Automated Reasoning for Mizar: Artificial Intelligence through Knowledge Exchange

Josef Urban*
Charles University in Prague

Abstract

This paper gives an overview of the existing link between the Mizar project for formalization of mathematics and Automated Reasoning tools (mainly the Automated Theorem Provers (ATPs)). It explains the motivation for this work, gives an overview of the translation method, discusses the projects and works that are based on it, and possible future projects and directions.

1 Introduction and Motivation

1.1 Why Mizar?

The Mizar proof assistant [Rud92, RT99] was chosen by the author for experiments with automated reasoning tools because of its focus on building the large formal Mizar Mathematical Library (MML) [RT03]. This formalization effort was started in 1989 by the Mizar team, and its main purpose is to verify a large body of mainstream mathematics in a way that is close and easily understandable to mathematicians, allowing them to build on this library with proofs from more and more advanced mathematical fields. This formalization goal has influenced:

- the choice of a relatively human-oriented formal language in Mizar
- the choice of the declarative Mizar proof style (Jaskowski-style natural deduction)
- the choice of first-order logic and set theory as unified common foundations for the whole library
- the focus on developing and using just one human-obvious [Rud87] first-order justification rule in Mizar
- and the focus on making the large library interconnected, usable for more advanced formalizations, and using consistent notation.

There are other systems and projects that are similar to Mizar in some of the above mentioned aspects. For example, building large and advanced formal libraries seems to be more and more common today, probably also because of the recent large formalization projects like Flyspeck [CLR06] that require a number of previously proved nontrivial mathematical results. In the work that is described here, Mizar thus should be considered as a suitable particular choice of a system for formalization of mathematics, which uses relatively common and accessible foundations, and produces a large formal library written in a relatively simple and easy-to-understand style. Some of the author's systems described below also work with other than Mizar data: for example, the learning from proofs has been experimentally instantiated to learn from Hales' proof of the Jordan Curve Theorem done in HOL Light[1] [HSA06], while MaLARea has already been successfully used for reasoning over problems from the large formal SUMO ontology[2] [NP01].

Rudnicki P, Sutcliffe G., Konev B., Schmidt R., Schulz S. (eds.);
Proceedings of the Combined KEAPPA - IWIL Workshops, pp. 1-16

*A large part of the work described here was supported by a Marie Curie International Fellowship within the 6^{th} European Community Framework Programme.

[1] `http://lipa.ms.mff.cuni.cz/~urban/holpademo.html`
[2] `http://www.ontologyportal.org/reasoning.html`

1.2 Benefits of cooperation

There are three kinds of benefits in linking proof assistants like Mizar and their libraries with Automated Reasoning technology and particularly ATPs:

- The obvious benefits for the proof assistants and their libraries. Automated Reasoning can provide a number of tools and strong methods that can assist formalization.
- The (a bit less obvious) benefits for the field of Automated Reasoning. For example, research in automated reasoning over very large libraries is painfully theoretical (and practically useless) until such libraries are really available for experiments. Mathematicians (and scientists, and other human "reasoners") typically know a lot of things about the domains of discourse, and use the knowledge in many ways that include many heuristical methods. It thus seems unrealistic (and limiting) to develop the automated reasoning tools solely for problems that contain only a few axioms, make little use of previously accumulated knowledge, and do not attempt to further accumulate and organize the body of knowledge.
- The benefits for the field of general Artificial Intelligence. These benefits are perhaps the least mentioned ones[3], however to the author they appear to be the strongest long-term motivation for this kind of work. In short, the AI fields of *deductive reasoning* and the *inductive reasoning* (represented by machine learning, data mining, knowledge discovery in databases, etc.) have so far benefited relatively little from each other's progress. This is an obvious deficiency in comparison with the human mind, which can both inductively suggest new ideas and problem solutions based on analogy, memory, statistical evidence, etc., and also confirm, adjust, and even significantly modify these ideas and problem solutions by deductive reasoning and explanation, based on the understanding of the world. Repositories of "human thought" that are both large (and thus allow the inductive methods), and have precise and deep semantics (and thus allow deduction) should be a very useful component for cross-fertilization of these two fields. Large formal mathematical libraries are currently the closest approximation to such a computer-understandable repository of "human thought" usable for these purposes. To be really usable, the libraries however again have to be presented in a form that is easy for existing automated reasoning tools to understand.

As mentioned above, Mizar is not the only system that can be used for experiments with automated reasoning and AI methods, and the work described here is related to a number of other works and projects. A lot of work on translating Mizar for automated reasoning tools was previously done in the ILF project [DW97]. The Isabelle/HOL system has been recently linked to first-order ATP systems [MP08, MP06], and this link is already widely used by the Isabelle community. The SAD proof verification system [VLP07] has been using ATP systems quite extensively. A translation of the Cyc knowledge base to first-order logic exists [RPG05], as well as a translation of the above mentioned SUMO ontology [PS07]. Machine learning and other complementary AI methods for formal mathematics have been studied, e.g., in [DFGS99, Len76, Col02, CMSM04, Faj88],

1.3 Structure of this paper

In Section 2 we give a brief overview of the Mizar extensions over pure first-order logic, like its type system and second-order constructs. Section 3 summarizes the methods that are used for translating the Mizar formalism and library to a pure first-order format suitable for automated reasoning tools. Section 4

[3] This may also be due to the frequent feeling of too many unfulfilled promises and too high expectations from the general AI, that also led to the current lack of funding for projects mentioning Artificial Intelligence.

shows several automated reasoning, proof assistance, and AI experiments and projects that are based on the current translation. Section 5 discusses some future possibilities.

