ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2007
Informatyka – Zeszyt 2

Grzegorz Bancerek[1]

EXPLORING MIZAR LIBRARY WITH MML Query

Abstract: MIZAR, a proof-checking system, is used to build the MIZAR Mathematical Library (MML). MML Query is a semantics-based tool for managing the mathematical knowledge in MIZAR including searching, browsing and presentation of the evolving MML content. The tool is becoming widely used as an aid for MIZAR authors and plays an essential role in the ongoing reorganization of MML.
In the paper, we briefly present MIZAR system including language, tools for logical verification and publishing, foundations of MML, its content and maintenance, and the problems raising when using the MML (information retrieval and rendering). We also present the possibilities offered by MML Query to solve these problems.

Keywords: MIZAR, sematic searching, repositories of formalized mathematics

1. Overview of the Mizar project

The MIZAR language is a language used for such a formalization of mathematics that is close to the vernacular used in mathematical publications. An implemented MIZAR verifier is available for checking correctness of MIZAR texts according to Jaśkowski natural deduction. The perpetual development of the MIZAR system (see [17]) has resulted in the MIZAR Mathematical Library (MML)—a centrally maintained library of formalized mathematics based on Tarski–Grothendieck set theory. Contributions to MML have been the main activity of the MIZAR project since the late 1980s. MML is organized as an interrelated collection of MIZAR articles. At this moment—December 2006—there are 959 articles in MML, occupying 70 MB, containing 43149 theorems and 8185 definitions. The most important facts included in MML are

- Jordan Curve Theorem (JCT), [16],
- Gödel Completeness Theorem (GCT), [12],
- Fundamental Theorem of Algebra (FTA), [18]
- Reflection Theorem, [6]

[1] Faculty of Computer Science, Białystok Technical University, Białystok

JCT is a substantial achievement of the project and is the result of a long-lasting cooperation between Shinshu University and the University of Białystok which was initiated by Yatsuka Nakamura in 1992 and has involved 16 people. About 70 articles[2] from the MML are devoted, directly or indirectly, to the JCT project.

Another large project within MML, called the CCL project [3,9], is aimed at formalization of the theory of continuous lattices as presented in *A Compendium of Continuous Lattices*, [14]. 58 MIZAR articles written by 16 authors cover about 65% of the main course of the book at the moment.

MML is commonly considered the biggest library of computer proof-checked mathematics. The scale of the development of MML could be measured by the number formalized theorems from the list The Hundred Greatest Theorems[3]. Currently, MML contains 35 of 100 theorems. It place the MIZAR system in fourth position in the ranking[4] maintained by Freek Wiedijk.

Notwithstanding the above, MML's coverage of mathematical knowledge is still minuscule. Even so, information retrieval in MML became a burning issue a long time ago. The lack of searching tools, which would be more advanced than some `grep`-based utilities, had delayed work in the CCL project when several authors formalized interrelated parts of the theory using various and barely compatible formalizations from the MML. This prompted in 2000 efforts aiming at development of a semantics-based searching tool for MML [3] and it was the origin of the MML Query system [10].

2. MML Query semantic searcher

The first release of MML Query was completed in 2001 and included basic queries enabling semantic searching of library items only. The second release completed in 2002 is described in [10]. It introduced a number of searchable resources and a variety of queries enabling more advanced searching. Additionally, beginnings of semantic presentation of MML were available in this release. The third release developed in 2004-2005 was inspired by the works aimed at presentation of the content of MML for different purposes:

- an application of MML Query in the Trial-Solution project [13] to generate semantically linked slicing of MIZAR articles,

[2] These articles are not devoted only to JCT but also to Brouwer Theorem, Urysohn Theorem, Tietze Theorem, etc.

[3] `http://personal.stevens.edu/~nkahl/Top100Theorems.html`

[4] Formalizing 100 Theorems, `http://www.cs.ru.nl/~freek/100/`

- translation of MML into the OMDoc format [15],
- semantic browsing in Emacs [11].

