

ATP and Presentation Service for Mizar Formalizations

Josef Urban · Piotr Rudnicki · Geoff Sutcliffe

the date of receipt and acceptance should be inserted later

Abstract This paper describes the *Automated Reasoning for Mizar* (MizAR) service, 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 service is based on a sound translation from the Mizar language to that of first-order ATP systems, and relies on the recent progress in application of ATP systems in large theories containing tens of thousands of available facts. We present the main features of MizAR services, followed by an account of initial experiments in finding proofs with the ATP assistance. Our initial experience indicates that the tool offers substantial help in exploring the Mizar library and in preparing new Mizar articles.

1 Motivation and System Overview

Computer supported formal mathematics [6] is becoming better known, widely used and experimented with. Projects like FlySpeck [5] and verification of tiny (but real) operating systems [10] are stimulating the development of interactive verification tools and interactive theorem provers (ITPs). Linked to this is the development of strong automated theorem proving (ATP) systems, used either independently to solve hard problems in suitable domains, or integrated with interactive tools. Such integrations have motivated recent research in the context of automated reasoning in large theories [11, 14, 24].

The Mizar project¹ is a long-term effort to collaboratively develop a formal computerized system representing important mathematical knowledge. The project is

Josef Urban, funded by NWO grants *MathWiki* and *Knowledge-based Automated Reasoning* Radboud University, Nijmegen

Piotr Rudnicki, supported by a NSERC grant
University of Alberta

Geoff Sutcliffe
University of Miami

¹ <http://mizar.org>. See [1] for introductory information on Mizar. For the rest of the paper we assume at least superficial familiarity with Mizar.

focused on building the Mizar Mathematical Library (MML) – a collection of mathematical ‘articles’ formalized and mechanically verified with the Mizar system. MML is the largest such library among similar projects.

This paper introduces the *Automated Reasoning for Mizar* (MizAR) service, which attaches several automated reasoning and presentation tools to the Mizar system. MizAR runs in the context of the MML expressed in the Mizar language – a language oriented toward mathematicians. An inspiration for MizAR is the established SystemOnTPTP ATP service [16], which uses the simpler TPTP language for expressing proving tasks. SystemOnTPTP provides a framework for finding proofs with many first-order ATP systems, offers various forms of proof presentation, supports discovery of new lemmas and independent proof verification. The MizAR service links to and re-uses parts of SystemOnTPTP. MizAR comprises the following main functionalities:

- Fast server-based verification of Mizar articles, and their HTML-based disambiguation linked to the whole cross-linked HTMLized MML – see Section 2.
- Generation of ATP problems in TPTP format for theorems and inferences in a given article, and invoking automated provers on them – see Sections 3, 4.
- Searching for useful premises for Mizar lemmas and inference steps – see Sections 5, 6.

The first version of the MizAR service was deployed in 2008 on a server of the Automated Reasoning Group in Prague. As of 2011 the service is independently hosted on two mid-range multi-core servers in Nijmegen² and in Edmonton³. A way for newcomers to explore MizAR is to use a web browser with an existing simple MML article, e.g., the `CARD_1` article⁴ about cardinal numbers [3] from the MML. More experienced Mizar users will typically interact with the service by launching commands from the Mizar Emacs authoring interface [18]. Some of the commands keep the communication fully in the Emacs session, while other commands offer a browser-based interaction.

2 Server-based Verification and HTMLization

The Mizar verifier acts as a compiler-like batch processor, verifying an entire article in one pass and reporting errors. The process of checking a whole article can be quite time-consuming for longer and more complex articles. In the omni-presence of the fast internet, there are several advantages of remote server-based verification of Mizar articles: (i) no need for a local installation of the entire system; (ii) browser-based access for simple tasks; (iii) large numbers of faster CPUs on the servers, offering great speedups through parallelization [20]. A dedicated server-based installation can also support modified, enhanced, or experimental versions of the verifier. For instance, an enhanced version of the verifier is useful when translating Mizar to ATP formats, and the Mizar parallelizer requires some Linux tools that might not be available on other platforms.⁵ An online service can also

² <http://mws.cs.ru.nl/~mptp/MizAR.html>

³ <http://mizar.cs.ualberta.ca/~mptp/MizAR.html>

⁴ http://mws.cs.ru.nl/~mptp/mml/mml/card_1.miz

⁵ As of 2011, Mizar is distributed for eight architectures, some of them targeted at PDAs.

easily include multiple versions of the Mizar library. This opens a path toward a wiki-like collaborative environment for formalizing mathematics in Mizar [21].

