Syntactic-semantic Form of Mizar Articles

Czesław Byliński[1] Artur Korniłowicz[2] Adam Naumowicz[2]

[1]Computer Networks Section, University of Bialystok, Poland

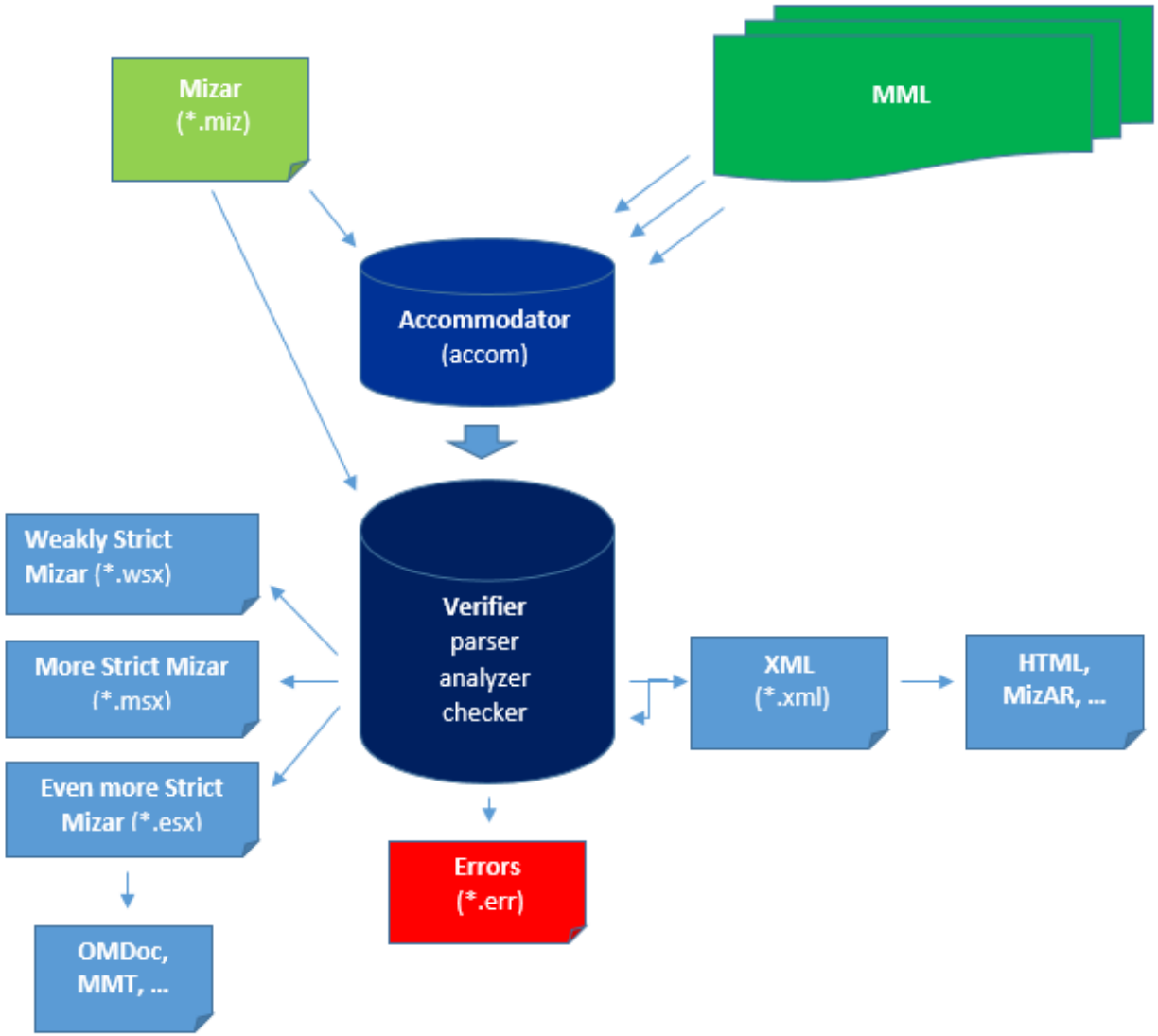
[2]Institute of Computer Science, University of Bialystok, Poland

