Escape to ATP for Mizar

Piotr Rudnicki, Josef Urban

Escape to ATP for Mizar

Piotr Rudnicki*
University of Alberta
Edmonton, Alberta, Canada
piotr@cs.ualberta.ca

Josef Urban†
Radboud University
Nijmegen, The Netherlands
josef.urban@gmail.com

Abstract

An interactive ATP service is a new feature in the Mizar proof assistant. The functionality of the service is in many respects analogous to the Sledgehammer subsystem of Isabelle/HOL. The ATP service requires minimal user configuration and is accessible via a few keystrokes from within Mizar mode in Emacs. In return, for a given goal formula, the ATP service, when it succeeds, finds premises sufficient to prove the goal. The "escape" to ATP uses a sound translation from Mizar's language to that of first-order provers, the same translation that has been used in the more batch oriented *Automated Reasoning for Mizar* (MizAR) web services presented in [16]. We briefly present the interactive ATP service followed by an account of initial experiments with the tool. We claim with some confidence that the tool will substantially ease the process of preparing new Mizar articles.

1 Introduction

We start by noting a substantial difference between the *context of discovering* a proof, when any help from ATPs is welcome, and the *context of justifying* a proof, when the proof is finished and we only want to check its correctness. The experience of maintaining the Mizar library[1] for more than 20 years, with its continual improvements, refactorings and reorganizations, indicates that in the context of justification we need a checker that runs very quickly. Many experiments in which the strength of the Mizar verifier was increased had to be abandoned because they tended to substantially increase the time of verifying the entire library. Such changes happen, for a variety of reasons, almost daily. On the other hand, when we are constructing a proof for a formula, we are ready to use any assistance, even if the available automated helpers run for quite some time.

Thus we need two, if not more, quite different proof checkers/verifiers/advisors. To ensure fast rechecking, the proofs stored in the large repository of formal mathematics need to be checkable almost instantly, and thus tend to be quite detailed. Such detailed proofs are not suitable for direct human consumption, but one can conceive of presentation tools that can browse through the proofs at various levels of abstraction. The question remains: how to construct the detailed proofs? For years, the common practice among Mizar authors has been to write such proofs by hand.

Until several years ago, the contents of the Mizar Mathematical Library (MML) was processable only by the 'secretive' Mizar processor, whose source code is still available only to the members of the Association of Mizar Users. Despite this difficulty, the contents of MML has been exported to other systems, independent of the Mizar processor [13]. For historical accuracy we cannot forget to mention the efforts of Ingo Dahn in trying to export the contents of MML to ATPs and other presentations [2]. Dahn's effort was inspired by the (then influential) *The QED Manifesto*[2] anonymously published in [1].

Pascal Fontaine, Aaron Stump (eds.); PxTP 2011, pp. 46-59

*Partially supported by NSERC.

†Partially supported by the NWO project "MathWiki: a Web-based Collaborative Authoring Environment for Formal Proofs"

[1] http://mizar.org

[2] http://en.wikipedia.org/wiki/QED_manifesto

Although the QED initiative did not result in much more than few written texts[3], the spirit of the project seems to be quite alive as this particular workshop demonstrates.

Over the last decade, the second author has developed software tools advising Mizar authors. The advice consists of suggesting premises relevant for proving a new conjecture in an article. The following tools are external to the Mizar processor and they can be launched from within the Mizar mode for Emacs.

- *M(ost—uch) of M(izar—ath) Matches* (MoMM [10]), released in 2002, is a subsumption tool based on Schulz's E Equational Theorem Prover[4]. Given a goal to prove, the tool quickly finds similar previous proof situations in the large Mizar library. Thanks to the sheer size of the library, subsumption, incorporating additional Mizar-like type mechanisms, turns out to provide useful (exact) advice in more than 50% of new proof situations [10], which is remarkably good compared with full ATP techniques.

