a return value. This alone, however, is not sufficient to inductively prove the validity of the above algorithm; the fact that ALGO_GCD (a, b), defined in relation to given arbitrary integers a, b, actually gives us the greatest common divisor of a, b is asserted below as a proposition and is proven. Reference ALGO_GCD.

Table 6: ALGO_GCD

```
theorem
for a, b being Element of INT holds
ALGO_GCD (a,b) = a gcd b
proof (WEB page reference)
end;
```

3. Teaching Materials and Practice Problems

As stated in Section 1 and explained in Section 2, the formal descriptions in

http://mizar.uwb.edu.pl/version
/current/html/ntalgo_1.html

are, similar to the definitions and theorems already in the Mizar library, a self-complete text without the need of outside references. Each theorem includes a complete proof. For any referenced or applied theorems, definitions, or terminology, links are posted to the reference source and the system of axioms in set theory can eventually be reached by following these links. We developed a textual commentary on this formalized account of the Euclidean algorithm as teaching material. This text commentary is stored in the general-use course management system Moodle. We highlight the substantial sections of this commentary below.

3.1 Feasibility of the Computational Algorithm

Given integers a, b, to show that the computational algorithm is always feasible, proving the existence, as well as uniqueness in relation to a, b, of natural number “it" as the return value of the function appearing in the formal definition Mizar Code 2-5 is necessary. For this, showing that the natural number progressions A and B satisfying the recurrence formula (2-2 reposted) can always be constructed from the

Table 7: Mizar Code 2-2

```
A . 0 = abs a
B . 0 = abs b
A . (i + 1) = B . i
B . (i + 1) = (A . i) mod (B . i)
( for all  natural number i )
```

given integers a, b is first necessary. The natural number progressions A, B are functions to themselves from the set of all natural numbers NAT, and the possibility of constructing A, B depends on the proof of existence of A, B by recursive functions based on the recurrence formula Mizar Code 2-2. To prove the existence of recursive functions, an existence theorem of functions, based on set theory axioms, is used. In the teaching materials, this is formalized using the scheme provided below, which corresponds to the recurrence formula Mizar Code 2-2 and general recursive definitions stored in the Mizar library. Reference Mizar Code RECDEF_2:sch 2. In the teaching materials, a detailed explanation is provided, mainly using the following lemma. (Mizar Code 3-1) For practice problems, to reinforce understanding of the recursive concept, a proof and description problem is given on the existence theorem of the number progression satisfying the recurrence formula that defines the Fibonacci sequence.

http://cai2.cs.shinshu-u.ac.jp/mizar
/moodle/mod/mizar/view.php?id=787

3.2 Stopping of the Computational Algorithm

In addition to showing the possibility of constructing (or the existence of) the natural number progressions A, B that

Table 8: Mizar Code RECDEF_2:sch 2

```
scheme :: RECDEF_2:sch 2
DoubleChoiceRec{
F1() -> non empty set ,
F2() -> non empty set ,
F3() -> Element of F1(),
F4() -> Element of F2(),
P1[ set , set , set , set , set ] } :
ex f being Function of NAT,F1()
ex g being Function of NAT,F2() st
( f . 0 = F3() \& g . 0 = F4()
\& ( for n being Element of NAT holds
P1[n,f . n,g . n,f . (n + 1),
  g . (n + 1)]))
provided
A1: for n being Element of NAT
for x being Element of F1()
for y being Element of F2()
ex x1 being Element of F1()
ex y1 being Element of F2()
st P1[n,x,y,x1,y1]
```

Table 9: Mizar Code 3-1

```
for a, b being Element of
INT ex A, B being sequence of NAT st
( A . 0 = abs a \& B . 0 = abs b
\& ( for i being Element of NAT holds
( A . (i + 1) = B . i \& B . (i + 1)
= (A . i) mod (B . i) ) ) )
```

satisfy the recurrence formula Mizar Code 2-2, it also needs to be shown that the repeated calculation algorithm stops at a finite number of iterations. For this, the existence of the set of all subscripts “i" of the natural number progression B satisfying the stopping condition B.i = 0 (2-3 reposted), as

Table 10: Mizar Code 2-3

```
{ i where i is natural number
 such that B.i = 0}
```

well as the existence of a minimum natural number belonging to this subset of natural numbers (2-4 reposted), must

