Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 174 (2007) 63–74
www.elsevier.com/locate/entcs

Presenting and Explaining Mizar

Josef Urban[1]

Dept. of Theoretical Computer Science
Charles University
Malostranské nám. 25, Praha, Czech Republic

Grzegorz Bancerek[2]

Faculty of Computer Science
Bialystok Technical University
ul. Wiejska 45A, Bialystok, Poland

Abstract

The Mizar proof language has both many human-friendly presentation features, and also firm semantical level allowing rigorous proof checking. Both the presentation features and the semantics are important for users, and an ideal Mizar presentation should be both human-friendly (i.e. very close to textbook presentations), and also allowing fast access to the detailed semantics and detailed proof explanations. This poses several questions, problems and choices when presenting original Mizar texts, presenting results of semantic queries over the Mizar library, and also when presenting texts produced directly on the semantical level, e.g. by automated theorem provers. This paper discusses solutions to these problems, and particularly implements an initial system for presenting detailed explanations of atomic Mizar inferences. This is done by the cooperation of the Mizar XML presentation tools, the MML Query system, and automated theorem provers working on the MPTP semantic translation of Mizar.

Keywords: Mizar, proof presentation, ATP, proof objects, MML Query, MPTP

1 Introduction

One of the main objectives in the development of the Mizar [10,9] proof language has always been its intuitive presentation and closeness to mathematical vernacular. The following features are worth mentioning in this context:

- It uses Jaskowski's natural deduction [8,5] for the high-level proof structure, complemented with "simple justification" ("by") steps, which are the atomic inferences checked by the fast Mizar refutational checker [21,7]. These atomic steps are fine-tuned to be of the "right" human-like granularity, i.e., they should be

[1] Email: urban@kti.ms.mff.cuni.cz
[2] Email: bancerek@mizar.org

1571-0661
doi:10.1016/j.entcs.2006.09.022

easy to understand to humans, but should not bother the reader with too much obvious details.

- The language is essentially first-order predicate theory, but it supports a number of linguistic features that make in more human-like. This includes, e.g., usage of adjectives and types and implicit usage of their hierarchies and dependencies (called "registrations" or "clusters" for adjectives), implicit usage of various properties (symmetry, reflexivity, projectivity, etc.), Mizar structures, implicit definitional expansions, etc.
- The language supports wide variety of notations and several kinds of symbol overloading, to allow faithful notation for different mathematical fields encoded in the large Mizar Mathematical Library (MML).

On the other hand, the main purpose of having formal proof languages is their mechanized proof checking. This means that all the above mentioned presentation features ultimately have to be transformed to a proof-checkable level with clear semantics. In Mizar, this is done in several compiler-like passes, which gradually transform the syntactic features to their semantic counterparts (possibly informing users about syntactic errors, etc.), and finally check on the semantic level the correctness of the proofs.

1.1 The semantic level of Mizar

The Mizar semantic level is characterized mainly by two transformations

- Formulas are transformed to the Mizar normal form (MNF), which uses only certain logical connectives ($\wedge$, $\neg$, $\top$, and $\forall$).[3]
- The disambiguation of all the notation (symbols and their patterns) into the "constructors". While the former are usually quite complicated and overloaded, constructors are the unique semantical elements (functors, predicates, etc.).

Both these transformations are many-to-one, and in some sense also many-to-many. Multiple user-level formulas can have the same MNF, and multiple user-level notations can end up being expressed in the same way on the constructor level. As for the many-to-many property, it is theoretically possible to have multiple MNF for one user-level formula, but in practice this does not happen, since the Mizar transformation algorithm is deterministic.[4] It is much more possible to have one user notation (symbols and their patterns) transformed to different constructors, since this heavily depends on the Mizar *environment* (e.g. type rules contributing to different ways of the overloading disambiguation).[5] The important consequence of this is that given a piece of a semantic-level Mizar text, there are usually multiple ways how it can be presented.

[3] The term "semantic correlate" introduced by Roman Suszko is usually used in the Mizar world for MNF.

[4] So if we *defined* MNF as the product of the Mizar transformation algorithm, it would indeed be just many-to-one.

