Normalization of Digital Mathematics Library Content

MathML Canonicalization

David Formánek, Martin Líška, Michal Růžička, and Petr Sojka

Masaryk University, Faculty of Informatics
Botanická 68a, 602 00 Brno, Czech Republic
david.formanek@mail.muni.cz, martin.liski@mail.muni.cz,
mruzicka@mail.muni.cz, sojka@fi.muni.cz

Abstract. Paper discusses the needs for data normalization in a Digital Mathematics Library (DML). Specifically, emphasis is given to canonicalizing formulae encoded in Presentation MathML notation which starts to be available in several DMLs and is used by DML applications. This is a prerequisite for advanced processing — namely math enabled fulltext searching or semantic filtering and automated classification. Different sources of MathML and their specifics are described. Several use cases of possible formulae canonicalization transformations are listed and discussed in detail. Findings are finally concluded and a design of a to-be-developed canonicalization tool is outlined.

Keywords: MathML normalization, canonicalization, digital mathematics libraries, DML, presentation MathML

1 Motivation

Modern Digital Mathematics Libraries (DML) such as EuDML [18,5] base their services on paper semantics, i.e. fulltext handling, including mathematical formulae, as well as basic metadata and Mathematics Subject Classification (MSC) codes. Mathematics literature is widely dispersed across a high number of publishers, making it very difficult to collect fulltexts from these heterogeneous sources. This situation is very different from other libraries, such as PubMed Central for biomedical and life sciences, where publishers have an agreed workflow using the NLM Journal Publishing Tag Set and tools developed with funding from the National Institutes of Health.

Full paper texts have to be 'homogenized', converted to some uniform representation, in order for math-aware full-text searches [15] and paper similarity computations [11,12] to work properly. These tasks are usually handled based on a bag-of-words representation of a document text — vector space model — every term (word, lemma) has its own dimension and the number of occurrences of a term reflects its value. Non-textual terms such as mathematical formulae are mostly not taken into account. This creates another challenge for DMLs, as

mathematical formulae are the essence of mathematical publications. There is an average of 380 mathematical formulae per arXiv paper in the MREC database [8]. It has been reported [21] that even a single histogram of mathematical symbols is sufficient for domain classification of a paper in the mathematical domain.

To reliably represent a paper for DML processing, including handling the mathematics, it is necessary to

1. select a canonical representation of the non-textual *structural* entities appearing in fulltexts (mathematical symbols, formulae, and equations); and
2. decide on equivalence classes for these entities (e.g., for which formulae should be considered equal for given DML tasks such as search, similarity computation, formulae editing, and conversion of math into Braille).

In this paper, we discuss the options for selecting the canonical representations of formulae to be used in DML tools, and the *canonicalization* process — the process — of computing this canonical representation from a variety of different sources and formats.

Our primary motivation is the natural requirement for our own (Web)MIaS system, which currently uses Presentation MathML [14] to operate correctly and offer an expected search behaviour to users regardless of the MathML input source. When a user posts a query to the system, the system must abstract it from the underlying notational differences in order for it to behave correctly. This requirement is increasingly emphasized with the growing number of different sources of MathML. Currently there are three sources (LaTeXML, Tralics, and user input; the number is expected to increase). If they are not correctly normalized the system misbehaves and it appears to users as if it simply does not work, however good the underlying design is.

We have used UMCL library [1,2] for canonicalization in our MIaS system sofar. However, we have found that the deficiencies of the software are so severe (change of formulae semantics, slowness,...) [7, chapter 5], and the need for canonicalization so important, that we have decided to design and implement new canonicalization tool from scratch.

This paper is structured as follows: in Section 2, different sources of mathematics are described and their differences are discussed. The core part of this paper is Section 3, where several use cases of possible canonical representation and canonicalization are documented and suggested. We conclude with Section 5, and present a plan for future work.

2 MathML Sources

To store mathematical formulae in our documents we have chosen MathML[1] — an XML-based language — as a widely used, formally defined, but still evolving standard. The widespread use of MathML and its XML base means of this

[1] More precisely, Presentation MathML, as there are currently significantly more real-life resources using this form of MathML than Content MathML.

language is supported by various tools in the whole document workflow. More importantly, MathML can be used as a common language among the advanced computer mathematical software packages that are extensively used by working mathematicians.