These investigations as well as continual development of the web interface to the MML Query system resulted in a text transformation processor MMLQT[5] which is able to interpret the MML Query language. The language in the third release was improved to satisfy requirements of MMLQT (ordered queries, version queries, and metadata queries) and to make searching with MML Query somewhat easier (non-expert searching, rough queries).

Currently, MML Query provides the following functionalities: semantic searching, semantic browsing, semantic presentation, collection of MML statistics,[6] and assistance for authoring MIZAR articles with Josef Urban's Mizar mode for Emacs [21,11]. In consequence, the tool facilitates individual authoring as well as collaborative work in larger projects by enabling adequate searching and uniform and unambiguous presentation. In particular, MML Query provides the possibility to create monographs–the uniform ordered semantic presentation of a specified piece of a theory which may be spread over the MML. These features of MML Query are also used in the ongoing reorganization of MML into the Encyclopedia of Mathematics in MIZAR.

3. MML Query language and its processing

The description of the syntax and semantics of the MML Query language is available on the web[7]. It was also presented in [10] and [4].

The parser of the MML Query language realize the idea of *non-expert queries* (see [4]). It means that first it tries to parse the query according to the syntax. If it is impossible, the parser tries to recognize the input as a part of a MIZAR formula. This is done by recognition of MIZAR symbols and their contexts. For each symbol and for each context the parser generates appropriate queries and composes them in a correct way. Finally, two versions are presented: a strict query and a rough query. Both queries are completed by ordering part which orders the result according to the number of concepts used. The strict query is executed first and if the result is empty, then the rough query is also executed. Such realization helps less experienced users to get better querying results. It also can save time for experts and gives a good base for editing more precise queries.

For example, parsing for the input `theorem 0 = 1` which is not a query according to the syntax of MML Query language generates the following strict query Q_1:

[5] MML Query Templates or MML Query Transformation

[6] Accessible on WWW at `http://mmlquery.mizar.org/`

[7] `http://mmlquery.mizar.org/`

```
1   number 0 occur and
2   (symbol '=' [[[notation | constructor] or vocabulary] | occur]) and
3   number 1 occur
4   | filter th
5   ordered by number of ref
```

and the following rough query Q_2:

```
rough max-max(
  number 0 occur,
  (symbol '=' [[[notation | constructor] or vocabulary] | occur]),
  number 1 occur
)
| filter th ordered by number of ref
```

The strict query Q_1 results in 1544 elements and then the rough query Q_2 is not executed (if it was executed the result would be the same). The elements are ordered by the number of constructors referred to (ordering part in line 5 in Q_1 above). It means that at the beginning we can find the elements using the smallest number of concepts. All elements are theorems (filtering query in line 4) which include number 0 (line 1), number 1 (line 3) and the equality (line 2). Actually, they might not include the equality but any other concept denotated with the symbol '='. But because all 23 notations from MML[8] using the symbol '=' concern the original equality[9] there is no such possibility. The first 4 theorems returned as a result of the query Q_1 are the following

```
CARD_1:87     1 = {0};

COMPTRIG:57   Arg 1 = 0;

NEWTON:18     0! = 1;

NEWTON:27     0 choose 0 = 1;
```

and concern, respectively, the von Neumann form of number 1, the angle of complex number 1, the factorial of number 0, and the Newton binomial coefficient $\binom{0}{0}$. The first column gives MML Query names of theorems which are also MML names used in MIZAR articles for referring (justifying reasoning steps).

[8] the query: `symbol '=' notation` returns 23 elements

[9] the query `symbol '=' notation | constructor | origin` results in one element `HIDDEN:pred 1` which is MML Query name for the original equality.