[5] And again, this is just many-to-one, if we fix the particular environment.

This semantic level directly serves for a number of purposes: It is used by Mizar itself for the proof checking and for storing the Mizar internal database. It is also used in the MML Query [2] searching and presentation system. It also serves as the basis for the formats used in the MoMM [16] system, the Mizar Proof Advisor and MPTP [14,17] systems, and for the format used for semantic browsing in the MizarMode [15,3].

It should be noted that this semantic level still expresses the *Mizar logic*, not the standard untyped first-order predicate logic used in current automated theorem provers (ATPs) like E [12], Vampire [11], SPASS [20,19], Otter [6] or Prover9. Further processing is needed when that logic is transformed to standard predicate logic: e.g., the Mizar types need to be encoded, all knowledge used implicitly by Mizar (e.g. type hierarchies) has to be expressed explicitly, etc. This is now done in a certain way (characterized mainly by encoding types as predicates) by the MPTP system, however there are also many possible choices in this transformation. Conversely, this transformation is again generally many-to-many, there will usually be multiple ways of encoding pure predicate logic in the Mizar logic.

1.2 Using the semantic level for linked presentation

Recently, the Mizar semantic level has been completely XML-ized [18], and XSLT tools [6] are being developed for creating linked HTML presentation of Mizar [7] from it. The XML-ized semantic format has been designed so that it is relatively easy to do the HTML linking of symbols and other Mizar resources, and it has been modified several times (usually by adding additional information as XML attributes) using the HTML presentation bottlenecks as a feedback. It currently allows quite faithful re-creation of the original Mizar presentation (see Section 3 for more details), while it also reveals a lot of information computed by the Mizar system (e.g. various formulas computed implicitly - for stating Mizar properties, correctness conditions, etc.), which are normally not accessible to Mizar authors. The main point of using pure XSLT for creating the HTML presentation is that all major browsers today support the XSLT language. This means that Mizar authors can now load the XML file (a by-product of the Mizar verification) directly into their browser whenever they need it during the authoring, and thus immediately get all the additional information contained there.

The XML-ized form of a Mizar article (and hence also the HTML presentation) however does not contain any explanation of the atomic "simple justification" ("by") steps. This kind of explanation was never needed for any purpose for the Mizar processing itself, and its addition (i.e., providing documentation mode for the proof checking of the "simple justification" steps) would involve a very large change of Mizar itself. This means that the users so far could not find out why a particular atomic step was accepted by Mizar. As mentioned above, these steps are designed to be "easy to understand but not unnecessarily verbose" for humans, which is

[6] http://kti.ms.mff.cuni.cz/cgi-bin/viewcvs.cgi/xsl4mizar/miz.xsltxt?view=markup
[7] http://merak.pb.bialystok.pl/mml/

however a very subjective matter depending on many factors. As with the normal natural-language proofs, sometimes the number of "obvious" facts used in an atomic Mizar step can be simply too high for a reader, making the "understanding search space" too large. Humans are also good at occasionally forgetting what should be obvious. One of Mizar's probably greatest contributions to the field of formalization of mathematics is its stress on the readability of proofs (i.e., unlike in the tactical provers, the language is not supposed to be "write-only"). While the readability is a very worthy goal per se (well, why shouldn't all of mathematics be presented in a readable, yet formally correct and mechanically checked way?), this feature of the language actually seems to be quite important in the maintenance of such a large repository of formal mathematics as is MML today, and for its refactoring (e.g., generalizing, reformulating of entire theories, etc.). For all these reasons, providing an optional finer explanation level, which helps to understand the more difficult steps when necessary, should be useful.

1.3 The rest of this paper

We describe our initial solution to the problem of providing and presenting the explanations of the Mizar atomic "simple justification" inference steps. This solution (cf. Diagram 1) uses the ATP technology (now the E-PROVER) for providing the actual explanations as ATP proof objects. The MPTP system is used to transform the Mizar "simple justification" inference steps to ATP problems, and the MML Query system is used to transform the ATP proof objects back into the Mizar notation. The proof objects transformed by MML Query are then linked to the appropriate places in the HTML presentation of Mizar articles, so that users can easily access them, when a particular atomic inference steps is not clear to them.