Mizar articles are written as text files according to the Mizar syntax, but their semantics is defined by the verifier in the context of the MML. Discovering all semantic details of such a formal article by hand from its raw textual form can be a challenging task, because of overloaded notation that is so common in mathematical practice. Fortunately, the internal format of MML is XML based [17], which can be automatically turned into an annotated HTML presentation. The HTML presentation, while close to the original text of the Mizar article, offers assistance in semantic browsing both through the MML as well as displaying semantics of an article still in development. Such assistance includes displaying the current goal (thesis) computed by the verifier at each point of a proof, disambiguation of overloaded mathematical symbols through hyperlinks, and explicit display of formulae hidden behind certain keywords announcing properties (such as projectivity, commutativity, antisymmetry, etc.) of constructors. The HTML presentation of Mizar texts forms the backbone to which other MizAR services are linked.

3 Translation to ATP Formats and Integrating ATP systems

MizAR provides access to ATP systems in the context of the large body of mathematics in the MML. The library is first translated into the MPTP (Mizar Problems for Theorem Provers) intermediate format [19], and then translated into the TPTP format that is the standard for many ATP systems. Complete static versions of the MML in the MPTP and TPTP formats are also stored on the MizAR server, and used for on-demand conversion of MML items (theorems, definitions, formulae encoding implicit Mizar type, etc.) into ATP problems in various usage scenarios. The conversion of MML items into MPTP and then into TPTP format requires a quite complex installation and setup (including SWI Prolog, Unix utilities, special XSL style sheets, the MML in the MPTP format, etc.) and therefore is better suited for processing on a dedicated server.

The HTMLization of an article and the generation of ATP problems are independent processes, and they constitute separate services that can run in parallel in different CPUs. For example, a call for ATP help issued from the Emacs interface would trigger only fast ATP processing, responding directly to Emacs. Notation disambiguation and proof explanation tasks would typically also trigger HTML processing, possibly linking additional ATP and explanation services (running on the translated article) to the HTML presentation.

The first version of MizAR used the E and SPASS provers by default, with an option to export the generated problems to the dozens of ATP systems and model finders available through the SystemOnTPTP interface. In 2010, the default ATP was changed to Vampire, motivated by its improved behavior on Mizar/MPTP problems, by its general compliance with the TPTP format, and particularly because of the direct integration of the SInE premise selection method [8].⁶ A recent comparison [22] of Vampire with the E and SPASS ATP systems on the set of all theorems from MML version 1011 is summarized in Table 1. This comparison is based on the *SMALL* versions of the MML problems, in which only the premises

⁶ This changes quickly: a fast SInE algorithm has been recently added also to the E prover.

explicitly provided by Mizar authors (plus some general implicit background facts) are used for constructing the problems. In this mode Vampire solves 20109 problems out of 51424 within a 30s CPU time limit per problem, which is significantly better than (untuned) E, solving 16191 of the problems. For more details and statistics of other usage scenarios, see [22]. The ATP systems are typically run (possibly in parallel) with different premise selections (even if some of them do their own premise selection internally), depending on the usage scenario. Some of the scenarios are explained below.