Execution of the rough query Q_2 starts from executing each partial query (lines 2-4). Each element from obtained results gets the number of partial queries it comes from. E.g., theorems `CARD_1:87` is the answer of each partial query and then it gets the number 3. The biggest number is set as `max` (for Q_2 it is 3) and all elements with numbers between `max` and `max` are returned as the result (all elements satisfying the conjunction of three partial queries). Rough query allows also to provide the limiting numbers directly, e.g., `rough 1-3 (...)`, and to use the word `count` which stands for the number of partial queries. The query `rough count (q1, ..., qn)` is an abbreviation of `rough count-count (q1, ..., qn)` and is equivalent to the conjunction `q1 and ... and qn`

Queries Q_1 and Q_2 above include *proper* parts and *ordering* parts. The *proper* part is obligatory in correct queries and *ordering* parts are optional. A correct query may also include *selection* or *presentation* parts. The *ordering* and *selection* parts may be nested inside the *proper* part if necessary but the *presentation* part may occur only at the end of whole query. For example, in the correct query

```
1   at least minus 3 * (
2      {TOPGEN_3:17,TOPGEN_3:30,CARD_2:44} | ref
3      ordered by number of occur reversed
4      select 0-6
5   )
6   ordered by number of ref
7   presented with version 4.66.942
```

an *ordering* part occurs twice (lines 3 and 6). The first *ordering* part and a *selection* part (line 4) are included in the main *proper* part (lines 1-5). A *presentation* part appears at the end (line 7). The *proper* part of the subquery from lines 2-4 is included in line 2. The theorems in line 2 are theorems from the list of 100 Greatest Theorems:

- The Denumerability of the Rational Numbers, [5]
- The Non-Denumerability of the Continuum, [5]
- The Number of Subsets of a Set (PS[10]), [2]

The sub-subquery in line 2 selects all constructors (concepts) referred to by these theorems. Next, the ordering rule `number of occur reversed` in line 3 orders them according to descending number of all occurrences in whole MML. So, the most popular constructors are at the beginning. The selection in line 4 takes first 7 of them. The rough variant of group query `at least minus 3 * ...` finds all elements from MML which refer to at least $4 = 7 - 3$ chosen constructors. Finally, the result is ordered

[10] Power Set

by the number of constructors referred to and presented according to the version 4.66.942 of MML.

Examples of less complicated (basic) queries are given below.

```
CARD_2:44 ref                                             (1)
{TOPGEN_3:17,TOPGEN_3:30,CARD_2:44}                       (2)
list of th from TOPGEN_3                                  (3)
(CARD_2:44 ref) butnot (list of constr from CARD_2)       (4)
(list of th from TOPGEN_3) | ref                          (5)
(TOPGEN_3:def 4 ref) & occur                              (6)
list of th where [ref | filter struct]                    (7)
list of constr where notation > 1                         (8)
list of th where positive ref <= negative ref             (9)
```

The queries listed above may be read as follows: (1) all constructors appearing in PS, (2) list of three theorems mentioned above, (3) all theorems from article TOPGEN_3, (4) all constructors from PS defined in articles other than CARD_2, (5) all constructors appearing in any theorem from article TOPGEN_3, (6) all items which refer to all constructors appearing in the definitional theorem of $\mathfrak{c}$ (continuum), (7) all theorems concerning structures, (8) all constructors with more than one notation, (9) all theorems that refer positively to a bigger number of constructors than they refer negatively. Other examples of queries can be found in [10] and [4].

4. MML Query Transformation

The presentation of the results of querying as well as the presentation of the content of MML for other purposes prompted the development of MML Query Transformation (MMLQT). At the moment, MMLQT is a text processor used for rendering MML content when browsing MML Query results, making MML statistics, or generating semantically linked abstracts [11] and OMDoc repository [15].

The transformation processor is based on a bunch of templates which include directives to the processor. Each purpose above has its own bunch of templates. MMLQT templates are text files which could be written in different styles, e.g., XML or LaTeX. The key role in XML style is played by the XML element `<mmlq>` which holds the directives. The transformation processor parses only the occurrences of `<mmlq>` and does not count any other XML elements—the text outside `<mmlq>` elements is simply rewritten. The directives are specified with the XML attribute `type` of `<mmlq>`

element. Namely, the value of the attribute determines the task to be performed by the processor. The performance depends on changing environment of the MMLQT processor which includes inter alia the current *argument* and the *version* of MML. Directives may change the environment directly (e.g., directive `change` for *argument* and `changever` for *version*) or indirectly (directive `foreach` for *argument*).