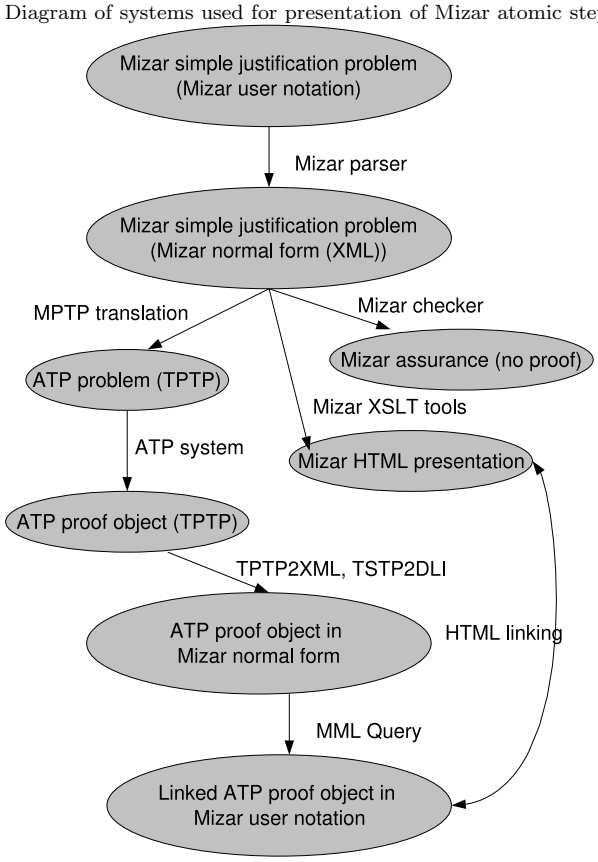
This kind of processing requires several of the above mentioned many-to-many transformations (mainly in the opposite order than mentioned above). We explain the general algorithm used by MML Query for presenting arbitrary semantic-level formulas in the user-friendly notation. This algorithm has been generally used for presenting the MML Query search results, and we are now using it also for the presentation of the text created by ATPs directly on the semantic level. The MML Query solution to this presentational problem is compared to the solution implemented in the HTML presentation of Mizar articles, and their suitability for different purposes is discussed.

2 Explaining and Presenting Mizar Simple Justifications

Readers can check the functionality for explaining the Mizar atomic inferences (implemented now for 35 initial Mizar articles) at the authors' web site [8]. This is a development version of the Mizar HTML presentation, very similar to the official

[8] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/

Fig. 1. Diagram of systems used for presentation of Mizar atomic steps



one at the Mizar site [9]. The main difference at the moment is the linking of the **by** keyword, which leads to the MML Query rendered ATP proof objects, also available at our site [10]. We provide a simple example below.

2.1 Simple example from user's perspective

Consider e.g. the first Theorem [11] in the Mizar article `ZFMISC_1` [4]:

```
theorem Th1: :: ZFMISC_1:1
  bool {} = {{} }
proof
  now
    let c1 be set ;
    ( c1 c= {} iff c1 = {} ) by XBOOLE_1:3;
    hence ( c1 in bool {} iff c1 in {{} } ) by Def1, TARSKI:def 1;
  end;
```

[9] http://mmlquery.mizar.org/mml/4.48.930/

[10] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/_by/

[11] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/zfmisc_1.html#T1

```
  hence bool {} = {{} } by TARSKI:2;
end;
```