2 Main Mizar extensions to pure first-order logic

2.1 Mizar types

Mizar is to a large extent a first-order system (using set-theoretical axioms), enhanced with a number of extensions that (are supposed to) make the formalization in Mizar more human-friendly. The largest of these extensions is the Mizar type system. Mizar allows defining *types* from first-order predicates. The difference between types and predicates in Mizar is not semantic: both are first-order predicates. The difference is technical and practical: representing some predicates as Mizar types allows the use of several type-related automations in Mizar. An example of this is automated use of type hierarchies, i.e., hierarchies saying that the descendant type is a subclass of the ancestor type. Table 1 shows such a hierarchy of type definitions started at the Mizar type "`Matrix of m,n,D`" in the article `MATRIX_1` [Jan91]. The exact meaning of the symbols and definitions mentioned there can be best explored in a linked presentation. A particular one (also used for linking with ATP tools) is available at `http://www.tptp.org/MizarTPTP`, where the following type hierarchy starts at `http://www.tptp.org/MizarTPTP/Articles/matrix_1.html#M1`.

These type declarations allow Mizar to automatically infer that any term with the type "`Matrix of m,n,D`" has also the following types, i.e., that the corresponding predicates hold about the term:

```
Matrix of D
tabular FinSequence of D*
FinSequence-like PartFunc of NAT,D*
Function-like Relation of NAT,D*
Subset of [:NAT,D*:]
Element of bool [:NAT,D*:]
set
```

It can be seen that these *Mizar types* consist of several parts: they can have one or more (unparameterized) *adjectives* like `tabular`, `Finsequence-Like`, and `Function-like`, and a (possibly parametrized) *type radix* like `Matrix of m,n,D`, `Matrix of D`, `FinSequence of D*`, `PartFunc of NAT,D*`, `Relation of NAT,D*`, etc. While only unidirectional widening hierarchies (like the one in Table 1) are allowed for the *type radices*, the mechanisms used for Mizar *adjectives* allow practically arbitrary Horn clauses (called *clusters* in Mizar), which results in fixpoint-like algorithms for computing the complete set of adjectives of a given term. For example, a term of the above mentioned type "`Matrix of m,n,D`" would automatically inherit all the adjectives of its type's ancestors (`tabular`, `Finsequence-like`, and `Function-like`), but it would also get all adjectives added by Mizar *clusters* to this initial set of adjectives. An example of such a cluster[4] is:

```
registration
cluster FinSequence-like -> finite set;
end;
```

This can be understood as the Horn clause:

```
finite :- FinSequence-like.
```

[4] `http://www.tptp.org/MizarTPTP/Articles/finseq_1.html#CC1`

Table 1: Hierarchy of type definitions for "Matrix of m,n,D"

```
definition
let D be non empty set;
let m, n be Nat;
mode Matrix of m,n,D -> Matrix of D means :Def3:  ::  MATRIX_1:def 3
len it = m & ( for p being FinSequence of D st p in rng it holds len p = n );
end;

definition
let D be set ;
mode Matrix of D is tabular FinSequence of D*;
end;

definition
let D be set;
redefine mode FinSequence of D -> FinSequence-like PartFunc of NAT,D;
end;

definition
let X, Y be set;
mode PartFunc of X,Y is Function-like Relation of X,Y;
end;

definition
let X, Y be set;
redefine mode Relation of X,Y -> Subset of [:X,Y:];
end;

definition
let X be set;
mode Subset of X is Element of bool X;
end;

definition
let X be set;
mode Element of X -> set means :Def2:  ::  SUBSET_1:def 2
it in X if not X is empty otherwise it is empty;
end;
```

which adds the adjective `finite` to the set of adjectives of any term which already has the adjective `Finsequence-like`. In this way the Mizar terms can get relatively large numbers of adjectives automatically, making significant portions of formal reasoning obvious to Mizar.

Mizar *structure types* are a special kind of Mizar types that are intended to encode mathematical structures, usually consisting of a carrier set and some operations on it. Typical examples are algebraic structures like groups, rings, and vector spaces, but there are also many other structures like topological and metric spaces. These structures have a special implementation in Mizar (providing additional automations) that differs from those of other Mizar types. This has both advantages and disadvantages, see [LR07] for a discussion and an alternative implementation of structures relying on the standard type mechanisms in Mizar.

2.2 Mizar second-order constructs

Mizar axiomatics is Tarski-Grothendieck set theory with strong axiom of choice. This is an extension of ZFC that adds arbitrarily large inaccessible cardinals to the universe. This extension is used to model some parts of category theory in Mizar. However, for practically all applications it is enough to think of Mizar's axiomatics as ZFC with strong choice. Particularly, the standard *Replacement (Fraenkel) Axiom scheme*[5] ("image of any set under any definable function is again a set") is used:

```
scheme :: TARSKI:sch 1
Fraenkel { A()-> set, P[set, set] }:
 ex X st for x holds x in X iff ex y st y in A() & P[y,x]
 provided for x,y,z st P[x,y] & P[x,z] holds y = z;
```

The expression `P[set, set]` here declares a "second-order" predicate variable. Its Mizar semantics is that it can be instantiated with any Mizar formula with two free variables of the type *set*. Once Mizar has to allow such second-order mechanisms for the axiomatics, it is advantageous for human authoring to allow them also for regular theorems. For example, the *Separation (Comprehension) scheme*[6] ("any definable subclass of a set is again a set"):

```
scheme :: XBOOLE_0:sch 1
 Separation { A()-> set, P[set] } :
   ex X being set st for x being set holds
                          x in X iff x in A() & P[x];
```

can be inferred from Replacement, but is much more often used in MML (490 uses of Separation vs. 24 uses of Replacement), and probably in normal mathematics too. The Replacement scheme is commonly used in mathematics to produce *Fraenkel (Abstract) terms*, i.e., terms of the form

```
{ N - M where M,N is Integer :  N < M }
```

Neither the schemes nor the Fraenkel terms can be directly expressed in pure first-order logic.

3 MPTP: Translating Mizar for Automated Reasoning tools

The translation of Mizar and MML to pure first-order logic is described in detail in [Urb03, Urb04, Urb07b, US07]. This section provides an overview of the translation using the MPTP (Mizar problems for Theorem Proving) system. In addition to the logical extensions mentioned above in Section 2, the translation from Mizar to pure first-order logic also has to deal with a number of practical issues related to the Mizar implementation, implementations of first-order ATP systems, and the most frequent uses of the translation system.