The base directive is `explain`. The role of this directive is presenting the meaning of an appropriate MML Query element according to the current presentation style which may differ in different purposes. The directive may have, for example, the following forms

```
<mmlq type="explain"/>
<mmlq type="explain" argument="CARD_2:44"/>
<mmlq type="explain" operation="ref"/>
<mmlq type="explain" query="list of notat from TOPGEN_3"/>
```

The first directive explains simply the meaning of *argument* and the second explains the meaning of theorem PS (`CARD_2:44`). The third explains the first element from the list of elements referred to by *argument*. The last explains the first notation from article `TOPGEN_3`. To explain all notations from `TOPGEN_3` we must use a little bit more complicated directive

```
<mmlq type="foreach" query="list of notat from TOPGEN_3">
  <mmlq type="explain"/>
</mmlq>
```

MMLQT templates could be created with the web interface Template Maker available at

`http://mmlquery.mizar.org/template-maker.php`

It has some support for editing directives in XML style and allows to render templates with the presentation style prepared for MML semantic browsing and statistics. It means that all symbols in rendered MIZAR formulae are linked to their definitions with suggestions for further browsing.

We may use Template Maker to render the meaning of 3 theorems mentioned in the previous section. It can be done by writing the following template

```
<mmlq type="foreach" query="{TOPGEN_3:17,TOPGEN_3:30,CARD_2:44}">
  <center><mmlq type="value"/></center>
  <mmlq type="explain"/>
  <hr>
</mmlq>
```

The template includes query (2) and the directive `value` which renders the MML Query name of *argument*. Moreover, two HTML elements, `<center>` and `<hr>`, are added to beautify the rendering:

```
                         CARD_2:44
theorem
for b1 being set holds
   exp(2,Card b1) = Card bool b1;
-------------------------------------------------------------
                        TOPGEN_3:17
theorem
Card RAT = alef 0;
-------------------------------------------------------------
                        TOPGEN_3:30
theorem
alef 0 in continuum;
-------------------------------------------------------------
```

The body of `<mmlq>` element with `foreach` directive is a subtemplate which is processed by MMLQT processor for each MML Query item retrieved with the query from XML attribute `query`.

MML Query statistics are generated by processing MMLQT templates. The main role in these templates is played by the directive `count` which renders the quantity of the result of appropriate query. For example, the main statistic page shows quantities of different resources in MML.

Version 4.76.959:

- 189 authors,
- 960 articles,
- 43149 theorems,
- 8185 definitions,
- 739 schemes,
- 9791 constructors: 106 aggregates, 1846 attributes, 6330 functors, 410 modes, 851 predicates, 142 selectors, and 106 structures

...

It is rendered from the following template:

```
Version <mmlq type="version" part="MML"/>:
<ul>
<li><mmlq type="link" template="authors.mqt">
       <mmlq type="count" query="list of article|author"/> authors
    </mmlq>,
<li><mmlq type="count" query="list of article"/> articles,
<li><mmlq type="count" query="list of th"/> theorems,
<li><mmlq type="count" query="list of def"/> definitions,
<li><mmlq type="count" query="list of sch"/> schemes,
<li><mmlq type="count" query="list of constr"/> constructors:
    <mmlq type="count" query="list of aggr"/> aggregates,
    <mmlq type="count" query="list of attr"/> attributes,
    <mmlq type="count" query="list of func"/> functors,
    <mmlq type="count" query="list of mode"/> modes,
    <mmlq type="count" query="list of pred"/> predicates,
    <mmlq type="count" query="list of sel"/> selectors, and
    <mmlq type="count" query="list of struct"/> structures
...
</ul>
```