Its (probably redundant) natural-language explanation is that the powerset of the empty set is a singleton containing just the empty set. The symbol `in` used inside the proof denotes the set-theoretical membership, and the symbol `c=` used below denotes the set-theoretical inclusion. Note that the theorem `XBOOLE_1:3` [12]

```
theorem E3: :: XBOOLE_1:3
  for b1 being set holds ( b1 c= {} implies b1 = {} )
```

is just an implication, not equivalence. So the user might want to know, why the first inference [13]

```
( c1 c= {} iff c1 = {} ) by XBOOLE_1:3;
```

is logically valid. Clicking on its **by** keyword will reveal the following E-PROVER's proof [14] rendered by MML Query [15] (we rather recommend to check this (above given link) directly in a browser, since the linking cannot be seen here in text form). The very first axiom there is `reflexivity_r1_tarski`, stating the reflexivity of the inclusion predicate. This is a Mizar *property*, which the system uses automatically, assuming that it is obvious to the readers. With this axiom, the inference easily follows:

```
MML Query rendering of ATP proof steps

axiom: reflexivity_r1_tarski
A:step 1
for x1, x2 being set holds
x1 c= x1
------------------------------------------------------
conjecture: e1_10_1
A:step 7
   (c11001 c= {}
iff
   c11001 = {})
------------------------------------------------------
axiom: t3_xboole_1
A:step 8
for x1 being set
      st x1 c= {}
   holds x1 = {}
------------------------------------------------------
inference: assume_negation(7)
A:step 9
(c11001 c= {} & c11001 <> {} or c11001 = {} & not c11001 c= {})
------------------------------------------------------
inference: variable_rename(1)
A:step 11
for x3, x4 being set holds
x3 c= x3
------------------------------------------------------
inference: split_conjunct(11)
A:step 12
x1 c= x1
------------------------------------------------------
inference: fof_nnf(9)
A:step 22
(c11001 c= {} implies c11001 <> {}) &
```

[12] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/xboole_1.html#T3

[13] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/zfmisc_1.html#E1:10_1

[14] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/_by/zfmisc_1/164_29.html

[15] Note that the constant $c1$ has been renamed by our system to $c11001$. This is actually $c1_{1001}$ in the HTML rendering, the subscript encodes the current Mizar proof level. This is a result of the MPTP system naming conventions, which need to provide unique name to every MPTP object.

```
 (c11001 c= {} or c11001 = {})
------------------------------------------------------
inference: split_conjunct(22)
A:step 23
(c11001 = {} or c11001 c= {})
------------------------------------------------------
inference: split_conjunct(22)
A:step 24
(c11001 = {} implies not c11001 c= {})
------------------------------------------------------
inference: fof_nnf(8)
A:step 25
for x1 being set
      st x1 c= {}
   holds x1 = {}
------------------------------------------------------
inference: variable_rename(25)
A:step 26
for x2 being set
      st x2 c= {}
   holds x2 = {}
------------------------------------------------------
inference: split_conjunct(26)
A:step 27
(x1 <> {} implies not x1 c= {})
------------------------------------------------------
inference: csr(24,27)
A:step 29
not c11001 c= {}
------------------------------------------------------
inference: sr(23,29)
A:step 30
c11001 = {}
------------------------------------------------------
inference: rw(rw(29,30),12)
A:step 31
contradiction
```

2.2 Simple example further explained

Now we will explain the particular stages of creating and rendering of the ATP explanation.

