MizarMode - An Integrated Proof Assistance Tool for the Mizar Way of Formalizing Mathematics

Josef Urban [1]

Dept. of Theoretical Computer Science
Charles University
Malostranske nam. 25, Praha, Czech Republic

Abstract

The Emacs authoring environment for Mizar (MizarMode) is today the authoring tool of choice for many (probably the majority of) Mizar authors. This article describes the MizarMode and focuses on the proof assistance functions and tools available in it.

We start with the explanation of the design principles behind the Mizar system, and show how these design principles - mainly the concentration on simple and intuitive human-oriented proofs - have helped Mizar in developing and maintaining a very large body of formalized mathematics.

Mizar is a non-programmable and non-tactical verifier: the proofs are developed in the traditional "write - compile - correct" software programming loop. While this method is in the beginning more laborious than the methods employed in tactical and programmable proof assistants, it makes the "proof code" in the long-run more readable, maintainable and reusable. This seems to be a crucial factor for a long-term and large-scale formalization effort.

MizarMode has been designed with the aim to facilitate this kind of proof development by a number of "code-generating", "code-browsing" and "code-searching" methods, and tools programmed or integrated within it. These methods and tools now include, e.g., the automated generation of proof skeletons, semantic browsing of the articles and abstracts, structured viewing, proof advice using trained machine learning tools like the Mizar Proof Advisor, deductive tools like MoMM, etc. We give an overview of these proof-assistance tools and their integration in the MizarMode, and also discuss some emerging and future extensions such as integration of external theorem proving assistance.

[1] mailto:urban@kti.ms.mff.cuni.cz

1 Introduction

The Mizar [MizarUrl,Rudnicki 1992,Rudnicki and Trybulec 1999] project is today probably the largest effort towards the computer assisted formalization of mathematics. The Mizar Mathematical Library (MML) contains at the moment of writing this article almost 900 formalized articles from many mathematical fields. With the current rate of growth it should reach 1000 articles by the end of 2006.

The Mizar system differs quite significantly from many other proof assistants. It has grown for more than 30 years in relative isolation, and there are many differences to other systems both in the philosophy and in the implementation. Due to these differences, Mizar is probably not considered to be a prototype of a "proof assistant" in the most common usage of this term. It is not interactive, tactical or programmable, it is "just" a verifier, and a set of utilities for a very well designed language and proof formalism. In the next section we explain the philosophy and implementation of Mizar a bit, and conjecture that the strong emphasis on the quality of the language and the readability of the proof presentation are in the long run very important for the development of a large body of formalized mathematics.

In interactive proof assistants a lot of "authoring assistance" (e.g., proof advice) can be directly implemented inside their kernels. The compiler-like operation of the Mizar verifier is the main reason why all such functions for Mizar have to be taken up by various external tools and authoring environments. In the remaining sections we give an overview of the Emacs authoring environment for Mizar (MizarMode), and the various proof-assistance tools that have been programmed or integrated in it exactly for this purpose.

2 The Mizar way of formalizing mathematics and its consequences for proof assistance

2.1 The emphasis on readability

Mizar focuses on the development of formal, yet sufficiently high-level and human-readable proofs. There are a growing number of natural language constructs in the Mizar language. Mizar is based on the Jaskowski [Jaskowski 1934], [Pelletier 1999] natural deduction style, which is quite intuitive and similar to standard textbook proofs, and usually clear even to people with no theoretical knowledge of natural deduction. The Mizar view is that the proof language should first concentrate on clarity, human readability and closeness to standard mathematical proofs, and generally be very careful about adding an excessive number of “tactics” or even allowing users to program their own extensions that make the resulting proofs hard to understand. Today, several other proof assistants that had originally started with the opposite view, i.e. with programmable tactics and other programmable extensions, have implemented a Mizar-like approach, e.g., the Isar flavor of Isabelle has recently become a standard for Isabelle. The proofs created by tactical proof assistants [2] often end up as one of the following extremes:

- the high-level proof script that can, e.g., employ complete theorem proving tactics, and can thus contain nontrivial lemmas without any further proof explanations
- the “machine language” to which the proof scripts can usually be expanded, containing the detailed trace of the proof attempts done by the tactics

