Mizar: State-of-the-Art and Beyond

Grzegorz Bancerek[1], Czesław Byliński[2], Adam Grabowski[3], Artur Korniłowicz[3], Roman Matuszewski[4], Adam Naumowicz[3(✉)], Karol Pąk[3], and Josef Urban[5]

[1] Association of Mizar Users, Białystok, Poland
bancerek@mizar.org
[2] Section of Computer Systems and Teleinformatic Networks, University of Białystok, Białystok, Poland
bylinski@mizar.org
[3] Institute of Informatics, University of Białystok, Białystok, Poland
{adam,arturk,adamn,karol}@mizar.org
[4] Department of Applied Linguistics, Faculty of Philology, University of Białystok, Białystok, Poland
romat@mizar.org
[5] Radboud University, Nijmegen, The Netherlands
josef.urban@gmail.com

Abstract. Mizar is one of the pioneering systems for mathematics formalization, which still has an active user community. The project has been in constant development since 1973, when Andrzej Trybulec designed the fundamentals of a language capable of rigorously encoding mathematical knowledge in a computerized environment which guarantees its full logical correctness. Since then, the system with its feature-rich language devised to approximate mathematics writing has influenced other formalization projects and has given rise to a number of Mizar modes implemented on top of other systems. However, the information about the system as a whole is not readily available to developers of other systems. Various papers describing Mizar features have been rather incremental and focused only on particular newly implemented Mizar aspects. The objective of the current paper is to give a survey of the most important Mizar features that distinguish it from other popular proof checkers. We also go a step further and describe most important current trends and lines of development that go beyond the state-of-the-art system.

1 Introduction

The Mizar [21,38] project is a long-term effort originally aimed at developing a computer environment to support mathematicians in preparing papers. Around 1973, Andrzej Trybulec, the leader of the project, has designed a language for writing formal mathematics. The implemented processor was intended to check written texts for logical consistency and correctness. For fifteen years numerous implementations of the system were developed in order to choose suitable underlying logic and expressive power of the language (PC – propositional calculus,

M. Kerber et al. (Eds.): CICM 2015, LNAI 9150, pp. 261–279, 2015.
DOI: 10.1007/978-3-319-20615-8_17

MSE – multi-sorted with equality, QC – quantifier calculus, etc.). An important issue was also to find the proper technology of automated cross-referencing between papers. The history of the first 30 years of MIZAR development has been described in [32]. All these experiments resulted in choosing the principles of the current MIZAR. The logical structure of the language is based on a variant of the classical first-order logic natural deduction system proposed by Jaśkowski [27]. The texts written in the language (called MIZAR articles) are organized into a data base – the MIZAR Mathematical Library, MML. The Tarski-Grothendieck set theory, which is basically ZFC set theory with the axiom of infinity replaced by Tarski's axiom of existence of arbitrarily large, strongly inaccessible cardinals forms the basis of doing mathematics in contemporary MIZAR.

Today there are a number of other projects committed to address problems related to computerized theorem proving developed at various research centers around the world. Most significantly: the HOL Light theorem prover[1], Isabelle[2], the Coq Proof Assistant[3], Metamath[4], ProofPower[5], Nqthm/ACL2[6], the PVS Specification and Verification System[7], and the Nuprl/PRL Project[8]. Each project is characterized by its own specifics implied by an assumed theoretical basis (e.g., type theory or set theory, classical logic versus intuitionistic logic or higher order logic) and main goals, towards which the project is geared (extracting programs from proofs, program verification, formalization of mathematics, automated theorem proving) [59]. Thanks to the discussions and research collaboration stemming from the QED [7] initiative, there has been a significant interplay between the projects. To name the most important cases of MIZAR's influence on other systems we can mention the Declare system developed by D. Syme [42], the MIZAR mode for HOL by J. Harrison [25], the Isar language for Isabelle by M. Wenzel [56], Mizar-light for HOL Light by F. Wiedijk [57], the declarative proof language (DPL) for Coq by P. Corbineau [16] and finally the `miz3` proof interface for HOL Light [61] that combines both the procedural and declarative style of writing proofs. The MIZAR way of writing proofs was also the model for the notion of 'formal proof sketches' developed by F. Wiedijk [58].

However, the information about the fundamental aspects of the MIZAR system as a whole has not been readily available. To have a better understanding of MIZAR, the developers of other systems have had to collect the information scattered in scarce user reference materials or various MIZAR research papers describing particular newly implemented MIZAR aspects. With this paper we intend to give a concise survey of the most important MIZAR features that distinguish it from other popular proof checkers and show its current lines of development. Hopefully, this will become beneficial for further collaboration with

[1] http://www.cl.cam.ac.uk/~jrh13/hol-light/.
[2] http://www.cl.cam.ac.uk/research/hvg/Isabelle/.
[3] http://coq.inria.fr.
[4] http://us.metamath.org.
[5] http://www.lemma-one.com/ProofPower/index/.
[6] http://www.cs.utexas.edu/users/moore/acl2/.
[7] http://pvs.csl.sri.com.
[8] http://www.nuprl.org.

other similar projects. The work is organized as follows. In Sect. 2 we present the basics of the Mizar language, in Sect. 3 we describe the main software components of the system. Section 4 presents the organization of MML. Next, we present the current Mizar development in Sect. 5 and conclude with a vision of the project for the near future in Sect. 6.

2 Language

The Mizar language encompasses both the grammar constructions that make use of a standardized list of reserved words and also the notation introduced by the authors of formalizations to encode concepts and notions.