2.2.1 Creation of the ATP problem

The problems are generated by the development version of the MPTP system. This is a system for translating Mizar into untyped first-order predicate logic, and encoding Mizar problems in a way suitable for solving by ATP systems. The recent (second) version of the system has been quite heavily tested (see [17]), and for problems which do not contain possible arithmetical evaluations (which will need further treatment) it now seems to provide all the information necessary for reproving Mizar inferences by ATPs. As mentioned above, MPTP encodes the Mizar types as predicates, and explicitly adds to problem specifications various kinds of information which is obvious to Mizar (like type hierarchy, or the reflexivity property mentioned above). The ATP problem specification (file named `zfmisc_1__164_29`, using the TPTP [13] format) is as follows:

```
% Mizar problem: e1_10_1,zfmisc_1,164,29
fof(reflexivity_r1_tarski, axiom,  (! [A, B] : r1_tarski(A, A)) ,
    file(tarski, r1_tarski), []).
fof(dt_k1_xboole_0, axiom, $true, file(xboole_0, k1_xboole_0), []).
fof(dt_c1_10_1, axiom, $true, file(zfmisc_1, c1_10_1), []).
fof(fc1_xboole_0, axiom, v1_xboole_0(k1_xboole_0),
    file(xboole_0, fc1_xboole_0), []).
fof(rc1_xboole_0, axiom,  (? [A] : v1_xboole_0(A)) ,
    file(xboole_0, rc1_xboole_0), []).
fof(rc2_xboole_0, axiom,  (? [A] :  ~ (v1_xboole_0(A)) ) ,
```

```
    file(xboole_0, rc2_xboole_0), []).
fof(e1_10_1, conjecture,
   (r1_tarski(c1_10_1, k1_xboole_0) <=> c1_10_1=k1_xboole_0) ,
   inference(mizar_bg_added, [],
   [reflexivity_r1_tarski, dt_k1_xboole_0, dt_c1_10_1, fc1_xboole_0,
   rc1_xboole_0, rc2_xboole_0, t3_xboole_1]), [file(zfmisc_1, e1_10_1)]).
fof(t3_xboole_1, axiom,
   (! [A] :  (r1_tarski(A, k1_xboole_0) => A=k1_xboole_0) ) ,
   file(xboole_1, t3_xboole_1), []).
```

This encoding is in more detail described in [17]. Note that there are more axioms than are actually needed for the proof above. This is because the MPTP algorithm for adding the "background knowledge" can be only approximative, and the main goal is to approximate it from the safe side, i.e. maintaining the completeness of the specification.

2.2.2 Creation of the ATP proof

The MPTP problems generated from the Mizar simple justifications are usually very easy for current ATP systems. We are using the latest version (0.91) of Stephan Schulz's E-PROVER, both because of its strength, and also because of its support of the TPTP standards, which allow us to use the TPTP tools for preprocessing and postprocessing. Running the E-PROVER through SystemOnTPTP [16] yields the following TSTP proof object:

```
fof(1, axiom,![X1]:![X2]:r1_tarski(X1,X1),
    file('/tmp/SystemOnTPTP15237/zfmisc_1__e1_10_1.p', reflexivity_r1_tarski)).
fof(7, conjecture,(r1_tarski(c1_10_1,k1_xboole_0)<=>equal(c1_10_1, k1_xboole_0)),
    file('/tmp/SystemOnTPTP15237/zfmisc_1__e1_10_1.p', e1_10_1)).
fof(8, axiom,![X1]:(r1_tarski(X1,k1_xboole_0)=>equal(X1, k1_xboole_0)),
    file('/tmp/SystemOnTPTP15237/zfmisc_1__e1_10_1.p', t3_xboole_1)).
fof(9, negated_conjecture,
    ~((r1_tarski(c1_10_1,k1_xboole_0)<=>equal(c1_10_1, k1_xboole_0))),
    inference(assume_negation,[status(cth)],[7])).
fof(11, plain,![X3]:![X4]:r1_tarski(X3,X3),
    inference(variable_rename,[status(thm)],[1])).
cnf(12,plain,(r1_tarski(X1,X1)),inference(split_conjunct,[status(thm)],[11])).
fof(22, negated_conjecture,
    ((~(r1_tarski(c1_10_1,k1_xboole_0))|~(equal(c1_10_1, k1_xboole_0)))
     &(r1_tarski(c1_10_1,k1_xboole_0)|equal(c1_10_1, k1_xboole_0))),
    inference(fof_nnf,[status(thm)],[9])).
cnf(23,negated_conjecture,(c1_10_1=k1_xboole_0|r1_tarski(c1_10_1,k1_xboole_0)),
    inference(split_conjunct,[status(thm)],[22])).
cnf(24,negated_conjecture,(c1_10_1!=k1_xboole_0|~r1_tarski(c1_10_1,k1_xboole_0)),
    inference(split_conjunct,[status(thm)],[22])).
fof(25, plain,![X1]:(~(r1_tarski(X1,k1_xboole_0))|equal(X1, k1_xboole_0)),
    inference(fof_nnf,[status(thm)],[8])).
fof(26, plain,![X2]:(~(r1_tarski(X2,k1_xboole_0))|equal(X2, k1_xboole_0)),
    inference(variable_rename,[status(thm)],[25])).
cnf(27,plain,(X1=k1_xboole_0|~r1_tarski(X1,k1_xboole_0)),
    inference(split_conjunct,[status(thm)],[26])).
cnf(29,negated_conjecture,(~r1_tarski(c1_10_1,k1_xboole_0)),
    inference(csr,[status(thm)],[24,27])).
cnf(30,negated_conjecture,(c1_10_1=k1_xboole_0),
    inference(sr,[status(thm)],[23,29,theory(equality)])).
cnf(31,negated_conjecture,($false),
   inference(rw,[status(thm)],
     [inference(rw,[status(thm)], [29,30,theory(equality)]),
      12,theory(equality)])).
cnf(32,negated_conjecture,($false),inference(cn,[status(thm)],
                                        [31,theory(equality)])).
cnf(33,negated_conjecture,($false),32,['proof']).
```