On the author end of the document workflow the MathML code can be 'hand made' using simple plain text editors such as MS Windows Notepad, or something more comfortable, such as specialized XML editors that are usually part of various integrated development environments. For example, the formula $x^2 + y^2$ can be written as follows:

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
    <msup>
        <mi>x</mi><mn>2</mn>
    </msup>
    <mo>+</mo>
    <msup>
        <mi>y</mi><mn>2</mn>
    </msup>
</math>
```

Listing 1: Example of the 'hand made' formula $x^2 + y^2$

However, the XML nature of MathML makes the coding of more complex formulae rather long for manual construction. Various software tools are more frequent sources of MathML. MathML can be generated as an output / data exchange format of complex specialized programs, such as Maple, Matlab, and Mathematica [9,20,22], or web services, such as the well known Wolfram Alpha [23], that are extensively used by mathematicians to support their work.

```
generate::MathML(x^2 + y^2,
                 Content = FALSE, Annotation = FALSE)
<math xmlns='http://www.w3.org/1998/Math/MathML'>
  <mrow xref='No7'>
    <msup xref='No3'>
      <mi xref='No1'>x</mi>
      <mn xref='No2'>2</mn>
    </msup>
    <mo>+</mo>
    <msup xref='No6'>
      <mi xref='No4'>y</mi>
      <mn xref='No5'>2</mn>
    </msup>
  </mrow>
</math>
```

Listing 2: Example of MathML export of the formula $x^2 + y^2$ by Matlab 7.9.0 MuPAD symbolic engine

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <msup>
   <mi>x</mi>
   <mn>2</mn>
  </msup>
  <mo>+</mo>
  <msup>
   <mi>y</mi>
   <mn>2</mn>
  </msup>
 </mrow>
</math>
```

Listing 3: Example of the MathML export of the Wolfram Alpha input query 'x^2 + y^2'

On the consumer end of the document workflow MathML can be used as an input for mathematical programs and services (Maple, Matlab, Mathematica, Wolfram Alpha, etc.) or simply displayed — usually as part of an XHTML web page — in a web browser with MathML support.

However, a large number of mathematical documents are produced using the TEX typesetting system and authored in TEX markup. Thus, it is necessary to be able to convert the TEX source code of mathematical formulae to the MathML language. Our main motivation is the WebMIaS system. For more complex input formulae, it would be uncomfortable for the user to manually construct queries in MathML, as the code would be very complicated. The well known LATEX syntax is far more appropriate for manual input. Therefore, we need a conversion from LATEX to MathML as part of the WebMIaS input routine.

There are several tools that are able to convert TEX markup to the MathML language. For example, arXMLiv [16] employs LATEXML [19]. The EuDML project and our WebMIaS [8] system internally use Tralics [6].

```
<math xmlns="http://www.w3.org/1998/Math/MathML"
    alttext="x^{2}+y^{2}" display="inline">
    <semantics>
        <mrow>
            <msup><mi>x</mi><mn>2</mn></msup>
            <mo>+</mo>
            <msup><mi>y</mi><mn>2</mn></msup>
        </mrow>
        <annotation encoding="application/x-tex">
            x^{2}+y^{2}
        </annotation>
    </semantics>
</math>
```

Listing 4: Example of LATEXML generated MathML of formula $x^2 + y^2$

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
    <mrow>
        <msup>
            <mi>x</mi> <mn>2</mn>
        </msup>
        <mo>+</mo>
        <msup>
            <mi>y</mi> <mn>2</mn>
        </msup>
    </mrow>
</math>
```

Listing 5: Example of Tralics generated MathML of formula $x^2 + y^2$

A frequent type of mathematical document in DML is the older papers that are unavailable in any digital-format or are available only in an 'end' format such as PDF that is suitable for reading and printing but is not appropriate for direct MathML processing. These documents can be a significant part of the DML content collection, so they are worth further processing.

Documents available in hard copy only can be scanned and processed using InftyReader [17] optical character recognition (OCR) software. InftyReader has a unique feature for detecting mathematical formulae in a scanned document. These formulae can be subsequently saved as MathML.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
    <msup>
        <mi mathvariant="italic">x</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
    <mo mathvariant="normal">+</mo>
    <msup>
        <mi mathvariant="italic">y</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
</math>
```