The other important role in statistics plays an *ordering* part together with a *selection* part of queries. They allow to choose top *n* MML Query items according to an appropriate criterion and to render them in an appropriate order. For example, the statistic page "The most popular theorems" uses the following query

```
list of thdef ordered by number of in by ref reversed  select 0-29
```

It shows top 30 (select 0-29) theorems and definitional theorems (thdef) and the choosing criterion is the number of items which refer to the theorem in their proofs. The relation `in by ref` is the inverse of the basic relation `by ref` which connect a MML Query item *i* with a theorem *t* if in the proof of *i* there is a reference to *t*. The most popular theorem is `TARSKI:def 1` which is definitional theorem of singleton set. It is referred in the proof by 3134 items.

Some statistics require a complicated ordering. For example, the page "The Longest Type Widening" requires the calculation of iteration degree for the following compound relation *W*:

```
[not = HIDDEN:mode 1
  | [ mothertype ref or prefix ref ]
    | [ mode or struct ]
]
```

The widening of MIZAR types is described in [7]. It is a transitive-reflexive closure of mother type and prefix relations. MML Query relations `mothertype ref` and `prefix ref` include mother type and prefix relations, respectively, but they concern also other constructors occurring in widening. Therefore, the relation W filters type constructors by `[ mode or struct ]` filter. The type `HIDDEN:mode 1` (*set*) is the widest type. Therefore, the relation W uses `not = HIDDEN:mode 1` relation as the type *set* is a mother type of itself. Summarizing, two types T_1 and T_2 are in the relation W if T_1 is different from the type *set* and T_2 occurs in the mother type or prefix of T_1 and T_2 is type constructor. The query resulting in top 20 types with longest widening is the following:

```
list of mode ordered by iteration of W reversed select 0-19
```

One of the longest widening starts from type `Leaf of T` where `T` is a `Tree`. It widens to `Element of Leaves T` which is `Element of T`. In further widening there occur a finite sequence of natural numbers, a partial function from the set $\mathbb{N}$ of natural numbers into $\mathbb{N}$, a subset of Cartesian product $\mathbb{N} \times \mathbb{N}$, and, finally, a set.

5. MML **Query interfaces**

The basic interface for MML Query is the program `mmlquery` written in the Perl language. It is a base for every other MML Query interface. It can be downloaded from MML Query home page together with zipped database files[11]. The full installation (see README file) requires 5.2GB of free disk space and is ready to use after unpacking and customizing a few variables. The program allows to call queries from the command line and to input them with or without support. The support requires Perl module `TermReadKey` which is a part of some Linux distributions[12] (e.g., Fedora Core 5) and is also available from CPAN Perl library[13].

To call a query from command line one has, for example, to type:

```
mmlquery --single --present=decode --query="list of th from CARD_2"
```

or simply

```
mmlq decode list of th from CARD_2
```

The first argument of `mmlq` determines a presentation style and others form a query. The program called in a terminal window for inputting queries with the command

[11] `http://mmlquery.mizar.org/mmlquery/downloads/`
[12] It may be installed with command: `yum install perl-TermReadKey`
[13] `http://www.perl.com/CPAN/`

```
mmlquery --input --present
```

shows the following message:

```
MML Query, version 1.4.01
Copyright (c) 2001-2007 Association of Mizar Users
Report bugs to Grzegorz Bancerek, bancerek@mizar.org
For help type \h or point your web browser to http://mmlquery.mizar.org/
Different presentation - see option \op present
Current presentation style = all_fields
Presentation styles: standard, all_fields, paths, decoded_exp, decoded,
                     mxa, html, emacs
mmlquery> _
```

The option `--input` turns on the input support of the interface. The input queries are interpreted and executed by the program and the result is presented with one of presentation styles. The simplest style `standard` gives only MML Query names. A more advanced style, `decoded`, gives reconstructed MIZAR formulae. The style `html` is used by the web interface[14] [10,4] and equips MML Query names with links for explanation and further browsing. The explanations and further browsing is rendered with MMLQT transformation. The style `emacs` is used by MIZAR mode for Emacs [21] when browsing GABs (Generated MIZAR abstracts, see [11]). GABs are produced with MMLQT transformation and offer integrated semantic browsing and demonstrate the information hidden in MIZAR articles. They offer also possibilities for querying with adequate constructors in provided interface.