- The Mizar Proof Advisor [9, p. 332] was released in 2003. The tool finds previous theorems and definitions from the library as hints (relevant premises) for proving a new conjecture. This advice is inexact—a proof is not searched for. The advisor is trained (in the sense of machine learning) on previous proofs and thus brings into the exact world of automated reasoning some heuristic methods similar to keyword-based web search.

Despite numerous regular presentations and other similar attempts by the second author, these tools have never become regularly used in the Mizar community. The following factors could have played a role:

- The Emacs mode for Mizar took some time to be adopted by a majority of Mizar users. Some core Mizar users still use editors other than Emacs.

- The proof advisor's precision depends on the domain in which it is applied, and the "Google"-like answer it provides may be quite difficult to post-process into a human-understandable complete proof, particularly when suggested connections and proof-ideas are non-obvious. Thus, using ATP to find a proof makes a big difference for users. The premises used in the final ATP proof are guaranteed to be sufficient, and because the user *knows* that the ATP is right, he might put extra effort into understanding the suggested ATP proof. In contrast, in the case of an imprecise AI advisor users might just resign even when the proof is easy to reconstruct from the "Google hits". This seems to be an interesting difference between the perceived utility of Googling for finding relevant high-level mathematical facts, and "Googling" in formal libraries with the purpose of finishing sufficiently simple low-level reasoning steps.

- While MoMM's overall performance on the MML was unusually good at the time of its invention in 2002, when large-theory ATP field was just being born, its performance depends on the domain in which it is applied, because it strongly depends on previously formalized knowledge. This might have discouraged the then small number of potential Mizar users ready to use Emacs and experiment with the then untested large-theory ATP techniques in 2002.

The last version of MoMM was released in 2006. Later, the second author's interest shifted to the more general task of employing AI techniques and ATPs in large formal knowledge bases.

In 2005 large parts of the Mizar software were reimplemented and the internal representation of Mizar articles became based on XML. The entire MML in this form is freely distributed and allows for processing of its contents by generic XML tools. This semantically complete XML representation of

[3] http://mizar.uwb.edu.pl/qed/

[4] http://www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.html

MML simplified the process of translating the library contents to the TPTP format and facilitated the creation of various utilities based on HTML representations; see the next section for a summary and consult [13, 16] for details. An early milestone of these translations is the MPTP system [11], which allows analysis of various mathematically oriented ATP benchmarks as described in [14]. Later, the XMLized MML opened the way for creating metasystems supporting reasoning in large theories [12, 17].

The XMLized version of MML prompted a crucial application of ATPs: the cross-verification of the Mizar library. The Mizar proof checker is a complicated piece of software whose sources are publicly unavailable. Its wide, successful usage was the only external reason for trusting Mizar-checked proofs. It was natural to attempt reproving all of the theorems in MML. As reported in [15], the initial experiments were encouraging: combinations of ATPs are able to reprove between 40% and 60% of MML theorems, depending on the area of mathematics covered. ATPs have been much more successful at reproving Mizar atomic inference steps where the success rate approaches 100%. It was only natural to ask whether the ATPs could prove—and not just reprove—the goals obtained by Mizar atomic inferences without using the guidance of which premises were used by human authors. This presents a challenge of how to select the potentially useful premises out of about 100,000 to choose from; this brings us closer to the topic of this paper.

The tools described in [16] offer a combination of several automated reasoning and proof presentation techniques for the Mizar system. One of the functions, which has been developed since 2008, offers a possibility to ask an ATP system for help in proving a conjecture in the context of a Mizar article. However, in order to call for the ATP assistance, one has to switch into an HTML presentation of the article.

Recently, functionality has been added to the Mizar mode in Emacs that allows the user to invoke ATP help, in several simple ways, from within an Emacs session. This functionality owes much to the recent dissemination and development phase of new ATP techniques for large theories such as SInE [4] and MaLARea [12, 17]. The techniques are now mature enough to handle with reasonable efficiency—at least comparable to specialized ATP techniques like MoMM—simple mathematical problems in the environment of something like 100,000 available facts (axioms).