Listing 6: Example of InftyReader generated MathML from a PDF document containing only formula the $x^2 + y^2$ in its body

Born-digital PDF documents with no available source codes can be processed using the MaxTract software [3,4], which that is under intensive development as part of the EuDML project. MaxTract generates LaTeX source / XHTML+MathML representation of the document based on an optical analysis of the positions of

characters on the page. The analysis is supported with information from the fonts embedded in the processed document.

```
<math display="block" xmlns="&mathml;">
 <mi>&#x0078;</mi>
 </math>

<p >

</p>

<p align="right" >
<math display="inline" xmlns="&mathml;">
 <mi>&#x0079;</mi>
 </math>
</p>

<p >

</p>
```

Listing 7: Example of XHTML + MathML generated by the development version of MaxTract from a PDF document containing only the formula $x^2 + y^2$ in its body

During the MathDex project, it became clear that the most time- and resources-consuming task in building a math search engine and database is the normalization and conversion of heterogeneous sources [10]. As shown in Listings 1 — 6, MathML can vary slightly due to the different ways a code was obtained, even for a trivial formula like $x^2 + y^2$.

In a DML project, there can be differences in the final MathML encoding even for semantically and structurally similar formulae, due to the origins of the MathML from different sources. In Section 3, several more complicated examples of possible ambiguities in MathML are discussed that have to be normalized to allow math searches and similarity computation.

3 Use Cases

Using our public working demo of the WebMIaS system we discovered several discrepancies in the form of MathML generated by the real-time TEX to MathML converter we currently use — Tralics — and by the MathML canonicalizer from the UMCL library. We employed the UMCL canonicalization module to try to normalize the users' MathML input and the MathML produced by the LATEXML converter contained in the arXMLiv collection. Then we went through the Presentation MathML specifications and gathered a list of possible reformatting rules we could perform.

The goal is to reduce the possible MathML scripts with the same semantics and mathematical structures to just one representation. To have such a canonicalized representation is convenient for many applications, as was described in Sections 1 and 2.

Analyzing the issues of possible inconsistencies and ambiguities of MathML-encoded formulae raised design and strategy questions. Conceptual decisions for handling different types of similar constructions and completely different formulae need to be made.

More specifically, for example, should we try to keep the MathML compact and reduce the number of nodes in transformations, or should we try to add nodes for better disambiguation? Another question is: should our future canonicalization tool produce valid MathML according to this schema? Unquestionably, this feature would be nice to have for many reasons and possible applications, but it certainly adds more requirements and takes much more effort to design and implement not only true/false validation, but also functional correctness validation.

Below are described proposals and discussions of transformations that can be performed with relatively minor difficulty. The list is not complete and is subject to further evaluation.

3.1 Removing Elements and Attributes

Many of the MathML elements used in Presentation MathML make little or no contribution to the semantics of the formula and therefore also to the formulae for indexing and searching. These are usually elements that alter the appearance of formulae in some way — space-like elements such as `mspace`, `mpadded`, `mphantom`, `maligngroup`, and `malignmark`. They may occasionally have some semantic meaning, but we prefer to canonicalize similar formulae into one representation rather than risk treating the same formulae as different. Therefore, these elements are best omitted. The content of the `mtext` element should be indexed as normal text before removal.

