J Autom Reasoning (2015) 55:257–268
DOI 10.1007/s10817-015-9331-7

Definitional Expansions in Mizar

In memoriam of Andrzej Trybulec, a pioneer of computerized formalization

Artur Korniłowicz[1]

Received: 14 October 2013 / Accepted: 25 May 2015 / Published online: 30 June 2015

Abstract The MIZAR VERIFIER uses definitional expansions for controlling proof structures. In this paper we propose another use of definitional expansions—enriching verified inferences by expansions of definitions of formulae included in the inferences and increasing the number of premises accessible by CHECKER. This introduces more knowledge to the reasoning, which helps to draw more conclusions. Some statistics about influence of such expansions on the MIZAR MATHEMATICAL LIBRARY are presented.

Keywords Proof assistant · Formal verification · Definitional expansion · MIZAR

1 Motivation

One of the aspects of the evolution of the language of mathematics is the development of a conceptual apparatus. An important question is which objects or collections of objects are worth to have own names. In other words, definitions of which objects should be introduced to the language. One extreme answer could be that everything should be named. However, some researchers treat definitions as an investment in mathematics, while theorems as profits from mathematics. Following this opinion, definitions should be set up carefully. Another question concerns communication between defined objects and their definitions. It is an especially important issue in the case of proof assistants, computer systems for formal proofs management, when their efficiency is desirable. Some related questions are: "How much automation should be implemented?", and "Which definitions should be expanded automatically to facilitate reasonings, but not to burden a given system too much, making it unusable in practice?"

✉ Artur Korniłowicz
arturk@math.uwb.edu.pl

1 Institute of Informatics, University of Białystok, K. Ciołkowskiego 1M, 15-245, Białystok, Poland

So far, in the MIZAR system [11, 23], definitional expansions were understood as a method of leading reasonings following natural deduction in the Jaśkowski style, with so called fixed variables (local constants) [17].

In this paper we propose another utilization of definitional expansions by the MIZAR VERIFIER. An enhancement of the computational power of MIZAR CHECKER by automatic expansions of definitions of the notions included in processed inferences is presented. Some dangers of using definitional expansions, mainly those related to the efficiency of the system, are discussed. Possible solutions and hints how to use and control the new mechanism effectively are shown as well.

The paper is structured as follows: Section 2 is a brief description of the MIZAR system focused on its basic modules. Section 3 is a short overview of definitions and redefinitions in MIZAR. Section 4 shows how using definitional expansions influences proof structures. Section 5 describes how MIZAR CHECKER processes definitional expansions. Simple examples are shown. Section 6 presents results of experiments how definitional expansions used by CHECKER impacts the MML. Section 7 presents related efforts in other systems. Section 8 discusses directions of further developments of expansions in the MIZAR system.

2 About Mizar

MIZAR [11, 23] is a proof assistant used for computerized verification of mathematical proofs. MIZAR is also a common name for the components of the system: a formal language derived from the mathematical vernacular and dozens of computer programs responsible for formal verification, optimization of proofs [24, 25]. An integrant part of the system is the Mizar Mathematical Library (MML), a collection of articles written in the MIZAR language. Knowledge stored in the database is used in various branches of science and education, e.g. for representing mathematics on WWW [16, 27], as an input for ATP systems [28], as an input for services classifying mathematics [14], and others.

The main programs of the MIZAR system used at the stage of writing new submissions are:

- ACCOMODATOR importing resources at the environment level (preamble) of the created article from the MML;
- VERIFIER checking the correctness of reasoning;
- EXPORTER exporting the results achieved in the given article to the MML for use in subsequent articles to be written in the future;
- ABSTRACTOR generating a brief representation of all statements in an article without their justifications.

The most important and most advanced program is VERIFIER consisting of several modules responsible for verifying different aspects of articles. Its main modules are:

- SCANNER and PARSER doing lexical analysis, checking validity of the article against the MIZAR grammar and creating parse-tree of a given article, [7, 8];
- ANALYZER doing type-checking, identification and disambiguation of used symbols, resolving hidden arguments;
- REASONER controlling the structure of reasonings;
- PRECHECKER transforming entry inferences into their disjunctive normal form and processing Skolem constants (constants generated by basing on existential premises);

- EQUALIZER computing the congruence closure of the universe of discourse collected from the given inference, [19];
- UNIFIER performing unification.