In total, the current version of the language comprises 112 reserved words and 29 special symbols. However, the language is open in the sense that authors are allowed to extend it by introducing their own new symbols for the notions they define. Currently there are 8239 symbols used in the Mizar Mathematical Library, including 722 predicate symbols, 1771 attribute symbols, 854 mode symbols, 4501 functor symbols, 36 left and right bracket symbols, 159 selector symbols, and 160 structure symbols[9]. In the Mizar terminology, predicates are constructors of formulas, attributes are constructors of adjectives, modes are constructors of types, functors are constructors of terms. Selector and structure symbols are used to declare structural types and their fields. Symbol overloading is highly used in Mizar to enable re-use of the symbols to denote different notions in much the same way it is done in pen-and-paper mathematics. For example the symbols * and + are used to denote 193 and 143 different operations in various fields of mathematics, respectively. The grammar of the Mizar language[10] provides means to formulate mathematical statements in a way that resembles a formal exposition in the natural language. There are constructs to represent various kinds of formulas (quantifiers, standard logical connectives), reasoning methods (straightforward and diffused reasoning, proof by exhaustion) and all sorts of natural deduction proof steps (assumptions, conclusions, references, etc.).

However, there are numerous examples of constructs that mathematicians employ in their works to make the text more concise and less explicit in the number of trivial details. The so-called de Bruijn factor of the present formalizations, that measures the ratio between the length of the formal text and its informal counterpart, is still too high (it has been calculated as around 4 for typical examples and higher in the case of more complicated texts) [33]. Still, making the formal text as short as possible is not the ultimate goal, the readability of the formalization is equally important. Notable examples of informal constructs that are responsible for the discrepancy between informal and formal mathematics are the use of analogy, references to the reader's intuition, or various forms of ellipses. We will be seeking for new useful constructions extending

[9] Generated with MML Query, http://mmlquery.mizar.org.

[10] The description of the Mizar grammar can be found at http://mizar.uwb.edu.pl/language/.

the formal language that would help to represent such linguistic structures. At this stage of the project, different sorts of ellipses, particularly those related to indexed variables, appear to be of high importance. Another open problem is how to introduce a convenient syntax for binding operators like integrals, sums, etc. The open, author-defined part of the Mizar language poses a number of separate research questions. Although the authors are allowed to use different constructs supported by the language to express the same notions, the choice they make can be crucial. This concerns in particular the use of predicates, attributes, and modes that is most natural and closest to mathematical tradition. To some extent, they can be used interchangeably, and using each of them might present specific benefit. For example, the use of attributes offers a lot of simplification in reasoning. Modes allow to express predicative statements about objects that can be categorized in a natural way. On the other hand, predicates are most primitive and generic, and can be used to expand any notion of the three kinds. The current Mizar language permits also the use of adjectives with visible arguments, e.g. `n-dimensional` (for an n-dimensional space), `X-defined` (for a function defined on some set X), `x-convergent` (for a sequence convergent to x) etc. In connection with the attribute clusters rounding-up automation [34], this enables a powerful Prolog-like computation mechanism.

3 Software

The heart of the Mizar system is its proof checker, but it is rarely used today as a standalone program. Over the years, Mizar has evolved into a complex proof assistant system [21] composed of many components that are used together to assist the user in various tasks related to formalizing mathematics. Apart from the core verification software, dedicated utilities support building the library and importing data from it for formalizations based on top of previous developments. A number of utilities distributed in the system's package help the users improve the quality of their formalizations by eliminating unnecessary or redundant parts. Several web-based services that allow to browse the available library as a semantically linked knowledge base, search its content with a dedicated query language, gather proof advice from automated theorem provers, or display the effects of the formalization in an automatically generated journal-style natural language representation. And although Mizar texts can in principle be prepared with the aid of any ASCII text editor, several dedicated editors have been independently implemented, with Mizar mode for Emacs being the most widely used and best integrated with the system's other elements [47]. Below we summarize the basic information about all these interlinked proof assistant components.

The Mizar verifier consists of several modules responsible for checking various aspects of the correctness of Mizar articles in a compiler-like manner. They are: Scanner, Parser, Analyzer, Reasoner, Checker, Schematizer.

Scanner reads the source file and slices it into tokens. Parser checks the syntactic correctness with respect to the grammar of the stream of tokens given

by SCANNER and produces the abstract representation of the article in the form of stacked blocks and items, to be used by ANALYZER which identifies constructors and notations used on the basis of the type information imported from the environment. REASONER is responsible for checking if a proof tactic used by the author corresponds to the formula being proved. The checking is based on the internal representation of formulas in a simplified "canonical" form – their *semantic correlates*. CHECKER works as a classical disprover, additionally taking into account the type information associated with all terms, the properties of the employed constructors, equality calculus, etc. CHECKER also uses special built-in automation procedures for processing selected objects like e.g. complex numbers (direct computation) or boolean operations on sets. SCHEMATIZER processes *schemes* – statements that go beyond first-order logic, using free second-order variables to form infinite families of theorems, e.g. the scheme of mathematical induction.

Before CHECKER can start its work the text should be run through ACCOMMODATOR which imports knowledge either from the MML or a locally created data base. After the verification, the contents of an article can be extracted by EXPORTER and possibly incorporated into the MML.

3.1 XML Layer

MML is one of the largest corpora of nontrivial computer-understandable mathematics. This makes it into a unique resource for all sorts of semantic experiments and assistance tools that go beyond shallow natural-language treatment of mathematical texts. Such tools include database-like semantic search as done by MMLQuery [12], full automated theorem proving (ATP) over MML as done by the MPTP export [45,49] and the Miz$\mathbb{AR}$ system based on it [28,52], or subsumption search based on ATP indexing data structures as used in MoMM [48] and in MathWebSearch [26]. Semantic parsing is also very useful for various functions provided by the Emacs authoring environment for MIZAR (MizarMode) [47], and e.g. its linking with MML Query [13].