We would like to stress that the Japanese Mizar community expressed deep interest in ATP techniques assisting in writing Mizar proofs. The growing popularity of the related Sledgehammer ATP subsystem [6] for Isabelle/HOL finally lead the first author to investigate the usability of the MizAR service (see Section 2) and thus provided the first valuable feedback to the second author. The first demonstration[5] of the service reported here was done at the the Joint Mathematical Meeting in New Orleans, January 2011. We report in this paper our initial experience with using the interactive ATP service which we have collected while developing a new Mizar article.

In Section 2 we summarize the automated reasoning online service called MizAR described in detail in [16]. Section 3 illustrates how the ATP service is invoked from inside Emacs and how the ATP reports back. Since this service for Mizar is quite new we have collected only some initial, but encouraging, experience which is presented in Section 4. The last section contains some conclusions but consists mainly of future plans.

2 Essence of the service

A host of several automated reasoning and proof presentation tools for the Mizar system are described by Urban and Sutcliffe in [16]. The collection of tools provides an evolving support for Mizar authors and potential users of the relatively rich Mizar library. The name chosen for this collection is *Automated*

[5] http://mws.cs.ru.nl/~urban/ams11/out4.ogv

Reasoning for Mizar with the suggested acronym MizAR and the main motivation for its development was Sutcliffe's SystemOnTPTP ATP service [8].

MML is distributed as a collection of source Mizar articles plus an internal representation of the library for use by the Mizar verifier. The standard Mizar distribution offers a stand alone installation of the Mizar verifier but no tools for browsing or searching the library besides common utilities like `grep`. In contrast, MizAR offers, among others, the following services:

- Web access to the whole cross-linked HTMLized presentation of MML, which allows for browsing the entire library just by following HTML links. The HTMLized version is presented in a form close to the original articles as stored in the library. This feature is especially important for new authors.
- Server-based verification of a Mizar article which frees the author from installing the system on the local machine. The increasing speed of the Internet, parallelization (currently in development) of the verifier, and access to multi-core servers, all make the remote verification an attractive alternative to installing Mizar locally.
- HTMLization of the article on which the author is currently working. Such assistance is crucial to a Mizar author: the author can inspect his text disambiguated with respect to the omnipresent overloading of Mizar notations, synonyms and antonyms. It is a key advantage over just reading a text version of an article.

  The HTMLized presentation is also furnished with links permitting an inspection of all the items that are tacitly taken into account by the Mizar processor during semantic analysis and inference checking.
- Translation of the article to MPTP (Mizar Problems for Theorem Provers) format.
- Generation of ATP problems in TPTP format for theorems and inferences in the article with the option of invoking some default ATP systems for solving the problems. This is the point that we are focusing on in this paper as we describe how this service is made available directly from inside Emacs.
- Generation of suggestions about potentially useful premises for proving a formula in the article being developed.

The ATP service described in this paper relies on MizAR being able to tackle an inference step and on success returning all the details about the sufficient premises. The current implementation, used in the experiments reported below, uses Vampire[6] as ATP and uses SInE[7][4] as the premises pre-selection heuristic. Three different premise pre-selections strategies are used, run in parallel. One of the versions uses the entire 100,000 MML items as axioms.