VERIFIER is a standalone program completely designed and implemented by the MIZAR developers. However, there are successful trials to involve external algorithms provided by other systems to perform some particular tasks during processing MIZAR texts [20, 21].

3 Definitions and Redefinitions in Mizar

The MIZAR language supports four kinds of expressions: *types* (e.g. `Group`), *terms* (e.g. `x+3`), *formulae* (e.g. `x in {x}`), and *adjectives* (e.g. `finite`). Every expression is an instantiation of a particular constructor, which serve for defining new notions extending the language. Constructors used for defining types are *modes*, for terms are *functors*, for formulae are *predicates*, and for adjectives are *attributes*.

All new notions are introduced within a *definitional block*. Within such a block there may appear:

- arguments (if required) with their corresponding types, called *loci*;
- possibly some assumptions for the definition;
- the main part of the definition: notation, result type (required for modes and functors) and definiens;
- a label used for referring to the definition;
- correctness conditions if they are needed to guarantee the soundness of the definition;
- properties of the introduced notion (e.g. commutativity of an operation, or reflexivity of a relation), if applicable.

For example, the inclusion of sets can be introduced as it is shown in Fig. 1 and the inverse of a relation in Fig. 2.

The main aim of using definitions is defining new objects and notions. Each definition determines a notation used to write down introduced concepts (the MIZAR language supports prefix, infix and suffix forms), as well as the meaning of defined notions (definientia). New definitions can encapsulate information and definiendum can serve for writing more concise and readable texts. But, the MIZAR system provides also more elaborate usage of definitions. Information provided by MIZAR definitions can be used by different modules of the system at different stages of verifying articles. For example, definientia determine possible proof skeletons controlled by REASONER, and properties increase computational power of EQUALIZER, [22].

Mathematical practice shows that one notion can be defined in several ways [12]. For example, lattices can be defined as algebraic structures with two binary operations satisfying some axioms or as posets with infima and suprema [15]. Moreover, notions declared for some arguments could be expressed in different ways when the arguments are of more precise types, e.g. properties of relations could be defined in terms of ordered pairs, but also in terms of simple elements.

From a technical point of view, every definition introduced in a MIZAR article defines a new constructor. For example, the inclusion of relations defined in terms of pairs and the inclusion of relations defined in terms of elements would be two different constructors (two different notions). Communication between these two constructors (explaining that at the end they are the same inclusion) could be realized by theorems, which would be explicitly referred to when such justification is required. In situations when the definiens depends on

Fig. 1 Definition of the inclusion of sets

```
definition
  let X,Y be set;
  pred X c= Y means :: TARSKI:def 3
  for x being object st x in X holds x in Y;
  reflexivity;
end;
```

types of arguments of the defined notion, the MIZAR art recommends using *redefinitions*. For example, the inclusion defined for sets (Fig. 1) can be redefined for relations as it is shown in (Fig. 3):

MIZAR VERIFIER supports some automations of processing of redefined notions. Admittedly, ANALYZER identifies the redefined version and the original version as two different notions, but CHECKER adjusts the redefinition to the original version making them equivalent.

Detailed explanation of using redefinitions is given in [11].

4 Definitional Expansions in the Reasoner

The MIZAR system supports logical reasoning in the Jaśkowski style of the natural deduction [17]. Structures of proofs are related to the structures of the formulae to be proven, one to one correspondence between statements and their proof skeletons can be observed. But sometimes it is better to trace a reasoning not following the structure of the statement, but the structure of the definition of the main symbol of the statement, that is to exploit a *definitional expansion*. An advantage of using definitional expansions can be seen, for example while comparing two alternative proofs of the monotonicity of the inverse of relations with respect to the inclusion of the relations:

```
P c= R implies P~ c= R~
proof
 assume
A1:  P c= R;
 let x;
 assume
A2:  x in P~;
 then consider a,b such that
A3: x = [a,b] by RELAT_1:def 1;
 [b,a] in P by A2,A3,RELAT_1:def 7;
 then [b,a] in R by A1;
 hence thesis by A3,RELAT_1:def 7;
end;
```

```
P c= R implies P~ c= R~
proof
 assume
A1:  P c= R;
 let a,b;
 assume [a,b] in P~;
 then [b,a] in P by RELAT_1:def 7;
 then [b,a] in R by A1;
 hence thesis by RELAT_1:def 7;
end;
```

where `RELAT_1:def 1` [30] is the definition of a relation and `RELAT_1:def 7` defines the inverse of a given relation. The left proof uses the expansion of the inclusion of relations `P` and `R` treated as just sets (Fig. 1), while the right proof uses the expansion of the inclusion

```
definition
  let R be Relation;
  func R~ -> Relation means :: RELAT_1:def 7
  for x,y being object holds [x,y] in it iff [y,x] in R;
  correctness;
  involutiveness;
end;
```