None of these two extremes is suitable as a basis for a large-scale and long-term formalization of mathematics. The basis has to be a simple and intuitive middle-level human-understandable proof language, which makes it possible to easily maintain, revise, generalize and learn from these proofs. This has been sufficiently shown in many large software projects: a vast majority of those which are successful strongly enforce clarity and understandability of code, which is crucial for peer review, maintenance, and refactoring by other project members. From this point of view, just the information that a tactical or fully automated prover has proven some nontrivial lemma is insufficient when we start to do proof or theory refactoring. Fifteen years of experience with developing MML have shown that such refactoring work is very frequent and important, sometimes even more important than adding new articles to the library.

2.2 Advantages of simple foundations

The accent on clarity and simplicity of the logic has direct advantages when integrating Mizar with external proof assistance tools. Mizar tries to “keep a low profile” in its logical foundations: instead of various “superlogics”, standard classical first-order logic and set theory are used. No complicated “theory translation” mechanisms are needed and it is perfectly possible to think of the whole Mizar Mathematical Library (MML) as of a single first-order theory. This means that a very powerful kind of external proof assistance is almost directly available to Mizar - the first-order automated theorem proving (ATP) technology. Below, we mention the MoMM tool [Urban 2004a], which has been integrated since its first implementation in 2003 into MizarMode

[2] See `http://www.cs.ru.nl/~freek/comparison/index.html` for a comparison done by Freek Wiedijk.

as an external proof advice. Another Mizar-to-ATP project is the MPTP [Urban 2004b] translation, which is more experimental and not yet integrated into MizarMode.

2.3 (Non)Interactivity

Though batch-like, the Mizar verifier is today practically an interactive tool. Thanks to its fast implementation, the full verification of a complete Mizar article takes on average about 6 seconds on current hardware, so the average verification time when developing a new article is about 3 seconds. In practice this number is actually below 1 second, thanks to a very simple "one-touch" method implemented in MizarMode. It tells the verifier to skip the proofs that have already been verified. This is typically used when the article grows so much that the complete verification makes the author impatient. The fast batch-like processing has some advantages.

- It makes the "random access" authoring possible and natural. Humans are not machines writing one perfect line after another: they like to go back and forth during writing texts as well as programs. It is quite hard to support this in an interactive framework, and such support will never be as robust as a complete recompilation.
- It makes large-scale theory refactoring possible. As noted above, such refactorings turn out to be very important. Complete verification of MML takes about one hour on standard hardware.
- It modularizes the various stages of formalizing mathematics into smaller and simpler subtasks. The task of the verifier is just to verify ("compile") the language, and as in any other compiler its development can focus on improving the language and adding supported features. There may be an unlimited number of various external proof-assistance tools, and they may differ in their task, implementation, speed, or support for the latest language features.

In the following sections we describe external proof-assistance tools integrated into MizarMode. Note how the external Mizar-independent implementation of tools like MoMM and the Mizar Proof Advisor allows them to do their specialized tasks much more efficiently (and thanks to the third-party toolkits, also more cheaply) than if they were implemented in any currently available programmable and interactive proof assistant.

3 Short overview and history of MizarMode and its integrated tools

The development of the current Mizar mode for Emacs started in 2000, during the author's work on a Mizar article [CARD_FIL]. About half a year earlier the first Linux version of Mizar (Mizar 6) had been released, making it possible to use a Unix-like environment for authoring Mizar articles. Until then, the most popular authoring environment for Mizar has been the Borland Turbo Pascal IDE running in MS-DOS. Apart from the "full" Mizar used since 1989 for building the MML, which has been available for a long time only for the x86 architecture running MS-DOS, there is another smaller implementation of Mizar, called Mizar-MSE. This implementation is much simpler and has been used at several places for logic courses. Since its sources have been publicly available, it has been ported to several programming languages and hardware architectures. Bob Beck from the University of Alberta wrote a simple Emacs mode for working with Mizar-MSE in 1991.