Most element attributes are similarly undesirable. Many are used for formatting, affecting only the appearance of rendered formulae (for example, the attributes `linebreak` and `indentalign` of the `mo` element). Others might have some slight semantic significance, but are very uncommon and usually not very important; we think these attributes should be removed. However, several exceptions exist. For instance, the element `mfrac` is used for fractions but its meaning changes with the attribute `linethickness` set to 0, which express a binomial coefficient. The attributes of the element `mfenced` are also important (see Listing 9). The attribute `mathvariant` can also influence formula semantics and therefore should be preserved in all possible elements. For example, the MIaS system makes use of this attribute so that hits with the assigned `mathvariant` font specifying the attribute are more relevant.

```
<mfrac>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> y </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
  <mrow>
    <mi> x </mi>
    <mphantom>
      <mo> + </mo>
      <mi> y </mi>
    </mphantom>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
</mfrac>
```

```
<mfrac>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> y </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
</mfrac>
```

Listing 8: Example of `<mphantom>` ommision

```
<mfrac linethickness="2"
       bevelled="true">
  <mi> a </mi>
  <mi> b </mi>
</mfrac>
```

```
<mfrac>
  <mi> a </mi>
  <mi> b </mi>
</mfrac>
```

Listing 9: Example of omission of unnecessary attributes in `mfrac`

3.2 Unifying Fences

There are two approaches to creating fenced formulae. One is more semantic and uses the `mfenced` element with the `open`, `close`, and `separator` attributes to describe delimiters and separators. The other places fence symbols directly within `mo` elements, and the fenced formula is enclosed in the `mrow` element to group the elements together. Although the first approach seems to be valid, we prefer the second one as it is more universal and allows easier conversion — e.g., converting addition to `mfenced` with attribute `separators` set to `+` would be invalid. As shown in Listing 10, `mfenced` elements are replaced by a more general `mrow` element, and fence and separator symbols are added as `mo` elements. Fenced elements are further enclosed in an `mrow` element so it can be treated as a single expression when needed. We could also consider unifying the symbols used as separators/delimiters.

```
<mfenced open="[">

  <mi> x </mi>

  <mi> y </mi>

</mfenced>
```

```
<mrow>
  <mo> [ </mo>
  <mrow>
    <mi> x </mi>
    <mo> , </mo>
    <mi> y </mi>
  <mrow>
  <mo> ) </mo>
</mrow>
```

Listing 10: Two ways of writing interval $[x, y)$

3.3 Mrow Minimizing

The `mrow` element is used for grouping other elements. Its most common use case is to obtain a given correct number of child elements of some parent element (e.g. `mfrac` needs two child elements). We can determine unnecessary occurrences of `mrow` by summing the number of its child elements and its siblings with respect to the number of required elements for the parent element. Parents requiring only one child element actually accept any number of elements that are treated as if they are inferred within a single `mrow` element. Hence, the grouping element is redundant and can be removed. In any case, the impact of the transformations to any form of processing canonicalized notation must be taken into account and the structure of the formulae cannot be violated. For instance, after removing the `mfenced` enclosing element we ought to wrap the fenced formula with an `mrow` if it is not.

```
<msqrt>
  <mrow>
    <mo> - </mo>
    <mn> 1 </mn>
  </mrow>
</msqrt>
```

```
<msqrt>

  <mo> - </mo>
  <mn> 1 </mn>

</msqrt>
```

Listing 11: Example of `<mrow>` removal after optimization $\sqrt{-1}$

3.4 Sub-/Superscripts Handling

The `msubsup` element used for attaching subscript and superscript to another element at the same time is redundant — the same thing can be expressed as

a combination of `msub` and `msup` elements. The order of the elements is important. When both elements are used, we prefer to place `msub` within `msup` (see Listing 12) because a subscript is usually more closely related to the base expression. A similar problem and solution is related to the elements triad of `munder`, `mover`, and `munderover`. Both `msubsup` and `munderover` can be used for limits of integration or bounds of summations; therefore, we should use only one canonical representant.

```
<msubsup>
  <mi> x </mi>

  <mn> 1 </mn>

  <mn> 2 </mn>
</msubsup>
```

```
<msup>
  <msub>
    <mi> x </mi>
    <mn> 1 </mn>
  </msub>
  <mn> 2 </mn>
</msup>
```

Listing 12: Two ways of expressing x_1^2

3.5 Applying Functions

There are many ways to express functions. Entity `&#x2061;` (function application) should be used but we cannot rely on that, so we suggest removing this operator for the purpose of unification. The opposite approach — adding the function application operator where it was omitted — could be rather tricky and could lead to ambiguities. The name of the function should occur in the `mi` element but it also can be considered as an operator and be placed in the `mo` element. The arguments of a function can be fenced with parentheses or an `mfenced` element or both. We chose canonical representation without an entity, with `mrow` and parentheses (see Listing 14). Other ambiguities can be caused by different invisible operators. For example, two identifiers in a subscript with no operator usually means multiplication but it can mean separation too.