Fig. 2 Definition of the inverse of a relation

```
definition
  let P,R be Relation;
  redefine pred P c= R means
  :: RELAT_1:def 3
  for a,b being object holds [a,b] in P implies [a,b] in R;
  compatibility;
end;
```

Fig. 3 Definition of the inclusion of relations

of relations `P` and `R` treated as pairwise sets (Fig. 3). It is quite obvious that the right proof is better than the left one, because introduction of variables `a` and `b` is forced by the proof skeleton, while in the left proof they are obtained by decomposition of the variable `x`.

5 Definitional Expansions in the Checker

As it has been shown in previous sections, MIZAR VERIFIER uses definitional expansions for controlling proof structures. In this section we propose another use of definitional expansions—enriching verified inferences by expansions of definitions of formulae included in the inferences and increasing the number of premises accessible by CHECKER.

5.1 Inference Checker

One of the basic tasks of proof checkers is checking obviousness of proof steps [9]. In the MIZAR context this relies on checking obviousness of a single inference

$$\frac{\phi_1, \phi_2, \ldots, \phi_n}{\psi}$$

where the formulae ϕ_i are premises and ψ is the conclusion. Because MIZAR is a disprover, the conclusion is negated and added to the premises, so such an inference is translated to

$$\frac{\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \wedge \neg\psi}{\bot}$$

Now we are at the stage where the proposed extension of the system—expanding of formulae—can be applied. The sign of each conjunct, say α, of the above formula is analyzed. If it is positive, i.e. when α is a premise, then the conjunct is expanded to $\alpha \wedge \hat{\alpha}$, and if the conjunct is negative, i.e. when $\neg\alpha$ is a premise, then it is expanded to $\alpha \vee \hat{\alpha}$, where $\hat{\alpha}$ is the definitional expansion of the α. Expanding a formula conjunctively or disjunctively depending on its sign (*dual instantiation*, [5]) creates more premises accessible to VERIFIER. Still, the satisfiability of the original formula is preserved, since α is satisfiable if and only if $\alpha \wedge \alpha$ is satisfiable, and α is satisfiable if and only if $\alpha \vee \alpha$ is satisfiable.

Now, the question is whether all accessible formulae should be expanded and whether expansions of initial formulae should be expanded, and so on. Similar questions had been already asked, for example, by Larry Wos, who considered definitional expansions as one of the 33 basic research problems of formal verification and automated reasoning [31, 32]. The answer to these questions is not obvious. Expanding everything to primitives introduces much knowledge to the reasoning, which can help to draw more conclusions. But clearly, in proof assistants used in practice [26, 29], where their efficiency is one of the crucial points (users should wait for answers from systems for reasonable amount of time), the fast growing size of fully expanded theories is an important factor in slowing down the processing [18].

Taking efficiency issues into account, in the implementation of the proposed algorithm within VERIFIER we decided to expand only one level and only formulae involving only constants. This means that general quantified formulae are not expanded.

5.2 Examples

To demonstrate the behavior of VERIFIER let's analyze two simple examples.
Example 1:

```
for A being set holds A c= A;
```

The above formula is translated internally to $\exists_A A \not\subseteq A$. Next, a Skolem constant A is introduced. The remaining statement to be proved is $A \not\subseteq A$. The statement uses only constants (A), so it can be expanded. Because the statement is negative, its expansion results in $\neg(A \subseteq A \vee \forall_x x \in A \Rightarrow x \in A)$, which is equivalent to $A \not\subseteq A \wedge \exists_x x \in A \wedge x \notin A$. Another Skolem constant x is introduced. And we get the contradiction $x \in A \wedge x \notin A$, which finishes the proof.
Example 2:

```
ex A being set st A c= A;
```

The above formula is internally transformed to $\forall_A A \not\subseteq A$. It is a general quantified formula, so VERIFIER does not expand it, and because there are no other variables around, it is not verified positively. A proof is required.

```
ex A being set st A c= A
proof
  take A = the set;
  thus A c= A; :: (*)
end;
```

The formula marked `(*)` is negated and internally transformed to $A \not\subseteq A$. It uses only constants (A is introduced by `take`), so it can be expanded. From now on, processing similar to that of Example 1 holds.