ITP 2021, July 1, 2021

Mizar Ecosystem in a Nutshell

1. Mizar language
   - subset of standard English (human-readable)
   - based on mathematical vernacular (natural deduction)
   - declarative rather than procedural proof style
2. Mizar system
   - simple command line interface
   - multi-pass verification process
   - efficient implementation
3. Mizar Mathematical Library (MML)
   - developed since 1989
   - 'articles' resembling mathematical papers
   - centrally maintained and peer-reviewed
   - constantly refactored

Strict Mizar Languages



Weakly Strict Mizar (WSM) and More Strict Mizar (MSM)

- Suitable for use independently of the Mizar system
- Generated by the two initial modules of the Mizar verifier:
  - parser – creates a description of the syntactic tree of the analyzed article
  - identification module (part of analyzer) – creates descriptions of the syntax of the Mizar article with additional information about the identifiers used: sentence and variable identifiers
- Do not contain information about the exact constructors used and being defined
- Do not provide unambiguous information that the Mizar system uses to process the whole MML

Even more Strict Mizar (ESM)

- New variant of the Mizar verifier's analyzer module responsible for the semantic analysis of the Mizar language structure
- Generating semantic information while preserving the syntactic information used in the original Mizar source text
  - semantic correlates with extra syntactic information
- Exact description of the syntactic structure of the article in connection with the resolved semantic information
  - all entered symbols
  - the semantics of definitions, expressions and theorems introduced and used in the article
  - enriched with absolute references to database items (within a particular MML version)
  - can be understood regardless of the local environment of a given article

Mizar Definitions

- Mizar allows defining mathematical notions of several basic categories: predicates, adjectives, types, operations and structures
- Various aspects of all these definitions are internally represented either as *formats*, *constructors*, or *patterns*.
  - ***Formats*** represent syntactic information about the kind of symbol used in the definition together with the number and position of arguments.
  - ***Constructors*** provide a numbering system for representing the semantics of mathematical objects (the unique numbering scheme is provided independently within each category of defined notions)
  - ***Patterns*** represent joint information about the used format, constructor, types of all arguments, and positions of visible arguments for a given definition
- The distinction between constructors and patterns is crucial for synonyms or redefinitions available in the Mizar language (they offer different contextual ways of representing semantically equal objects)
- There are more patterns than constructors since not all definitions introduce new constructors

Example: Definitions of the First Projection and Domain

```
definition
  let X be set;
  func proj1 X -> set means
:: XTUPLE_0:def 12
  x in it iff ex y st [x,y] in X;
  correctness;
end;
```

```
notation :: RELAT_1
  let R be Relation;
  synonym dom R for proj1 R;
end;
```

```
definition :: RELSET_1
  let X be set;
  let R be X-defined Relation;
  redefine func dom R -> Subset of X;
end;
```

ESM Representation

```
<Functor-Definition>
 <Redefine occurs="true"/>
 <InfixFunctor-Pattern formatdes="02[0(1)1]" formatnr="45" spelling="dom" position
      ="113\18"
  origpatternnr="23" absoluteorigpatternMMLId="RELAT_1:1" patternnr="35"
  absolutepatternMMLId="RELSET_1:1" constr="K48"
  absoluteconstrMMLId="RELSET_1:1" origconstrnr="42"
  absoluteorigconstrMMLId="XTUPLE_0:9">
  <Loci/>
  <Loci>
   <Locus idnr="22" varidkind="Identifier" spelling="R" position="112\23" origin="
        Constant"
    kind="Constant" serialnr="40" varnr="2"/>
  </Loci>
 </InfixFunctor-Pattern>
 <Type-Specification>
  <Standard-Type nr="5" formatnr="34" patternnr="4"
       absolutepatternMMLId="SUBSET_1:2"
   spelling="Subset" sort="Expandable-Type" position="113\30">
   <Arguments>
    <Simple-Term idnr="11" spelling="X" position="113\35" origin="ReservedVar"
     sort="Constant" serialnr="3" varnr="1"/>
   </Arguments>
  </Standard-Type>
 </Type-Specification>
```

