System Description: XSL-Based Translator of Mizar to LaTeX

Grzegorz Bancerek[2], Adam Naumowicz[1](✉), and Josef Urban[2]

[1] University of Bialystok, Bialystok, Poland
adamn@math.uwb.edu.pl
[2] Czech Technical University in Prague, Prague, Czech Republic

Abstract. We describe a new version of the Mizar-to-LaTeX translator. The system has been re-implemented as XSL stylesheets instead of as Pascal programs, allowing greater flexibility. It can now be used to generate both LaTeX/PDF and HTML with MathJax code. We also experimentally support generation of full proofs. Finally, the system is now available online and through the Mizar Emacs interface.

1 Introduction

The Mizar project [4] has since its inception in 1973 focused on formal representation of mathematics that would be as close possible to its representation in natural language. After the Mizar language emerged and a larger number of articles have been collected in the Mizar Mathematical Library (MML) [3,7], the Mizar team started to experiment with translating the Mizar articles into LaTeX. This translation has been evolving for the last three decades [1,5,10].

Here we describe the most recent large re-implementation of the system which switches from custom Pascal programs to more flexible XSL processing, done primarily by the first author over the last eight years. This in turn builds on the XML-ization of the Mizar processing, done over the past decade [9,11]. The system can now produce both LaTeX/PDF and HTML using MathJax for displaying formulas and terms. We experimentally support also translation of full proofs. The system is used for in-house production of the journal Formalized Mathematics[1] [8] and it is newly also available as a remote PDF/HTML service for Mizar authors via the Emacs authoring environment for Mizar [12], similarly to the MizAR [13] ATP service for Mizar.

G. Bancerek—Deceased.

A. Naumowicz—Partially supported by the project N° 2017/01/X/ST6/00012 financed by the National Science Center, Poland and the COST Action EUTypes CA15123. The Mizar processing has been performed using the infrastructure of the University of Bialystok High Performance Computing Center.

J. Urban—Supported by the ERC Consolidator grant no. 649043 AI4REASON, and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15 003/0000466 and the European Regional Development Fund.

[1] https://www.degruyter.com/view/j/forma.

F. Rabe et al. (Eds.): CICM 2018, LNAI 11006, pp. 1–6, 2018.
https://doi.org/10.1007/978-3-319-96812-4_1

2 Summary of the Old Mizar-to-LaTeX Translation

The previous incarnation of the translator, as well as its history, are described in detail in [1]. The complete process was carried out by a set of eight custom programs (`fmparse`, `newfmfrm`, `addfmfrm`, `fmfrm`, `resvar`, `fmnotats`, `fmanalyz` and `jformath`) run in succession after preparing the article with the standard Mizar accommodator utility and before producing the final output with LaTeX using BibTeX metadata provided by the user/editor.

Let us only briefly recall that the translation of the non-formula text parts was done as a static mapping of the keywords. For example the `commutativity` property for functors was expressed by the sentence *Let us observe that the functor is commutative.* The translation of atomic formulas, terms, and types was done according to a database of LaTeX *translation patterns* for the Mizar symbols with arities (a *format* in the Mizar terminology). Each pattern consists of a control header (deciding e.g. about using the math mode in LaTeX, bracketing, etc.) and one or more proper patterns that map the format with arguments to LaTeX in various contexts. Following is an example of such a pattern for a typical `PartUnion` functor:[2]

```
O1 0 2
OPartUnion
mol@s#1#2; \bigcup_{\beta{<_{#2}}#1}\beta
```

This means that the Mizar term `PartUnion(B,R)` would be displayed as $\bigcup_{\beta<_R B}\beta$. Note that in this case the translation is quite nontrivial and it reveals information about how the symbol `PartUnion` is defined in Mizar.

All top-level elements of an article, i.e. theorems, definitions, schemes, reservations and global shortcuts were presented in the rendering, but proofs were not translated as a rule. Single letter variables occurring in an article were preserved while others were abbreviated into single letters with indices.

3 Description of the New Technology

The new technology of automated translation currently used for Mizar texts published in the *Formalized Mathematics* journal is based on XSL translation templates applied to the XML representation of the weakly-strict Mizar [9] encoding of the original Mizar input file (`*.wsx` file). However, the semantic representation generated by the Mizar verifier (`*.xml` file) is also used to decode links to external articles. All bibliographic metadata are first translated to special XML format and merged with information extracted from the Mizar article. Global (for the journal) LaTeX translation patterns are also kept in the XML `pub.xml` file. In the following sections we describe the basic functions of the main XSL stylesheets. They have been designed to perform well-defined simple iterative tasks within the process of generating the final LaTeX rendering for

[2] http://mizar.uwb.edu.pl/version/current/html/pcomps_2.html#K1, see also [6].