5.3 Some Technical Details

In this section we discuss some technical details important for active users of the MIZAR proof assistant writing MIZAR articles.

5.3.1 Environment Directives

Environment directive `definitions` is used for importing two different kinds of information from the database: definitional expansions used by REASONER and expansions of terms defined by `equals` used by EQUALIZER. It would be quite natural for CHECKER to reuse the same definitional expansions as REASONER, but experiments with using definitional expansions by CHECKER have shown that in some cases there were too many expansions to be exploited, which slows down the performance of the system. It was decided to introduce a new environment directive `expansions`, which imports definitions only for CHECKER.

It allows better controlling efficiency of the system. Moreover, the meaning of the original directive `definitions` has been changed. Its new version imports definitional expansion for REASONER only. Expansions of terms defined by `equals` are imported by a new directive `equalities`.

5.3.2 Predicative Formulae and Attributive Formulae

MIZAR syntax provides three kinds of atomic formulae: predicative, attributive (of the structure "*term* `is` *adjective*") and qualified (of the structure "*term* `is` *type*"). Because types can be defined using adjectives as well, in first implementations and tests of the new treatment of definitional expansions, it was decided not to expand qualified formulae (again for efficiency reasons). Only predicative formulae and attributive formulae are expanded.

5.3.3 Permissiveness

Another limitation of using definitional expansions by CHECKER are permissive definitions, that is, definitions with assumptions required to prove their correctness conditions. Expansions of such definitions demand justification of the assumptions. To guarantee that all conditions of permissive definitions are fulfilled, VERIFIER searches for statements confirming the definitions among all accessible premises. But in a general case it is not certain that adequate premises are present in the search space. This is the reason as to why permissive definitions are not expanded by CHECKER automatically.

5.3.4 Modified Modules

Full implementation of the proposed extension of the system required modification of several modules of VERIFIER. Because one new word (`expansions`) has been introduced to the MIZAR language, SCANNER and PARSER implement a new parsing rule to process the environment directive. Moreover, units preparing XML representation of articles [27] are adjusted. Creating a new environment directive caused modification of ACCOMODATOR importing definitions from the database. Essential changes are done in PRECHECKER. Procedures selecting definitions which can be expanded (non permissive definitions of predicates and attributes) from among all definitions accessible at a given part of the article are implemented. Procedures processing attributive formulae and predicative formulae are modified. Whenever PRECHECKER meets a predicative formula or attributive formula, it expands the used in the formulae notions and creates appropriate conjunctions or disjunctions in accordance with the description in Section 5.1. Such formulae are next processed by EQUALIZER and UNIFIER in the standard way.

6 Experiments

Modules responsible for processing definitional expansions by MIZAR CHECKER were incorporated into the MIZAR Version 8.1.01. The implemented software was tested on the MML Version 4.128.1067.

6.1 Profits

Definitional expansions utilized in the MIZAR system have big influence on the MML. All the improvements resulting from the application of definitional expansions have been implemented in the library.

In next subsections we describe main gains obtained as a result of using definitional expansions in CHECKER.

6.1.1 *External References*

To refer to some facts stored in the MML, authors can use labels of these facts. In all MIZAR articles there were 603 558 references to external facts. RELPREM, which is a special program dedicated to finding unnecessary references, reported 40 689 such cases. It means that authors had to 40 689 times (6.7% of all references) wonder how to justify some statements and remember (or search using some tools [2, 3]) the labels of useful theorems or definitions, whereas now CHECKER can prove these statements automatically.

6.1.2 *Trivial Proofs*

Proofs of some theorems stored in the MML have similar, and in some cases just the same, structure. An example is

```
{} c= A
proof
  let x; :: begins definitional expansion of inclusion
  thus thesis; :: thus x in {} implies x in A;
end;
```

Fixed variable x is introduced just to force using a particular definitional expansion, the expansion of the inclusion of sets. Such proofs are no longer required. TRIVDEMO, a tool for detecting unnecessary proofs, reported 2099 such cases.