For instance, the HTMLized presentation of the Mizar text snippet

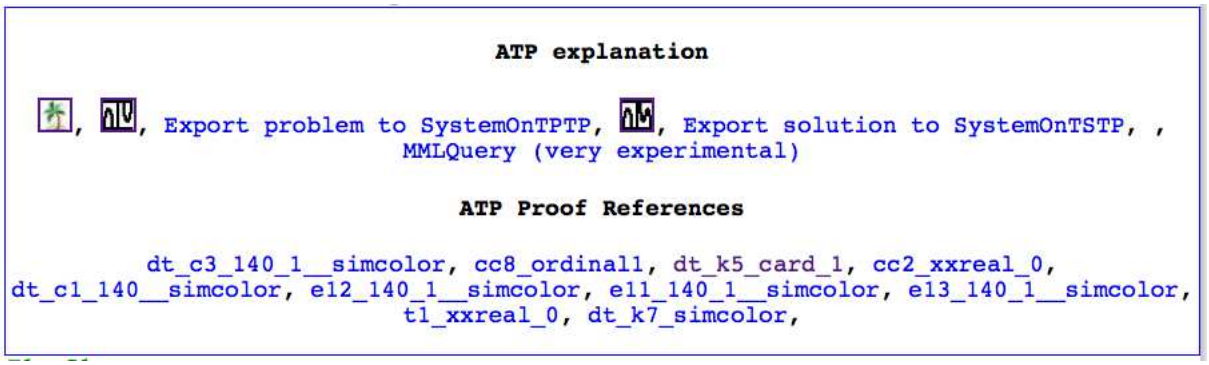
[6] `http://www.vprover.org/`
[7] `http://www.cs.man.ac.uk/~hoderk/`.

```
F1: card C >= cnG by Lchro;
G1: card C <= cnG by D1, E1, MYCIELSK:8;
H1: card C = cnG by F1, G1, XXREAL_0:1;
```

looks like

```
F1: card C >= chromatic# G by Lchro;
G1: card C <= chromatic# G by D1, E1, MYCIELSK:8;
H1: card C = chromatic# G by F1, G1, XXREAL_0:1;
```

where behind almost every token there is a link to its denotation. Clicking on the last by token produces



Again, behind each displayed token of text, there is a link to its denotation. For example, clicking at `dt_k5_card_1` takes us to

```
definition
 let X be finite set ;
 :: original: card
 redefine func card X -> Element of omega ;
 coherence
 proof end;
end;
```

which is a redefinition of the functor `card` for finite sets returning a natural number (`Element of omega`). The list of Mizar items underneath the heading `ATP Proof References` above includes all the items that the ATP needed in order to prove the goal—including the items which the Mizar processor handles tacitly with the explicit references `F1, G1, XXREAL_0:1`. The items used tacitly in an inference step by the Mizar verifier help those authors who are puzzled when an inference step is accepted and it is not immediately clear why.

At the moment, the ATP service uses some other naming convention for Mizar items than the purely textual Mizar sources. The problem stems from an asymmetry: some items in Mizar sources cannot be explicitly referenced in the textual form as they do not have Mizar names. These items (clusters) are tacitly processed by the Mizar verifier. However, all information-bearing Mizar items exported to Mizar-'unaware' ATPs have to be referable by name. The work on standardization of names for Mizar items is being continued.

Another, and from our viewpoint more substantial, form of assistance that the ATP service can be asked for is to prove an inference step that the Mizar verifier does not accept, or, which is a frequently the case, the author does not immediately know how to justify. The current MizAR service provides the assistance but the returned output uses the HTMLized presentation of the Mizar text and the naming of items convenient for ATPs rather than for a human reader. In the next Section, we present a solution to this problem whereby the ATP help is interactively returned in the Emacs session and uses the plain text version of Mizar notation.

3 Access from Emacs

The interactive ATP service is accessible from the Mizar-mode in Emacs and all the findings are reported within the same Emacs session. (We apologize to non-Emacs speakers for using technical jargon.) In the Mizar menu under `Remote solving` we set the remote server, several are available in Białystok, Nijmegen and Edmonton. Then, we need to switch on `by; triggers ATP completion`.

Mizar inference steps are presented to the verifier by stating the goal followed by the keyword `by` and then a list of labels 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. The acceptance of an inference is decided by the rather complex and still evolving Mizar checker[8] [7].

Finding the necessary references requires detailed knowledge of the library and even then, in more complicated cases it can be a time-consuming process. We have chosen a particularly simple example to illustrate the steps performed when finding sufficient premises with the assistance of the ATP.

When the ATP service is switched on, we invoke the service with typing `by;` after the goal we want to be assisted with.