Obviously, different ATP systems (or even the same ATP run with different settings) can produce different refutational proofs, and these proofs will generally also differ from the hypothetical "Mizar proof" (i.e., the steps done by the Mizar checker

[16] http://www.cs.miami.edu/~tptp/cgi-bin/SystemOnTPTPFormMaker

to justify the conjecture). Certainly, we might try to optimize the search in order to find e.g., the shortest (and thus hopefully the "best understandable") proofs. On the other hand, as noted above, these inferences are supposed to be quite easy for humans, and they turn out to be quite easy also for ATPs, so such (potentially quite resource-intensive) optimization is probably not worth its cost. A much better way to make the proofs more understandable is to spend some effort on their presentation, which is what we do in the following steps.

2.2.3 Preparing the ATP proof for MML Query rendering

The TSTP proof object is first transformed to the XML encoding of TSTP, by Petr Pudlák's and Geoff Sutcliffe's tool TPTP2XML. The XML listing would be too long to be included here, and it is available on our web site [17]. Then we apply our XSLT stylesheet `tstp2dli.xsl` [18] which translates the TPTP notation to the MML Query DLI (Decoded Library Item) format. MML Query is only used to process the DLI-encoded formulas, so another task of `tstp2dli.xsl` is to take care of the linking of formula names to the Mizar HTML pages, and linking of the ATP step references. The result is again on our web site [19].

2.2.4 Presentation with MML Query

As mentioned above, the MML Query DLI format is a notation for the Mizar semantic level. Therefore two tasks have to be done by MML Query:

- translate the formulas in MNF back into a human-friendly notation (i.e., usually compress it by using more logical connectives)
- translate Mizar constructors back into a human-friendly notation (i.e., find suitable Mizar symbols and their patterns, which correspond to the given constructor form)

First we explain the MNF transformation. A conjunction in MNF used in MML Query may have more than two conjuncts (the result of the associativity of conjunction in Mizar). Such situation is denoted here by $\&(\ldots)$. The reconstruction of richer set of connectives ($\exists$, $\vee$, $\Rightarrow$, and $\Leftrightarrow$) tries to reverse the Mizar transformation algorithm and could be described by following rules:

$$\begin{aligned} \neg\forall_x\varphi &\rightarrow \exists_x\neg\varphi \\ \neg\&(\varphi_1,\ldots,\varphi_{n-1},\neg\varphi_n) &\rightarrow \&(\varphi_1,\ldots,\varphi_{n-1}) \Rightarrow \varphi_n \\ \neg\&(\neg\varphi_1,\ldots,\neg\varphi_n) &\rightarrow \varphi_1 \vee \ldots \vee \varphi_n \end{aligned}$$