Since parsing the advanced human-like MIZAR language is notoriously hard [15], a natural solution taken in 2004 was to make a complete and well-described separation between the parsing and semantic analysis stages (PARSER and ANALYZER) on the one hand, and proof checking and all kinds of other (possibly external) utilities and tools on the other hand. This resulted in a large reimplementation of MIZAR described in [46]. MIZAR started to use XML natively as its internal format produced by the early parsing and analysis processing stages. This format has been gradually extended, and now it contains a very complete semantically disambiguated form of a MIZAR article, as well as a description of the presentation-level syntax that allows various HTML-based presentations combining semantic information and tools with deeply hyperlinked MIZAR texts. The whole MIZAR internal library (items reusable in other articles) is now distributed in this format, and complete articles are translated to the format just by running the MIZAR verifier. An additional suite of open-source XSL-based

translators from the native Mizar XML format to richer or more targeted formats such as MPTP and HTML is developed and maintained by the XSL4Mizar project.[11]

3.2 MPTP and MizAR

There are several goals of the MPTP – Mizar Problems for Theorem Proving – project [45,49], translating the MML to ATP formats. In short, the cooperation of modern ATP systems with large libraries of formalized mathematics is good both for the formalization efforts, providing strong proof assistance, cross-verification, automated theory refactorings, etc., and also for the ATP research, providing a large number of testing problems, allowing research of automated optimization on various mathematical domains and dealing with large knowledge bases, etc. Such cooperation is also the best candidate for merging the deductive (e.g., ATP) and inductive (e.g., machine learning) methods of Artificial Intelligence, because mathematics is (by definition) the most deductively developed science, and once we have a sufficient amount of such data, inductive methods can be applied too and combined with the deductive methods in novel ways [29,50,54,55].

The MPTP translation starts with the native Mizar XML layer, which is first by XSL programs transformed to the extended TPTP (MPTP) Prolog-like format, which adds to TPTP Mizar-like dependent types with attributes and subtyping, second-order constructs such as Fraenkel terms, and supposition-style proofs [44] based on Jaśkowski's natural deduction. The Prolog utilities then process this format, producing TPTP problems and proofs in various ways, typically either for large-scale ATP experiments over the whole translated MML, or in a fast interactive way used by the MizAR (Automated Reasoning for Mizar) system.

MizAR is an online "cloud-based" remote-solving system which integrates several automated reasoning, artificial intelligence, and presentation tools with Mizar and its authoring environment. The service provides ATP assistance to Mizar authors in finding and explaining proofs, and offers generation of Mizar problems as challenges to ATP systems. The system can be used on Mizar goals directly from MizarMode just by typing `by;` after a goal that needs to be solved. This triggers the fast MPTP processing on the server and its parallelized solving using a combination of AI and ATP systems that give the author a 40 % chance of proving a top-level Mizar theorem without any interactive assistance [28]. Another common way how to work with the system is via its web interface[12], where articles can be uploaded, remotely verified, hyperlinked, explored and interactively used for ATP experiments. Such remote-processing functionality is already close to the ideas of formal wikis for Mizar [4,51], whose proper merging with MizAR is one of our next goals.

[11] http://github.com/JUrban/xsl4mizar.
[12] http://mizar.cs.ualberta.ca/~mptp/MizAR.html.

3.3 MML Query

MML Query is based on semantic on-line tool for searching, browsing and presentation of the evolving MML content [10]. The tool offers functionality that outranks commonly used grep-based utilities that often fail because of homonyms and overloading of symbols (and formats) heavily used in the MML. MML Query can also be used to build monographs – the uniform ordered semantic presentation of a specified piece of a theory which may be spread over the MML. The MML Query system also provides a text transformation processor MMLQT (MML Query Templates or MML Query Transformation) which is able to interpret the MML Query language to create ordered queries, version queries, and metadata queries, and to make searching with MML Query somewhat easier (non-expert searching, rough queries).

3.4 Formalized Mathematics Preview

Automatically generated natural language renderings of MIZAR articles can be previewed using a dedicated on-line service[13], which is also used for proof-reading papers in the *Formalized Mathematics* journal, see Sect. 4.3. The translation process [9] might be considered as a rewriting system, where the original MIZAR text is first reduced into an abstract form, which is later augmented with the information coming from the semantic analysis, and then the pattern translation of formulas and formats follows. The translation works on the basis of general and specific patterns. Several alternative patterns might be used for a given translation object. The current implementation of the translation system supports theorems, definitions, schemes, reservations and skeletal proof steps with references to selected most important facts. Finally, some metadata (a user provided summary in English and division into named sections) are incorporated to produce a form that resembles a standard journal paper.

Although the process of generating the journal papers is mechanized, the final result depends on the author or editor. The authors are encouraged to use the previewing facility and suggest any changes to the way their new notions are automatically translated using default patterns.

4 Mizar Mathematical Library

MML is a repository of articles covering various branches of mathematics and computer science. As of now, it contains over 1200 articles, over 10 thousand definitions, and approximately 50 thousand theorems. This collection has been written by over 250 authors. The acquired repository of formalized mathematics is considered one of the largest databases of this type [60]. All articles have been verified by the MIZAR checker and contain mathematical notions systematically defined on top of common axiomatics based on the Tarski-Grothendieck set theory. TG is a non-conservative extension of Zermelo-Fraenkel set theory and is

[13] http://fm.uwb.edu.pl/proof-read/.

distinguished from other axiomatic set theories by the inclusion of Tarski's axiom which states that for each set there is a Grothendieck universe it belongs to. Tarski's axiom implies the existence of strongly inaccessible cardinals, providing a richer ontology than that of conventional set theories such as ZFC.

4.1 Notable Formalizations

Through the years, when the MIZAR project evolved, the development of the repository of MIZAR texts (including the MIZAR language itself) was stimulated by large formalization projects. Among them, the most notable one was the formalization of *Compendium of Continuous Lattices* by Gierz et al. (mentioned in the second edition of the book issued under the title *Continuous Lattices and Domains*) in the years 1995–2003. This collective work of over a dozen of MIZAR authors resulted in 36 articles from the `WAYBEL` series reflecting faithfully the content of the book and 22 articles in the `YELLOW` series bridging the gap between the existing and desired state of the MML [11].

Another example of long-lasting cooperative work of a bigger group of authors was the formalization of the proof of the Jordan Curve Theorem continued from the very beginnings of MML until its successful finale – Artur Korniłowicz's "Jordan Curve Theorem" [30]. This may be considered as a part of a more general project: a formal encoding of general topology – also influential throughout the years. Among recently growing parts of mathematics represented in the MML we can list also functional analysis, lattice theory, and group theory.