Table 11: Mizar Code 2-4

```
i0= min { i where i is natural number such that B.i = 0}
```

be shown. For this, the well-known theorem “a minimum element exists for a non-empty set of natural numbers" is used. In addition, in terms of the existence itself of the set Mizar Code 2-3, there is a formal proof based on the axiom schema of separation in set theory. To show that this set is not empty, we use proof by contradiction and begin with the assumption that a natural number “i," which gives B.i = 0, does not exist. It is then shown that the natural number progressions A, B from the recurrence formula Mizar Code 2-2 are decreasing progressions, and as a result, A.i and B.i must have been negative values; thus, contradicting the fact that A.i and B.i are natural numbers. In the teaching material,

this section is explained using the following lemma, (Mizar Code 3-2) and the meaning of proof by contradiction is also

Table 12: Mizar Code 3-2

```
for a, b being Element of INT
for A, B being sequence
of NAT st A.0
= abs a \& B.0
= abs b \&
( for i being Element of NAT holds
(A . (i + 1) = B . i \& B . (i + 1)
= (A . i) mod (B . i))) holds
{ i where i is Element of NAT :
B . i = 0 } is
non empty Subset of NAT
```

discussed. For practice problems, we provided fill-in-the-blank questions based on the description of this proof and a proof by contradiction proof problem involving a proposition related to simple set operations.

http://cai2.cs.shinshu-u.ac.jp/mizar
/moodle/mod/mizar/view.php?id=792

3.3 Uniqueness of the Return Value in Relation to a, b

Under the formal definition Mizar Code 2-5, the uniqueness of natural number “it" (the return value of function ALGO_GCD (a, b)) in relation to a, b, comes down to its uniqueness in relation to initial values A.0 = abs (a) and B.0 = abs (b) of the natural number progressions A and B satisfying the recurrence formula Mizar Code 2-2. For this, it must be proven that the two pairs of natural number progressions A, B1 and A2, B2, both satisfying the same initial conditions

```
A1.0= abs( a), B1.0= abs(b),
A2.0= abs( a), B2.0= abs(b),
```

and having been constructed by the recurrence formula Mizar Code 2-2, are consistent as functions to themselves from the set of all natural numbers NAT- in other words, as functions, they must satisfy verbatims A1 = A2 and B1 = B2. To this end, it must be shown that the verbatims A1.i = A2.i and B1.i = B2.i hold for any arbitrary natural number “i" and that mathematical induction is used in relation to “i." There are a number of mathematical induction schemes stored in the Mizar library, and in the teaching material presented here, we used the following Mizar Code 3-3 In the teaching material, commentary is provided on this section, which includes a number of different forms of mathematical induction. As practice problems, we provided a fill-in-the-blank problem based on the proof in this section, and, to ensure that students become more familiar with mathematical induction proofs, we also provided a proof problem involving progressions of natural numbers, with the proposition shown here: Figure 2

Table 13: Mizar Code 3-3

```
scheme
NatInd{ P1[ Nat] } :
for k being Nat holds P1[k]
provided
A1: P1[ 0 ] and
A2: for k being Nat st P1[k]
holds P1[k + 1]
```

http://cai2.cs.shinshu-u.ac.jp/mizar
/moodle/mod/mizar/view.php?id=784

The students are asked to fill in a section of the proof in the blank space. The system performs a check, and students can continue studying by themselves until no errors remain.

3.4 Proving that the Return Value, ALGO_GCD (a, b), Provides the Greatest Common Divisor of a, b

As previously mentioned, to deductively prove the validity of the algorithm, it needs to be proven that ALGO_GCD (a, b), defined in terms of the given arbitrary integers, actually gives the greatest common divisor of a, b; for this, a proof is provided with the proposition shown below. (Mizar Code NATLGO 1:2) The greatest common factor “a gcd b" of

Table 14: Mizar Code NATLGO_1:2

```
theorem
for a, b being Element of INT holds
ALGO_GCD (a,b) = a gcd b
```

a, b is defined in the Mizar library as follows : (Mizar Code a gcd d) “Nat" and “Integer" are the variable forms of