The reconstruction of $\Leftrightarrow$ is a bit more complicated. If formula has the form

$$\neg\&(\varphi_1,\ldots,\varphi_k,\varphi_{k-1},\ldots,\neg\varphi_n) \wedge \neg\&(\psi_1,\ldots,\psi_m)$$

and

$$\&(\neg\&(),\varphi_1,\ldots,\varphi_k) \equiv \&(\psi_1,\ldots,\psi_m)$$

[17] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html.930/_by_xml/zfmisc_1/164_29.xml
[18] http://kti.ms.mff.cuni.cz/cgi-bin/viewcvs.cgi/xsl4mizar/tstp2dli.xsltxt?view=markup
[19] http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/_by_dli/zfmisc_1/zfmisc_1__164_29.dli

then it is transformed to

$$\&(\varphi_1, \ldots, \varphi_k) \Leftrightarrow \neg\&(\varphi_{k+1}, \ldots, \varphi_n)$$

The equivalence $\equiv$ is the smallest equivalence satisfying the conditions of double negation and single conjunction:

$$\neg\neg\varphi \equiv \varphi \qquad \text{and} \qquad \&(\varphi) \equiv \varphi$$

The compression of quantifiers is also applied and formulas of the form $\forall_x(\varphi \Rightarrow \psi)$ are presented as `for x st` φ `holds` ψ. The indenting and breaking of long formulas is applied for better readability. The above rules assume that transformed formulas do not include double negations nor nested or single conjunctions (this is the case of formulas generated from Mizar). So, the transformation of an arbitrary formula expressed with $\wedge$, $\neg$, $\top$, and $\forall$ requires additional pruning at beginning. This has been implemented to handle our ATP-generated data.

The translation of Mizar constructors into the human-friendly notation uses the "rule of first available notation". Generally, multiple synonyms (or antonyms) can exist as user symbols corresponding to one Mizar constructor (in other words, when a Mizar author introduces a synonym, it exists only on the presentation level). Since in the general case (like the one when data come from ATPs) we have no other information than the constructor format, it is reasonable to present the constructor using just the first human symbol found in the MML, which corresponds to that constructor. More special rules can be (and are) used by MML Query, when additional information is available, e.g., about the Mizar articles from which the constructor encoding comes.

The result of the final MML Query rendering step on our example is already shown above, it is the final explanation of the "simple justification" inference that is presented to the reader.

3 MML Query presentation versus the XML-based HTML presentation of full articles

The above described by-explanation system, which we have implemented for the 35 initial Mizar articles, uses two different techniques for presentation of the Mizar semantic level. The presentation of the by-explanations is done by the MML Query "artificial intelligence" reconstruction of a possible user notation (described in 2.2.4). With no other information added to the constructor encoding, there is really no other choice. On the other hand, for the XML-based HTML presentation of full articles, to which the MML Query explanations are linked, a lot of further information is available, namely from the Mizar parser, which was used to process the original Mizar text. Even though the purpose of the Mizar XML format is to primarily contain the semantic information, the XML format allows for easy addition of the original presentation information. This feature has been added to Mizar some time

ago, and the XML produced by recently distributed Mizar versions already contains this additional presentational information.

It is therefore quite likely, that in many cases the presentations of the same Mizar formula by these two systems will differ. We don't think that in the particular case of the system presented above this is necessarily harmful. While the goal of the XML-based HTML presentation is to achieve high resemblance to the original article, with as much additional semantic information as possible, the goal of the MML Query presentation is to present pieces of Mizar text in a uniform and predictable way. In this sense the MML Query presentation can be thought of as a tool for strong uniform formatting of Mizar. While this difference may be initially surprising for a reader, when he first uses the by-explanation functionality, this might also lead to his deeper understanding of the presentation process, and of Mizar itself. Should this become a serious problem, we can always offer to the user to apply the MML Query formatting to the whole XML-based article presentation.