The challenge which is stimulating not only for the MIZAR system, but also for other proof-assistants is the "Top 100 mathematical theorems" – the collection of important or interesting facts proposed at the edge of centuries by Paul and Jack Abad as "The Hundred Greatest Theorems". On the page maintained by Freek Wiedijk http://www.cs.ru.nl/F.Wiedijk/100/ one can find systems of computer formalization of mathematics ordered by the number of the items from that list which have been proven in these systems' libraries, covering 91 % of items altogether. Currently, among nine systems listed on the Wiedijk's page, the MIZAR system comes in second place with the total number of 62 items formalized.

4.2 MML Structure

The articles composing the library can be roughly divided into five parts. Although the parts are not formally separate, each of them requires slightly different management procedures:

- the axiomatics – currently containing three files with MML identifiers `HIDDEN` (introducing primitive notions: the root type `object`, membership relation `in`), `TARSKI_0` and `TARSKI_A` – basically axioms of Tarski-Grothendieck set theory (actually Tarski's axiom A is the only exportable item in `TARSKI_A`);
- classical part, currently 323 items, not using the notion of a structure – pure set-theoretic part;

– structural part – all the other articles; this part deals with the notion of a structure, e.g. algebraic structures such as groups, fields, vector spaces, lattices, etc.;
– Encyclopedia of Mathematics in MIZAR (EMM) – currently 14 files with MML identifiers starting with `X`; a collection of monographs;
– the formal model of random access Turing machines, started by Andrzej Trybulec and Yatsuka Nakamura in 1992.

The division into MML's classical and structural parts is an ongoing process as some "classical" items are still being formalized. The process of such changes of the library, called library revisions [23], is coordinated by the Library Committee of the Association of MIZAR Users [5].

4.3 Formalized Mathematics

Although the MIZAR language is developed to be as close as possible to the language used in mathematical papers [22], it is still an artificial language, limited in scope by a preset list of reserved words and allowed grammar constructions. For a more complete popularization of formalized results, it is beneficial to make the content of the repository accessible also in the form of conversational (colloquial or erudite) English, which would enable access to the base by persons not familiar with the MIZAR language. An example of such accessibility is generating articles for the journal *Formalized Mathematics* (ISSN 1426–2630, established in 1990) from the formalized articles contained in the MML [9]. Every submission to MML is first reviewed in a standard journal manner by at least two (usually three) independent specialists; the reviews are on the double-blind basis. MIZAR articles accepted to the MIZAR library are then automatically translated into more human-readable LaTeX format and published in *Formalized Mathematics*. The journal is published quarterly – with thirty as the approximate number of MIZAR articles per volume.

5 Current Developments

All of the project's components undergo implementation and design changes directed towards creating a better proof assistant environment. Most importantly, the verification system is being made stronger, the language more user-friendly, the library better-organized and presentation methods more semantically oriented.

5.1 Stronger Checker

The principal method of verification of informal mathematical papers is peer review. The reviewers, who work in the same field, are capable of understanding the mathematical text even if parts, deemed by the author obvious, are omitted, or the author refers to an analogy to other examples of reasoning. The reviewers are also willing to accept minor errors (e.g., typographical) or imprecisions

stemming from the impossibility of attaining a coherent presentation of many notions and facts scattered throughout the vast literature. Nevertheless, from a computer system perspective, whose task is to semantically represent a given mathematical text, the above mentioned imprecisions are not acceptable.

In MIZAR automation is used to fill the gaps in the user provided declarative proofs, and its main role is to justify proof steps considered trivial by the human being. The current version of the checker provides several mechanisms that increase automation: *reductions* that reduce terms to their proper subterms [31]; *identifications* that identify notions defined within different theories; *properties of functors* that generate particular equalities representing chosen properties of terms (e.g. involutiveness, projectivity, commutativity, idempotence); *properties of predicates* that generate particular formulas representing chosen properties of relations (e.g. reflexivity, irreflexivity, symmetry, asymmetry, connectedness) [37]; and *definitional and functional expansions*. Moreover, there are attempts interfacing external dedicated computational systems (computer algebra systems, solvers, automated theorem provers) to strengthen the MIZAR notion of obviousness [2,35,36].

5.2 Improving Language Readability

The readability of MIZAR proof scripts is considered one of the most important factors of the formalization quality, but in practice enlargement of the database is rather orthogonal to the improvement of the formalization quality. Considering the current size of the library, manual editorial work on improving its legibility becomes infeasible. Therefore several aspects of proof legibility have been identified that can be approached in an automated fashion. Examples of such tasks include finding and removing inessential reasoning fragments or redundant premises in the justification of proof steps, analyzing the order of proof steps and reorganizing proof scripts in the MML according to a consistent style [40], or extracting fragments of reasoning in the form of lemmas or encapsulated nested proofs [24,39].

5.3 Library Reorganization

Initially, the MML development was mainly geared towards volume parameters. Of prime importance was the mathematical result and the growth of the collection of the formalized theorems and proofs. First of all, attention was paid to the local quality of formalization, not to preserving the integrity of the knowledge in the whole base [41]. This approach permitted the accumulation of knowledge, but it did not guarantee taking full advantage of its amount. Currently, the MML development focuses on deciding the structure of the repository in such a way as to enable natural expansions while continuing easy access to the entire accumulated knowledge. The basic problems encountered while managing the development of this large repository are related to the preservation of the integrity of the information it contains [41]. For example:

– independently introducing by different authors different (sometimes incompatible) notations to denote the same (semantically equivalent) notions;
– independently developing the same theory by means of different notion apparatus;
– thematic dispersion of related knowledge in various sections of the repository.

Methods of finding out this type of situations in the MIZAR library are being worked on [39]. Integrity criteria and dedicated algorithms are implemented to assist error detection and the process of refactoring the database [23]. The need for database refactoring complies with the principles of database creation, where duplication and redundancy of information is avoided. At the initial stages of the MML creation, the focus was mainly on collecting as much formalized knowledge as possible to test various aspects of the system. Processing diverse data involved various modules of the system, which was crucial for determining directions of its further development by pointing out its stronger and weaker features. It also allowed to accumulate a considerable amount of formal knowledge. With the present size of the database, managing the library and also its applicability for users, especially new ones, requires developing and adopting a new approach.

