Mining for Formalization Environments in the Mizar Mathematical Library

Adam Naumowicz

Institute of Informatics
University of Białystok, Poland
adamn@mizar.org

ITP 2017, Brasilia, September 28, 2017
What is Mizar and the Mizar Mathematical Library?

- Mizar is a system for encoding and proof-checking mathematics invented by Andrzej Trybulec (†2013) and developed since 1970s.
- Its language tries to mimic standard mathematical practice.
- Its verification engine is designed to preserve human understanding of proof steps.
- It is being used to build a centralized library of formalized mathematical knowledge based on simple axioms (of set theory) and special focus on reusability - Mizar Mathematical Library (MML).
- Current MML makes use of 8863 symbols (1933 attributes, 4825 functors, 37 left brackets, 37 right brackets, 936 modes, 752 predicates, 175 selectors, and 168 structures).
- There are plenty of symbols which are used in multiple contexts, e.g. the popular asterisk symbol * is used to denote almost two hundred formally different operations.
The level of reusability in other large formalization libraries is currently significantly smaller than in the Mizar library.

Isabelle-based Archive of Formal Proofs (as presented at CICM2015):

- 106 out of its 215 articles were isolated nodes of the imports graph, i.e. they were not related to other articles in the library.
- Among the articles that had some non-trivial dependence, the maximal number of reused articles in one article was equal to 4, and the maximal number of articles directly depending on some other article reached 9.

MML ver. 5.41.1289:

- Only one isolated node (for technical reasons preserved in the library, (article SCHEMS_1), There is an article which imports data from 121 other articles (JORDAN)
- The elementary properties of subsets from SUBSET are imported in as many as 1250 articles.
### Average Number of Directives Per Article in Current MML

<table>
<thead>
<tr>
<th>Directive</th>
<th>Avg. number of article names</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructors</td>
<td>12.53760</td>
</tr>
<tr>
<td>definitions</td>
<td>3.41117</td>
</tr>
<tr>
<td>equalities</td>
<td>3.84251</td>
</tr>
<tr>
<td>expansions</td>
<td>2.77502</td>
</tr>
<tr>
<td>notations</td>
<td>26.87900</td>
</tr>
<tr>
<td>registrations</td>
<td>15.49810</td>
</tr>
<tr>
<td>requirements</td>
<td>4.05275</td>
</tr>
<tr>
<td>schemes</td>
<td>2.56943</td>
</tr>
<tr>
<td>theorems</td>
<td>24.95810</td>
</tr>
<tr>
<td>vocabularies</td>
<td>29.31650</td>
</tr>
</tbody>
</table>
Less Fine-grained ("standardized") import Directive

The original environment declaration of an example article in the current MML looks like this:

```plaintext
environ

vocabulary XBOOLE_0, SUBSET_1, TARSKI, ORDERS_2, WAYBEL_0, XXREAL_0,
    ZFMISC_1, RELAT_1, MCART_1, LATTICE3, RELAT_2, LATTICES, YELLOW_0,
    EQREL_1, REWRITE1, ORDINAL2, FUNCT_1, STRUCT_0, YELLOW_3;
notations TARSKI, XBOOLE_0, ZFMISC_1, XTUPLE_0, SUBSET_1, RELAT_1, RELAT_2,
    RELSET_1, MCART_1, DOMAIN_1, FUNCT_2, BINOP_1, STRUCT_0, ORDERS_2,
    LATTICES, YELLOW_0, WAYBEL_0;
constructors DOMAIN_1, LATTICE3, ORDERS_3, WAYBEL_0, RELSET_1, XTUPLE_0;
registrations XBOOLE_0, SUBSET_1, RELSET_1, STRUCT_0, LATTICE3, YELLOW_0,
    ORDERS_2, WAYBEL_0, RELAT_1, XTUPLE_0;
requirements SUBSET, BOOLE;
definitions LATTICE3, RELAT_2, TARSKI, WAYBEL_0, ORDERS_2;
expansions LATTICE3, RELAT_2, WAYBEL_0, ORDERS_2;
theorems FUNCT_1, FUNCT_2, FUNCT_5, LATTICE3, MCART_1, ORDERS_2, RELAT_1,
    RELAT_2, RELSET_1, TARSKI, WAYBEL_0, YELLOW_0, YELLOW_2, ZFMISC_1,
    XBOOLE_0, BINOP_1, XTUPLE_0;
schemes FUNCT_7, RELAT_1;
```

The "standardized" version looks as follows:

```plaintext
environ

vocabulary XBOOLE_0, SUBSET_1, TARSKI, ORDERS_2, WAYBEL_0, XXREAL_0,
    ZFMISC_1, RELAT_1, MCART_1, LATTICE3, RELAT_2, LATTICES, YELLOW_0,
    EQREL_1, REWRITE1, ORDINAL2, FUNCT_1, STRUCT_0, YELLOW_3;
requirements SUBSET, BOOLE;
imports RELAT_1, TARSKI, XBOOLE_0, XTUPLE_0, ZFMISC_1, SUBSET_1, FUNCT_1,
    RELAT_2, RELSET_1, MCART_1, FUNCT_2, BINOP_1, DOMAIN_1, FUNCT_5, FUNCT_7,
    STRUCT_0, LATTICE3, YELLOW_0, ORDERS_2, ORDERS_3, WAYBEL_0, YELLOW_2;
```
Formalization Environments vs. Importing Directives

- It is worthwhile to consider developing even more high-level environment importing directives - ready-made formalization environments built upon the current dependence structure of MML articles.

- A similar approach has previously been employed to semi-automatically generate the, so called, “encyclopedic” Mizar articles (XBOOLE*, XREAL*, and XCMPLX* series) which contain all basic definitions and theorems related to boolean operations on sets, real and complex numbers, respectively, originally scattered all around the MML articles.

- Since the symbols used in the MML are derived from standard English mathematical terminology or use common mathematical notions familiar from their \LaTeX representations, the symbols are naturally a good starting point for building a formalization environment.
Methodology and Tools for MML Mining and Environment Building

- A vocabulary file for a given symbol can be identified with a Mizar findvoc utility or MML symbol searching (http://webmizar.cs.shinshu-u.ac.jp/mmlfe/current/).

- To find (and then import) a definition that makes use of that symbol one can use the MML Query system (http://mmlquery.mizar.org/mmlquery/three.html) with a query like this:

  symbol + notation | constructor

  to select all constructors denoted with the + symbol.
We can restrict the query to a specified list of articles (which itself can be a result of another query):

\[
\text{symbol} + \text{notation} \mid \text{constructor} \mid \text{NAT}_1:* \\
\]

(in this case the constructors defined in one article - \text{NAT}_1).

Selecting e.g. specific theorems (and so the containing articles for the environment) may look like this:

\[
\text{(NAT}_1: \text{func 2 occur}) \text{ and (NAT}_1: \text{func 1 occur}) \mid \text{th} \\
\]

where we search for all theorems with occurrences of the + and * operations defined for natural numbers.
Ultimate Goal: Integrating Mizar Authoring Software

Emacs-mode for Mizar

MML symbol searching

MML Query

HTML-ized Mizar browsing

A. Naumowicz

Mining for Formalization Environments in the Mizar Mathematical Library