PDF and MathJax-enabled HTML presentation. Let us note that the presented XSL translation is not a 1-1 reimplementation of former Pascal-based code. The current multi-pass method has been implemented from scratch to make use of available XML representation formats, whereas the former was bound to the internal structures of the Mizar verifier.

3.1 addformat.xsl

This is the main stylesheet responsible for selecting information to be translated from the weakly-strict Mizar representation. Identified symbols are matched with their format specification to be later replaced by concrete translations.

3.2 addtranslation.xsl

This script augments the processed file with available translation patterns. An example of a concrete pattern (for the previously mentioned PartUnion functor) as extracted from the `pub.xml` file looks as follows:

```
<Translation
  voc="PCOMPS_2" kind="O" symbolnr="1" symbol="OPartUnion" argsnr="2"
  leftargsnr="0" rightargsnr="4" format="O1 0 2" header="mol@s#1#2;"
  priority="8" forcing="rqw" context1="l" context2="" TeX-mode="m">
  <pattern>
    \bigcup_{\beta{&lt;_{<X pos="lower" locus="2"/>}}<X locus="1"/>}\beta
  </pattern>
</Translation>
```

The system also proposes formats for new definitions introduced in the current article. The `unknown_patterns.xsl` stylesheet handles unknown patterns.

3.3 varrepr.xsl

The task of this stylesheet is to replace any identifiers that contain names of Greek letters into corresponding LaTeX symbols. Longer variable identifiers are given standardized representations with subscripts (e.g. `AA` becomes A_1).

3.4 multipred.xsl

This procedure is technically split into several passes of `multipred.xsl`, `multipred2.xsl` and `multipred3.xsl` run in succession together with `prune.xsl`. The goal is to locate in the input text a list of constructs with a qualifying format that can be printed together in a shortened form. For example, in the PCOMPS_2 article, instead of the literal translation: "G_9 is cover of P_6, and G_9 is finer than F_9" (cf. [6]) the script produces a shortened phrase "G_9 is cover of P_6 and finer than F_9".

3.5 transitive.xsl

This stylesheet improves the quality of the translation by generating output of the form: "$x < y < z$" instead of "$x < y$ and $y < z$", i.e. it joins consecutive predicates with shared arguments as is usually done in informal mathematics.

3.6 compress.xsl

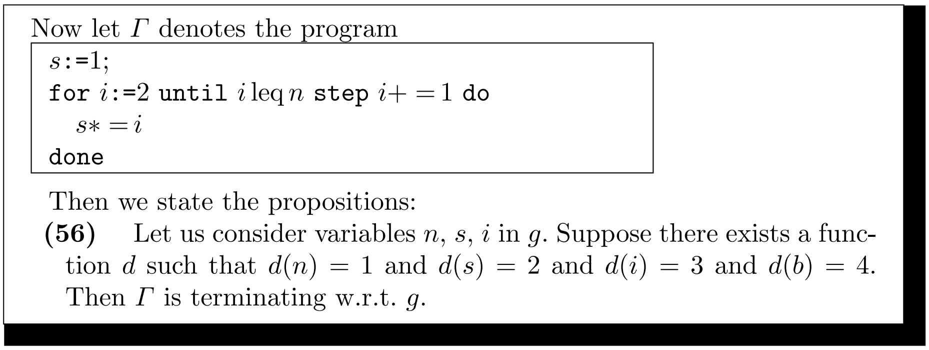
There are several independent stylesheets that compress and thus make more natural for the reader the occurrences of particular constructs in a common context. These are: `compress_let.xsl`, `compress_for.xsl`, `compress_assume.xsl`, `brackets.xsl`, `compress_func.xsl`, and `compressres.xsl` for generalizations, quantifiers, assumptions, brackets, functor definitions with common arguments, and variable reservations, respectively.

3.7 recognize-programs.xsl

The set of templates `recognize-programs.xsl`, `recognize-programs2.xsl` and `recognize-programs3.xsl` is used to generate custom encoding of specific complex terms - representation of programs. For example, the LaTeX translation of the following Mizar theorem statement from the AOFA_I00 article about an algorithm defined in terms of a custom *if-while* algebra:

```
theorem
  for n,s,i being Variable of g st ex d being Function st d.n = 1 & d.s
  = 2 & d.i = 3 & d.b = 4 holds s:=1\;for-do(i:=2, i leq n, i+=1, s*=i)
  is_terminating_wrt g
```

is rendered in PDF as follows (cf. [2]):


Now let Γ denotes the program

```
s:=1;
for i:=2 until i leq n step i+ = 1 do
   s* = i
done
```

Then we state the propositions:

(56) Let us consider variables n, s, i in g. Suppose there exists a function d such that $d(n) = 1$ and $d(s) = 2$ and $d(i) = 3$ and $d(b) = 4$. Then Γ is terminating w.r.t. g.