4 Design Considerations

The design and implementation decisions of the canonicalization application depend on the purpose of new canonicalizer. Even though the use of the math content by different tools might be similar, the experience shows that we hardly could 'fit one size' for all applications. Thus the main design imperative is the modularity, simplicity, extensibility and flexibility, so that the canonicalizer might be easily modified when the need of the applications change. With different data the canonicalizer might change even for different types of math-aware search.

```
<mi> f </mi>
<mo> &#x2061; </mo>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

```
<mi> f </mi>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

Listing 13: Using or not using the operator for function application

```
<mi> sin </mi>

<mo> &#x2061; </mo>

<mi> x </mi>
```

```
<mi>sin</mi>
<mrow>
  <mo>(</mo>
  <mi>x</mi>
  <mo>)</mo>
</mrow>
```

Listing 14: Adding parentheses to sine function argument

Examples in subsections of previous section form set of modules that do the necessary MathML tree transformations as recursive procedures on MathML trees.

According to the expected size of the input data set, effectiveness, the speed of the canonicalization application is also a critical parameter — in our MREC [8] corpora there is 168,000,000 formulae to canonicalize. Thus, use of standard XSL transformations does not seem to be appropriate, for example, as UMCL example showed.

Another key decision is handling of invalid input MathML and question of valid MathML on the output as mentioned in Section 3.

As the (Web)MIaS system as well as other core parts of EuDML system (Lucene) do use the Java platform is seems to be natural to use Java also for the implementation of canonicalization application.

5 Conclusions and Future Work

We consider MathML canonicalization important for proper functioning of several math-aware applications that handle documents in DMLs. We have defined the problems and enumerated the most important use cases as modules of newly designed canonicalizer.

We are currently working on finishing the design and implementation of a first version of application that will be used for the task of math indexing in MIaS

system employed in EuDML project. By evaluation of this task we will verify our design decisions and plan to use it for another tools working with math fulltext data (semantic similarity tools as gensim [12]).

Acknowledgements This work was partially supported by the European Union through its Competitiveness and Innovation Programme (Information and Communication Technologies Policy Support Programme, 'Open access to scientific information', Grant Agreement No. 250503, a project of the European Digital Mathematics Library, EuDML).

References

1. Archambault, D., Berger, F., Moço, V.: Overview of the "Universal Maths Conversion Library". In: Pruski, A., Knops, H. (eds.) Assistive Technology: From Virtuality to Reality: Proceedings of 8th European Conference for the Advancement of Assistive Technology in Europe AAATE 2005, Lille, France. pp. 256–260. IOS Press, Amsterdam, The Netherlands (Sep 2005)
2. Archambault, D., Moço, V.: Canonical MathML to Simplify Conversion of MathML to Braille Mathematical Notations. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) Computers Helping People with Special Needs, Lecture Notes in Computer Science, vol. 4061, pp. 1191–1198. Springer Berlin / Heidelberg (2006), http://dx.doi.org/10.1007/11788713_172
3. Baker, J.B., Sexton, A.P., Sorge, V.: A linear grammar approach to mathematical formula recognition from PDF. In: Proceedings of the Conferences in Intelligent Computer Mathematics, CICM 2009. LNAI, vol. 5625, pp. 201–216. Springer (2009)
4. Baker, J.B., Sexton, A.P., Sorge, V.: Towards reverse engineering of PDF documents. In: Sojka, P., Bouche, T. (eds.) Towards a Digital Mathematics Library, DML 2011. pp. 65–75. Masaryk University Press, Bertinoro, Italy (July 2011), http://hdl.handle.net/10338.dmlcz/702603
5. Borbinha, J., Bouche, T., Nowiński, A., Sojka, P.: Project EuDML—A First Year Demonstration. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011. Lecture Notes in Artificial Intelligence, LNAI, vol. 6824, pp. 281–284. Springer-Verlag, Berlin, Germany (Jul 2011), http://dx.doi.org/10.1007/978-3-642-22673-1_21
6. Grimm, J.: Producing MathML with Tralics. In: Sojka [13], pp. 105–117, http://dml.cz/dmlcz/702579
7. Jarmar, M.: Conversion of Mathematical Documents into Braille. Master's thesis, Faculty of Informatics (Jan 2012), https://is.muni.cz/th/172981/fi_m/?lang=en
8. Líška, M., Sojka, P., Růžička, M., Mravec, P.: Web Interface and Collection for Mathematical Retrieval. In: Sojka, P., Bouche, T. (eds.) Proceedings of DML 2011. pp. 77–84. Masaryk University, Bertinoro, Italy (Jul 2011), http://www.fi.muni.cz/~sojka/dml-2011-program.html
9. Maplesoft, a division of Waterloo Maple Inc.: MathML – Maple Help (Apr 2012), http://www.maplesoft.com/support/help/Maple/view.aspx?path=MathML
10. Munavalli, R., Miner, R.: MathFind: A Math-Aware Search Engine. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 735–735. SIGIR '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1148170.1148348