3.1 MPTP 0.1

The first version of MPTP is described in detail in [Urb03, Urb04]. This version was used for initial exploration of the usability of ATP systems on the Mizar Mathematical Library (MML). The first important number obtained was the 41% success rate of ATP-reproving of about 30000 MML theorems from other Mizar theorems and definitions selected from the corresponding MML proofs.

No previous evidence about the feasibility and usefulness of ATP methods on a very large library like MML was available prior to the experiments done with MPTP 0.1[7], sometimes leading to overly

[5] `http://www.tptp.org/MizarTPTP/Articles/tarski.html#S1`

[6] `http://www.tptp.org/MizarTPTP/Articles/subset_1.html#S1`

[7] A lot of work on MPTP was inspired by the previous work done in the ILF project [DW97] on importing Mizar. However it seemed that the project had stopped before it could finish the export of the whole MML to ATP problems, and provide some initial overall statistics of ATP success rate on MML.

pessimistic views on such a project. Therefore the goal of this first version was to relatively quickly achieve a "mostly-correct" translated version of the whole MML that would allow assessment of the potential of ATP methods for this large library. Many shortcuts and simplifications were therefore taken in this first MPTP version, naming at least the following:

- Mizar formulas were directly exported to the DFG [HKW96] syntax used by the SPASS [WBH+02] system. SPASS seemed to perform best on MPTP problems, probably because of its handling of sort theories. SPASS also has a built-in efficient clausifier [NW01], which the other efficient provers like E [Sch02] and Vampire [RV02] did not have at that time (CNF was the main category of the CASC competition until 2006).
- One simple method of handling sorts (encoding as predicates and relativization) was chosen for the export, yielding standard (untyped) first-order formulas, from which the original type information could not be recovered and used for different encodings.
- Mizar proofs were not exported. Only the lists of MML references (theorems and definitions) used for proof of each MML theorem were remembered for re-creation of ATP problems corresponding to Mizar proofs (see Section 3.2.4 for overview of the problem creation in MPTP). The proof structure and internal lemmas[8] were forgotten.
- Such lists of MML references were *theoretically* sufficient[9] as premises for re-proving of about 80% (27449 out of 33527) of theorem proofs - i.e., the Mizar proofs use only these MML references and some implicit (background) facts like type hierarchy, arithmetical evaluations, etc. In the Mizar proofs of the remaining ca 20% (6078) of theorems, Mizar *schemes* and top-level non-theorem lemmas[10] were used. These two kinds of propositions were completely ignored, making these theorems not eligible for ATP reproving.
- The export of Mizar *structure* types was incomplete (some axioms were missing), *abstract terms* were translated incorrectly, and the background theory computed for problems could sometimes be too strong, possibly leading to MML-invalid (cyclic) proofs. All these shortcuts were justified by the low frequency of such cases in MML.

Many of these simplifications made further experiments with MPTP difficult or impossible, and also made the 41% success rate uncertain (some ATP proof could succeed, and some could fail because of these simplifications). The lack of proof structure prevented measurement of ATP success rate on all internal proof lemmas, and experiments with unfolding lemmas with their own proofs. Additionally, even if only several abstract terms were translated incorrectly, during such proof unfoldings their effect could spread much wider. Experiments like finding new proofs, and cross-verification of Mizar proofs (described below) would suffer from constant doubt about the possible amount of error caused by the incorrectly translated parts of Mizar, and debugging would be very hard.

3.2 MPTP 0.2

During the work on MPTP 0.1 and other Mizar-related systems, it became clear that the old internal format used by Mizar (designed long ago, when memory and storage were expensive) was quite hard

[8] An *Internal Lemma* is a lemma proved inside a proof of a MML theorem. It can be proved either by *Simple Justification* (Mizar keyword "`by`") or it can have its own structured subproof (Mizar keywords "`proof ... end`").

[9] This means that the proof should be found by a complete ATP system with unlimited resources. As mentioned above, the *practical* ATP success was 41%.

[10] The vast majority of Mizar propositions proved at the top-level (i.e., not inside a proof of another proposition) are exported from Mizar as theorems reusable in other articles. This is however not mandatory.

to extend for new Mizar constructs and utilities. A new extensible and richer format seemed to be needed for Mizar itself, and for systems like MPTP, MMLQuery [BR03], MoMM [Urb06b], MizarMode [Urb06a, BU04], each of which had its own special-purpose exporter from Mizar doing very similar things.

This resulted in quite a large reimplementation of Mizar described in [Urb05]. Mizar started to use XML[11] natively as its internal format produced during parsing. The internal format was significantly extended, and it now contains a very complete semantically disambiguated form of a Mizar article, together with a large amount of presentational information that allows quite faithful re-creation of Mizar articles from this internal format. Because of the completeness of this format, and thanks to the widespread availability of XML parsers, the need for special-purpose Mizar exporters for various systems and the problem of their maintenance were largely eliminated. This allows quite simple combinations of Mizar-based systems, for example both the HTML and the MPTP/TPTP parts of the MizarTPTP presentation at `http://www.tptp.org/MizarTPTP` [UTSP07] are produced from the same XML form of Mizar articles, with quite simple changes to the XSL stylesheets producing the HTML.

After this necessary upgrade of Mizar, MPTP 0.2 was started from scratch, using a simple XSL stylesheet (instead of the Pascal exporter) to export the Mizar XML into an extended TPTP-like [SS98] format, and using Prolog (instead of Perl) for generating ATP problems in TPTP. In MPTP 0.1 the translated formulas were already in DFG format, and Perl (treating formulas mostly just as strings) was enough to generate the ATP problems. This is no longer true in MPTP 0.2: the extended TPTP-like format is (so far) not directly usable by ATPs, because it encodes Mizar types and abstract terms (and some other things) in a generic way, allowing different translations. Prolog was therefore needed to implement structural functions on the formulas, such as the type relativization. The MPTP 0.2 codebase has grown from ca. 900 lines of XSLTXT[12] (a compact human-friendly version of XSL) and 1500 lines of Prolog in 2005 to ca. 2000 lines of XSLTXT and 5500 lines of Prolog in 2008. The basis of the system remains the same, but the codebase has grown as a number of different functionalities (used by the projects mentioned below) have been added.