Table 1 Evaluation of E, SPASS, and Vampire on all MML *SMALL* problems in 30s

description	proved	countersatisfiable	timeout or memory out	total
E 1.1-004	16191	4	35229	51424
SPASS 3.7	17550	12	33862	51424
Vampire 0.6	20109	0	31315	51424
together	22607	12	28817	51424

4 Solving Problems with the Use of the Whole MML

An obvious use-case of MizAR is when a new conjecture is attacked with the help of the whole MML library, containing about a hundred thousand premises. While there are several complementary AI approaches to premise selection, and experimenting with them is interesting and potentially very rewarding, the default method for this use-case is the Mizar-tuned Vampire/SInE system, which is capable of loading the whole translated MML and selecting promising premises in seconds.

When a user asks the service to solve a problem using the whole MML, the service creates a TPTP problem for the task by including the file containing the whole translated MML (available statically on the server in TPTP format), and adding all the propositions from the current article that are available before the proposition for which a proof is sought. Other (typically leaner) premise selections can be created in parallel by analogous mechanisms, producing several versions of the problem that are handed over to the ATP systems in parallel. The current implementation uses four different premise selections: (i) using the full library, (ii) using only premises from the articles imported by the current one, (iii) using only premises from the current article, and (iv) using only the premises explicitly given by the user for the problem. As noted above, the current choice is to use only Vampire/SInE, and parallelize with respect to the different premise selections. This is rather accidental: arbitrary (parallel) combinations of ATP systems and premise selection methods are possible, and limited only by the time limit and the number of free CPUs on the server.

As soon as a proof is found by at least one of the methods in the current pool, the TPTP output is searched for the necessary axioms, and they are presented to the user either in HTML or in Emacs (see below).

5 ATP-supported Authoring in Emacs

Even though MizAR is a web-based service in the spirit of SystemOnTPTP, this does not mean that it requires a browser to use. The above mentioned *whole-library*

solving functionality is available directly from the Emacs authoring environment for Mizar [18], providing fast authoring support without any need for switching to a browser. The implementation uses the Emacs Lisp `url` module and a `http-post` request sent directly to the MizAR server. This communication channel also allows other remote functions, in particular it is possible to call MizAR only for remote (parallelized) verification using the raw speed of the server.

A basic use of MizAR is illustrated by the following example. Inference steps are presented to the Mizar-verifier by stating the goal followed by the keyword ‘by’ with a list of premises. For example,

```
A: x in L ...
...
D: {x} c= L by A, ZFMISC_1:37;
```

where the label `ZFMISC_1:37` refers to a fact imported from MML.⁷ Finding the necessary references requires detailed knowledge of MML, and in more complicated cases it is a time-consuming process. With MizAR available, it is possible to try find sufficient premises (like `ZFMISC_1:37`) by invoking the service with typing ‘by;’ after the goal for which assistance is desired. The query is posted to the MizAR server while the Emacs buffer changes to

```
A: x in L ...
...
D: {x} c= L ; :: ATP asked ...
```

This communication is asynchronous, allowing multiple queries. The ATP answer is provided within seconds, depending on preset time limits and the server’s load. The premises used in the ATP solution are used to replace the original “by;” (or a failure is reported). In this example the result is

```
A: x in L ...
...
D: {x} c= L by A,ENUMSET1:69,ZFMISC_1:37;
```

after which (still in Emacs) the standard Mizar utility `relprem` that detects unnecessary premises in an inference can be invoked. In this example `relprem` detects that `ENUMSET1:69` is unnecessary, and its removal yields the inference step that started this example.

The above example is an inference step taken from a proof of a very simple theorem in the `SCMYCIEL` article [15]

```
theorem Sub3:
for G being SimpleGraph, L being set, x being set
st x in L & x in Vertices G
holds x in Vertices (G SubgraphInducedBy L)
```

Proving the whole theorem is too hard for the ATP service at the moment, and calling MizAR fails with the message: `Sub3: ... Unsolved`. As an alternative, the following obvious intermediate proof steps can be tried.