MML Query interface is called in one of three modes: `--plain`, `--command`, and `--admin`. The plain mode is used only for querying. The command mode allows additionally

- to make abbreviations for MML Query items (e.g., `set Set = HIDDEN:mode 1`),
- to make abbreviations for MML Query relations and results,
- to set resource relations which are used in *non-expert queries* as default relations,
- to set explanation functions which are used in the result window of the web interface.

The command mode is set as default for the web interface. The interface in MIZAR mode for Emacs maintained by Josef Urban keeps also a running process of the `mmlquery` program in command mode and with `emacs` presentation style. The MIZAR mode takes over the results of queries and presents them integrated with other features.

[14] `http://mmlquery.mizar.org/mmlquery/three.html`

The admin mode is used to make indexing of all resources. It provides the means to change the MML Query database and to process additional MIZAR resources such as bibliographic files and translation patterns of the journal *Formalized Mathematics*. Translation patterns include English phrases or $\mathcal{AMS}$ TEX symbols for each MIZAR symbol which can be used in searching. E.g., the MIZAR symbol `VectSp` which corresponds to MML Query item `VECTSP_1:modenot 6` is translated in *Formalized Mathematics* as *vector space*. The last form is probably the best for beginners. (More about automatic translation in *Formalized Mathematics* could be found in [1])

6. Further work

Further work concerns the development of *non-expert queries* and the improvement of the functionality of MML Query interfaces. These tasks require better integration of MML Query with the MIZAR system. To strengthen the integration, MML Query tools must use the internal XML format of MIZAR articles which is newer than MML Querys̀ own format. Successful porting to MIZAR XML format requires an extension of this format used for the first stage of MIZAR text processing, since some information important for searching is lost with current implementation.

Another direction in further work concerns investigations on data mining and theorem prover techniques to extract from MML new basic relations which could improve searching.

References

[1] Bancerek, G.: Automatic translation in Formalized Mathematics, Mechanized Mathematics and Its Applications, 5(2): 19-31, 2006, MIZAR Japan.

[2] Bancerek, G.: Cardinal Arithmetics, *Formalized Mathematics*, 1(3):543–547, 1990.

[3] Bancerek, G.: Development of the theory of continuous lattices in MIZAR, in M. Kerber and M. Kohlhase (eds), *Symbolic Computation and Automated Reasoning*, 65-80, A. K. Peters, 2001.

[4] Bancerek, G.: Information retrieval and rendering with MML Query, in J.M. Borwein, W.M. Farmer (eds), Proceedings of MKM 2006, Wokingham, UK, *LNAI*, 4108: 266-279, 2006.

[5] Bancerek, G.: On Constructing Topological Spaces and Sorgenfrey Line, *Formalized Mathematics*, 13(1):171–179, 2005.

[6] Bancerek, G.: The Reflection Theorem, *Formalized Mathematics*, 1(5):963-972, 1990.

[7] Bancerek, G.: On the structure of Mizar types. Electronic Notes in Theoretical Computer Science, Vol. 85 (7), Elsevier, 2003.

[8] Bancerek, G., Endou, N., Shidama, Y.: Lim-inf Convergence and its Compactness. *Mechanized Mathematics and Its Applications*, 2(1): 29-35, 2002.

[9] Bancerek, G., Rudnicki, P.: A Compendium of Continuous Lattices in MIZAR, *Journal of Automated Reasoning*, 29(3-4): 189-224, 2002.

[10] Bancerek, G., Rudnicki, P.: Information retrieval in MML, in A. Asperti, B. Buchberger, J. H. Davenport (eds), Proceedings of MKM 2003, Bertinoro, *LNCS*, 2594: 119-131, 2003.

