XML-based representation of Mizar Mathematical Library

Artur Korniłowicz
arturk@math.uwb.edu.pl

Joint work with Czesław Byliński and Adam Naumowicz

Institute of Computer Science
University of Bialystok

EuroProofNet Workshop on the development, maintenance, refactoring and search of large libraries of proofs
Tbilisi, Georgia, September 23–24, 2022 [online]

Plan

- About Mizar language
- Internal languages: WSX, MSX, ESX
- Mizar Mathematical Library
- Translation to other formats

Abstract

The Mizar Mathematical Library is a collection of mathematical papers written in the Mizar language and fully computer-verified by the Mizar proof checker. From many years the Mizar Mathematical Library is a subject of translation into other systems for its cross-verification or representation of its content in various formats. To facilitate this task, several internal XML-based formats have been designed to be processable by generic tools independent from the Mizar system. These formats represent various syntactic-semantic information accessible at different stages of proof verification by the Mizar checker.
In this talk, we present constructions of the Mizar language and describe how they are represented in the XML formats. We also present the content of the Mizar Mathematical Library with a focus on its initial articles and discuss the management model of the library.

Definedable items

- Predicates
- Types
- Adjectives
- Operations
- Structures (several notions are defined internally)

- Synonyms and antonyms
- Redefinitions
- Registrations, reductions, identifications, properties

Representation of items

All defined mathematical notions are represented using:

- Formats – symbols and place of arguments (overloading)
- Patterns – notions
- Constructors – types of arguments, types of result types and meaning

Example

```
definition
  let A,B be object;
  func operation1(A,B) -> object means ...
end;

notation
  let A,B be object;
  synonym A synonym1 B for operation1(A,B);
end;

definition
  let A,B be set;
  redefine func A synonym1 B -> set;
  ...
end;
```

XML elements and attributes – definition

```
<Functor-Definition MMLId="EXAMPLE:1">
 <Redefine occurs="false"/>
  <InfixFunctor-Pattern formatdes="O1[O(2)2]" spelling="operation1"
     patternnr="10" absolutepatternMMLId="EXAMPLE:1"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origconstrnr="0">
  <Loci/>
  <Loci>
   <Locus spelling="A" kind="Constant" serialnr="1" varnr="1"/>
   <Locus spelling="B" kind="Constant" serialnr="2" varnr="2"/>
  </Loci>
 </InfixFunctor-Pattern>
 <Type-Specification>
  <Standard-Type formatnr="2" patternnr="1" absolutepatternMMLId="HIDDE
   <Arguments/>
<Definiens kind="Simple-Definiens" shape="Formula-Expression">
```

XML elements and attributes – synonym

```
<Func-Synonym>
 <InfixFunctor-Pattern formatdes="O2[1(2)1]" spelling="synonym1"
     patternnr="11" absolutepatternMMLId="EXAMPLE:2"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origpatternnr="10" absoluteorigpatternMMLId="EXAMPLE:1"
     origconstrnr="0">
  <Loci><Locus spelling="A" kind="Constant" serialnr="3" varnr="1"/></L
  <Loci><Locus spelling="B" kind="Constant" serialnr="4" varnr="2"/></L
 </InfixFunctor-Pattern>
 <Pattern-Shaped-Expression>
 <InfixFunctor-Pattern formatdes="O1[0(2)2]" spelling="operation1"
     patternnr="10" absolutepatternMMLId="EXAMPLE:1"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origconstrnr="0">
  <Loci/>
  <Loci>
   <Locus spelling="A" origin="Constant" kind="Constant" serialnr="3" va
   <Locus spelling="B" origin="Constant" kind="Constant" serialnr="4" va
  </Loci>
 </InfixFunctor-Pattern>
 </Pattern-Shaped-Expression>
 </Func-Synonym>
```

XML elements and attributes – comparison

Definition

```
<Functor-Definition MMLId="EXAMPLE:1">
  <InfixFunctor-Pattern formatdes="01[0(2)2]" spelling="operation1"
     patternnr="10" absolutepatternMMLId="EXAMPLE:1"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origconstrnr="0">
```

Synonym

```
<Func-Synonym>
 <InfixFunctor-Pattern formatdes="02[1(2)1]" spelling="synonym1"
     patternnr="11" absolutepatternMMLId="EXAMPLE:2"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origpatternnr="10" absoluteorigpatternMMLId="EXAMPLE:1"
     origconstrnr="0">
 <InfixFunctor-Pattern formatdes="01[0(2)2]" spelling="operation1"
     patternnr="10" absolutepatternMMLId="EXAMPLE:1"
     constr="K10" absoluteconstrMMLId="EXAMPLE:1"
     origconstrnr="0">
```

XML elements and attributes – redefinition

```
<Functor-Definition>
 <Redefine occurs="true"/>
 <InfixFunctor-Pattern formatdes="O2[1(2)1]" spelling="synonym1"
      patternnr="12" absolutepatternMMLId="EXAMPLE:3"
      constr="K11" absoluteconstrMMLId="EXAMPLE:2"
      origpatternnr="11" absoluteorigpatternMMLId="EXAMPLE:2"
      origconstrnr="10" absoluteorigconstrMMLId="EXAMPLE:1">
  <Loci>
   <Locus spelling="A" kind="Constant" serialnr="5" varnr="1"/>
  </Loci>
  <Loci>
   <Locus spelling="B" kind="Constant" serialnr="6" varnr="2"/>
  </Loci>
 </InfixFunctor-Pattern>
 <Type-Specification>
```