⁷ See http://mizar.uwb.edu.pl/version/7.11.07_4.160.1126/html/zfmisc_1.html#T37 for the exact statement of `ZFMISC_1:37` in MML version 7.11.07_4.160.1126 . For other theorems cited in this paper, replace the article name and theorem number accordingly. For definitions, replace ‘T’ by ‘D’.

```

proof
  let G be SimpleGraph, L be set, x be set such that
  A: x in L and
  B: x in Vertices G;
  C: {x} in G ;
  D: {x} c= L ;
  E: {x} in (G SubgraphInducedBy L) ;
  thus x in Vertices (G SubgraphInducedBy L) ;
end;

```

None of the sentences labeled C, D, E or the proof conclusion are obvious to Mizar. Again, ATP systems can be used by replacing semicolons with ‘by;’. The replies from ATP come almost immediately, with the final result as follows.

```

C: {x} in G by B,SCMYCIEL:5;
D: {x} c= L by A,ENUMSET1:69,ZFMISC_1:37;
E: {x} in (G SubgraphInducedBy L) by C,D,BOOLE:7,SCMYCIEL:14;
  thus x in Vertices (G SubgraphInducedBy L) by SCMYCIEL:func 5,E,BOOLE:7,SCMYCIEL:5;

```

Such replies from the ATP often need some post-editing to satisfy the Mizar checker:

- Some references returned by the ATP service, like `SCMYCIEL:func 5`, mention typing items, which are implicit⁸ to Mizar and cannot be explicitly referred to.
- Some references, like `BOOLE:7`, encode Mizar automations (called *requirements*), i.e., theorems added automatically to proof search by Mizar. They typically do not have to be used explicitly in Mizar because their automated use is switched on by a global directive.
- Some references are spurious for the Mizar verifier (caused by the non-minimized ATP proof search) and they can be removed, e.g. `ENUMSET1:69` above. Some reference minimization can be done with the `relprem` utility.
- The article is named `SCMYCIEL`, and references to lemmas from this article use this name. These references have to be renamed to the corresponding local names.

Most of this post-editing can be automated. Note that studying the references found by the ATP is instructive as the automated service sometimes (particularly with a large library) finds solutions quite different from what the author had in mind. After the post-editing, the final result accepted by the Mizar verifier is:

```

C: {x} in G by B,Vertices0;
D: {x} c= L by A,ZFMISC_1:37;
E: {x} in (G SubgraphInducedBy L) by C,D;
  thus x in Vertices (G SubgraphInducedBy L) by E,Vertices0;

```

An initial evaluation of this authoring assistance is provided below, in Section 7.

6 Access from HTML

ATP and other services can be called from the MizAR web interface, which is similar to that of SystemOnTPTP. The services can be invoked also from the HTML presentation of the user’s article, created either through the web interface or by

⁸ The rich Mizar type system becomes explicit when translated to untyped first-order logic.

launching a web browser directly from Emacs. The HTML presentation contains links to the ATP services that are associated with the Mizar keywords ‘by’ and ‘from’, indicating logical justification in Mizar. Consider for example, the Mizar justification

```
hence ( f is one-to-one & proj1 f = X & proj2 f = A )
  by A1, A2, WELLORD1:def 7, WELLORD2:16, WELLORD2:def 1;
```

in the last line of the proof of theorem Th4 in the CARD_1 article. Such justifications may involve many implicit Mizar facts and mechanisms that make the raw Mizar text hard to understand. The process of translation to TPTP reveals all this implicit information and the ATP proofs can show explicitly how this information is used. For the Mizar justification above, clicking on the ‘by’ keyword calls the default ATP system on the corresponding ATP problem. If a proof is found, the interface is refreshed with an explanation box that includes a list of the references used in the proof. In this case the references shown to the user are

```
dt_c2_6__mtest_1, dt_k2_wellord1, dt_k1_wellord2, dt_c5_6__mtest_1,
e7_6__mtest_1, e2_6__mtest_1, t16_wellord2, d1_wellord2, e8_6__mtest_1,
e6_6__mtest_1, d7_wellord1,
```

These references are reported using the MPTP syntax and are linked dynamically to the corresponding places in the article’s HTML or in the HTML-ized MML. Note that the ATP proof reports more references than in the original Mizar inference. The extra references are mainly typing statements used implicitly by Mizar.