Example: Mizar Structure

```
reserve S1,S2 for 1-sorted;

definition
  let S1;
  let S2 be 1-sorted;
  struct (ModuleStr over S1, RightModStr over S2) BiModStr over S1,S2
  (#
    carrier -> set,
    addF, multF -> BinOp of the carrier,
    ZeroF, OneF -> Element of the carrier,
    lmult -> Function of [:the carrier of S1, the carrier:], the carrier,
    rmult -> Function of [:the carrier, the carrier of S2:], the carrier
  #);
end;
```

Mizar Structures Expanded

Apart from the representation of visible syntactic elements, the XML data structure of ESM now contains the following extra elements enclosed within the `<Structure-Patterns-Rendering>` container

1. `<AggregateFunctor-Pattern>` representing the patterns for tuple terms encoded as `<Aggregate-Term>` (aggregates represent concrete full structures. For example rings of integers with addition and multiplication as ring operations)
2. `<ForgetfulFunctor-Pattern>` representing the patterns for substructure terms `<Forgetful-Functor-Term>` (forgetful terms can represent full structures or substructures which are ancestors (direct or indirect) of the structure, for example `the 1-sorted of B` is an indirect and `the ModuleStr of B` is a direct ancestor of some `B` of type `BiModStr`)

Mizar Structures Expanded

3. `<Strict-Pattern>` defining a special adjective `strict` (for example a ring is not a strict group, because it contains more selectors than those of a group `strict` is generated automatically when a definition of a structure occurs; regular adjectives, like `empty`, `finite`, etc. must be defined within regular definitional blocks)
4. `<Selectors-List>` / `<SelectorFunctor-Pattern>` representing the patterns of terms of the category encoded as `<Selector-Term>` (selector terms represent one field of a given structure, for example `the carrier of B`)

Applications: Linked MML

- A collection of HTML files generated from Mizar source files for intuitive browsing through the library and exploring the inter-linked notions
  - available since 1995
  - reimplemented in 2004 by J. Urban to use the internal XML-based format (`.xml` files) representing the result of the Mizar verifier's analyser pass
- Enriched with various elements invisible in the content of the original Mizar texts, and generated during the verification
  - richer than the article seen at the level of the Mizar text
  - definitional theorems (the internal representation of definitions in the form of equivalence theorems)
  - expanded attribute clusters (sets of attributes appearing in the text extended with automatically calculated consequences based on registrations available in the environment)