Table 15: Mizar Code a gcd d

```
definition
let a, b be Integer;
func a gcd b -> Nat means
( it divides a \& it divides b \&
( for m being Integer st m divides a
\& m divides b holds m divides it ));
existence
uniqueness
commutativity ;
end;
```

natural numbers and integers, respectively. The pronoun “it" represents “a gcd b" itself (as previously noted) and equates to the return value of the function in programming language. In addition, the predicate “x divides y," defined in terms of integers x, y, is defined as follows. (Mizar Code x divides y) To prove “ALGO_GCD (a, b) = a gcd b" of the theorem above, it first needs to be shown that

```
A.i gcd B.i = A.(i+1) gcd B.(i+1)
```

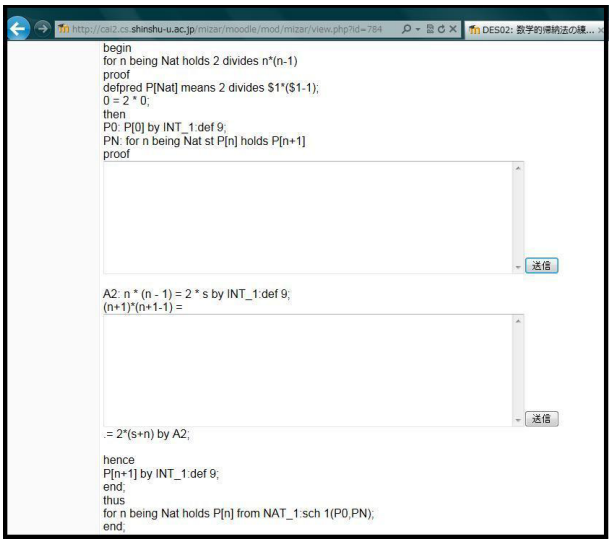

Fig. 1: An example from the teaching material

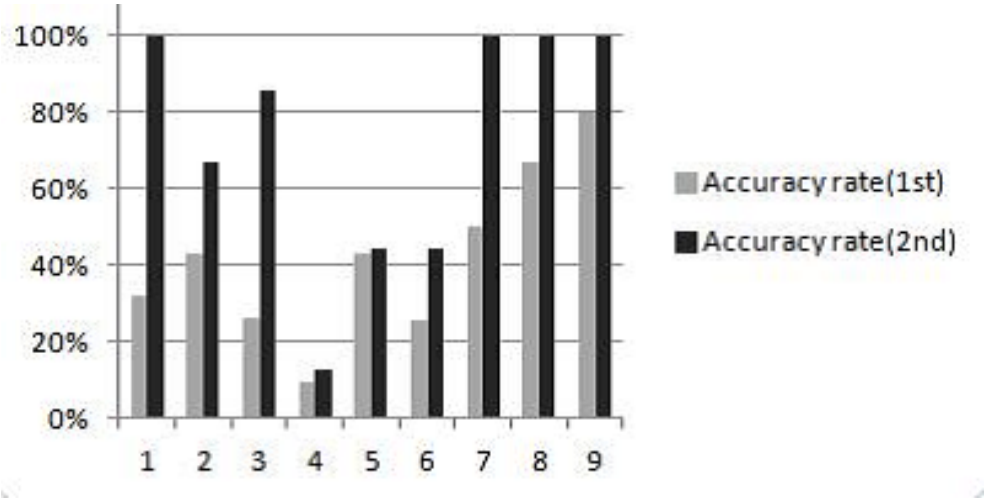

Fig. 2: score of accuracy rate

Table 16: Mizar Code x divides y

```
definition
let i1, i2 be Integer;
pred i1 divides i2 means
ex i3 being Integer st i2 = i1 * i3;
reflexivity
end;
```

holds for any arbitrary “i" of the natural number progressions A, B, obtained by the recurrence formula Mizar Code 2-2. It then needs to be shown that because of this,

```
A.0 gcd B.0 = A.i gcd B.i
```

holds, and finally, when the stopping condition B.i = 0 is satisfied,

```
A.i gcd B.i = A.i gcd 0 = A.i
```

holds. For this, we use (Mizar Code NAT_D:28) stored in

Table 17: Mizar Code NAT_D:28

```
theorem :: NAT_D:28
for m, n being Nat st m > 0 holds
n gcd m = m gcd (n mod m)
```

the library, and the lemma (Mizar Code 3-4) along with

Table 18: Mizar Code 3-4

```
LM6: for a being Element of
NAT holds a gcd 0 = a
```

mathematical induction. For this section, teaching materials

are still being created, and we are planning to include commentary on both mod operations and various properties of the gcd.