The Beck's Emacs mode became a starting point for the implementation of the Emacs mode for Mizar 6. First, standard Emacs features like syntax highlighting and indenting were added, and the functions for running the Mizar verifier and other Mizar utilities were adjusted to Mizar 6. This became the first publicly released version of the new Mizar mode for Emacs. A bit later in the year 2000, tag-based browsing of MML references (theorems, definitions and schemes) was added, support for Mizar-like error messaging and first functions for making "overviews" of some parts of Mizar articles (reservations and theorems) were implemented. While these features are nothing unusual in the Emacs world, the tag-based browsing of references was (at least for the author as the first user) already a big help for studying the proofs in MML[3].

Mizar references are unique (each has its own unique name), and have a standardized spelling which allows the use of standard tag-creation tools (etags, ctags) for creating the tag tables for Emacs. None of this is true for Mizar symbols, which complicates the use of standard tag-based methods for symbol browsing. The first attempt at tag-based browsing of symbols was done in 2001, using a Perl script (already requiring some knowledge of internal Mizar workings) for tag-creation, and providing several methods for dealing with the symbol overloading. A solution for the precise browsing of overloaded constructs has however only been implemented in 2003. We describe it below.

[3] Note that while there has been an HTML-linked presentation of the Mizar abstracts for quite a long time, even at the time of writing this article (end of 2004), there is not yet an HTML presentation of Mizar proofs, and the Emacs browsing is still the only method how to follow references in them. This is hopefully going to change in quite a short time, as a full XML representation of articles is automatically available in the upcoming Mizar version 7.2 .

From 2002 the number of users of MizarMode started to grow, a public CVS was set up, it was ported to the Windows version of GNU Emacs, and it was included in the Mizar distribution. Structured viewing of Mizar articles was added by customizing the general Hide/Show Emacs module for Mizar. The first step towards semantic disambiguation of the overloaded Mizar symbols was taken, and the newly available MMLQuery semantic searching tool [Bancerek and Rudnicki 2003] linked in. Speedbar and Imenu support for listing and fast access to items in long articles were added. At the end of 2002 the first version of the MoMM deductive proof-assistance tool [Urban 2004a] (see below) was publicly released, already with full MizarMode integration.

In 2003 the Mizar Proof Advisor [Urban 2004b] (see below) was made accessible in the MizarMode, precise semantic browsing using the Generated Abstracts (GABs) was added [Bancerek and Urban 2004], and the autocreation of proof skeletons for arbitrary formulas was implemented. The first version of the MizarMode manual [Urban 2003] was written. After this, the addition of new tools has slowed down, and in 2004 mainly improvements of the current tools, and their integration and adjustments to Mizar 7, have been done. MizarMode has been used for the courses of logic and formalized mathematics taught at the University of Bialystok and at the Bialystok Technical University in 2004 and some initial support for this has been implemented. There will be some necessary changes related to the upcoming XML-ized Mizar version 7.2, and possibly some presentation and semantic searching additions made available by the XML interface. We plan a tighter integration with the MMLQuery tool and possibly also integration of the tools creating TEX papers from Mizar articles. The largest planned addition is integration of automated theorem provers working on the MPTP translation [Urban 2004b], coming probably after the release of the next MPTP version.

4 Deductive proof assistance in MizarMode with the MoMM tool

MoMM [4] [Urban 2004a] is both an experiment with using and extending ATP indexing technology for maintenance and data-mining in large mathematical databases like MML, and a practical tool providing authors of Mizar articles with deductive proof advice. The experiment consists in taking a very large body of formal mathematics, transforming it into an ATP-like format and using ATP indexing technology to interreduce it, find out whether it is at least in some very weak sense consistent, find the strongest versions of the theorems in that body of formal mathematics, the most useful theorems, etc. This initial idea leads to further experiments with various improvements of the ATP

[4] This acronym now stands for *Most of Mizar Matches*. This name is motivated by the interreduction results described in [Urban 2004a].

indexing techniques, e.g., strengthening the ATP-like subsumption indexing to efficiently handle the type hierarchies common in formalized mathematics, various speed and memory optimizations, etc.