In particular, methods to identify notations independently introduced by different authors that denote semantically equivalent notions are investigated. There are known cases of such definitions in the current library. For example, the notion of the exponentiation operation for numbers is denoted in separate formalizations as '`power(x,n)`', '`x to_power n`', '`x |^ n`'. In principle, the authors should be allowed to use the notation they prefer. However, this is a typical source of confusion, duplication, redundancy and an obstacle to efficient search for applicable facts in the library for other authors. Exploring more such cases requires statistical analysis as well as semantic matching of information. The considerations are based on the analysis of definitions in the simplest cases, but also on the analysis of the usage of selected notions in common contexts.

MIZAR developers have also detected cases where the same theories were independently developed by means of a different notion apparatus and have found ways for the best utilization of the independently developed results. For example, in the current library the group is a triple structure with its carrier, a binary operation and a pre-selected element serving as the group's unity. On the other hand, there is also the corresponding theory based on the ordered pair structure (the carrier and an operation) and the group's unity definable by means of an attached extra axiom expressed as an adjective ('unital'). A more complicated case is e.g. the development of lattice theory in terms of ordered sets on the one hand, and as an algebraic theory with two lattice operations on the other hand. For resolving such cases we can consider approaches based on selecting one from the concurrent developments and applying it to eliminate the rest, but also, as in the latter example, with finding ways to provide interoperability of different methods by identifying and encapsulating core components of all developments.

The development of the MML knowledge base has been incremental in its nature. Over two hundred authors who have contributed to its current content represent different backgrounds and skill levels (from students to university

professors). As a result, the library suffers from thematic dispersion of related knowledge in various sections of the repository. For instance, numerous facts concerning simple set theory have been developed while proving some properties of digital circuits in a series of articles loosely connected with basic Boolean properties of sets. Responsible for that is partly the size of the library which makes it difficult for a researcher to grasp it as a whole, but also the authors' tendency or preference for specific approaches to developing mathematics. We investigate new methods based on knowledge exploration that can alleviate the problems. The research is directed towards more efficient, semantics based methods of searching for the information contained in the knowledge base. The main goal is to find ways how to unify (and generalize if needed) all relevant facts once a case of such dispersed knowledge is found in the library. This can be achieved by creating new specialized articles in selected fields, to enrich the repository and to test new language constructions and system properties, and also to define new directions of the development of the base.

5.4 More Semantic Representations

For the needs of the many forms of presenting the contents of the Mizar library, used to popularize formalized knowledge within the mathematical community and on the Internet [22], translation rules that concern the improvement of the quality of article presentation, are being developed. Current research includes: the development of the system of transformation rules for the translation process using the XML/XSLT technology which will result in the design of a more flexible and easier modifiable software tool chain; forming a richer base of translation patterns including new categories for subjected phrases, patterns of mathematical formulae for new constructions, and variations of translation patterns dependent on the mathematical context; working out methods of presentation that take full advantage of the semantically linked information contained in Mizar articles; an improvement of translation of proofs by extracting the references from proofs and a shallow translation of the proof (with the extraction applied for subproofs); the automatic generation of preliminaries for an article and each of its sections based on statistic analysis of the notation and terminology used and the theorems referred to and the subject classification automatically developed for the MML.

Currently, the Mizar language and logic is mainly oriented at human users. The large number of human-friendly linguistic and logical features makes it unsuitable as a direct input for today's automated theorem systems, which work in relatively simple formalisms such as untyped first-order logic, and use simple Prolog-style languages such as the TPTP standard. Suitable layers and interfaces for correct bi-directional communication with such automated systems are being worked on, in particular we can mention the work with the TPTP format and its Mizar-oriented MPTP extension [49]. Since 2005 Mizar has been using an XML-based semantic internal layer, and this layer has been gradually enhanced to serve also a number of external applications. Objects on this layer are fully semantically disambiguated, i.e., there is no use of overloading, all term and formula constructors are linked to their definitions, and full types of terms

are computed. This layer is already used for exporting the Mizar formulas to ATP systems, but it is machine-oriented and so far cannot be used for importing the ATP proofs and for writing human-readable texts. Another issue is that this layer so far does not contain complete information about how proofs were done in Mizar. This makes it difficult to replay the Mizar proofs in other systems, and also to learn from such proofs. Many Mizar mechanisms, such as the use of registrations, identities, requirements, and sometimes also definitions, are implicit, and they become explicit only during the process of verification.

The Mizar developers have started research focused on producing a version of a "strict" semantic Mizar layer [14], where no variables are reserved, all implicit mechanisms (registrations, identities, definitional expansions, etc.) used in the proofs can be made explicit before each proof (or even formula), and overloaded symbols are replaced by their unique synonyms. Such unique synonyms will be introduced either automatically, analogously to the current semantic names, or by suitable syntactic conventions in the Mizar language.

This should allow at least an initial import of the ATP proofs and their verification in Mizar, which is currently not possible, because such proofs may merge very different parts of the MML. Such different parts of the library are currently only mutually consistent on the semantic level, but it is a very nontrivial task (similar, e.g., to merging the notation of two different mathematical theories) to combine such parts also with respect to the overloaded notational mechanisms. Such a layer should in turn allow to construct a chain of Mizar presentation improving utilities (similar to those that already exist for maintaining the MML) that will work on the verified proofs in this layer, and try to make the proofs more human-readable and mathematical by introducing common (possibly overloaded) notation, common names for variables (using reservations), common type mechanisms (such as registrations) for multiple proofs, etc.

Such an approach seems useful not just for importing the semantically encoded proofs produced by ATP systems, but also as a method for automatic merging of different parts of the MML. This is a common task that naturally arises when maintaining a large mathematical library like the MML, and which currently requires a lot of human effort, again because of clashing notational conventions. Such merged developments may first be exported into the "strict" semantic layer and verified there for correctness. After that, the presentation-improving utilities can attempt to automatically construct a common human-friendly notational layer for such merged articles.