In the following subsections we summarize the main translation methods and functionalities used currently by MPTP 0.2.

3.2.1 Type translation in MPTP 0.2

MPTP extends the TPTP language to allow *parametrized types* like "`Matrix of n,m,D`", and "`tabular FinSequence of D*`" mentioned in Section 2. For example the following Mizar formula:

```
for G being infinite Graph, W1 being Walk of G
  ex W2 being Subwalk of W1 st W1=W2;
```

is translated into this extended TPTP format:

```
! [G : (~ finite & graph), W1 : walk(G)]: (? [W2 : subwalk(W1)] : (W1=W2))
```

This differs from the (proposed) typed TPTP standard which only allows atomic sorts. MPTP currently only implements the predicate encoding (relativization) of types. Type declarations with arity n are transformed into predicates with arity $n+1$, and, e.g., the above formulas becomes:

```
! [G] :
  ( ( ~ finite(G) & graph(G) )
 => ! [W1] :
```

[11] See `http://lipa.ms.mff.cuni.cz/~urban/Mizar.html` for specification of the Mizar XML format.
[12] `http://www.zanthan.com/ajm/xsltxt/`

```
    ( walk(G,W1)
   => ? [W2] :
        ( subwalk(W1,W2) & (W1=W2) ) ) )
```

The implicit type hierarchies and fixpoint automations used in Mizar for adjectives are replaced by explicit inclusion of the corresponding formulas into the ATP problems. For example, the cluster

```
cluster FinSequence-like -> finite set;
```

mentioned in Section 2 is translated as:

```
! [X] : 'FinSequence-like'(X) => finite(X)
```

and explicitly added to the ATP problem when needed (see Section 3.2.4). Because pretty Mizar symbol names are largely overloaded (there are e.g. more than 100 different meanings of the symbol '+' in MML), their unique disambiguated internal naming is used in MPTP. That means that the above formula will actually look like this:

```
! [X] : v1_finseq_1(X) => v1_finset_1(X)
```

An attempt was started by the author to provide unique descriptive names for Mizar symbols[13], and these names have already been used for creating the MPTP Challenge problems (see below). However, there are around 10000 Mizar symbols, so cooperation from other Mizar and MPTP users is needed to incrementally improve this unique descriptive naming.

3.2.2 Translation of abstract terms in MPTP 0.2

The special `all`/3 predicate is used for encoding abstract terms in MPTP 0.2. For instance the following abstract term mentioned in Section 2.2

```
{ N - M where M,N is Integer : N < M }
```

is encoded as:

```
all([M:integer,N:integer], minus(N,M), less(N,M))
```

Abstract terms are very similar to lambda terms, which are sometimes called anonymous functions. Therefore the process of removing abstract terms and inventing fresh names for them was called *deanonymization* in [Urb07b]. It seems that a similar procedure is used in the export of lambda terms in Isabelle/HOL to first-order logic, with the name *lambda-lifting*. The procedure is very similar to Skolemization, and that is why this syntactic extension could eventually become handled by standard ATP clausifiers, or even dealt with in calculi which implement delayed transformation to normal forms (e.g., tableaux or [GS03]). It means that a new functor symbol is introduced, corresponding to the abstract term in the following way:

```
![X]:(in(X,all_0_xx) <=> ?[N:integer,M:integer]:
                           (X = minus(N,M) & less(N,M))).
```

Here `all_0_xx`/0 is the newly introduced "Fraenkel" functor for the abstract term given above, the first number in it (`0`) is its arity and the second number (`xx`) just a serial numbering of such symbols with the same arity. Obviously Fraenkel functors with nonzero arity can arise if their context includes quantified variables, this is similar to Skolemization. The predicate `in`/2 (set-theoretic membership) has to be available for this encoding.

[13] `http://wiki.mizar.org/cgi-bin/twiki/view/Mizar/NiceConstructorNames`

As with Skolemization, a lot of optimizing steps can be done during deanonymization. If one abstract term is used twice, only one Fraenkel functor is necessary. This has the additional advantage that the equality of such terms is obvious, while for different Fraenkel functors the Extensionality axiom (*"two sets are equal if they contain the same elements"*) has to be used to find out that they encode the same term. Abstract terms are used often inside Mizar proofs, and proof-local constants often occur in them. A fairly efficient optimization is implemented by extracting the abstract terms from the proof context, i.e., by generalizing the proof-local constants appearing inside a term before the definition of the corresponding Fraenkel functor is created. When this is done for a whole Mizar article, the number of Fraenkel definitions can be reduced very significantly, sometimes by a factor of 10 or even 20. This extraction from the proof context turns out to be necessary for reproving Mizar theorems whose proofs contain abstract terms, because for such reproving attempts only proof-external symbols and references can be used, and Fraenkel definitions containing proof-local constants would not be accessible, possibly making the reproving task incomplete. Such article-global generation of Fraenkel definitions is a standard preprocessing step done immediately after the article is loaded (for MPTP processing) into Prolog, and the abstract terms are replaced globally in all formulas by the corresponding Fraenkel functors before any ATP problems are generated. Some of the generated Fraenkel functors are used very frequently, which suggests that they probably deserve their own proper definition as Mizar functors.

3.2.3 Translation of schemes in MPTP 0.2

The MPTP treatment of schemes (see Section 2.2) is similar to that of the abstract terms: the instances that are already present in the MML are used. This is sufficient for first-order reproving, and with a sufficiently large body of mathematics like MML (and thus sufficiently many first-order scheme instances), it could also be "reasonably sufficient" for proving new things (though obviously incomplete in general in this case).