A byproduct of this ATP explanation feature is the cross-verification of Mizar atomic inferences. With a recent version of MPTP and the strong ATP systems available at the time, over 99% of Mizar atomic inferences could be cross-verified [23]. Such functionality is valuable as a debugging tool for Mizar developers, and also for the developers of the MPTP translation layer.

Another interactive mode of use is for generating problems and finding proofs that are too hard for the Mizar checker, and experimenting with the ATP strength in the mathematician-oriented Mizar language instead of having to encode the problems directly in the low-level TPTP language. Users can do this within MizAR by providing a set of premises on the right-hand side of the ‘by’ keyword and letting ATP systems try to find a proof. If the default ATP systems are not successful, the user can use the links and icons in the explanation box to inspect the ATP problem, and launch the SystemOnTPTP interface to try the ATP systems available there. In a similar way, one can use ATP systems and model finders for detecting countersatisfiability of Mizar-formulated problems.

It is hard to enumerate all the ITP-ATP use-cases that are possible through MizAR. For instance, the user might prefer to use a SAT solver (for attacking propositional problems), instantiation-based systems like iProver (strong in effectively propositional problems), or to experiment with SMT solvers. A reliable ITP-to-ATP translation saves the developers of ITPs a large amount of work by allowing them to be always on top of the state-of-the-art in ATP research. Similarly, a link from the ITP user interface to an ATP user interface saves the developers of ITP user interfaces (in this case the first author) years of work done by the developers of ATP user interfaces. While a basic direct implementation makes sense in both of these cases, fully reimplementing every new ATP method (or user interface to it) inside ITPs and their interfaces can hardly catch up with the rapid development of ATP systems and their interfaces.

A special kind of service that is particular to ITPs with large libraries is premise selection based on (possibly expensive) AI-based preprocessing of the libraries. If no ATP system can find a proof for a MizAR-generated ATP problem, finding relevant premises from MML can help. When ATP fails to find a proof, the ‘Suggest hints’ link can be used to ask MizAR to suggest a set of potential premises. This invokes a Bayesian advisor that has been trained on the whole MML (i.e., on all of the proofs in it). See [24, 2] for the details of how such machine learning is organized in the context of a large deductive repository like MML, and for detailed statistics on how it improves existing premise selection methods. This service is very fast, taking typically less than a second. The hints are again HTML-ized and inserted into an explanation box, as shown in Figure 1. A similar hint function is accessible also from the Emacs mode.

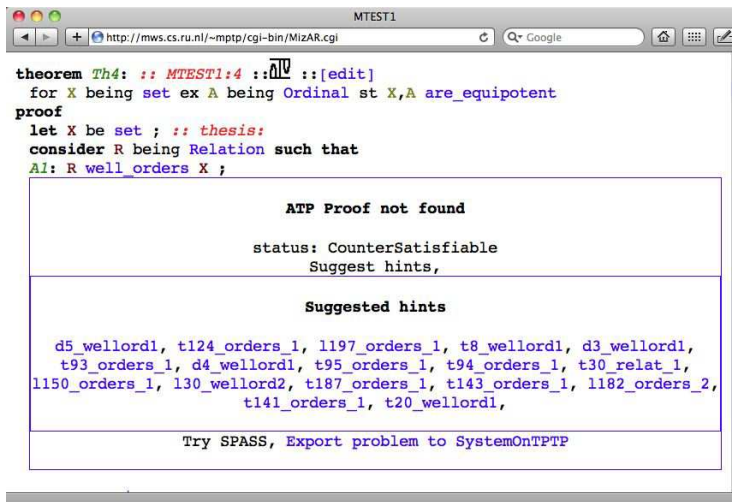


Fig. 1 Explanation box offering hints

7 Initial evaluation of the ATP-supported authoring

In January 2011 the second author was developing a new formalization of simple graphs as 1-dimensional simplicial complexes. The goal of the article was to develop enough theory to prove the construction of the Mycielskian of a graph. This construction was used by Mycielski [12] to prove existence of triangle-free graphs with arbitrarily large chromatic number.