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

The server reacts immediately and displays

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

and the ATP answer is provided quickly, generally less than a minute if not several seconds, depending on preset time limits and how busy the server is. The reported solution replaces the original `by;` with the found premises (or report of a failure). In our simple case we get

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

Within the same Emacs session, we invoke the standard Mizar utility relprem that detects premises relevant for the Mizar verifier. We learn that referring to `ENUMSET1:69` is not necessary and after removing it, we obtain the result with which we started our illustration.

[8] `http://www.cs.ru.nl/~freek/mizar/by.ps.gz`

The above example is one step taken from a proof of a very simple theorem

```
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)
```

which turns out to be too much for the ATP service to handle at the moment and calling `by;` returns `Sub3: ... by Unsolved`.

This 'theorem' comes from an attempt to represent simple graphs as at-most-1-dimensional simplicial complexes presented in Section 4.

Mizar proofs are typically written with a mix of top-down and bottom-up strategy, depending on the author, as the Mizar system allows constructing the proofs both ways. In this small example, we tried to write the formulae of the intermediate proof steps, that we found immediately obvious to ourselves.

```
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 is obvious to the Mizar verifier. We could have searched for the necessary premises by hand but instead we employed the ATP escape by replacing the semicolon with `by;`. The replies from the ATP server were coming while we were still typing, with the final result as follows:

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

At this initial stage of the project, the replies from the ATP server need some postediting in order to satisfy Mizar verifier.

- The article in which we work is named `SIMCOLOR` and references to lemmas from this article use the name. These references have to be renamed to the corresponding local names.
- Some references returned by the ATP service mention Mizar items, like `SIMCOLOR:func 5`, which are implicitly processed by Mizar and cannot be explicitly referred to.
- Some references, like `BOOLE:7`, are processed implicitly by the Mizar verifier and do not have to be referenced.
- Finally, some references are spurious for the Mizar verifier and they can be removed, e.g. `ENUMSET1:69` above. Why these references are returned by the ATP is sometimes puzzling.

All this postediting can (and will) be automated. Note that studying the references found by the ATP is instructive as the automated service sometimes finds solutions quite different from what the author had in mind.

With the postediting by hand 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;
```

4 Initial experience

Automated assistance for Mizar authors has been developed by the second author for quite some time as mentioned in Section 2. Yet it was only in January 2011 that it was noticed that such assistance can be easily incorporated into the Mizar mode in Emacs, offering on the-fly advice for justifying proof steps. At the time the first author was developing a new formalization of simple graphs, (the sixth in the Mizar library!) as 1-dimensional simplicial complexes[9]. In this formalization, we start with a set of vertex labels with vertices being singletons (0-dimensional simplexes) and edges being unordered pairs (1-dimensional simplexes).

We started using the ATP service on a Mizar article which was being created and even the formalization of basic definitions was still not ironed out. 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 [5] to prove existence of triangle-free graphs with arbitrarily large chromatic number.

The formalization uses only very basic notions of set theory: 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. We were interested in how the ATP service performs under such conditions since the Mizar library contains thousands of facts using only these basic notions.

4.1 The first little surprise

The Mizar checker provides a proof placeholder similar to Sorry in Isabelle and to Admitted in Coq, which makes the Mizar verifier temporarily treat a stated fact as proven. In this case, one writes `@proof` instead of `proof`. However, this practice soon becomes tedious; more importantly, it cannot be used in contexts where a proof is syntactically illegal and a simple justification with `by` must be used. One helpful trick (thanks to Freek Wiedijk) is to state at the beginning of the article

```
FAKE: contradiction @proof end;
```

and then use `by FAKE` to justify any goal as for the Mizar verifier anything follows from a contradiction. Mizar articles destined for the library must not contain `@proof`, of course.

We have employed the `FAKE` trick, but it has spoiled the ATP server: with `contradiction` available as an axiom, the ATP was able to prove everything. We could have resorted to using `@proof`, but we believe that the Mizar authoring environment needs to provide a built in tool for temporarily preventing the verifier from checking an inference step. This can be implemented with `@by` as an analog of `@proof` or (less conveniently) using some pragma comments.