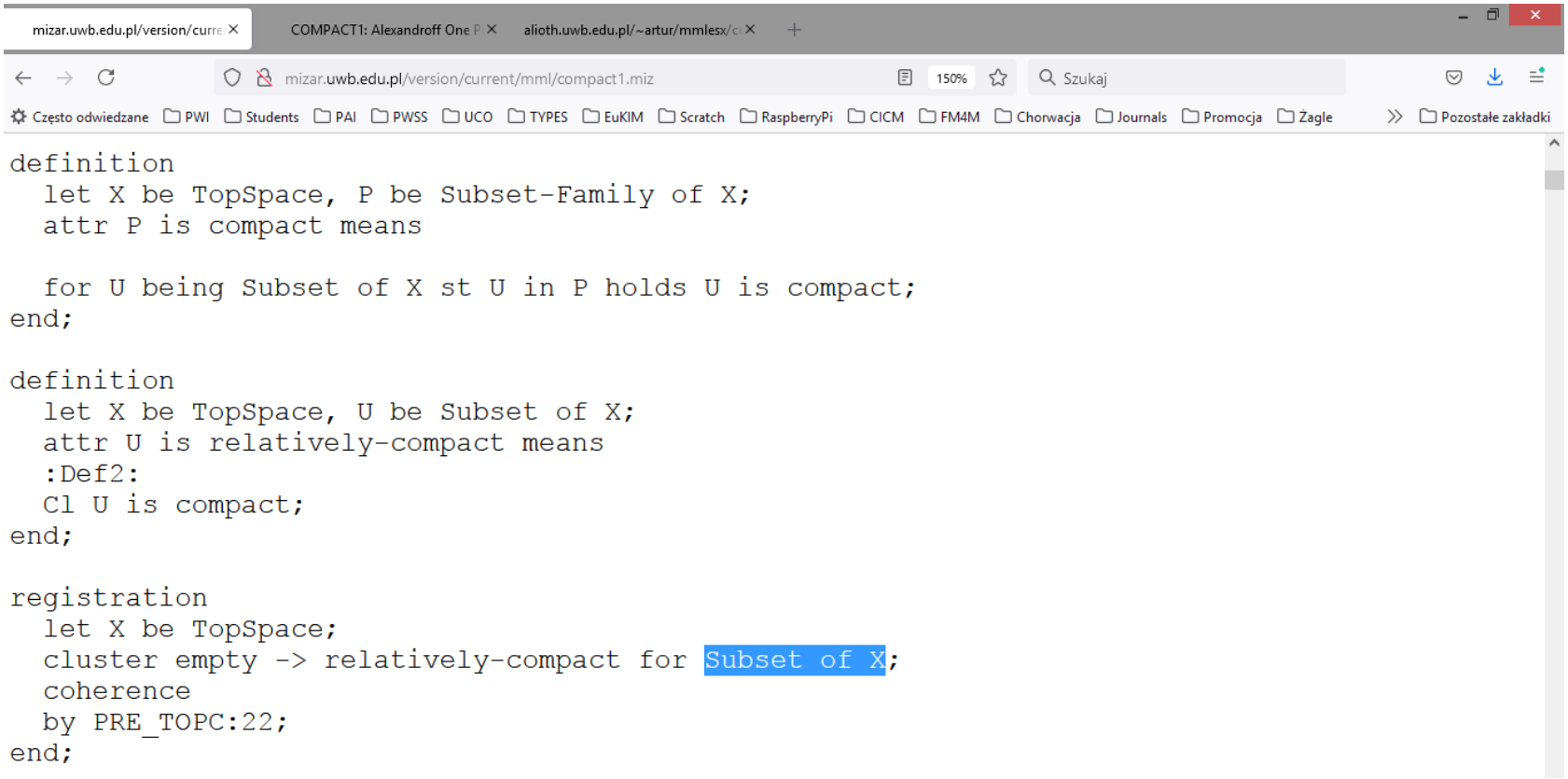
Applications: Linked MML – Shortcomings

- Necessity of multiple 'recalculations' of numerical representations of objects
  - formats
  - patterns
  - constructors
- Loosing the original structure of logical formulas
  - as a result of translating these formulas into semantic correlates needed by the checker
  - expanding local variables introduced with the `set` construct