7.1 Axiom debugging

The MizAR service was used on a Mizar article that was in an early stage of development, such that even the formalization of basic definitions was still not ironed out. Only very basic notions of set theory are used in the article: empty set, singleton, ordered and unordered pairs, union of a family of sets, subset, finite set, cardinality of a finite set, partition of a set, Cartesian product, basic properties

of relations and functions, and not much more. It was of interest to see how the MizAR service performs under such conditions, since the Mizar library contains thousands of facts using only these basic notions.

An initial surprise came when the ATP was able to prove almost anything. It turned out that the following statement:

```
theorem SG1:
  for G being SimpleGraph holds {} in G @proof end;
```

was to blame.⁹ While developing a new formalization we frequently state similar, simple facts in a top-down manner, and leave them unproven while focusing on more interesting pieces. We had carelessly stated that $\{\}$ is in every simple graph, even in an empty graph. With an easy contradiction derivable from the axioms, a (refutational) ATP can justify anything. After correcting this to:

```
theorem SG1:
  for G being non empty SimpleGraph holds {} in G @proof end;
```

the ATP was still surprisingly successful. A similar unproven statement

```
theorem SG0:
  for G being SimpleGraph holds G = { {} } \\/ Vertices G \\/ Edges G @proof end;
```

turns out to be false, as it fails when G is empty. The presence of this unproven fact allowed the ATP to prove many other facts in a rather unexpected way. This was again corrected by requiring G in $SG0$ to be nonempty and later led us to revise the definition of `SimpleGraph`. Thus, the ATP helped to straighten out the basic definitions before we did more proofs that we deemed interesting, but which would have been based on unproven, contradictory lemmas about empty graphs.

7.2 Deciphering ATP proofs

There are times when ATP manages to find a proof for a fact that is worth including in the MML as an exportable (reusable) item. Such items are marked `theorem` in Mizar. Here is an example

```
theorem Aux1a:
  for x, X being set holds not [x,X] in X

  for which the ATP returns the following list of premises:
  by ENUMSET1:69,ZFMISC_1:12,ZFMISC_1:38,TARSKI:def 5,ORDINAL1:3;
```

Even though this is very far from deep ATP proofs, the example shows that it may be a bit of a challenge to convert the resolution proof found by ATP into a sequence of inference steps that are understandable to humans and acceptable by the Mizar verifier. After examining the ATP proof we constructed a detailed justification by hand, using the `proof construct`.

```
theorem Aux1a:
  for x, X being set holds not [x,X] in X
  proof
    let x, X be set such that
  A: [x,X] in X;
  B: [x,X] = { {x,X}, {x} } by TARSKI:def 5;
  C: {x,X} in { {x,X}, {x} } by ZFMISC_1:38;
  D: X in {x,X} by ZFMISC_1:38;
    thus contradiction by A, B, C, D, ORDINAL1:3;
  end;
```

⁹ The `@proof` syntax tells Mizar to skip checking of such proof block.

Later a more natural proof of this little fact was found, directly using the definition of an unordered pair. Directing the ATP to use premises preferred by the user would be an interesting future research.

7.3 Overall ATP efficiency and experience

Of the few hundred non-trivial inferences that were tried in the SCMYCIEL article, ATP managed to solve around 40%, which, is surprisingly close to the success rate of Sledgehammer on non-trivial goals [13]. On the other hand, ATP can re-prove 86% of the inferences if it is told which premises were used by humans. This means that more precise narrowing of potential premises is a vital issue for the ATP service, which could particularly benefit from learning from the large number of previous proofs [2]. As mentioned in Section 6, an earlier experiment using several ATP systems has shown that with smarter premise selection more than 99% of atomic inferences can be re-proved [23].¹⁰ The interactive ATP service helped in several ways:

- ATP managed to directly prove some lemmas that require a structured proof for the Mizar verifier. This is not a big surprise, as the Mizar verifier uses only pattern matching and very limited resolution. The feedback from the ATP system was quite helpful, as it is much easier to write a detailed proof when one knows the facts that suffice for the proof.
- ATP turned out to be a search tool in a rather unexpected way. More than once the ATP system indicated that a local lemma had been formed, from which the given formula followed in one step, while we were about to write several inference steps.
- ATP systems found proofs quite different from what the user had in mind. Sometimes it found large sets of premises when some small collection of premises sufficed. The converse also happened.
- When the ATP system finds a proof, it returns all Mizar items that it used. The feedback also includes those Mizar items that are tacitly processed by the verifier, and they cannot be referenced in a Mizar text. This feedback information led us to a better understanding of the task at hand.

8 Conclusions, Related and Future Work

The Mizar service allows authors to use a number of auxiliary tools on Mizar articles. The use-cases range from using HTMLization to disambiguate complicated Mizar texts, using ATP systems to find new proofs, explaining Mizar inferences, and finding counterexamples, to using AI-based techniques for proof advice. The system features both a web and an Emacs interface, allowing flexible switching between *reading and exploration* mode and *authoring* mode.

Related work goes back at least to Dahn’s [4] work in 90’s on ILF and its Mizar-to-ATP bridge, Harrison’s and Hurd’s work on Meson [7] and Metis [9] used in HOL (Light), and the recent work by Paulson et al. [13] on linking Isabelle/HOL

¹⁰ Precisely, 99.8% atomic inferences (6751 out of 6765) were re-proved automatically. The 14 remaining problems were also proved by ATPs after manual premise selection.

with ATP systems. A detailed comparison of systems bridging ITP with ATP systems is beyond the scope of this paper.

There are many directions for future work in this setting, some of them mentioned above. Several versions of the MML are now present on the servers in text, HTML, MPTP, and TPTP formats, but are not directly editable by the users. Giving the user the ability to edit the supporting MML leads in the direction of formal mathematical wikis, with all the interesting persistence, versioning, linking, user-authentication, and dependency problems to solve. Merging the current Mizar wiki development with the services presented here is obvious future work. This should form a rich collaborative platform for formal mathematics, with a large number of services providing strong automated support, and exposing the functionalities that make formal mathematics so interesting. We foresee a large amount of work on making the system stronger, more attractive and responsive.

- We would like to speed up all MizAR services through more parallel processing once the necessary hardware is available.
- There seems to be no end to improving techniques for hint selection in large libraries. We consider such techniques crucial to the success of bridging ITP and ATP systems.
- There is an urgent need for converting the very verbose (typically refutational) proofs found by ATP systems into structured and simple to check proofs.¹¹ As ATP systems are getting stronger and more useful for finding proofs, this problem is becoming more pressing.

Our initial experience with the interactive ATP service for Mizar authors is encouraging. Despite very large library context, we get decent automated help both in finding justification for proof steps, as well as in ‘debugging’ the conceptual framework of a new formalization.

References

1. Grabowski A., Kornilowicz A., and Naumowicz A. Mizar in a Nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
2. J. Alama, D. Kühlwein, E. Tsivtsivadze, J. Urban, and T. Heskes. Premise Selection for Mathematics by Corpus Analysis and Kernel Methods. *ArXiv e-prints*, August 2011.
3. G. Bancerek. The Ordinal Numbers. *Journal of Formalized Mathematics*, 1(1):91–96, 1990.
4. Ingo Dahn and Christoph Wernhard. First-order Proof Problems Extracted from an Article in the MIZAR Mathematical Library. In M.P. Bonacina and U. Furbach, editors, *Int. Workshop on First-Order Theorem Proving (FTP’97)*, pages 58–62, 1997.
5. T. Hales. A Proof of the Kepler Conjecture. *Annals of Mathematics*, 162(3):1065–1185, 2005.
6. T. Hales, editor. *A Special Issue on Formal Proof of Notices of the AMS*, volume 55(11). American Mathematical Society, 2008.
7. J. Harrison. Optimizing Proof Search in Model Elimination. In M. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, number 1104 in Lecture Notes in Artificial Intelligence, pages 313–327. Springer-Verlag, 1996.