3.8 article2latex.xsl and article2html.xsl

Depending on the output format, based on the preparatory tasks performed by the previously mentioned stylesheets, one may choose to generate either LaTeX code for a self-contained PDF article, or code embedded in an HTML document to be rendered by MathJax. If LaTeX code is chosen, we finally run `pdflatex` using several Mizar-specific headers.

3.9 Remote Service and Processing Times

Since the translation toolchain has many components and relies on a number of custom tools and their versions installed, we make it available as an online service. This is similar to the solution taken for the MizAR [13] ATP service for Mizar. The Mizar users can now send their articles from Emacs to the service for PDF and HTML translation of their current work independently of the publication process of *Formalized Mathematics*. This is done by selecting the *Mizar→Remote solving→Produce PDF online* and *Mizar→Remote solving→Produce MathJax online* menu options, respectively.[3] The remote processing of a basic Mizar article such as `XBOOLE_1` takes about 15s and the whole MML can be processed locally on the server overnight. This is quite comparable to the speed of the earlier Pascal-written version of the toolchain.

4 Conclusion and Future Work

The described re-implementation of the Mizar to LaTeX translation system is based on the flexible and easily extensible XML/XSL technology rather than custom Pascal programs tied to the specific implementation of the core Mizar system. Thanks to this approach, a set of shared scripts and stylesheets can be used for the production of the printed editions of the journal *Formalized Mathematics* but also as an on-demand remote PDF/HTML service for Mizar authors. Users of the Emacs authoring environment for Mizar can now experiment with the PDF and HTML presentation of their current work based on the shared LaTeX translation.

Future work on translation will focus on adding more natural language based linguistic features to the generated mathematical text. In particular, the human-friendly presentation of the structure of (nested with varying levels of importance) proofs will be investigated. It is now much easier to experiment with more natural translation patterns as a step by step improvement of the translation by just modifying the `addformat.xsl` stylesheet which controls filtering relevant information from the original Mizar article and either the `article2latex.xsl` or `article2html.xsl` templates that generate the final specific rendering in LaTeX or MathJax/HTML format.

References

1. Bancerek, G.: Automatic translation in Formalized Mathematics. Mech. Math. Appl. **5**(2), 19–31 (2006)
2. Bancerek, G.: Mizar analysis of algorithms: algorithms over integers. Formaliz. Math. **16**(2), 177–194 (2008). https://doi.org/10.2478/v10037-008-0024-0
3. Bancerek, G., Bylinski, C., Grabowski, A., Kornilowicz, A., Matuszewski, R., Naumowicz, A., Pak, K.: The role of the Mizar Mathematical Library for interactive proof development in Mizar. J. Autom. Reason. **61**(1–4), 9–32 (2018). https://doi.org/10.1007/s10817-017-9440-6
4. Bancerek, G., et al.: Mizar: state-of-the-art and beyond. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V. (eds.) CICM 2015. LNCS (LNAI), vol. 9150, pp. 261–279. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20615-8_17

[3] Both menu items trigger the `mizar-tex-remote` function in the current `mizar.el` Emacs Lisp mode for Mizar available at https://github.com/JUrban/mizarmode/blob/master/mizar.el.

5. Bancerek, G., Carlson, P.: Mizar and the machine translation of mathematics documents. http://www.mizar.org/project/banc_carl93.ps
6. Borys, L.: On paracompactness of metrizable spaces. Formaliz. Math. **3**(1), 81–84 (1992). http://fm.mizar.org/1992-3/pdf3-1/pcomps_2.pdf
7. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Four decades of Mizar - foreword. J. Autom. Reason. **55**(3), 191–198 (2015). https://doi.org/10.1007/s10817-015-9345-1
8. Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) Calculemus/MKM - 2007. LNCS (LNAI), vol. 4573, pp. 235–249. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73086-6_20
9. Naumowicz, A., Piliszek, R.: Accessing the Mizar library with a weakly strict Mizar parser. In: Kohlhase, M., Johansson, M., Miller, B., de Moura, L., Tompa, F. (eds.) CICM 2016. LNCS (LNAI), vol. 9791, pp. 77–82. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42547-4_6
10. Trybulec, A., Rudnicki, P.: A collection of TEXed Mizar abstracts. http://www.mizar.org/project/TR-89-18.pdf
11. Urban, J.: XML-izing Mizar: making semantic processing and presentation of MML easy. In: Kohlhase, M. (ed.) MKM 2005. LNCS (LNAI), vol. 3863, pp. 346–360. Springer, Heidelberg (2006). https://doi.org/10.1007/11618027_23
12. Urban, J.: MizarMode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. J. Appl. Log. **4**(4), 414–427 (2006)
13. Urban, J., Rudnicki, P., Sutcliffe, G.: ATP and presentation service for Mizar formalizations. J. Autom. Reason. **50**, 229–241 (2013). https://doi.org/10.1007/s10817-012-9269-y