Applications: ESM-based Linked MML

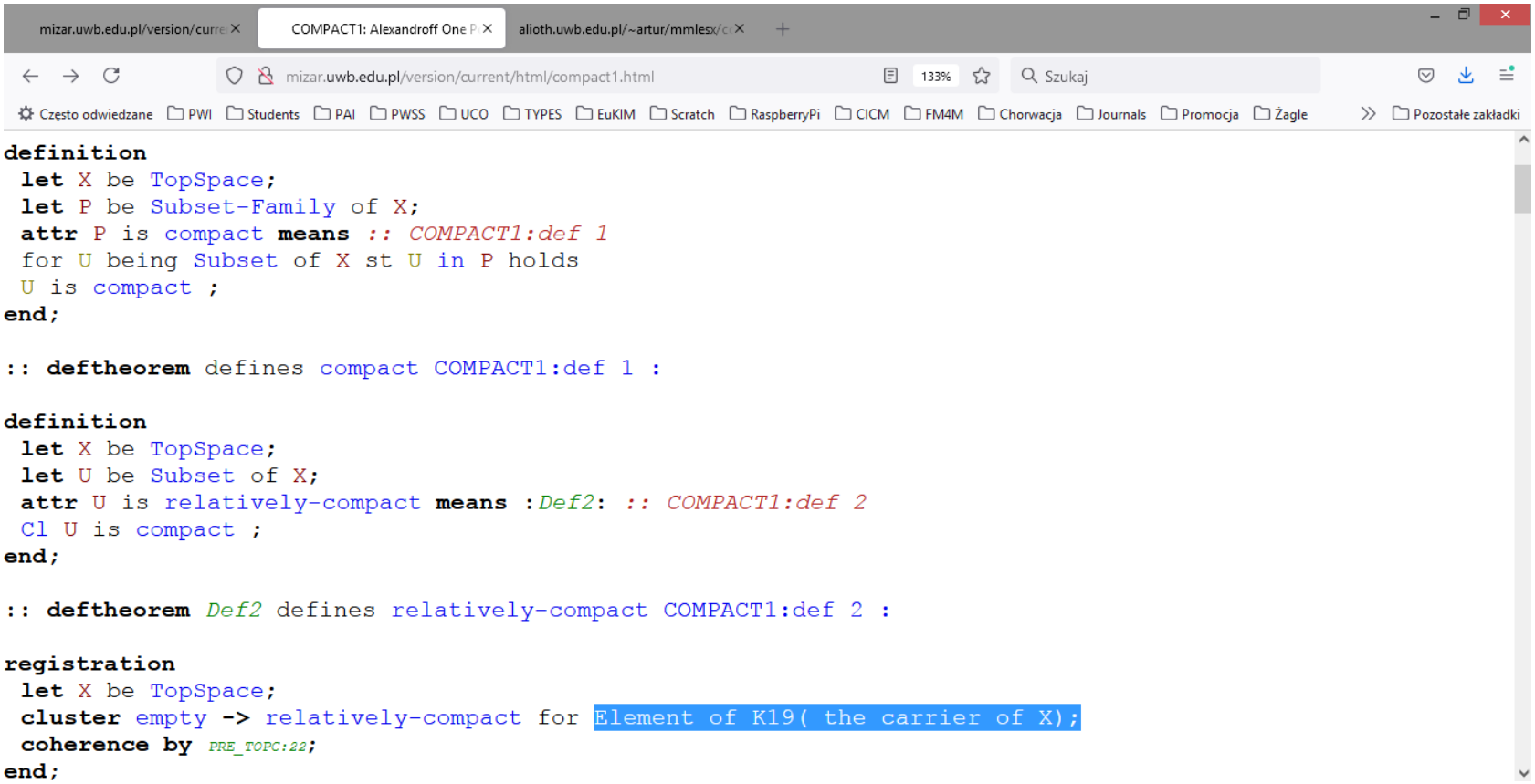
- Rendering the semantic connections between linked notions
- Preserving the original syntactic structure of the underlying Mizar text
- Easily implementing a system of links (anchor names) for the definitions of:
  - all basic concepts, i.e. predicates, adjectives, types, operations and structures
  - local predicates and local functors
  - local labels and references to external statements, definitions, and schemas