[9] Using simplicial complexes was first suggested by Veblen [18, p. 2] according to Harary [3, p. 7] who was wondering why the approach had been popular only within pure mathematics. From our short experience, the explanation could be that for a pen-and-pencil graph theorist the advantage of using this simple framework is not clear.

4.2 Empty graphs

With the FAKE statement removed, the ATP was still able to prove everything. This time, the following statement

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

was to blame. While developing a new formalization we frequently state similar, simple facts 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. We tried the following fix:

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

which only removed a symptom. A similar unproven statement

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

is also unprovable, as it fails when G is empty. However, the presence of this unproven fact allowed the ATP to prove many other facts in a rather unexpected way. We again tried a quick fix by requiring G in SG0 to be nonempty.

However, we ran out of quick fixes when the ATP managed to prove a lot based on a contradiction stemming from:

```
theorem Vertices2:
for G being SimpleGraph holds G = {} iff Vertices G = {} @proof end;
```

As with our then definition of SimpleGraph (which we do not show here) the 'only if' of the above formula leads to a contradiction but in discovering this fact the ATP used a dozen other facts from MML. This time, the quick and dirty fix from above would not work, as the theorem

```
theorem Vertices2:
for G being non empty SimpleGraph holds G = {} iff Vertices G = {} @proof end;
```

looks rather suspicious even to a naked eye. This was the moment when we changed the definition of simple graph to the following:

```
definition
  mode SimpleGraph is 1-at_most_dimensional subset-closed
                                (finite-membered non empty set);
end;
```

With this definition, the only simple graph with no vertices is {{}}, i.e. the singleton of an empty set. We call such a graph void and a graph with no edges edgeless. The ATP helped us to straighten the basic definition before we did more proofs which we deemed interesting but which would have been based on unproven, contradictory lemmas about empty graphs.

4.3 Empty sets

After rectifying the issue of contradictions stemming from empty simple graphs, the ATP was still able to prove surprising results by finding a contradiction elsewhere. Consider:

```
theorem M0e2a:
for G being SimpleGraph, u being Element of Vertices G
 holds {[u, union G], union G} in Mycielskian G
@proof end;
```

The fault is again in misusing empty sets. With the definition of `SimpleGraph` stated at the end of Section 4.2, a simple graph can be non empty and still have no vertices[10]. The construction of a Mycielskian of `G` adds vertices to the graph which are labeled, in our approach, either by `union G` or an ordered pair with the first element being a vertex of `G`.

In Mizar, typing `u being Element of Vertices G` means that `u in Vertices G` but only when `Vertices G` is non empty. The contradiction stemmed from `M0e2a` combined with another unproven (but valid) fact that

```
theorem M0e2:
for G being SimpleGraph, u being set
 st { union G, u } in Mycielskian G
  holds ex x being set st x in Vertices G & u = [x, union G]
@proof end;
```

In the case of a `void` graph, we still can introduce objects that can be typed `Element of Vertices G` although `Vertices G` is empty. Then `M0e2a` gives us an edge in the Mycielskian of `G` and `M0e2` results in an `x` in `Vertices G` although there are none. To fix the problem we changed `M0e2a` to

```
theorem M0e2a:
for G being SimpleGraph, u being set
 st u in Vertices G holds { [u, union G], union G } in Mycielskian G
@proof end;
```

With the ATP service, we managed to discover inconsistencies of our formalization quite early on rather than much later and without investing time to prove more interesting facts based on shaky assumptions.

The contradiction-finding capability of ATP systems in large theories is a fresh development which is desirable in many contexts. The large-theory component that we use—SInE [4]—won the SUMO reasoning prize at CASC[11] in 2008 by finding a contradiction in an early first-order presentation of the large SUMO library.