Apart from importing and merging the semantically encoded mathematical parts produced by ATP systems, another application of such a layer is in splitting articles and producing a small independent article for each Mizar theorem and definition. This is currently difficult to do automatically, due to mechanisms like reservations, etc. Having a small separate article for each Mizar item again means that such small articles can be subjected to the number of existing Mizar utilities, in particular those that detect the minimal set of (both notational and semantic) dependencies of an article [1,6]. Detecting such a minimal set is useful for various applications, ranging from training of premise-selection

tools for large theory ATP systems, to experimental reverse mathematics assisted by ATP systems and automatically producing the strongest possible version of the theorems in the MML.

6 Future Mizar

Based on the successful long-term development of the MIZAR project, we are encouraged to believe that the project will eventually evolve into a widely-used computerized environment which could make the accumulated formalized mathematical knowledge accessible to a broad spectrum of users and in the future become a modern encyclopedia of mathematics.

For the realization of this long-term goal, it is imperative that first an effective information system for mathematics is formed, bridging the existing knowledge with computer capabilities of processing and searching for information. The fundamental element of this system is a language to represent mathematics in a computerized form. The specifics of the project is to define this language in such a way as to fulfill the above function. It is essential for this language to allow a uniform style in which mathematics will be done, at the same time not restricting the freedom of terminology usage and diverse methods of formalization. Furthermore, the formalization language should be close to the natural language, which would allow additional control of correctness of formalized texts, in particular the definitions of notions.

The key consideration will be defining criteria of readability of mathematical texts and proofs in a formalized form enabling the development of the base of mathematical knowledge, its accessibility and processing at various levels by a possibly wide group of users. To illustrate the readability of developments carried out with the use of the most popular state-of-the-art systems we can look e.g. at the statement of the Fundamental Theorem of Algebra and compare it to its Wikipedia entry: *Every non-constant single-variable polynomial with complex coefficients has at least one complex root.*

Coq:

```
forall f : CCX, nonConst _ f -> {z : CC | f ! z [=] Zero}.
```

HOL Light:

```
|- !a n. a(0) = Cx(&0) \/ ~(!k. k IN 1..n ==> a(k) = Cx(&0))
  ==> ?z. vsum(0..n) (\i. a(i) * z pow i) = Cx(&0)
```

Isabelle:

```
~constant(poly p) ==> z::complex. poly p z = 0
```

MIZAR:

```
for p being Polynomial of F_Complex st len p > 1 holds
  p is with_roots;
```

From the above samples it can be seen that, no matter which system we consider, there is still a significant difference in the readability of the formal and informal (natural language) representation. Improving the readability of formalized texts would allow better communication with the mathematical community and

their greater engagement in the project. The participation of active mathematicians is particularly important for validating and standardizing the definitions of notions deposited in the base [43]. Involving more working mathematicians, who would be able to share their firsthand experience with using the language of mathematical publications on a daily basis, would result in the development and accessibility of a better language and system to formalize mathematics, and several forms of access to a wider audience of mathematical knowledge collected in the MIZAR Mathematical Library [18,19,33]. The accomplishment would be for diverse fields of science and education to benefit from such computer verified knowledge. The pre-processed database will also be used for research aimed at developing automated theorem proving systems (provers).

The ultimate, long-term goal, towards which the work on MIZAR is directed, is to construct a modern encyclopedia of mathematics. We believe that the MIZAR project is well positioned to start a new generation of encyclopedia. All major scientific encyclopedias are available in an electronic form and many, such as Wikipedia or Scholarpedia, solicit input from independent contributors, but the entered data is not verified. The information contained in the huge MIZAR Mathematical Library repository, verified, checked and cross-linked, can be used to build an encyclopedia, which is mathematical at first and later expanded to other sciences, an encyclopedia of entirely different merit, with exclusively formalized and verified data. As a source for citations of mathematical definitions and theorems, an MML based encyclopedia would be invaluable and unique for human users. On the other hand, the rich source of formal mathematical knowledge contained in the MML can be used to develop automated theorem proving methods and systems trained over the mathematics data, and to assist further development of mathematics over such large formal corpora [52,53]. Such automated methods can help with searching the large library, constructing new proofs automatically [20], finding alternative proofs [3], and help with re-structuring the proofs and theories.

References

1. Alama, J.: Mizar-items: exploring fine-grained dependencies in the Mizar mathematical library. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) MKM/Calculemus 2011. LNCS, vol. 6824, pp. 276–277. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-22673-1_19
2. Alama, J.: Escape to Mizar from ATPs. In: Fontaine, P., Schmidt, R.A., Schulz, S. (eds.) Third Workshop on Practical Aspects of Automated Reasoning, PAAR-2012, Manchester, UK, 30 June–1 July 2012. EPiC Series, vol. 21, pp. 3–11. EasyChair (2012). http://www.easychair.org/publications/?page=1559779348
3. Alama, J.: Eliciting implicit assumptions of Mizar proofs by property omission. J. Autom. Reasoning **50**(2), 123–133 (2013). http://dx.doi.org/10.1007/s10817-012-9264-3
4. Alama, J., Brink, K., Mamane, L., Urban, J.: Large formal wikis: issues and solutions. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) MKM/Calculemus 2011. LNCS, vol. 6824, pp. 133–148. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-22673-1