Standard Mizar



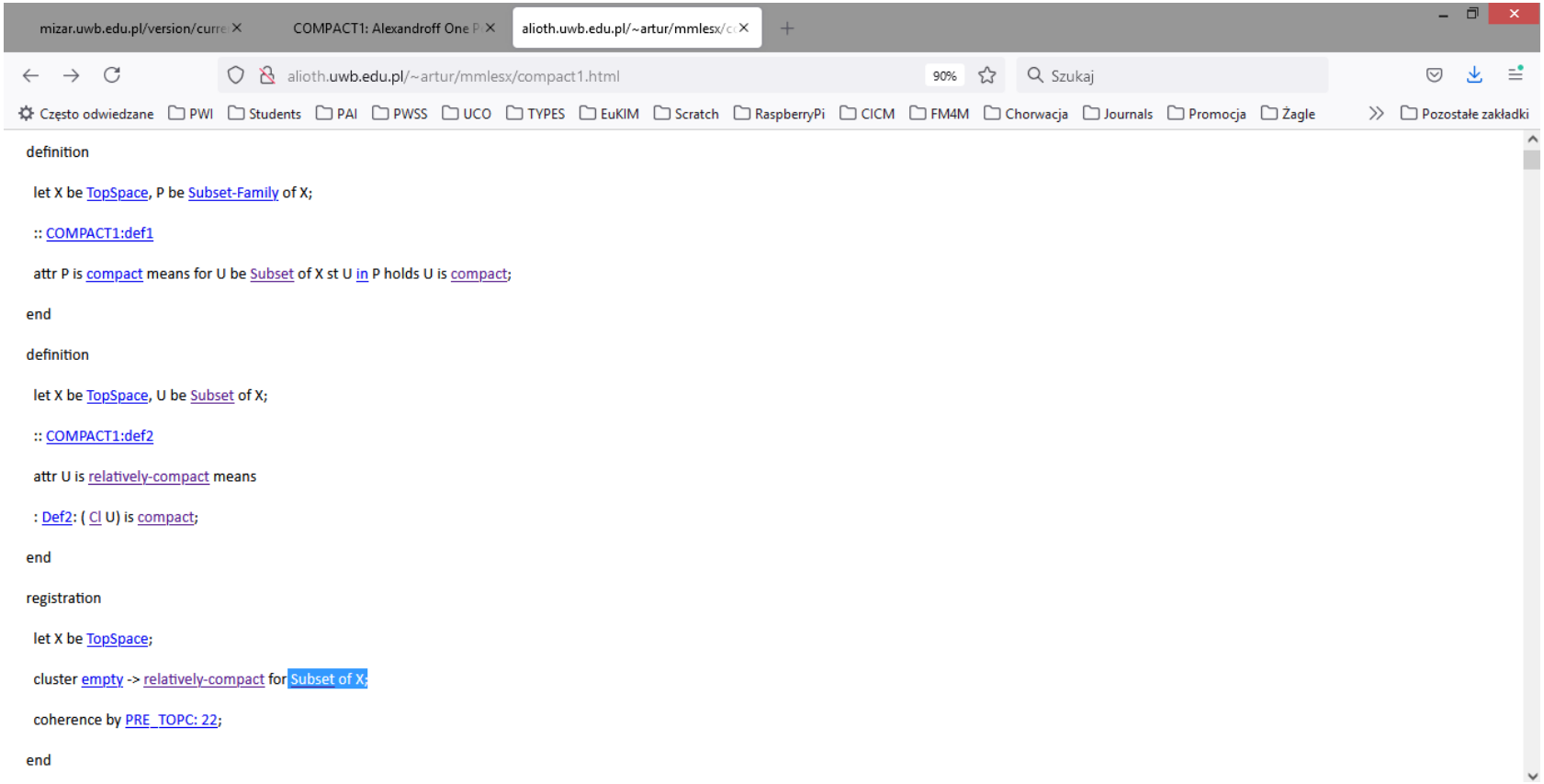

```
definition
  let X be TopSpace, P be Subset-Family of X;
  attr P is compact means

  for U being Subset of X st U in P holds U is compact;
end;

definition
  let X be TopSpace, U be Subset of X;
  attr U is relatively-compact means
  :Def2:
  Cl U is compact;
end;

registration
  let X be TopSpace;
  cluster empty -> relatively-compact for Subset of X;
  coherence
  by PRE_TOPC:22;
end;
```