The current state of experiments shows that it is possible, on standard hardware and with memory requirements which are today quite easily met (about 140 MB of RAM), to have all (about 45000) MML theorems and definitions fully loaded in an ATP indexing data structure (a perfect discrimination tree with some improvements), and available for real-time subsumption queries by Mizar authors. The users have the option to load only some of the available theorems and definitions, e.g., only those used in the articles upon which they want to build. MoMM is also capable of loading lemmas exported from the layered structure of the Jaskowski natural deduction proofs, and treat them in the same way as the main Mizar theorems. The interreduction experiments show that sometimes such lemmas are even stronger than the regular theorems, or are repeated so often that they should be upgraded to be regular theorems. The number of lemmas exportable from MML is quite high - almost 900000 clauses. The full interreduction of all these lemmas is still possible and quite fast (about 30 minutes), but the loading times and memory consumption for the resulting interreduced clausebase is (at the moment of writing this article) probably outside the reach of the average Mizar author (about 1 GB RAM). If formalized mathematics continues to grow only linearly (which is now a sad fact), the exponential growth of computing power will make a fully loaded locally run MoMM an option for an average user in a year or two. Until then it is possible to load MoMM only with lemmas from selected articles.

The integration of MoMM into MizarMode is done as follows:

(1) The MoMM distribution containing suitably interreduced clause bases for each MML article is downloaded and installed.
(2) MoMM is started by the author of an Mizar article from the Emacs menu, and fast-loaded with suitable clause bases.
(3) A modified Mizar verifier is used during the verification instead of the standard one. The difference is that we additionally generate MoMM queries from the currently authored article (typically from the parts which are not accepted by the Mizar checker, i.e. lacking sufficient justification). This modified verifier will probably become redundant after the planned XML changes in Mizar - the queries will be probably rather generated from the XML files by MizarMode.
(4) The generated queries are associated with their counterparts in the Mizar article, and the author can use the interactive functions in MizarMode for sending the queries to the MoMM process.
(5) If such a query is successful (i.e., a match was found by MoMM) and it is a MML theorem, it can be used for direct justification of the corresponding Mizar formula. If the match is an unexported Mizar lemma, the author

is presented with its exact position in the MML and its justification can be copied into his article.

Example
A user writes a formula that he wants to justify, possibly with a partial justification, like
`A3:` ψ `by A1,A2;`
The modified Mizar verifier tries to verify that ψ really directly follows from A1 and A2, or fails. If it fails, it generates an MoMM query. Here it would be one or more clause(s) created by transforming the formula "`(A1 & A2) implies` ψ" to CNF. The user can see that the verification failed, and can send the generated query to the MoMM process, loaded with some MML theorems and lemmas. MoMM tries to subsume the generated clause(s) with the theorems and lemmas loaded. If it succeeds, e.g., with some theorem Th1, the inference
`A3:` ψ `by A1,A2,Th1;`
is then usually accepted by the Mizar verifier.

The ATP technology (MoMM is based on the E prover [Schulz 2002] by Stephan Schulz) is obviously capable of much more than just subsumption, and the subsumption test is now already extended by some equality transformations. However we want to keep MoMM as a fast and real-time tool that handles the Mizar type hierarchy in a way similar to Mizar, and keeps as many theorems and lemmas loaded as possible. For full theorem proving attempts, the Mizar-to-ATP export has to be different (this is being done in the MPTP project): the initial number of theorems and lemmas used for a proof attempt has to be minimized to the least relevant set of premises (to prevent unnecessary inferences) and the proof attempts will typically take longer time. Also proof transformations will be required. Nevertheless, it is quite probable that MoMM will be extended with some options for limited real-time theorem proving, e.g., in some restricted "backchaining" (tableau-like) mode. The mutual matching (inter-reduction) results on MML show that already now that for more than half of MML inferences MoMM can provide advice, i.e. the brute-force "store and index everything" approach is quite successful when loaded with such a large body of mathematics as MML.

5 Proof assistance in MizarMode with machine learning systems trained on MML

Having a large repository of formalized mathematics like MML suggests combining the exact deductive methods used in theorem provers, with the experience learned (induced) from previous proofs. Unlike deductive reasoning, inductive reasoning is not "exact" in a strict logical sense. However, previous experience and the knowledge gained from it seems to be an important part of the methods employed by humans in mathematics and problem solving in general. This inductive/deductive combination was tried and evaluated on a large scale using the first version of MPTP. The results are described in [Urban 2004b]. Even without the deductive counterpart, systems trained by machine learning and data mining methods on MML proofs can provide Mizar authors with useful proof assistance. One of the results of the machine learning experiments over MML is the Mizar Proof Advisor [5], which is now also integrated into MizarMode.