4 Conclusions and Future Work

We have provided an initial implementation of a system explaining the Mizar atomic "simple justification" inferences, and demonstrated it on 35 initial Mizar articles. For this, we have used or newly developed a chain of tools working on the Mizar semantic level. The systems could be already now extended to a much larger portion of MML, since the ATP success rate on solving nonnumerical Mizar "simple justification" problems is very high (above 90% with 4s timelimit and one ATP used in a push-button manner, and above 99% with more sophisticated, but still fully automated methods described in Section 4.3. of [17]). However we still do not have a satisfactory algorithm for serious transformation of the ATP solutions on heavily typed Mizar articles to the Mizar typed semantic level. It is possible already now to render such solutions with MML Query as mentioned above, and it probably would be useful purely for the explanation purpose. However such rendering will be incorrect from the Mizar parser's point of view, which requires that variables are qualified with proper types when typed functors or predicates are applied to them. A transformation of the ATP untyped solutions to Mizar-acceptable typed solutions is probably feasible, and it is an interesting line of further research. Obviously all the tools participating in our chain can be improved too. One interesting idea is to have all of the systems participating in the chain working in real time, passing the solutions to each other. Such functionality will probably be developed quite soon, and generally used for providing ATP support to Mizar authors. From this point of view, providing the current by-explanation functionality can be thought of as a test-bed, making the way for more ambitious ATP-for-Mizar applications.

5 Acknowledgments

Thanks to Stephan Schulz, Geoff Sutcliffe, and Petr Pudlák for developing and providing the ATP systems and metasystems participating in our tool chain. Geoff

Sutcliffe has also first suggested the "click for ATP" functionality in the Mizar HTML presentation. This work was supported by a Marie Curie International Fellowship within the 6^{th} European Community Framework Programme.

References

[1] Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors. *Mathematical Knowledge Management, Third International Conference, MKM 2004, Bialowieza, Poland, September 19-21, 2004, Proceedings*, volume 3119 of *Lecture Notes in Computer Science*. Springer, 2004.

[2] Grzegorz Bancerek and Piotr Rudnicki. Information retrieval in MML. In *MKM*, volume 2594 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2003.

[3] Grzegorz Bancerek and Josef Urban. Integrated semantic browsing of the Mizar Mathematical Library for authoring Mizar articles. In Asperti et al. [1], pages 44–57.

[4] Czeslaw Bylinski. Some basic properties of sets. *Formalized Mathematics*, 1(1), 1989.

[5] S. Jaskowski. On the rules of suppositions. *Studia Logica*, 1, 1934.

[6] W. W. McCune. Otter 3.0 reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne, Illinois, 1994.

[7] Adam Naumowicz and Czeslaw Bylinski. Improving mizar texts with properties and requirements. In Asperti et al. [1], pages 290–301.

[8] F. J. Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20:1 – 31, 1999.

[9] Piotr Rudnicki and Andrzej Trybulec. On equivalents of well-foundedness. *J. Autom. Reasoning*, 23(3-4):197–234, 1999.

[10] P. Rudnicki. An overview of the Mizar project. In *1992 Workshop on Types for Proofs and Programs*, pages 311–332. Chalmers University of Technology, Bastad, 1992.

[11] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *Journal of AI Communications*, 15(2-3):91–110, 2002.

[12] S. Schulz. E – a brainiac theorem prover. *Journal of AI Communications*, 15(2-3):111–126, 2002.

[13] G. Sutcliffe and C.B. Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[14] Josef Urban. MPTP - motivation, implementation, first experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

[15] Josef Urban. MizarMode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. *Journal of Applied Logic*, 2005. Article In Press, doi:10.1016/j.jal.2005.10.004, available online at http://ktiml.mff.cuni.cz/~urban/mizmode.ps.

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

[17] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 2006. Accepted for Publication.

[18] Josef Urban. XML-izing Mizar: making semantic processing and presentation of MML easy. In Michael Kohlhase, editor, *MKM 2005*, volume 3863 of *Lecture Notes in Artificial Intelligence*, pages 346–360. Springer, 2006.

[19] Christoph Weidenbach, Uwe Brahm, Thomas Hillenbrand, Enno Keen, Christian Theobald, and Dalibor Topic. SPASS version 2.0. In *CADE*, pages 275–279, 2002.

[20] C. Weidenbach. *Handbook of Automated Reasoning*, volume II, chapter SPASS: Combining Superposition, Sorts and Splitting, pages 1965–2013. Elsevier and MIT Press, 2001.

[21] Freek Wiedijk. CHECKER - notes on the basic inference step in Mizar. available at http://www.cs.kun.nl/~freek/mizar/by.dvi, 2000.