Structures – syntax

```
Structure-Definition = "struct" [ "(" Ancestors ")" ]
  Structure-Symbol [ "over" Loci ] "(#" Fields "#)" ";"

Ancestors =
  Structure-Type-Expression { "," Structure-Type-Expression }

Fields = Field-Segment { "," Field-Segment }

Field-Segment =
  Selector-Symbol { "," Selector-Symbol } "->" Type-Expression
```

Structures – example

```
definition
  let FS1,FS2 be 1-sorted;
  struct (ModuleStr over FS1, RightModStr over FS2)
          BiModStr over FS1,FS2
  (#
  carrier -> set,
  addF -> BinOp of the carrier,
  ZeroF -> Element of the carrier,
  lmult -> Function of [:the carrier of FS1, the carrier:],
                         the carrier,
  rmult -> Function of [:the carrier, the carrier of FS2:],
                         the carrier
  #);
end;
```

Structures – Aggregate (ring of integers)

```
definition
  struct (addLoopStr,multLoopStr_0) doubleLoopStr
  (#
    carrier -> set,
    addF, multF -> BinOp of the carrier,
    OneF, ZeroF -> Element of the carrier
  #);
end;
```

```
doubleLoopStr (# INT, addint, multint, In(1,INT), In(0,INT) #)
```

Structures – Forgetful functor (topological part of topological groups)

```
struct 1-sorted (# carrier -> set #);

struct (1-sorted) TopStruct (# carrier -> set,
  topology -> Subset-Family of the carrier #);

struct (1-sorted) multMagma (# carrier -> set,
  multF -> BinOp of the carrier #);

struct (multMagma, TopStruct) TopGrStr (# carrier -> set,
  multF -> BinOp of the carrier,
  topology -> Subset-Family of the carrier #);

let G be TopologicalGroup;
the TopStruct of G is strict;
```

Structures – Selector term (zero element in a linear space)

```
definition
  struct (addLoopStr) RLSStruct
  (#
    carrier -> set,
    ZeroF -> Element of the carrier,
    addF -> BinOp of the carrier,
    Mult -> Function of [:REAL, the carrier :], the carrier
  #);
end;
```

```
let R be RealLinearSpace;
the ZeroF of R is Element of R;
```

Structures – only opening elements

```
<Loci-Declaration>
<Ancestors>
<Structure-Pattern>
<Field-Segments>
  <Field-Segment>
    <Selectors>
      <Selector>
    <Standard-Type>
<Structure-Patterns-Rendering>
  <AggregateFunctor-Pattern>
  <ForgetfulFunctor-Pattern>
  <Strict-Pattern>
  <Selectors-List>
     <SelectorFunctor-Pattern>
```

Verifier

- Parser $\rightsquigarrow$ `.wsx`
- Variables Identifier $\rightsquigarrow$ `.msx`
- Analyzer $\rightsquigarrow$ `.esx, .tix, .xml`
- Reasoner
- Checker

ESX files

ESX repository – /abstr /mml

https://github.com/arturkornilowicz/esx_files.git

MML – structure

- HIDDEN – primitives, automatically imported to all articles
- TARSKI – axiomatics (set theory)
- articles (set theory, relations, functions, logic, calculus, number theory, algebraic structures (groups, rings, algebras, . . .), topology theory, category theory, geometry, mathematical models of computations, . . .)

When the order of processing of articles is important, then they should be processed according to the file **mml.lar** from the Mizar distribution.

HIDDEN.ABS

```
definition   mode object;   end;

definition   mode set -> object;   end;

definition   let x,y be object;
  pred x = y;
  reflexivity;  symmetry;
end;

notation   let x,y be object;
  antonym x <> y for x = y;
end;

definition   let x be object, X be set;
  pred x in X;
end;
```

Translation to other formats

- Probably no problems with the presentation of the content of MML
- Some problems with the translation of proofs – use of implicit premises in inferences
  - built-in requirements (e.g. the correspondence between adjectives like "positive" vs. the formula "> 0")
  - registrations (correspondence between adjectives)
  - reductions of terms to subterms
  - term identifications
  - properties (reflexivity, irreflexivity, symmetry, asymmetry, connectedness, commutativity, idempotence, projectivity, involutiveness)
  - definitional expansions
  - flexary connectives (extra expansions are generated)

Logical connectives

- negation (`not`)
- conjunction (&)
- flexary conjunction (& ... &)
- disjunction (`or`)
- flexary disjunction (`or` ... `or`)
- implication (`implies`)
- bi-implication (`iff`)

Bibliography

- G. Bancerek, Cz. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pąk, J. Urban *"Mizar: State-of-the-art and Beyond"* Intelligent Computer Mathematics, International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings., (M. Kerber et al. Ed(s).), LNCS vol. 9150, Springer, 2015, pp. 261–279.
- G. Bancerek, Cz. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pąk *"The Role of the Mizar Mathematical Library for Interactive Proof Development in Mizar"* J. Automat. Reason. 61 (2018), no. 1–4, pp. 9–32.
- C. Byliński, A. Korniłowicz, A. Naumowicz *"Syntactic-semantic Form of Mizar Articles"* Interactive Theorem Proving, 12th International Conference ITP, Jun 29 – Jul 1, 2021, (L. Cohen et al. Ed(s).), LIPIcs vol. 193, Schloss Dagstuhl Leibniz-Zentrum fuer Informatik, 2021, pp. 11:1–11:17.
- C. Rothgang, A. Korniłowicz, F. Rabe *"A New Export of the Mizar Mathematical Library"* Intelligent Computer Mathematics, 14th International Conference CICM, Jul 26–31, 2021, Timisoara, Romania, (F. Kamareddine, et al. Ed(s).), vol. 12833, Springer, 2021, pp. 205–210.

Thank you very much!