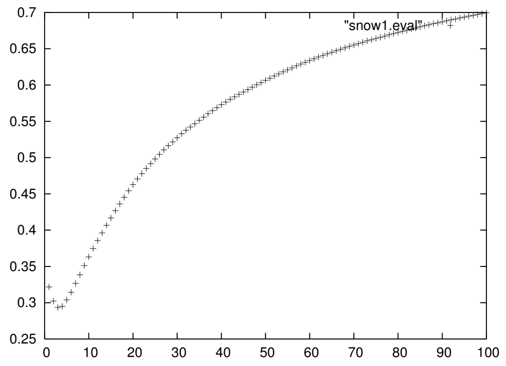
The typical problem faced by a Mizar author is to find the theorems and definitions that might be relevant for the proof of his formula. Since there are now more than 40000 theorems and definitions in MML, this task is highly nontrivial. Even if the author knows (or - as it often turns out in formalized mathematics - *thinks* that he knows) the high-level mathematical proof, it is still a problem to find out how to construct the proof using the MML. If exact deductive methods, like using MoMM, are applicable, they are obviously the best. However, for many formulas this is not the case. The MMLQuery tool can be used for various database-like queries, typically for getting theorems with similar symbols. The Mizar Proof Advisor learns from the references in the previous proofs - it is neither constrained by any deductive laws nor by paying attention whether the symbols in the advised hints overlap with the proved formula or not.

There are a number of "classical" machine learning methods available for such learning (neural nets, decision trees, Markov models, singular value decomposition, Bayes nets, etc.). There are also more sophisticated first-order learning methods, like Inductive Logic Programming, or even systems aspiring to full-scale AI discovery like the Artificial Mathematician [Lenat 1979,Lenat 1982]. Obviously any of these methods can be experimented with and their performance on MML can be compared to each other. For the implementation of the Mizar Proof Advisor the SNoW machine learning toolkit [Carlson et all 1999] was chosen. SNoW is used mainly for natural language processing tasks, and is designed to work efficiently in domains where the number of features and targets is very large, which is useful for MML with its thousands of definitions and tens of thousands of theorems. SNoW implements several "classical" learning algorithms, and also comes with a preprocessor for efficient emulation of some first-order learning methods. The simple setup for the training of the Proof Advisor is to select some suitable features characterizing Mizar formulas, and try to learn the association of such features to the theorems and definitions occurring in their proofs. In the simplest case, which is now implemented, the formula features are just its signature, i.e. the symbols occurring in it. The task may then be reformulated as "learning the relevancy ordering of MML theo-

[5] `http://lipa.ms.mff.cuni.cz/~urban/posdemo.html`

rems and definitions for a given MML subsignature". Preliminary evaluation of the machine learning methods available in the SNoW toolkit suggest using naïve Bayesian learning. Its suitability for the given setting was also confirmed to the author in several discussions with machine-learning people. However, as noted above, we plan to do a more thorough evaluation of various machine learning methods on MML and also of their suitability for combining with automated deductive methods. The following graph taken from [Urban 2004b] shows the results of 10-fold cross-validation of the Bayesian learning run by SNoW on MML 773. The numbers show the average ratio of the actual proof references which where correctly guessed by the trained system on unknown test data as the hint limit grew from 1 to 100. The final value of 0.7 for 100 hints means that on previously unknown formulas, about 70 percent of references needed for their MML proof will be present among the first 100 hints provided by the trained Advisor.

Fig. 1. Ratio of necessary references in SNoW hints



The usage of the Mizar Proof Advisor in the MizarMode is very simple. The author sends the complete formula for which he wants to get proof advice to the server running the Advisor [6]. The server extracts the formula features (which are now just the symbols occurring in it) and queries the Advisor with them. The Advisor replies with a list of names of MML theorems and definitions ordered by the expected relevance for proving formulas characterized by the given features. The names are obviously browsable in the MizarMode, so the users can directly check them and try to think for themselves how they could be useful for the proof.