4.4 Deciphering the ATP advice

There are times when the ATP manages to find a proof for a fact that we deem worthwhile to be included as an exportable item, these 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;
```

and at the moment the author is left with the task of converting the list into a sequence of smaller inference steps acceptable by the Mizar verifier. Sometimes it is quite a puzzling task but with the trustworthy verdict from the ATP we know we can do it. It may help to consult the feedback obtained from MizAR in the form of the ATP Proof References (see Section 2) which in this case displays

```
                    ATP Proof References
reflexivity_r1_tarski, t38_zfmisc_1, t69_enumset1, d5_tarski, l1_simcolor,
                commutativity_k2_tarski, t3_ordinal1,
```

[10] It is common practice among graph theorists to require the set of vertices to be non empty but this seems to be a matter of tradition when graph is considered as a structure with a non empty carrier.

[11] http://www.ontologyportal.org/reasoning.html

We see that the ATP used the fact that the subset relation is reflexive. (c= is the first relational symbol defined in TARSKI.)

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

With this information we quickly see how to construct a more detailed justification by hand using the proof construct.[12]

```
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;
```

It turns out that the current ATP service reports using some premises that are not necessary for the Mizar verifier, like ENUMSET1:69,ZFMISC_1:12. Minimizing the set of premises returned by the ATP is a topic of ongoing work. As mentioned earlier we strongly favor refactoring into structured, detailed, and human-obvious proofs. We foresee heavy use of standard Mizar tools like the detectors of irrelevant premises and spurious inference steps, both of which preserve human-obviousness.

We ended up using the following facts from MML as premises:

```
definition let x,y;
  func [x,y] equals
:: TARSKI:def 5
  { { x,y }, { x } };
end;

theorem :: ZFMISC_1:38
  {x1,x2} c= Z iff x1 in Z & x2 in Z;

theorem :: ORDINAL1:3
  not ( X in Y & Y in Z & Z in X);
```

Later we realized that a more natural proof of this little fact can use the definition of the unordered pair,

```
definition
```

[12] 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 structural proofs which are perceived as obvious by humans. The work in this direction has been started by translating detailed (untyped) Otter proofs directly into Mizar format, and compressing them using the Mizar proof-compression utilities in a fix-point loop, https://github.com/JUrban/ott2miz.