The implementation uses Mizar (compiled with a special scheme reporting directive) to print the particular instantiations of the second-order functor and predicate variables. These second-order variables in schemes are encoded using a fresh functor or predicate symbol, so, e.g., the Separation (Comprehension) scheme (called `s1_xboole_1` in MPTP) is encoded in the extended language in this way:

```
? [B1: $true]: ![B2: $true]:
(in(B2,B1) <=> ( in(B2,f1_s1_xboole_0) & p1_s1_xboole_0(B2)))
```

i.e., in pure TPTP as:

```
? [B1]: (![B2]: (in(B2,B1) <=> ( in(B2,f1_s1_xboole_0) & p1_s1_xboole_0(B2))))
```

Here `f1_s1_xboole_0` and `p1_s1_xboole_0` are the fresh symbols encoding the second-order variables. This handling is semantically sufficient for reproving of Mizar schemes, because nothing (except their declared type restrictions) is known about these fresh first-order symbols (and therefore the proof can be instantiated to any particular first-order functors and predicates of the proper types). However, additional treatment is necessary when schemes are applied inside other proofs. In these cases, these symbols first have to be replaced with the instantiations reported by Mizar. For each scheme a number of its instances are thus obtained, again (as in the case of abstract terms) possibly containing some proof-local constants. Again, for reproving theorems, these instances have to be extracted from the proof context by generalizing the proof-local constants, and again this is done globally for a whole article and before any ATP problems are generated. Obviously, there is the objection that relying on Mizar and Prolog to carry out the second-order instantiation is a weak point of possible ATP cross-verification of Mizar proofs. The correctness of this procedure is, however, easy to check, by checking that the original scheme proofs (with the fresh first-order symbols) also work for the particular scheme instances (with the first-order symbols instantiated to the Mizar-supplied instances).

3.2.4 Problem creation and axiom selection in MPTP

There are several ways in which MPTP can create ATP problems. While the translation of the above mentioned Mizar constructs is usually fixed, the main degree of freedom is the selection of suitable premises from the translated MML for a particular conjecture. The most common MPTP task (used, e.g., for the re-proving experiments described in Section 4.1) is to generate ATP problem for a given MML theorem in such a way that all the MML theorems and definitions explicitly used in the MML proof are included, together with additional "background" formulas encoding the knowledge that can be used implicitly by Mizar. These background formulas encode the type hierarchy, definitional expansions, arithmetical evaluations, properties of Mizar functors and predicates like commutativity and antisymmetry, etc. The addition of such formulas into the ATP problems is done in a fixpoint algorithm watching the current set of symbols in the problem (initialized with the symbols contained in the formulas that are used explicitly in the MML proof), and adding these implicit facts when they might be needed. Several versions of this "enriching" algorithm exist in MPTP. The more background facts are added, the more complete the problem is (and the harder it can be for ATPs to re-prove the problem if the added axiom is redundant[14]). The default MPTP version is intended to be complete in the sense that an ATP problem corresponding to an existing Mizar proof will contain all the background axioms needed to re-play the Mizar proof by ATPs. This is useful for debugging, however stricter (heuristical) versions are available too. Additionally, this is an instance of the general "axiom selection" problem, for which specialized systems like MaLARea [Urb07a, USPV08] are being used.

4 Experiments and projects based on the MPTP

MPTP has so far been used for

- experiments with re-proving Mizar theorems and simple lemmas by ATPs from the theorems and definitions used in the corresponding Mizar proofs
- experiments with fully automated re-proving of Mizar theorems, i.e., the necessary axioms being selected fully automatically from the whole available MML
- finding new ATP proofs that are simpler than the original Mizar proofs
- ATP-based cross-verification of the Mizar proofs
- ATP-based explanation of Mizar atomic inferences
- inclusion of Mizar problems into the TPTP problem library, and unified web presentation of Mizar together with the corresponding TPTP problems
- creation of the MPTP $100 Challenges for reasoning in large theories in 2007, and subsequent creation of the MZR category of the CASC Large Theory Batch (LTB) competition in 2008
- a testbed for AI systems like MaLARea targeted at reasoning in large theories and combining inductive techniques like machine learning with deductive reasoning

[14]For example, the SPASS prover has recently re-proved the Lagrange's theorem in Mizar (`http://www.tptp.org/MizarTPTP/Articles/group_2.html#T177`) from 25 premises. However the default MPTP background-adding algorithm results in inclusion of another 135 formulas into the ATP problem, making the problem impossible to prove by existing standard ATPs.

4.1 Re-proving experiments

As mentioned in Section 3.1, the initial large-scale experiment done with MPTP 0.1 indicated that 41% of the Mizar proofs could be automatically found by ATPs, if the user provides the same theorems and definitions that are used in the Mizar proofs, plus the corresponding background formulas. As already mentioned, this number was far from certain, e.g., out of the 27449 problems tried, 625 were shown to be CounterSatisfiable by SPASS (pointing to various oversimplifications taken in MPTP 0.1). The experiment was therefore repeated with MPTP 0.2, but with only 12529 problems that come from articles that do not use internal arithmetical evaluations done by Mizar. These evaluations were not handled by MPTP 0.2 at the time these experiments were conducted, being the last (known) part of Mizar that could be blamed for possible ATP incompleteness. The E prover version 0.9 and SPASS version 2.1 were used for this experiment, with a 20s time limit (due to limited resources). The results (reported in [Urb07b]) are given in Table 2. 39% of the 12529 theorems were proved by either SPASS or E, and no countersatisfiability was found.