[6] Local installation of the Mizar Proof Advisor has just been requested by some users, so it will probably be made available in the same way as the MoMM distribution. The hardware requirements for the full loading of the Bayes net trained on the complete MML are about 100MB RAM and about 30 seconds on an average computer, so this will be easily available to average Mizar users.

6 Code generation with proof skeletons

One of the supposed advantages of tactical interactive systems over Mizar authoring is easier writability (with serious consequences to readability and maintenance mentioned above). For example, when a formula has the form of an implication, many tactical systems make it possible to say something like *Implication-Intro* which automatically changes the proof state by adding the antecedent(s) of the formula to some list of available lemmas and replacing the thesis with its consequent. Such a step would in Mizar schematically look like this:

```
P implies Q
proof
   assume P;
   thus Q;
end;
```

As the authors of "Mizar-like" modes for various tactical proof assistants have realized, one can attempt to have the best of both worlds, i.e. it is possible to write *Implication-Intro* in some interactive shell, and have the proof expanded and displayed in the same explicit way, as shown above in the Mizar notation. For Mizar we can do the same, only coming to the "best of both worlds" from the other side. The proof skeletons now implemented in the MizarMode are a one-key automated mechanism for constructing the skeleton of Jaskowski-style proofs for any given formula. E.g., for the following formula (theorem GOEDELCP:1 from the article on Goedel Completeness Theorem)

```
theorem
  Th1: CX is negation_faithful implies (CX |- p iff not CX |- 'not' p)
```

the automatically produced proof skeleton looks this way:

```
theorem
  Th1: CX is negation_faithful implies (CX |- p iff not CX |- 'not' p)
  proof
    assume A1: CX is negation_faithful;
    hereby
      assume A2: CX |- p;
      thus not CX |- 'not' p;
    end;
    assume A3: not CX |- 'not' p;
    thus CX |- p;
  end;
```

There may obviously be different opinions about how a particular proof should be done, while the default skeleton-creating command can implement just one default way of producing the skeleton steps. That's why a large part of the skeleton creation process is customizable. Users can both experiment with their own "tactics" on the parsed formula-trees, and influence, e.g., whether and how the autogenerated labels will be created. All this is done on the editor level, with automatic access to undo-history, etc. The user decides when the verifier will be called and on what.

7 Semantic disambiguation and browsing

The MML today contains thousands of definitions from a range of different fields of mathematics. To be able to easily both read and write MML articles, simple notation is needed. However short symbols are scarce, and they are often used for different purposes in different domains of mathematics. This is called *ad hoc overloading*. Sometimes one symbol is even used with different meanings in the same domain, so the disambiguation is typically done according to operator parameters. This is called *parametric polymorphism*. The Mizar language supports both these kinds of overloading. However, the disambiguating process done by Mizar can be quite nontrivial. In more advanced domains this can make the user unsure as to how the system has resolved his formulas, or make him wonder why the system reports a type checking error.

This language structure makes it necessary to use Mizar-based tools (e.g., a Mizar-like parser) to disambiguate the exact meanings of symbols in some Mizar text. Using simpler browsing tools, like the standard tag-based browsing which is sufficient for the uniquely-named references (theorems, definitions and schemes) can be useful, but cannot generally achieve the required precision.

To make the disambiguation accessible to the authors, we take advantage of the fact that different processing stages of the Mizar verifier use intermediate files for passing information to the next stages. The intermediate files usually contain information about positions in the original article, so that proper error messaging is possible. Thus, for our purpose, it suffices to collect the disambiguated (constructor) format from the appropriate intermediate file, and associate it with the corresponding position in the original article. This association is done using the Emacs mechanism of text properties immediately after processing, which means that even the editing actions that change positions will usually not influence the correspondence between the text in the article and its disambiguated counterpart. In the MizarMode this mechanism is called the *Constructor Explanations*. When switched on by the user, a disambiguated form of formulas is available after Mizar processing for any formula in the article justified by simple justification. This limitation to formulas with simple justification is not serious: it is always possible to add a simple justification to any formula just for this purpose. This limitation will be completely removed after the XML-zation of the intermediate files. With a simple command the disambiguated form (constructor explanation) of a given formula is put in the special buffer *Constructors list*, where it can be used to get various useful information about the formula. The available operations now include