XML-based HTML




```
definition
 let X be TopSpace;
 let P be Subset-Family of X;
 attr P is compact means :: COMPACT1:def 1
 for U being Subset of X st U in P holds
 U is compact ;
end;

:: deftheorem defines compact COMPACT1:def 1 :

definition
 let X be TopSpace;
 let U be Subset of X;
 attr U is relatively-compact means :Def2: :: COMPACT1:def 2
 Cl U is compact ;
end;

:: deftheorem Def2 defines relatively-compact COMPACT1:def 2 :

registration
 let X be TopSpace;
 cluster empty -> relatively-compact for Element of K19( the carrier of X);
 coherence by PRE_TOPC:22;
end;
```

ESM-based HTML



Applications: *Formalized Mathematics*

- *Formalized Mathematics* (FM) is a journal publishing papers based on regular Mizar formalizations accepted for inclusion into the Mizar Mathematical Library following a round of human peer-review
- After acceptance, the underlying Mizar scripts are automatically translated into a LaTeX format and the resulting generated natural language (English) texts become available as traditional mathematical papers downloadable as PDF files
- The current implementation of the software responsible for the translation is based on a number of `XSLT` style-sheets which convert `.wsx` files representing parse-trees of given Mizar articles into a series of XML files containing human readable meta-text with increasing level of semantic detail

Applications: *Formalized Mathematics* – Translation Patterns

- Authors of Mizar formalizations can propose their preferred *translation patterns* to FM editors
- These patterns allow changing formal and often technical-looking Mizar statements into more natural representation resembling standard mathematical notation using traditional symbols or a fixed placement and order of arguments
- In the current FM translation method based primarily on the `.wsx` files some different notions introduced in the Mizar script are indistinguishable without special processing by more advanced modules of the Mizar verifier
- Example: the multiplication of complex numbers and the multiplication of elements of a ring (both written as infix operations utilizing the same symbol)

Two Multiplications in ESM Format

```
<Functor-Definition MMLId="E1:1">
 <InfixFunctor-Pattern formatdes="043[1(2)1]" formatnr="80"
     spelling="*"
    position="24\9" patternnr="224" absolutepatternMMLId="E1:2"
        constr="K465"
    absoluteconstrMMLId="E1:1" origconstrnr="0">
```

```
<Functor-Definition MMLId="E1:2">
 <InfixFunctor-Pattern formatdes="043[1(2)1]" formatnr="80"
     spelling="*"
    position="33\9" patternnr="225" absolutepatternMMLId="E1:3"
        constr="K466"
    absoluteconstrMMLId="E1:2" origconstrnr="0">
```

- Both operations have the same `formatnr="80"`,
- They can now be uniquely identified by values of the `patternnr` and `constr` attributes

Summary and Conclusions

- Even more Strict Mizar (ESM) is an extended XML-based data format simplifying the access to mathematical notions formalized in Mizar and available as part of the Mizar Mathematical Library
- Easy access to both syntactic and semantic data of the underlying Mizar scripts
- The extra information should allow creating various applications of the Mizar library requiring fullest possible information to be retrieved from the formalization files
- Improved structure of already existing Mizar XML file formats (WSM and MSM)

Future Work

- Still more semantic information needed for knowledge representation systems like OMDoc and MMT
- Extending the XML representation with extra (inferred) type information (Type Inference XML)
  - redefinitions
  - existential witness constants (`take`) and term shortcuts (`set`)
  - annotating visible/hidden arguments
- Diffused statements expansion (`now`)
- Current goal reconstruction (`thesis`)