[11] Bancerek, G., Urban, J.: Integrated Semantic Browsing of the Mizar Mathematical Library for Authoring Mizar Articles, in Asperti, A., Bancerek, G., Trybulec, A. (eds), Proceedings of MKM 2004, Białowieża, *LNCS*, 3119: 44-57, 2004.

[12] Braselmann, P., Koepke, P.: Gödel's Completeness Theorem, *Formalized Mathematics*, 13(1):49-53, 2005.

[13] Dahn, I.: Management of Informal Mathematics Knowledge–Lessons Learned from the Trial-Solution Project, in Bai, F., Wegner, B., Electronic Information and Communication in Mathematics, LNCS 2730:29-43, 2003.

[14] Gierz, G., K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, D. S. Scott. *A Compendium of Continuous Lattices*, Springer-Verlag, Berlin, 1980.

[15] M. Kohlhase (joint work with Bancerek, G.). Towards MIZAR Mathematical Library in OMDOC format. Electronic Proceedings of the Workshop *Mathematics on the Semantic Web*, Eindhoven, 2003, `http://www.win.tue.nl/dw/monet/proceedings/kohlhase-mizarinomdoc.ps`

[16] Korniłowicz, A.: Jordan Curve Theorem, *Formalized Mathematics*, 13(4):481-491, 2005.

[17] R. Matuszewski, Rudnicki, P.: MIZAR: the first 30 years. *Mechanized Mathematics and Its Applications*, **4**(1):3–24, 2005.

[18] R. Milewski. Fundamental Theorem of Algebra, *Formalized Mathematics*, 9(3):461-470, 2001.

[19] Rudnicki, P., Ch. Schwarzweller, A. Trybulec. Commutative Algebra in the Mizar System. *Journal of Symbolic Computation*, **32**:143–169, 2001.

[20] Rudnicki, P., A. Trybulec. On equivalents of well-foundedness. *Journal of Automated Reasoning*, 23(3-4):197–234, 1999.

[21] J. Urban. MizarMode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. Journal of Applied Logic, 2005, `doi:10.1016/j.jal.2005.10.004`

[22] J. Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. International Journal on Artificial Intelligence Tools, 15(1): 109–130, 2006.

[23] J. Urban. XML-izing Mizar: making semantic processing and presentation of MML easy. in M. Kohlhase (ed), Proceedings of MKM 2005, Bremen, *LNCS*, 3863: 346-360, 2006.

[24] J. Urban, Bancerek, G.: Presenting and Explaining Mizar. in S. Autexier and C. Benzmüller (eds), Proceedings of UITP 2006, IJCAR'06 Workshop, Seattle, 97-108, 2006.

[25] Wiedijk, F.: *Mizar: An Impression.* `http://www.cs.kun.nl/~freek/notes`.

Partially supported by Białystok Technical University grant W/WI/1/06

EKSPLORACJA BIBLIOTEKI MIZARA Z UŻYCIEM MML QUERY

Streszczenie: MIZAR(komputerowy system weryfikacji dowodów) jest używany do budowania MIZAR Mathematical Library (MML). MML Query jest narzędziem opartym o semantykę tekstu mizarowego służącym do zarządzania wiedzą w systemie MIZAR włączając w to wyszukiwanie, przeglądanie i prezentację ewolującej zawartości MML. Narzędzie to stało się szeroko używane jako pomoc dla autorów mizarowych oraz odgrywa istotną rolę w nieustających reorganizacjach MML.
W artykule zaprezentowane są elementy systemu MIZAR i MML Query jak język, narzędzia logicznej weryfikacji i automatycznej publikacji, podstawy biblioteki MML, jej zawartość i konserwacja oraz problemy pojawiające się przy używaniu MML (odzyskiwanie informacji oraz jej przedstawianie). Ponadto, są przedstawione możliwości MML Query w rozwiązywaniu tych problemów.

Słowa kluczowe: MIZAR, wyszukiwanie semantyczne, repozytoria sformalizowanej matematyki

Artykuł zrealizowano w ramach pracy badawczej W/WI/1/06