- Browsing the semantically disambiguated MMLQuery abstract (GABs, see below)
- Asking the MMLQuery tool for the meaning of a constructor
- Using tag tables produced from MML to see the definition of a constructor in a normal (i.e. text only) Mizar abstract
- Sending the whole formula to the Mizar Proof Advisor (see above)

The MMLQuery abstracts (GABs) are now the preferred browsing method. They are very similar to the normal (text only) Mizar abstracts, however they also contain additional information thatdisambiguates all the symbols present in them to the appropriate constructors. This is quite similar to, e.g., HTML-linked abstracts (which can be created by the MMLQuery in a very similar way), but the encoding (a customization of the text/enriched MIME Content-type [RFC-1896,RFC-1563,RFC-1523], which we currently call *text/mmlquery*) is much more suitable for fast loading and browsing of the MMLQuery abstracts in the MizarMode, and it is also compatible with the *Constructor Explanations*. The MMLQuery abstracts are not yet distributed with Mizar. They can be downloaded from Grzegorz Bancerek's site[7]. The detailed description of their implementation, as well as of the browsing and searching functions implemented on them, is given in [Bancerek and Urban 2004].

8 Some minor features and planned extensions of the MizarMode

There are a number of minor features making the authoring in MizarMode more convenient. Usually they take advantage of the huge library of Emacs functions and minor modes, accumulated in this more-than-editor over the decades. The structured (outlined) displaying of articles is very easily achieved by a simple customization of the minor Hide/Show mode. For instance, all proofs can be completely hidden by running the "Hide All" command, which practically amounts to displaying the abstract of the article. Similarly, the proof presentation level can be set and the subproofs folded and unfolded. The Imenu and Speedbar minor modes are customized for providing overview of various items in the Mizar articles. Again, this can be used to easily look up all function or predicate definitions in a given article, etc.

[7] `http://merak.pb.bialystok.pl/mmlquery/downloads/`

There are now several ways of running the Mizar verifier from Emacs, either the traditional way with the numbers of error messages inserted directly into the edited buffer to visually indicate the error position, or the newer method suggested by Freek Wiedijk, which uses the standard "compile mode" of Emacs. Some options have been recently implemented to facilitate education with Mizar at the University of Bialystok and at the Bialystok Technical University. This usually means that some more "dangerous" or more complicated options of Mizar processing (e.g., creation of the local environment for an article) are locked in MizarMode, and students are thus protected against accidentally running into a situation that they cannot handle. A simple initial support also exists for communication with a server providing exercises, etc.

The next planned extension (or rather an update) is related to the above mentioned XML-ization of the internal Mizar files. The reason for doing this has been motivated exactly by the need to have better communication with various external tools, and as noted above, this will probably make the specialized MoMM verifier redundant, or make the *Constructor Explanations* available for all formulas in the article. This can be also a basis for implementing more presentation functions.

The largest planned addition is integration of automated theorem provers working on the MPTP translation [Urban 2004b]. This will probably turn out to be as easy as the integration of MoMM or the Mizar Proof Advisor. The main bulk of work consists of making the MPTP system in such a way that ATP systems will be actually useful in providing advice. The initial experiments described in [Urban 2004b] promise that this will be possible. The hope is that this will further boost the ease of formalization in Mizar and, accelerate the further growth of the world's largest library of formalized mathematics.

9 Acknowledgments

Though MizarMode has been developed almost exclusively for several years by the author, it builds on the work of many other people. It started with the initial Mizar-MSE mode by Bob Beck, customizes several minor modes available in GNU Emacs, and if customization is not possible, it simply borrows a lot of code from other Emacs modes. Thanks to all these developers for making their work available in such a nicely customizable way. Having to develop just a fraction of the MizarMode functionality from scratch would take much more time and probably would not be possible for the author (and definitely would not be such a rewarding fun). Thanks also for the many suggestions and feedback from all MizarMode users and users of the systems integrated in it. As for the tools integrated into MizarMode, thanks to Stephan Schulz and Geoff Sutcliffe for the basis on which the MoMM tool is implemented, thanks to prof. Dan Roth and his colleagues for the SNoW toolkit, and thanks to Grzegorz Bancerek for his MMLQuery and cooperation in implementing the semantic browsing features in MizarMode.