6.1.3 *Obvious Theorems*

An important gain from using definitional expansions by CHECKER is that some theorems became obvious. The information brought by new premises causes activating some mechanisms (e.g. generating Skolem constants) of the system that enable us to deduce required conclusions.

An example of a theorem, which is now obvious, but in the MML was referred to 6061 times, is transitivity of the inclusion of sets. The statement without expansions is

$$\frac{A \subseteq B \;\wedge\; B \subseteq C}{A \subseteq C}$$

Expanding three inclusions generates the formula

$$\frac{\begin{array}{c} A \subseteq B \;\wedge\; (\forall_x x \in A \Rightarrow x \in B) \\ \wedge \\ B \subseteq C \;\wedge\; (\forall_x x \in B \Rightarrow x \in C) \end{array}}{A \subseteq C \;\vee\; \forall_x x \in A \Rightarrow x \in C}$$

which after negating the conclusion and moving it to the premises, the new formula is

$$\frac{\begin{array}{c} A \subseteq B \ \wedge \ (\forall_x x \in A \Rightarrow x \in B) \\ \wedge \\ B \subseteq C \ \wedge \ (\forall_x x \in B \Rightarrow x \in C) \\ \wedge \\ A \nsubseteq C \ \wedge \ (\exists_x x \in A \wedge x \notin C) \end{array}}{\bot}$$

The conjunct $\exists_x x \in A \wedge x \notin C$ allows introducing a constant x such that $x \in A$ and $x \notin C$. Because $x \in A$, then $x \in B$, and then $x \in C$, which results in the contradiction with $x \notin C$.

6.2 Problems and their Possible Solutions

In this section we describe main problems predicted during the design stage of extending the functionality of MIZAR CHECKER by processing definitional expansions, which were indeed met during its implementation into the MIZAR system and into the experiments with the MML.

6.2.1 Redefined Notions

Some mathematical notions can be defined using a different apparatus. For example, equality or inclusion of functions can be defined in terms of function assignments or ordered pairs (functions are relations) or simply elements (functions are sets as well). It means that one definition could be expanded in different ways and could generate several formulae for each notion. This would cause fast growing of the number of processed statements. Practical systems should have some mechanisms preventing such time consuming processing. In the case of the MIZAR system it can be controlled by the environment of articles, described in Section 5.3.

6.2.2 Better Object Typing

One of the axioms adopted by the MML states that every object is a set. Moreover, the type `set` is the root of the type tree of all types defined in the MML. Therefore, all notions defined or redefined for sets, e.g. inclusion, equality, etc., can be applied to every term used in the MML. If one writes, for example, $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$, the equality is expanded into two inclusions $\frac{1}{4} + \frac{1}{4} \subseteq \frac{1}{2}$ and $\frac{1}{2} \subseteq \frac{1}{4} + \frac{1}{4}$, both definitely true, but rather meaningless.

To avoid such expansions, a new type (`object`) was defined in the MML and declared as the root of the type tree. Moreover, `object` was put as the default type of objects that intentionally are not meant as sets (e.g. numbers, structures), which isolates them from definitions and redefinitions declared for sets.

7 Related Work

Definitional expansions of mathematical concepts, methods and criteria of their exploitation in theorem provers and proof checkers are one of the important subjects of study in the automated reasoning domain [31, 32]. Different techniques, such as peeking (where expanding of the definition of a predicate P used in the conjecture to be proved is based on occurrences of symbols of predicates used both in the conjecture and in the definiens of the predicate P [6]) or gazing (an abstraction-based technique for choosing formulae from a theory for use in a proof, where formulae are selected on the basis of a global plan built in an abstract space [4, 10]), were implemented and tested in several systems.

A conclusion that could be learned from experiences of all known systems is that definitional expansions are a very strong mechanism playing an important role in proving and proof searching. However, they should be used carefully. Too deep expanding can lead to producing a huge number of premises, make the search space unmanageable and systems useless in practice [5, 18].

8 Further Work

Further development of the usage of definitional expansions in the MIZAR system can go in two directions: a) strengthening the power of CHECKER by expansion of greater number of definitions, and b) exploiting already implemented features in the MML.