Table 2: Reproving of the theorems from non-numerical articles by MPTP 0.2 in 2005

description	proved	countersatisfiable	timeout or memory out	total
E 0.9	4309	0	8220	12529
SPASS 2.1	3850	0	8679	12529
together	4854	0	7675	12529

These results have thus, to a large extent, confirmed the optimistic outlook of the first measurement in MPTP 0.1. In later (so far unreported) experiments, this ATP performance has been steadily going up; see Table 3 for results from a 2007 run with a 60s timelimit. This is a result of better pruning of redundant axioms in MPTP, and also of ATP development, which obviously was influenced by the inclusion of MPTP problems into the TPTP library, forming a significant part of the FOF problems in the CASC competition since 2006. Together, in this increased timelimit, the newer versions of E and SPASS solved 6500 problems, i.e., 52% of them all. With the addition of Vampire and its customized Fampire version (which alone solves 51% of the problems), the combined success rate went up to 7694 of these problems, i.e., to 61%. The caveat is that the methods for dealing with arithmetics are becoming stronger and stronger in Mizar, and so far it is not clear how to handle them efficiently in ATPs. The MPTP problem creation for problems containing arithmetics is thus currently quite crude, and the ATP success rate on such problems will likely be significantly lower than on the nonarithmetical ones.

Table 3: Reproving of the theorems from non-numerical articles by MPTP 0.2 in 2007

description	proved	countersatisfiable	timeout or memory out	total
E 0.999	5661	0	6868	12529
SPASS 2.2	5775	0	6754	12529
E+SPASS together	6500	-	-	12529
Vampire 8.1	5110	0	7419	12529
Vampire 9	5330	0	7119	12529
Fampire 9	6411	0	6118	12529
all together	7694	-	-	12529

4.2 Finding new proofs and the AI aspects

MPTP 0.2 was also used to try to prove Mizar theorems fully automatically, i.e., the choice of premises for each theorem was done automatically, and all previously proved theorems were eligible. Because giving ATPs thousands of axioms is usually hopeless[15], the axiom selection was done by symbol-based machine learning from previously available proofs. The results (reported in [Urb07b]) are given in Table 4. 2408 of the 12529 theorems were proved either by E 0.9 or SPASS 2.1 from the axioms selected by the machine learner, and the combined success rate of this whole system was thus 19%. These experiments have not been repeated so far, as the combination of machine learning and other axiom selection methods have recently been under heavy development in the MaLARea system.

Table 4: Proving new theorems with machine learning support by MPTP 0.2 in 2005

description	proved	countersatisfiability	timeout or memory out	total
E 0.9	2167	0	10362	12529
SPASS 2.1	1543	0	10986	12529
together	2408	0	10121	12529

This experiment demonstrates a very real and quite unique benefit of large formal mathematical libraries for conducting novel integration of AI methods. As the machine learner is trained on previous proofs, it recommends relevant premises from the large library that (according to the past experience) should be useful for proving new conjectures. A variety of machine learning methods (neural nets, Bayes nets, decision trees, nearest neighbor, etc.) can be used for this, and their performance evaluated in the standard machine learning way, i.e., by looking at the actual axiom selection done by the human author in the Mizar proof, and comparing it with the selection suggested by the trained learner. However, what if the machine learner is sometimes more clever than the human, and suggests a completely different (and perhaps better) selection of premises, leading to a different proof? In such a case, the standard machine learning evaluation (i.e., comparison of the two sets of premises) will say that the two sets of premises differ too much, and thus the machine learner has failed. This is considered acceptable for machine learning, as in general, there is no deeper concept of *truth* available, there are just training and testing data. However in our domain we do have a method how to show that the trained learner was right (and possibly smarter than the human): we can run an ATP system on its axiom selection. If a proof is found, it provides a much stronger measure of correctness. Obviously, this is only true if we know that the translation from Mizar to TPTP was correct, i.e., conducting such experiments really requires that we take extra care to ensure that no oversimplifications were made in this translation.

In the above mentioned experiment, 329 of the 2408 (i.e., 14%) proofs found by ATPs used less premises than the original MML proof, often suggesting a shorter proof. An example of such proof shortening is discussed in [Urb07b], showing that the newly found proof is really valid. Instead of arguing from the first principles (definitions) like in the human proof, the combined inductive-deductive system was smart enough to find a combination of previously proved lemmas (properties) that justify the conjecture more quickly.

[15] This is changing as we go: the new CASC-LTB category will hopefully spark interest in ATP systems dealing efficiently with large numbers of unnecessary axioms.

4.3 ATP-based explanation, presentation, and cross-verification of Mizar proofs

While proofs of whole Mizar theorems can be quite hard for ATP systems, re-proving the Mizar atomic justification steps (called *Simple Justifications* in Mizar) turns out to be quite easy for ATPs. The combination of E and SPASS usually solves more than 95% of such problems, and with smarter automated methods for axiom selection a 99.8% success rate (14 unsolved problems from 6765) was achieved in [US07]. This makes it practical to use ATPs for explanation and presentation of the (not always easily understandable) Mizar simple justifications, and to construct larger systems for independent ATP-based cross-verification of (possibly very long) Mizar proofs. In [US07] such a cross-verification system is presented, using the GDV [Sut06] system (which was extended to process Jaskowski-style natural deduction proofs that make frequent use of assumptions (suppositions)). MPTP was used to translate Mizar proofs to this format, and GDV together with the E, SPASS, and MaLARea systems was used to automatically verify the structural correctness of proofs, and 99.8% of the proof steps needed for the 252 Mizar problems selected for the MPTP Challenge (see below). This provides the first practical method for independent verification of Mizar, and opens the possibility of importing Mizar proofs into other proof assistants. A web presentation allowing interaction with ATP systems and GDV verification of Mizar proofs has been set up at `http://www.tptp.org/MizarTPTP` (described in [UTSP07]), and a static presentation using the MML Query system to translate the ATP proofs back to Mizar notation exists at `http://lipa.ms.mff.cuni.cz/~urban/xmlmml/html_bytst/` (described in [UB07]). A new online service integrating these functionalities is being built at `http://octopi.mizar.org/~mptp/MizAR.html`.