11. Řehůřek, R., Sojka, P.: Automated Classification and Categorization of Mathematical Knowledge. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) Intelligent Computer Mathematics—Proceedings of 7th International Conference on Mathematical Knowledge Management MKM 2008. Lecture Notes in Computer Science LNCS/LNAI, vol. 5144, pp. 543–557. Springer-Verlag, Berlin, Heidelberg (Jul 2008)
12. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en, software available at http://nlp.fi.muni.cz/projekty/gensim
13. Sojka, P. (ed.): Towards a Digital Mathematics Library. Masaryk University, Paris, France (Jul 2010), http://www.fi.muni.cz/~sojka/dml-2010-program.html
14. Sojka, P., Líška, M.: Indexing and Searching Mathematics in Digital Libraries (Mar 2011), submitted to MKM 2011
15. Sojka, P., Líška, M.: Indexing and Searching Mathematics in Digital Libraries – Architecture, Design and Scalability Issues. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011. Lecture Notes in Artificial Intelligence, LNAI, vol. 6824, pp. 228–243. Springer-Verlag, Berlin, Germany (Jul 2011), http://dx.doi.org/10.1007/978-3-642-22673-1_16
16. Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., Miller, B.: Transforming Large Collections of Scientific Publications to XML. Mathematics in Computer Science 3, 299–307 (2010), http://dx.doi.org/10.1007/s11786-010-0024-7
17. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY — An integrated OCR system for mathematical documents. In: Vanoirbeek, C., Roisin, C., Munson, E. (eds.) Proceedings of ACM Symposium on Document Engineering 2003. pp. 95–104. ACM, Grenoble, France (2003)
18. Sylwestrzak, W., Borbinha, J., Bouche, T., Nowiński, A., Sojka, P.: EuDML—Towards the European Digital Mathematics Library. In: Sojka [13], pp. 11–24, http://dml.cz/dmlcz/702569
19. The LaTeXML project: The LaTeXML Developer Portal (Apr 2012), https://trac.mathweb.org/LaTeXML/
20. The MathWorks, Inc.: MuPAD – Matlab (May 2012), http://www.mathworks.com/discovery/mupad.html
21. Watt, S.M.: Mathematical Document Classification via Symbol Frequency Analysis. In: Sojka, P. (ed.) Towards Digital Mathematics Library—Proceedings of DML 2008. pp. 29–40. Masaryk University, Birmingham, UK (Jul 2008), http://www.fi.muni.cz/~sojka/dml-2008-program.xhtml
22. Wolfram: Mathematica Import/Export Format : MathML (Apr 2012), http://reference.wolfram.com/mathematica/ref/format/MathML.html
23. Wolfram Alpha LLC: Wolfram Alpha (Apr 2012), http://www.wolframalpha.com/