Ad a) In the current implementation of using definitional expansions by CHECKER, only predicative formulae and attributive formulae that are explicitly stated in a given inference are expanded. A possible extension of CHECKER could analyze types of all variables considered in the inference and expand all adjectives involved in the types. However, such an approach could produce too many expansions (in the MML there are known cases when one type consist of more than 30 adjectives). A prospective protection against too large enlargement of inferences could be introducing some quantitative parameters controlling the number of adjectives included in the types.

Ad b) Extensions of functionalities of MIZAR VERIFIER are always tested on every MIZAR article collected in the MML. As it was shown in Section 6 definitional expansions processed by MIZAR CHECKER have important influence on the MML—some facts became obvious, some references unnecessary. These circumstances caused quite many simplifications of proofs. To induce more similar simplifications, a task for the future could be scanning the entire MML and detecting theorems that could be reformulated as redefinitions of particular notions. Such redefinitions could be then expanded and the expansions could generate more unnecessary references. On the other hand, when the processing of definitional expansions causes much of a slowdown of the performance of the system, e.g. due to over-redefined notions, such notions could be detected and their redefinitions could be restated as theorems. In both cases, a refactoring of the MML [13] would be required while maintaining licensing its content [1].

Acknowledgments I would like to thank Professor Andrzej Trybulec† for 20 years of our cooperation, for his continuous support of my work, for every inspiring and instructive talk. I thank Him for every lecture and every seminar, for every visit at His home where we discussed MIZAR issues. I am thankful for promoting my master thesis and Ph.D. thesis. I thank Him for His sense of humor, for all anecdotes and jokes.

References

1. Alama, J., Kohlhase, M., Mamane, L., Naumowicz, A., Rudnicki, P., Urban, J.: Licensing the Mizar Mathematical Library. In: Davenport, J.H. et al. (eds.) Proceedings of Calculemus/MKM 2011, LNCS, vol. 6824, pp. 149–163. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-22673-1_11 (2011)
2. Bancerek, G.: Information retrieval and rendering with MML query. In: Borwein, J. et al. (eds.) Mathematical Knowledge Management, LNCS, vol. 4108, pp. 266–279. Springer Berlin Heidelberg. doi:10.1007/11812289_21 (2006)
3. Bancerek, G., Urban, J.: Integrated semantic browsing of the Mizar Mathematical Library for authoring Mizar articles. In: Asperti, A. et al. (eds.) MKM 2004, Bialowieza, Poland, September 2004, Proceedings, LNCS, vol. 3119, pp. 44–57. Springer. doi:10.1007/978-3-540-27818-4_4 (2004)
4. Barker-Plummer, D.: Gazing: An approach to the problem of definition and lemma use. J. Autom. Reasoning **8**(3), 311–344 (1992)
5. Bishop, M., Andrews, P.B.: Selectively instantiating definitions. In: CADE-15, Lindau, Germany, July, 1998, Proceedings, LNCS, vol. 1421, pp. 365–380. Springer (1998)
6. Bledsoe, W.W.: The UT interactive prover. Memo ATP-17B, Mathematics Department. University of Texas (1983)
7. Byliński, C., Alama, J.: New developments in parsing Mizar. In: Jeuring, J. et al. (eds.) Intelligent Computer Mathematics 11th International Conference LNAI vol. 7362, pp. 427–431 Springer-Verlag Berlin Heidelberg (2012). doi:10.1007/978-3-642-31374-5_30
8. Cairns, P., Gow, J.: Using and parsing the Mizar language. Electronic Notes in Theoretical Computer Science **93**, 60–69 (2004). doi:10.1016/j.entcs.2003.12.028. http://www.sciencedirect.com/science/article/pii/S1571066104000131
9. Davis, M.: Obvious logical inferences. In: Proceedings of the Seventh International Joint Conference on Artificial Intelligence, pp. 530–531 (1981)
10. Giunchiglia, F., Walsh, T.: Theorem proving with definitions. In: Proceedings of AISB 89, pp. 433–435 (1989)
11. Grabowski, A., Naumowicz, A.: Mizar in a nutshell. J. Formal. Reasoning, Special Issue: User Tutorials I **3**(2), 153–245 (2010)
12. Grabowski, A., Schwarzweller, C.: Translating mathematical vernacular into knowledge repositories. In: Proceedings of MKM'05, pp. 49–64. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/11618027_4 (2006)
13. Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: Proceedings of Calculemus '07 / MKM '07, pp. 235–249. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-540-73086-6_20 (2007)
14. Grabowski, A., Schwarzweller, C.: Towards automatically categorizing mathematical knowledge. In: Ganzha, M. et al. (eds.) FedCSIS 2012, Wroclaw, Poland, September 2012, Proceedings, pp. 63–68 (2012)
15. Grätzer, G.: General Lattice Theory. Academic Press, New York (1978)
16. Iancu, M., Kohlhase, M., Rabe, F., Urban, J.: The Mizar Mathematical Library in OMDoc Translation and applications. J. Autom. Reasoning **50**(2), 191–202 (2013). doi:10.1007/s10817-012-9271-4
17. Jaśkowski, S.: On the Rules of Suppositions in Formal Logic. Studia Logica. Nakładem Seminarjum Filozoficznego Wydziału Matematyczno-Przyrodniczego Uniwersytetu Warszawskiego (1934). http://books.google.pl/books?id=6w0vRAAACAAJ
18. Kieffer, S., Avigad, J., Friedman, H.: A language for mathematical knowledge management. In: Grabowski, A. et al. (eds.) Computer Reconstruction of the Body of Mathematics, Studies in Logic, Grammar and Rhetoric, vol. 18(31), pp. 51–66. Białystok (2009)
19. Korniłowicz, A.: On rewriting rules in Mizar. J. Autom. Reasoning **50**(2), 203–210 (2013). doi:10.1007/s10817-012-9261-6
20. Naumowicz, A.: Interfacing external CA systems for Grobner bases computation in Mizar proof checking. Int. J. Comput. Math. **87**(1), 1–11 (2010). doi:10.1080/00207160701864459
21. Naumowicz, A. In: Watt, S.M. et al. (eds.): SAT-enhanced Mizar proof checking (2014). doi:10.1007/978-3-319-08434-3_37
22. Naumowicz, A., Byliński, C.: Improving Mizar texts with properties and requirements. In: Asperti, A. et al. (eds.) MKM 2004 Proceedings, LNCS, vol. 3119, pp. 290–301. doi:10.1007/978-3-540-27818-4_21 (2004)