4. Evaluation of the Teaching Material and Tasks for the Future

As a trial, approximately 20 undergraduate and graduate computer science students studied and solved practice problems from the above teaching materials. Table 19 below outlines the topic and score of the practice problems for each subsection in Section 3. In addition, after one week, to verify whether there is an effect on education, we conducted a second experiment with some of the same problems but different values as parameters. For Questions 5-7 of subsection 3.1, we asked students to sketch a number of simple proofs using regular mathematical expressions and in the second experiment, the questions required knowledge of how to express proof lines in the Mizar language. Fig.3

Table 19: Score of accuracy rate

Section	Content		1st	2nd
3.1	The definition induction	Q.1	32.2%	100%
		Q.2	43.7%	66.7%
		Q.3	25.9%	85.7%
		Q.4	9.3%	12.5%
		Q.5	42.8%	44.4%
		Q.6	25.0%	17.4%
		Q.7	26.1%	11.4%
3.2	Proof by contradiction	Q.1	35%	30.8%
		Q.2	25.3%	44.4%
3.3	Mathematical of a recursive function	Q.1	50%	100%
		Q.2	66.7%	100%
		Q.3	80.0%	100%

shows that the accuracy rate of problem solving increased for most questions, affirming the potential of the system as an effective educational tool. For the proof-sketching questions(3.1:5-7), students had some trouble translating mathematical knowledge into the Mizar language, indicating the need for a longer introductory training period for learning Mizar expressions. As mentioned in Section 1, the novelty of this teaching material lies in the fact that a formal mathematical language and a processing system are used to help students write their proofs and a computer is used to automate the explanation of mistakes in an attempted proof. The formal descriptive language used in the teaching materials and practice problems of this study are for students who are majoring in computer science or related fields, who are already familiar with programming languages, should not have any trouble. However, for the student who is relatively unfamiliar with programming languages, continued efforts are needed to lessen the burden of learning the formal language, so that they can focus on understanding and mastering content. In the future, considering measures such as replacing the terminology of formal descriptions with Japanese language might be worth.

References

[1] Takaya Ido, Hiroyuki Okazaki, Hiroshi Yamazaki, Yasunari shidama, Content development for distance education for advance university mathematics by mizar, IEICE Technical Report ET2012-55 (2012-11), pp.13-17,2012.

[2] Piotr Rudnicki, An Overview of the MIZAR Project, WORKSHOP ON TYPES FOR PROOFS AND PROGRAMS, pp.311-329,1992.

[3] Tatsuo Miwa, Katsumi Wasaki, Noboru Endo, Yasunari Shidama, A Development of CMS/Moodle Assignment Module for Interactive Mathematical Exercises by using Mizar Proof Checking System in Formalized Mathematics, IEICE Tech. Rep., vol. 108, no. 247, ET2008-41, pp.11-16, 2008.

[4] H.J.Hoover and P.Rudnicki, Mizar MSE:making freshman logic formal, http://www.cs.ualberta.ca/˜piotr/Mizar-MSE/leaflet.ps, 1992.

[5] Hirofumi FUKURA, A practical training methods for the formal logic with proof checker "Mizar-MSE" -Education and training that uses "baby Mizar"-, Proceedings of the Technical Symposium and General Assembly of Mizar JAPAN(Spring 2007), pp.13-19,2007.

[6] Hirofumi FUKURA, Integrated environment for proof checker Mizar-MSE, Proceedings of the Technical Symposium and General Assembly of Mizar JAPAN(Autumn 2008), pp.1-9,2008.

[7] Yatuka Nakamura, Toshihiko Watanabe, Yasushi Tanaka, Pauline Kawamoto, Mizar Lecture Notes (4th Edition, HTML), http://markun.cs.shinshu-u.ac.jp /kiso/projects/proofchecker/mizar /Mizar4/printout/mizar4en_prn.pdf 14 December 2012.

[8] V. Shoup, A Computational Introduction to Number Theory and Algebra, http://www.shoup.net/ntb/ 14 December 2012.