```
    let y;
    let z; func { y, z } -> set means
  :: TARSKI:def 2
    x in it iff x = y or x = z;
    commutativity;
   end;
```

instead of ZFMISC_1:38 and the reflexivity of c=, to state what belongs to an unordered pair.

It is an interesting problem for future research of how to direct the ATP into using premises of our preference rather than what the ATP finds first.

4.5 ATP service effectiveness

Our work on the SIMCOLOR article is being continued. Of the few hundred nontrivial inferences that we tried, the ATP managed to solve around 40% which, surprisingly, is close to the success rate of Sledgehammer on nontrivial goals reported by Paulson and Blanchette [6]. On the other hand, the ATP which we used can reprove 86% of the inferences if it is told which premises where used by humans. This means that more precise narrowing of potential premises is a vital issue for the ATP service. From other experiments using several ATPs in parallel and longer time limits one can reprove more than 99% of atomic inferences in the Mizar library [15].

The interactive ATP service has helped us in several ways:

- The ATP managed to prove some lemmas using the simple by justification 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 was quite helpful, as it is much easier to write a detailed proof when one knows the facts that suffice for the proof.

- The ATP turned out to be a search tool in a rather unexpected way. More than once, the ATP indicated that we had already formed a local lemma from which a given formula follows in one step while we were about to write several inference steps.

- The ATP found proof justifications quite different from what we had in mind. Sometimes, it found large sets of premises when some other premise sufficed. The inverse also happened as stated in the previous item.

- When the ATP finds a justification it returns all the Mizar items that are deemed necessary for the automated prover. The feedback also includes those Mizar items which are tacitly processed by the Mizar verifier and they are not and now cannot be referenced in a Mizar text. This feedback information led us to a better understanding of the task at hand.

5 Conclusion and future plans

Our initial experience with the interactive ATP service for Mizar authors is encouraging. We get both help in justifying single inference steps as well as in 'debugging' the conceptual framework of a new formalization, as pointed out in Section 4. We find the help of the ATP service very valuable in the *context of discovering* a proof.

The plans for the future development of the ATP service for Mizar include:

- Employing several ATP systems in parallel. E prover and SPASS have been part of the MizAR service since its beginning. Recent exhaustive experiments [14] over MML show that the internal

strategy scheduling of Vampire is adequate and a more important source of gains is parallelization of different axiom selection scenarios.

- Improving the techniques for narrowing the selection of potentially useful premises such that the ATP systems can become more efficient.
- Creating strong knowledge-based ATP meta-systems able to suggest intermediate lemmas, suitable concepts, and reliable proof structures through learning from the large proof corpus in a similar way as learning of ATP premise selection. That ATPs with suitable heuristic premise selection methods can significantly help in large theories like MML has been known since 2003 [9]. We perceive a continuation of this effort as a chance for large-theory formal mathematics to boost development of strong semantic AI systems that are hard to envisage in less formal domains.
- Automatically translating the findings of the ATP service to a sequence of basic inference steps acceptable by the Mizar verifier. This would substantially contribute to creating proofs at the level of detail suitable in the *context of justification*, necessary for smooth maintenance of the growing library.
- Designing syntax such that the Mizar author can restrict the parts of MML which are searched for relevant premises. At the moment the entire MML of some 100,000 items is searched. Such a feature would facilitate automating searches for alternative proofs.

We foresee future work on this project as a contribution to web-based collaborative platforms for development of mathematics, currently within the MathWiki[13] initiative.

Acknowledgments: We would like to thank one of the reviewers for many helpful comments, suggestions and corrections. Also, we would like to thank Lorna Stewart and Jesse Alama for their help in improving English of the paper.

References

[1] The QED manifesto. In Alan Bundy, editor, *CADE*, volume 814 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 1994.

[2] Ingo Dahn and Christph 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.

[3] Frank Harary. *Graph theory*. Addison-Wesley, 1969.

[4] Kryštof Hoder and Andrej Voronkov. Sine qua non for large theory reasoning. In *Proceedings of CADE-23*, 2011. Accepted.

[5] Jan Mycielski. Sur le coloriage des graphes. *Colloquium Mathematicum*, 3:161–162, 1955.

[6] 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.

[7] Piotr Rudnicki. Obvious inferences. *Journal of Automated Reasoning*, 3:383–393, 1987.

[8] Geoff Sutcliffe. System description: SystemOnTPTP. In David A. McAllester, editor, *CADE*, volume 1831 of *Lecture Notes in Computer Science*, pages 406–410. Springer, 2000.

[9] Josef Urban. MPTP — motivation, implementation, first experiments. *J. Autom. Reasoning*, 33(3-4):319–339, 2004.

[10] Josef Urban. MoMM — fast interreduction and retrieval in large libraries of formalized mathematics. *International Journal on Artificial Intelligence Tools*, 15(1):109–130, 2006.

[13] `http://www.fnds.cs.ru.nl/fndswiki/Research/MathWiki`

[11] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2006.

[12] Josef Urban. MaLARea: a metasystem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *ESARLT*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[13] Josef Urban. Automated reasoning for Mizar: Artificial intelligence through knowledge exchange. In Piotr Rudnicki, Geoff Sutcliffe, Boris Konev, Renate A. Schmidt, and Stephan Schulz, editors, *LPAR Workshops*, volume 418 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[14] Josef Urban, Kryštof Hoder, and Andrei Voronkov. Evaluation of automated theorem proving on the Mizar mathematical library. In Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, editors, *ICMS*, volume 6327 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 2010.

[15] 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.

[16] Josef Urban and Geoff Sutcliffe. Automated reasoning and presentation support for formalizing mathematics in Mizar. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *AISC/MKM/Calculemus*, volume 6167 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2010.

[17] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí Vyskocil. MaLARea SG1- machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2008.

[18] Oswald Veblen. *Analysis Situs*, volume V. AMS Colloquium Publications, 193.