4.4 Use of MPTP for ATP challenges and competitions

The first MPTP problems were included into the TPTP library in 2006, and were already used for the 2006 CASC competition [Sut07]. In 2006, the MPTP $100 Challenges[16] were created and announced. This is a set of 252 related large-theory problems needed for one half (on of two implications) of the Mizar proof of the general topological Bolzano-Weierstrass theorem. Unlike the CASC competition, the challenge had an overall timelimit (252 * 5 minutes = 21 hours) for solving the problems, allowing complementary techniques like machine learning from previous solutions to be experimented with transparently in runtime. The challenge was won a year later by the leanCoP [OB03] system, having already revealed several interesting approaches to ATP in large theories: goal-directed calculi like connection tableaux (used in leanCoP), model-based axiom selection (used, e.g., in SRASS [SP07]), and machine learning of axiom relevance (used in MaLARea). The MPTP Challenge problems were again included into the TPTP and used for the CASC competition in 2007. In 2008, the CASC-LTB (Large Theory Batch) division appeared for the first time, with a similar setting to the MPTP Challenges, and additional large-theory problems from the Cyc and SUMO ontologies. A set of 245 relatively hard Mizar problems was added to the TPTP for this purpose, coming from the most advanced parts of the Mizar library. The problems come in four versions, containing different amounts of the previously available MML theorems and definitions as axioms. The largest versions thus contain over 50000 axioms. This has practically led to the inclusion of the MML into TPTP.

4.5 Development of larger AI metasystems like MaLARea on MPTP data

In Section 4.2 it is explained how the deeply defined notion of mathematical *truth* (implemented through ATPs) can improve the evaluation of learning systems working on large semantic knowledge bases like the translated MML, and how such systems can be used to find new mathematical proofs. This is, however, only one part of the AI fun made possible by such large libraries being available to ATPs.

[16] `http://www.tptp.org/MPTPChallenge/`

Another part is that newly found proofs can be recycled, and used again for learning in such domains. This closed loop between using deductive methods to find proofs, and using inductive methods to learn from existing proofs and suggest new proof directions, is the main idea behind the MaLARea [Urb07a, USPV08] meta-system. There are many kinds of information that such an autonomous meta-system can try to use and learn. The second version of MaLARea has just started to use structural and semantic features of formulas for their characterization, and for improving the axiom selection. Extracting lemmas from proofs and adding them to the set of available premises, creating new interesting conjectures and defining new useful notions (see [Len76, Col02, Faj88] for some previous interesting work on this), finding optimal strategies for problem classes, guiding the ATPs more closely than just by selecting axioms, inventing policies for efficient governing of the overall inductive-deductive loop: all these are interesting AI tasks that become relevant in this large-theory setting, and that seem to be highly relevant for the ultimate task of *doing mathematics* and perhaps even generally *thinking* automatically.

5 Future work

There is large amount of work to be done on practically all the projects mentioned above. The MPTP translation is by no means optimal (and especially proper encoding of arithmetics needs more experiments and work). Import of ATP proofs to Mizar practically does not exist (there is a basic translator taking Otter proof objects to Mizar, however this is very distant from the readable proofs in MML). MPTP has mostly been used for offline problem generation so far, and its use for online ATP services requires further work. With sufficiently strong ATP systems, the cross-verification of the whole MML could be attempted, and work on importing detailed ATP proofs into other proof assistants could be started. The work with second-order constructs is in some sense incomplete, and either a translation to (finitely axiomatized) NBG set theory, or usage of higher-order ATPs would be interesting from this point of view. More challenges and interesting presentation tools can be developed, for example an ATP-enhanced wiki for Mizar would be quite interesting. The heuristical and machine learning methods, and combined AI metasystems, have a very long way to go, some future directions are mentioned above. This is no longer only about mathematics: all kinds of more or less formal large knowledge bases are becoming available in other sciences, and automated reasoning could become one of the strongest methods for general reasoning in sciences when sufficient amount of formal knowledge exists. Strong ATP methods for formal mathematics could also provide useful semantic filtering for larger systems for automatic formalization of mathematical papers. This is a field that has been so far deemed to be rather science fiction than a real possibility.

References

[BR03] Grzegorz Bancerek and Piotr Rudnicki. Information retrieval in MML. In *MKM*, volume 2594 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2003.

[BU04] Grzegorz Bancerek and Josef Urban. Integrated semantic browsing of the Mizar Mathematical Library for authoring Mizar articles. In *MKM*, pages 44–57, 2004.

[CLR06] Thierry Coquand, Henri Lombardi, and Marie-Françoise Roy, editors. *Mathematics, Algorithms, Proofs, 9.-14. January 2005*, volume 05021 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[CMSM04] S. Colton, A. Meier, V. Sorge, and R. McCasland. Automatic Generation of Classification Theorems for Finite Algebras. In M. Rusinowitch and D. Basin, editors, *Proceedings of the 2nd International Joint Conference on Automated Reasoning*, number 3097 in Lecture Notes in Artificial Intelligence, pages 400–414, 2004.

[Col02] S. Colton. The HR Program for Theorem Generation. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 285–289. Springer-Verlag, 2002.

[DFGS99] J. Denzinger, M. Fuchs, C. Goller, and S. Schulz. Learning from Previous Proof Experience. Technical Report AR99-4, Institut für Informatik, Technische Universität München, 1999. (also to be published as a SEKI report).

[DW97] Ingo Dahn and Christoph Wernhard. First order proof problems extracted from an article in the MIZAR Mathematical Library. In Maria Paola Bonacina and Ulrich Furbach, editors, *Int. Workshop on First-Order Theorem Proving (FTP'97)*, RISC-Linz Report Series No. 97-50, pages 58–62. Johannes Kepler Universität, Linz (Austria), 1997.

[Faj88] Siemion Fajtlowicz. On conjectures of graffiti. *Discrete Mathematics*, 72(1-3):113–118, 1988.

[GS03] H. Ganzinger and J. Stuber. Superposition with Equivalence Reasoning. In F. Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction*, number 2741 in Lecture Notes in Artificial Intelligence, pages 335–349. Springer-Verlag, 2003.