5. Alama, J., Kohlhase, M., Mamane, L., Naumowicz, A., Rudnicki, P., Urban, J.: Licensing the Mizar mathematical library. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) MKM 2011/Calculemus 2011. LNCS, vol. 6824, pp. 149–163. Springer, Heidelberg (2011). http://dx.doi.org/10.1007/978-3-642-22673-1_11
6. Alama, J., Mamane, L., Urban, J.: Dependencies in formal mathematics: applications and extraction for Coq and Mizar. In: Campbell, J.A., Jeuring, J., Carette, J., Dos Reis, G., Sojka, P., Wenzel, M., Sorge, V. (eds.) CICM 2012. LNCS, vol. 7362, pp. 1–16. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-31374-5_1
7. Anonymous: the QED manifesto. Bundy, A. (ed.) CADE 1994. LNCS, vol. 814. Springer, Heidelberg (1994)
8. Strotmann, A.: The categorial type of OpenMath objects. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 378–392. Springer, Heidelberg (2004)
9. Bancerek, G.: Automatic translation in formalized mathematics. Mech. Math. Appl. **5**(2), 19–31 (2006)
10. Bancerek, G.: Information retrieval and rendering with MML query. In: Borwein, J.M., Farmer, W.M. (eds.) MKM 2006. LNCS (LNAI), vol. 4108, pp. 266–279. Springer, Heidelberg (2006). http://dx.doi.org/10.1007/11812289_21
11. Bancerek, G., Rudnicki, P.: A compendium of continuous lattices in Mizar: formalizing recent mathematics. J. Autom. Reason. **29**(3–4), 189–224 (2002)
12. Bancerek, G., Rudnicki, P.: Information retrieval in MML. In: Asperti, A., Buchberger, B., Davenport, J.H. (eds.) MKM 2003. LNCS, vol. 2594, pp. 119–132. Springer, Heidelberg (2003)
13. Bancerek, G., Urban, J.: Integrated semantic browsing of the Mizar mathematical library for authoring Mizar articles. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 44–57. Springer, Heidelberg (2004)
14. Bylinski, C., Alama, J.: New developments in parsing Mizar. In: Campbell, J.A., Jeuring, J., Carette, J., Dos Reis, G., Sojka, P., Wenzel, M., Sorge, V. (eds.) CICM 2012. LNCS (LNAI), vol. 7362, pp. 427–431. Springer, Heidelberg (2012)
15. Cairns, P.: Informalising formal mathematics: searching the Mizar library with latent semantics. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 58–72. Springer, Heidelberg (2004)
16. Corbineau, P.: A declarative language for the Coq proof assistant. In: Miculan, M., Scagnetto, I., Honsell, F. (eds.) TYPES 2007. LNCS, vol. 4941, pp. 69–84. Springer, Heidelberg (2008). http://dx.doi.org/10.1007/978-3-540-68103-8_5
17. Botana, F.: A symbolic companion for interactive geometric systems. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) MKM 2011 and Calculemus 2011. LNCS, vol. 6824, pp. 285–286. Springer, Heidelberg (2011)
18. Futa, Y., Okazaki, H., Shidama, Y.: Formalization of definitions and theorems related to an elliptic curve over a finite prime field by using Mizar. J. Autom. Reason. **50**(2), 161–172 (2013). http://dx.doi.org/10.1007/s10817-012-9265-2
19. Gow, J., Cairns, P.: Closing the gap between formal and digital libraries of mathematics. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec. Studies in Logic, Grammar and Rhetoric, University of Białystok, vol. 10(23), pp. 249–263 (2007). http://mizar.org/trybulec65/
20. Grabowski, A.: Efficient rough set theory merging. Fundamenta Informaticae **135**(4), 371–385 (2014). http://dx.doi.org/10.3233/FI-2014-1129
21. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a nutshell. J. Formaliz. Reason. Spec. Issue: User Tutor. I **3**(2), 153–245 (2010)

22. Grabowski, A., Schwarzweller, C.: Translating mathematical vernacular into knowledge repositories. In: Kohlhase, M. (ed.) MKM 2005. LNCS (LNAI), vol. 3863, pp. 49–64. Springer, Heidelberg (2006). http://dx.doi.org/10.1007/11618027_4
23. Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/CALCULEMUS 2007. LNCS (LNAI), vol. 4573, pp. 235–249. Springer, Heidelberg (2007). http://dx.doi.org/10.1007/978-3-540-73086-6_20
24. Grabowski, A., Schwarzweller, C.: Towards automatically categorizing mathematical knowledge. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) Proceedings of Federated Conference on Computer Science and Information Systems - FedCSIS 2012, Wroclaw, Poland, 9–12 September 2012, pp. 63–68 (2012)
25. Harrison, J.: A Mizar mode for HOL. In: von Wright, J., Harrison, J., Grundy, J. (eds.) TPHOLs 1996. LNCS, vol. 1125. Springer, Heidelberg (1996). http://dl.acm.org/citation.cfm?id=646523.694700
26. Iancu, M., Kohlhase, M., Rabe, F., Urban, J.: The Mizar mathematical library in OMDoc: translation and applications. J. Autom. Reason. **50**(2), 191–202 (2013). http://dx.doi.org/10.1007/s10817-012-9271-4
27. 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
28. Kaliszyk, C., Urban, J.: MizAR 40 for Mizar 40 (2013). CoRR abs/1310.2805
29. Kaliszyk, C., Urban, J., Vyskočil, J.: Machine learner for automated reasoning 0.4 and 0.5 (2014). Accepted to PAAR 2014, CoRR abs/1402.2359
30. Korniłowicz, A.: Jordan curve theorem. Formaliz. Math. **13**(4), 481–491 (2005)
31. Korniłowicz, A.: On rewriting rules in Mizar. J. Autom. Reason. **50**(2), 203–210 (2013). http://dx.doi.org/10.1007/s10817-012-9261-6
32. Matuszewski, R., Rudnicki, P.: Mizar: the first 30 years. In: Mechanized Mathematicsand Its Applications, Special Issue on 30 Years of Mizar, vol. 4, no. 1, pp. 3–24 (2005)
33. Naumowicz, A.: An example of formalizing recent mathematical results in Mizar. J. Appl. Logic **4**(4), 396–413 (2006). http://www.sciencedirect.com/science/article/pii/S1570868305000686
34. Naumowicz, A.: Enhanced processing of adjectives in Mizar. In: Grabowski, A., Naumowicz, A. (eds.) Computer Reconstruction of the Body of Mathematics. Studies in Logic, Grammar and Rhetoric, University of Białystok, vol. 18, no. 31, pp. 89–101 (2009)
35. Naumowicz, A.: Interfacing external CA systems for Grobner bases computation in Mizar proof checking. Int. J. Comput. Math. **87**(1), 1–11 (2010). http://dx.doi.org/10.1080/00207160701864459
36. Naumowicz, A.: SAT-enhanced Mizar proof checking. In: Watt, S.M., Davenport, J.H., Sexton, A.P., Sojka, P., Urban, J. (eds.) CICM 2014. LNCS, vol. 8543, pp. 449–452. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-319-08434-3_37
37. Naumowicz, A., Byliński, C.: Improving Mizar texts with *properties* and *requirements*. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 290–301. Springer, Heidelberg (2004). http://dx.doi.org/10.1007/978-3-540-27818-4_21
38. Naumowicz, A., Korniłowicz, A.: A brief overview of Mizar. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 67–72. Springer, Heidelberg (2009). http://dx.doi.org/10.1007/978-3-642-03359-9_5

