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.

Formalization Reusability

- 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

Directive	Avg. number of article names
constructors	12.53760
definitions	3.41117
equalities	3.84251
expansions	2.77502
notations	26.87900
registrations	15.49810
requirements	4.05275
schemes	2.56943
theorems	24.95810
vocabularies	29.31650

Less Fine-grained (“standardized”) `import` Directive

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

```
environ

 vocabularies 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,
      LATTICE3, 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:

```
environ
 vocabularies 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 LATEXrepresentations, 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.

Methodology and Tools for MML Mining and Environment Building

- We can restrict the query to a specified list of articles (which itself can be a result of another query):

  `symbol + notation | constructor | NAT_1:*`

  (in this case the constructors defined in one article - `NAT_1`).

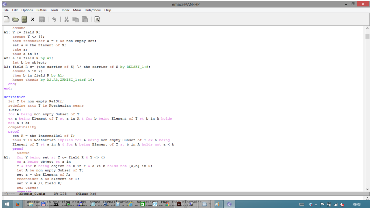
- Selecting e.g. specific theorems (and so the containing articles for the environment) may look like this:

  `(NAT_1:func 2 occur) and (NAT_1:func 1 occur) | th`

  where we search for all theorems with occurrences of the + and * operations defined for natural numbers.

Ultimate Goal: Integrating Mizar Authoring Software

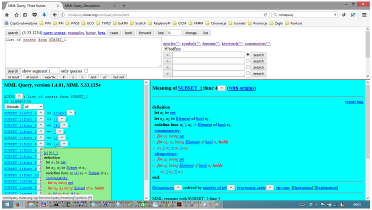
Emacs-mode for Mizar



```
    assume
A1: Y c= field R;
    assume Y <> {};
    then reconsider X = Y as non empty set;
    set a = the Element of X;
    take a;
    thus a in Y;
A2: a in field R by A1;
    let b be object;
A3: field R c= (the carrier of S) \/ the carrier of S by RELSET_1:8;
    assume b in Y;
    then b in field R by A1;
    hence thesis by A2,A3,ZFMISC_1:def 10;
  end;
end;

definition
  let T be non empty RelStr;
  redefine attr T is Noetherian means
  :Def2:
  for A being non empty Subset of T
  ex a being Element of T st a in A & for b being Element of T st b in A holds
  not a < b;
  compatibility
  proof
    set R = the InternalRel of T;
    thus T is Noetherian implies for A being non empty Subset of T ex a being
    Element of T st a in A & for b being Element of T st b in A holds not a < b
    proof
      assume
A1:   for Y being set st Y c= field R & Y <> {}
      ex a being object st a in
      Y & for b being object st b in Y & a <> b holds not [a,b] in R;
      let A be non empty Subset of T;
      set a = the Element of A;
      reconsider a as Element of T;
      set Y = A /\ field R;
      per cases;
```

MML Query



(5.33.1254) query syntax, examples, home, beta,

```
list of constr from SUBSET_1
```

articles^^, symbols^^, formats^^, keywords^^, constructors^^,

MML Query, version 1.4.01, MML 5.33.1254

```
QUERY ^ : list of constr from SUBSET_1
16 element(s)
```

Decode all

```
SUBSET_1:attr 1 ^ => proper ^;
SUBSET_1:func 1 ^ => {} ^;
SUBSET_1:func 2 ^ => [#] ^;
SUBSET_1:func 3 ^ => ` ^;
SUBSET_1:func 4 ^ => \/ ^;
SUBSET_1:func 5 ^ => \+\ ^;
SUBSET_1:func 6
SUBSET_1:func 7
SUBSET_1:func 8
SUBSET_1:func 9
SUBSET_1:func 10
SUBSET_1:mode 1
SUBSET_1:mode 2
SUBSET_1:pred 1
```

[x] [*] [_]

definition
let a_1 be set;
let a_2, a_3 be Subset of a_1;
redefine func a_2 \+\ a_3 -> Subset of a_1;
commutativity;
:: *for a_1 being set*
:: *for a_2, a_3 being Subset of a_1 holds*
:: *a_2 \+\ a_3 = a_3 \+\ a_2;*

mmlquery.mizar.org/cgi-bin/mmlquery/meaning?symbol=[#]

Meaning of SUBSET_1:func 4 ^ (with origins)

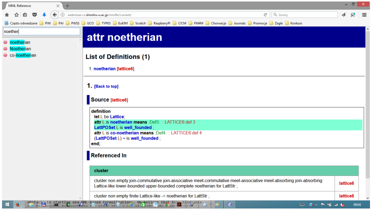
report bug

definition
let a_1 **be** set;
let a_2, a_3 **be** Element of bool a_1;
redefine func a_2 \/ a_3 -> Element of bool a_1;
commutativity;
:: *for a_1 being set*
:: *for a_2, a_3 being Element of bool a_1 holds*
:: *a_2 \/ a_3 = a_3 \/ a_2;*
idempotence;
:: *for a_1 being set*
:: *for a_2 being Element of bool a_1 holds*
:: *a_2 \/ a_2 = a_2;*
end;

Occurrences ^ ordered by number of ref ^, processing order ^, environ, [Operations] [Explanation]

MML versions with SUBSET_1:func 4

MML symbol searching



noether

- noetherian
- Noetherian
- co-noetherian

attr noetherian

List of Definitions (1)

1. noetherian [lattice6]

1. [Back to top]

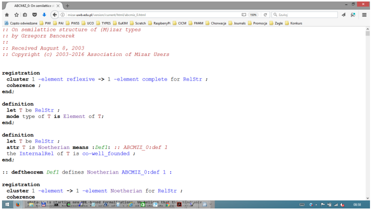
Source [lattice6]

```
definition
  let L be Lattice;
  attr L is noetherian means :Def3: :: LATTICE6:def 3
  LattPOSet L is well_founded ;
  attr L is co-noetherian means :Def4: :: LATTICE6:def 4
  (LattPOSet L) ~ is well_founded ;
end;
```

Referenced In

cluster	
cluster non empty join-commutative join-associative meet-commutative meet-associative meet-absorbing join-absorbing Lattice-like lower-bounded upper-bounded complete noetherian for LattStr ;	lattice6
cluster non empty finite Lattice-like -> noetherian for LattStr ;	lattice6

HTML-ized Mizar browsing


ABCMIZ_0: On semilattice st
mizar.uwb.edu.pl/version/current/html/abcmiz_0.html
150%
Szukaj
Często odwiedzane PWI PAI PWSS UCO TYPES EuKIM Scratch RaspberryPi CICM FM4M Chorwacja Journals Promocja Zagle Konkurs
08:58

```
:: On semilattice structure of {M}izar types
:: by Grzegorz Bancerek
::
:: Received August 8, 2003
:: Copyright (c) 2003-2016 Association of Mizar Users


registration
 cluster 1 -element reflexive -> 1 -element complete for RelStr ;
 coherence ;
end;

definition
 let T be RelStr ;
 mode type of T is Element of T;
end;

definition
 let T be RelStr ;
 attr T is Noetherian means :Def1: :: ABCMIZ_0:def 1
 the InternalRel of T is co-well_founded ;
end;

:: deftheorem Def1 defines Noetherian ABCMIZ_0:def 1 :

registration
 cluster 1 -element -> 1 -element Noetherian for RelStr ;
 coherence
```