23. Naumowicz, A., Korniłowicz, A., et al.: A brief overview of Mizar. In: Berghofer, S. (ed.) Proceedings of TPHOLs'09, *LNCS*, vol. 5674, pp. 67–72. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-03359-9_5 (2009)
24. Pąk, K.: Methods of lemma extraction in natural deduction proofs. J. Autom. Reasoning **50**(2), 217–228 (2013). doi:10.1007/s10817-012-9267-0
25. Pąk, K.: Improving legibility of natural deduction proofs is not trivial. Logical Methods in Comput. Sc. **10**(3), 1–30 (2014). doi:10.2168/LMCS-10(3:23)2014
26. Trybulec, A., Korniłowicz, A., Naumowicz, A., Kuperberg, K.: Formal mathematics for mathematicians. J. Autom. Reasoning **50**(2), 119–121 (2013). doi:10.1007/s10817-012-9268-z
27. Urban, J.: XML-izing Mizar: Making semantic processing and presentation of MML easy. In: Kohlhase, M. (ed.) MKM 2005, Bremen, Germany July 2005, LNCS, vol. 3863, pp. 346–360 Springer. doi:10.1007/11618027_23 (2005)
28. Urban, J., Hoder, K., Voronkov, A.: Evaluation of automated theorem proving on the Mizar Mathematical Library. In: Fukuda, K. et al. (eds.) ICMS 2010, Kobe, Japan. *LNCS*, vol. 6327, pp. 155–166. Springer. doi:10.1007/978-3-642-15582-6_30 (2010)
29. In: Wiedijk, F. (ed.): The Seventeen Provers of the World, Foreword by Dana S. Scott, *LNCS*, vol. 3600. Springer (2006)
30. Woronowicz, E.: Relations and their basic properties. Formalized Mathematics **1**(1), 73–83 (1990). http://fm.mizar.org/1990-1/pdf1-1/relat_1.pdf
31. Wos, L.: Automated Reasoning: 33 Basic Research Problems. Prentice-Hall, Englewood Cliffs. N.J (1987)
32. Wos, L.: The problem of definition expansion and contraction. J. Autom. Reasoning **3**(4), 433–435 (1987)