¹¹ We consider it a good feature that Mizar does not allow complicated, fragile, and slow proof finding procedures as a part of the core proof checking. The twenty years of experience with daily large-scale theory refactoring of MML has taught the Mizar community that such fragility and slowness should be avoided. We strongly believe that the way from automatically found proofs to proofs in the MML leads through suitable semi-automated refactoring into structured proofs which are perceived as obvious by humans. A related recent effort is described in [25].

8. K. Hoder and A. Voronkov. Sine Qua Non for Large Theory Reasoning. In V. Sofronie-Stokkermans and N. Bjørner, editors, *Proceedings of the 23rd International Conference on Automated Deduction*, number 6803 in Lecture Notes in Artificial Intelligence, pages 299–314. Springer-Verlag, 2011.
9. Joe Hurd. First-order proof tactics in higher-order logic theorem provers. In Myla Archer, Ben Di Vito, and César Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, September 2003.
10. G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: Formal Verification of an OS Kernel. In T. Anderson, editor, *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, pages 207–220. ACM Press, 2009.
11. Jia Meng and Lawrence C. Paulson. Lightweight Relevance Filtering for Machine-generated Resolution Problems. *Journal of Applied Logic*, 7(1):41–57, 2009.
12. Jan Mycielski. Sur le coloriage des graphes. *Colloquium Mathematicum*, 3:161–162, 1955.
13. Lawrence C. Paulson and Jasmin C. Blanchette. Three Years of Experience with Sledgehammer, a Practical Link between Automated and Interactive Theorem Provers. In *8th IWIL*, 2010. Invited talk.
14. A. Pease and G. Sutcliffe. First Order Reasoning on a Large Ontology. In J. Urban, G. Sutcliffe, and S. Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories*, number 257 in CEUR Workshop Proceedings, pages 59–69, 2007.
15. Piotr Rudnicki and Lorna Stewart. Simple graphs as simplicial complexes: the Mycielskian of a graph. *Formalized Mathematics*, 20(2):169–183, 2012.
16. G. Sutcliffe. SystemOnTPTP. In D. McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction*, number 1831 in Lecture Notes in Artificial Intelligence, pages 406–410. Springer-Verlag, 2000.
17. J. Urban. XML-izing Mizar: Making Semantic Processing and Presentaion of MML Easy. In M. Kohlhase, editor, *Proceedings of the 4th International Conference on Mathematical Knowledge Management*, number 3863 in Lecture Notes in Computer Science, pages 346–360, 2005.
18. J. Urban. MizarMode - An Integrated Proof Assistance Tool for the Mizar Way of Formalizing Mathematics. *Journal of Applied Logic*, 4(4):414–427, 2006.
19. Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2006.
20. Josef Urban. Parallelizing Mizar. *CoRR*, abs/1206.0141, 2012.
21. Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A Wiki for Mizar: Motivation, Considerations, and Initial Prototype. In *AISC/MKM/Calculus*, volume 6167 of *LNCS*, pages 455–469, 2010.
22. Josef Urban, Kryštof Hoder, and Andrei Voronkov. Evaluation of automated theorem proving on the Mizar mathematical library. In *ICMS*, volume 6327 of *LNCS*, pages 155–166, 2010.
23. Josef Urban and Geoff Sutcliffe. ATP-based cross-verification of Mizar proofs: Method, systems, and first experiments. *Mathematics in Computer Science*, 2(2):231–251, 2008.
24. Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí Vyskocil. MaLAREa SG1- machine learner for automated reasoning with semantic guidance. In *IJCAR*, volume 5195 of *LNCS*, pages 441–456, 2008.
25. Jirí Vyskocil, David Stanovský, and Josef Urban. Automated proof compression by invention of new definitions. In Edmund M. Clarke and Andrei Voronkov, editors, *LPAR (Dakar)*, volume 6355 of *Lecture Notes in Computer Science*, pages 447–462. Springer, 2010.