[HKW96] R. Hähnle, M. Kerber, and C. Weidenbach. Common Syntax of the DFG-Schwerpunktprogramm Deduction. Technical Report TR 10/96, Fakultät für Informatik, Universät Karlsruhe, Karlsruhe, Germany, 1996.

[HSA06] John Harrison, Konrad Slind, and Rob Arthan. Hol. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*, pages 11–19. Springer, 2006.

[Jan91] Katarzyna Jankowska. Matrices. Abelian group of matrices. *Formalized Mathematics*, 2(4):475–480, 1991.

[Len76] D. Lenat. *An Artificial Intelligence Approach to Discovery in Mathematics*. PhD thesis, Stanford University, Stanford, USA, 1976.

[LR07] Gilbert Lee and Piotr Rudnicki. Alternative aggregates in Mizar. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Calculemus/MKM*, volume 4573 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2007.

[MP06] J. Meng and L. Paulson. Lightweight Relevance Filtering for Machine-Generated Resolution Problems. In G. Sutcliffe, R. Schmidt, and S. Schulz, editors, *Proceedings of the FLoC'06 Workshop on Empirically Successful Computerized Reasoning, 3rd International Joint Conference on Automated Reasoning*, volume 192 of *CEUR Workshop Proceedings*, pages 53–69, 2006.

[MP08] Jia Meng and Lawrence C. Paulson. Translating higher-order clauses to first-order clauses. *J. Autom. Reasoning*, 40(1):35–60, 2008.

[NP01] I. Niles and A. Pease. Towards A Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, pages 2–9, 2001.

[NW01] A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 335–367. Elsevier Science, 2001.

[OB03] J. Otten and W. Bibel. leanCoP: Lean Connection-Based Theorem Proving. *Journal of Symbolic Computation*, 36(1-2):139–161, 2003.

[PS07] 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*, 2007.

[RPG05] D. Ramachandran, Reagan P., and K. Goolsbey. First-orderized ResearchCyc: Expressiveness and Efficiency in a Common Sense Knowledge Base. In Shvaiko P., editor, *Proceedings of the Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2005.

[RT99] Piotr Rudnicki and Andrzej Trybulec. On equivalents of well-foundedness. *J. Autom. Reasoning*, 23(3-4):197–234, 1999.

[RT03] Piotr Rudnicki and Andrzej Trybulec. On the integrity of a repository of formalized mathematics. In *MKM*, volume 2594 of *Lecture Notes in Computer Science*, pages 162–174. Springer, 2003.

[Rud87] P. Rudnicki. Obvious Inferences. *Journal of Automated Reasoning*, 3(4):383–393, 1987.

[Rud92] P. Rudnicki. An Overview of the Mizar Project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, pages 311–332, 1992.

[RV02] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*,

15(2-3):91–110, 2002.

[Sch02] S. Schulz. E: A Brainiac Theorem Prover. *AI Communications*, 15(2-3):111–126, 2002.

[SP07] G. Sutcliffe and Y. Puzis. SRASS - a Semantic Relevance Axiom Selection System. In F. Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction*, number 4603 in Lecture Notes in Artificial Intelligence, pages 295–310. Springer-Verlag, 2007.

[SS98] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[Sut06] G. Sutcliffe. Semantic Derivation Verification. *International Journal on Artificial Intelligence Tools*, 15(6):1053–1070, 2006.

[Sut07] G. Sutcliffe. The 3rd IJCAR Automated Theorem Proving Competition. *AI Communications*, 20(2):117–126, 2007.

[UB07] Josef Urban and Grzegorz Bancerek. Presenting and explaining Mizar. *Electr. Notes Theor. Comput. Sci.*, 174(2):63–74, 2007.

[Urb03] J. Urban. Translating Mizar for First Order Theorem Provers. In A. Asperti, B. Buchberger, and J.H. Davenport, editors, *Proceedings of the 2nd International Conference on Mathematical Knowledge Management*, number 2594 in Lecture Notes in Computer Science, pages 203–215. Springer-Verlag, 2003.

[Urb04] J. Urban. MPTP - Motivation, Implementation, First Experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

[Urb05] J. Urban. XML-izing Mizar: Making Semantic Processing and Presentaion of MML Easy. In M. Kohlhase, editor, *Proceedings of the 4th Integrated Conference on Mathematical Knowledge Management*, volume 3863 of *Lecture Notes in Computer Science*, pages 346–360, 2005.

[Urb06a] J. Urban. MizarMode - An Integrated Proof Assistance Tool for the Mizar Way of Formalizing Mathematics. *Journal of Applied Logic*, 4(4):414–427, 2006.

[Urb06b] Josef Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. *International Journal on Artificial Intelligence Tools*, 15(1):109–130, 2006.

[Urb07a] J. Urban. MaLARea: a Metasystem for Automated Reasoning in Large Theories. In J. Urban, G. Sutcliffe, and S. Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories*, pages 45–58, 2007.

[Urb07b] J. Urban. MPTP 0.2: Design, Implementation, and Initial Experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2007.

[US07] J. Urban and G. Sutcliffe. ATP Cross-verification of the Mizar MPTP Challenge Problems. In N. Dershowitz and A. Voronkov, editors, *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 4790 in Lecture Notes in Artificial Intelligence, pages 546–560, 2007.

[USPV08] 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, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456, 2008.

[UTSP07] J. Urban, S. Trac, G. Sutcliffe, and Y. Puzis. Combining Mizar and TPTP Semantic Presentation Tools. In *Proceedings of the Mathematical User-Interfaces Workshop 2007*, 2007.

[VLP07] K. Verchinine, A. Lyaletski, and A. Paskevick. System for Automated Deduction (SAD): A Tool for Proof Verification. In F. Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction*, number 4603 in Lecture Notes in Artificial Intelligence, pages 398–403. Springer-Verlag, 2007.

[WBH+02] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobald, and D. Topic. SPASS Version 2.0. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 275–279. Springer-Verlag, 2002.