References

[Bancerek and Rudnicki 2003] Bancerek G. and Rudnicki P. [2003], Information Retrieval in MML, In Andrea Asperti, Bruno Buchberger, James Davenport (eds.), Mathematical Knowledge Management, Proceedings of MKM 2003, LNCS 2594.

[Bancerek and Urban 2004] Bancerek G., Urban J. [2004], Integrated semantic browsing of the Mizar Mathematical Library for authoring Mizar articles. In editors Andrea Asperti, Grzegorz Bancerek and Andrzej Trybulec - Proceedings of the Third International Conference on Mathematical Knowledge Management, MKM 2004, Lecture Notes in Computer Science 3119, pp. 44 - 57, Springer-Verlag.

[Carlson et all 1999] Carlson A. J., Cumby C. M., Rosen J. L. and Roth D. [1999], SNoW User's Guide. UIUC Tech report UIUC-DCS-R-99-210.

[Jaskowski 1934] Jaskowski, S. (1934) On the Rules of Suppositions in Formal Logic" Studia Logica v.1.

[Lenat 1979] Lenat D. B. [1979], On automated scientific theory formation: A case study using the AM program. In J. Hayes, D. Michie and L.I. Mikulich (Eds.), Machine Intelligence 9 (Halstead, New York, 1979) 251-283.

[Lenat 1982] Lenat, D. B. [1982] The nature of heuristics. In Artificial Intelligence, Vol. 19, No. 2, October 1982, North-Holland, Amsterdam.

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

[Pelletier 1999] Pelletier F. J. [1999], A Brief History of Natural Deduction. History and Philosophy of Logic, vol. 20 (1999), pp. 1 - 31.

[RFC-1523] Borenstein, N., "The text/enriched MIME Content-type", 09/23/1993. `ftp://ftp.rfc-editor.org/in-notes/rfc1523.txt`

[RFC-1563] Borenstein, N., "The text/enriched MIME Content-type", 01/10/1994. `ftp://ftp.rfc-editor.org/in-notes/rfc1563.txt`

[RFC-1896] Resnick, P., Walker, A., "The text/enriched MIME Content-type", January 1996, `ftp://ftp.rfc-editor.org/in-notes/rfc1896.txt`

[Rudnicki 1992] Rudnicki P. [1992], An Overview of the Mizar Project, Proceedings of the 1992 Workshop on Types for Proofs and Programs, Chalmers University of Technology, Bastad.

[Rudnicki and Trybulec 1999] Rudnicki, P. and Trybulec, A. [1999], On Equivalents of Well-foundedness. An experiment in MIZAR, Journal of Automated Reasoning, Vol. 23, pp. 197 - 234, Kluwer Academic Publishers, 1999.

[Schulz 2002] Schulz S. [2002], E – A Brainiac Theorem Prover, Journal of AI Communications, Vol. 15, pp. 111-126.

[CARD_FIL] Urban J. [2000], Basic Facts about Inaccessible and Measurable Cardinals. Journal of Formalized Mathematics, Volume 12, 2000.

[Urban 2003] Urban J. [2003], MizarMode: Emacs Authoring Environment for Mizar, available online at `http://kti.mff.cuni.cz/~urban/MizarModeDoc/html/`

[Urban 2004a] Josef Urban. MoMM - Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. Accepted to editors Geoff Sutcliffe, Stefan Schultz and Tanel Tammit - International Journal of AI Tools, Special Issue on Empirically Successful First Order Reasoning, World Scientific Publishing. Available online at `http://ktiml.mff.cuni.cz/~urban/MoMM/momm.ps`

[Urban 2004b] Josef
Urban. MPTP - Motivation, Implementation, First Experiments. Accepted to editors Ingo Dahn, Deepak Kapur and Laurent Vigneron - Journal of Automated Reasoning, First-Order Theorem Proving Special Issue. Kluwer Academic Publishers (supposed publication: end of 2004). Available online at `http://kti.ms.mff.cuni.cz/~urban/MPTP/mptp-jar.ps.gz`.