39. Pąk, K.: Methods of lemma extraction in natural deduction proofs. J. Autom. Reason. **50**(2), 217–228 (2013). http://dx.doi.org/10.1007/s10817-012-9267-0
40. Pąk, K.: Improving legibility of natural deduction proofs is not trivial. Logic. Methods Comput. Sci. **10**(3), 1–30 (2014). http://dx.doi.org/10.2168/LMCS-10(3:23)2014
41. Rudnicki, P., Trybulec, A.: On the integrity of a repository of formal mathematics. In: Asperti, A., Buchberger, B., Davenport, J.H. (eds.) MKM 2003. LNCS, vol. 2594, pp. 162–174. Springer, Heidelberg (2003)
42. Syme, D.: DECLARE: a prototype declarative proof system for higher order logic. Technical report, University of Cambridge (1997)
43. Trybulec, A., Korniłowicz, A., Naumowicz, A., Kuperberg, K.: Formal mathematics for mathematicians. J. Autom. Reason. **50**(2), 119–121 (2013). http://dx.doi.org/10.1007/s10817-012-9268-z
44. Urban, J., Sutcliffe, G., Trac, S., Puzis, Y.: Combining Mizar and TPTP semantic presentation and verification tools. Stud. Logic Gramm. Rhetor. **18**(31), 121–136 (2009)
45. Urban, J.: MPTP - motivation, implementation, first experiments. J. Autom. Reason. **33**(3–4), 319–339 (2004)
46. Urban, J.: XML-izing Mizar: making semantic processing and presentation of MML easy. In: Kohlhase, M. (ed.) MKM 2005. LNCS (LNAI), vol. 3863, pp. 346–360. Springer, Heidelberg (2006)
47. Urban, J.: MizarMode – an integrated proof assistance tool for the Mizar way of formalizing mathematics. J. Appl. Logic **4**(4), 414–427 (2006). http://dx.doi.org/10.1016/j.jal.2005.10.004
48. Urban, J.: MoMM – fast interreduction and retrieval in large libraries of formalized mathematics. Int. J. Artif. Intell. Tools **15**(1), 109–130 (2006). http://ktiml.mff.cuni.cz/urban/MoMM/momm.ps
49. Urban, J.: MPTP 0.2: design, implementation, and initial experiments. J. Autom. Reason. **37**(1–2), 21–43 (2006). http://dx.doi.org/10.1007/s10817-006-9032-3
50. Urban, J.: BliStr: The Blind Strategymaker (2014). Accepted to PAAR 2014, CoRR abs/1301.2683
51. Urban, J., Alama, J., Rudnicki, P., Geuvers, H.: A wiki for Mizar: motivation, considerations, and initial prototype. In: Autexier, S., Calmet, J., Delahaye, D., Ion, P.D.F., Rideau, L., Rioboo, R., Sexton, A.P. (eds.) AISC 2010. LNCS, vol. 6167, pp. 455–469. Springer, Heidelberg (2010)
52. Urban, J., Rudnicki, P., Sutcliffe, G.: ATP and presentation service for Mizar formalizations. J. Autom. Reason. **50**(2), 229–241 (2013). http://dx.doi.org/10.1007/s10817-012-9269-y
53. Urban, J., Sutcliffe, G.: ATP-based cross-verification of Mizar proofs: method, systems, and first experiments. Math. Comput. Sci. **2**(2), 231–251 (2008). http://dx.doi.org/10.1007/s11786-008-0053-7
54. Urban, J., Sutcliffe, G., Pudlák, P., Vyskočil, J.: MaLARea SG1 - machine learner for automated reasoning with semantic guidance. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 441–456. Springer, Heidelberg (2008)
55. Urban, J., Vyskočil, J., Štěpánek, P.: MaLeCoP machine learning connection prover. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS, vol. 6793, pp. 263–277. Springer, Heidelberg (2011)
56. Wenzel, M., Wiedijk, F.: A comparison of Mizar and Isar. J. Autom. Reason. **29**(3–4), 389–411 (2003). http://dx.doi.org/10.1023/A:1021935419355

57. Wiedijk, F.: Mizar light for HOL light. In: Boulton, R.J., Jackson, P.B. (eds.) TPHOLs 2001. LNCS, vol. 2152, pp. 378–394. Springer, Heidelberg (2001)
58. Wiedijk, F.: Formal proof sketches. In: Berardi, S., Coppo, M., Damiani, F. (eds.) TYPES 2003. LNCS, vol. 3085, pp. 378–393. Springer, Heidelberg (2004)
59. Gamboa, R.: ACL2. In: Wiedijk, F. (ed.) The Seventeen Provers of the World. LNCS (LNAI), vol. 3600, pp. 55–66. Springer, Heidelberg (2006)
60. Wiedijk, F.: Formal proof–getting started. Not. Am. Math. Soc. **55**(11), 1408–1414 (2008)
61. Wiedijk, F.: A synthesis of the procedural and declarative styles of interactive theorem proving. Logic. Methods Comput. Sci. **8**(1:30), 1–26 (2012)