MathTools: An Open API for Convenient MathML Handling

André Greiner-Petter[1(✉)], Moritz Schubotz[1], Howard S. Cohl[2], and Bela Gipp[1]

[1] Department of Computer and Information Science, University of Konstanz, Box 76, 78464 Konstanz, Germany
{andre.greiner-petter,moritz.schubotz,bela.gipp}@uni-konstanz.de
[2] Applied and Computational Mathematics Division, National Institute of Standards and Technology, Mission Viejo, CA 92694, USA
howard.cohl@nist.gov

Abstract. Mathematical formulae carry complex and essential semantic information in a variety of formats. Accessing this information with different systems requires a standardized machine-readable format that is capable of encoding presentational and semantic information. Even though MathML is an official recommendation by W3C and an ISO standard for representing mathematical expressions, we could identify only very few systems which use the full descriptiveness of MathML. MathML's high complexity results in a steep learning curve for novice users. We hypothesize that this complexity is the reason why many community-driven projects refrain from using MathML, and instead develop problem-specific data formats for their purposes. We provide a user-friendly, open-source application programming interface for controlling MathML data. Our API allows one to create, manipulate, and efficiently access commonly needed information in presentation and content MathML. Our interface also provides tools for calculating differences and similarities between MathML expressions. The API also allows one to determine the distance between expressions using different similarity measures. In addition, we provide adapters for numerous conversion tools and the canonicalization project. Our toolkit facilitates processing of mathematics for digital libraries without the need to obtain XML expertise.

Keywords: MathML · API · Toolkit · Java

1 Introduction

MathML has the ability to represent presentational and content information. Several optional features and markup options make MathML a highly versatile, but complex format. MathML's ability to mix presentation and content markup for an expression (hereafter called fully descriptive MathML) makes MathML increasingly important for mathematical digital libraries. For example, Listing 1 shows a simple example of a cross-referenced parallel markup MathML document generated from the LaTeX string `\frac{a}{b}`.

F. Rabe et al. (Eds.): CICM 2018, LNAI 11006, pp. 104–110, 2018.
https://doi.org/10.1007/978-3-319-96812-4_9

Listing 1. Parallel markup MathML with examples of cross-references.

```
<math><semantics>
  <mfrac id="p.2" xref="c.1">
  <mi id="p.1" xref="c.2">a</mi>
  <mi id="p.3" xref="c.3">b</mi></mfrac>
<annotation-xml encoding="MathML-Content"><apply>
  <divide id="c.1" xref="p.2" />
  <ci id="c.2" xref="p.1">a</ci>
  <ci id="c.3" xref="p.3">b</ci></apply></annotation-xml>
<annotation encoding="application/x-tex">\frac{a}{b}</annotation>
</semantics></math>
```

Although MathML is an official recommendation of the World Wide Web Consortium since 1998, has been an ISO standard (ISO/IEC 40314) since 2015, and is also part of HTML5, it is still a rarely used format. For example, the prominent browser Microsoft Internet Explorer does not support MathML. Google Chrome supported MathML only in version 24 and dropped it again in newer versions in favor of MathJax[1]. Furthermore, we were only able to identify few databases that use fully descriptive MathML. Most databases use basic MathML instead, such as the DLMF [4] which only provides presentational MathML. Numerous tools that process math allow for MathML as an input or export format. Most of these tools avoid MathML as an internal data representation. Instead, many systems create their own problem-specific data representation and implement custom tools to parse MathML for internal processing. We hypothesize that the complexity of MathML and the steep learning curve required to master MathML are the main reasons why community-driven projects usually avoid the use of content MathML in their mathematical databases. This workaround causes problems in regard to reusability, restricts applicability, and decreases efficiency for processing mathematics.

During our research of a MathML benchmark [17], we realized that parsing MathML data is an error-prone endeavor. Even small changes, such as missing default namespaces, can cause the parsing process to fail.

With MathTools, we provide a collection of tools that overcome the issues of complexity and simplifies access to MathML data. MathTools also contains an open API for easy access to useful services that we used in several of our previous projects, such as for similarity calculations or for LaTeX to MathML conversion tools [3,9,10,12,15–19]. The tools we have developed are able to parse fully descriptive MathML and to a certain degree, invalid MathML. Furthermore, we provide easy access to several useful features implemented in MathML related projects. These features include similarity metrics, distance calculations, and Java imports for state-of-the-art conversion tools.

2 Related Work

Many digital libraries avoid using MathML and process mathematics using custom internal formats instead. The conversion tool LaTeXML [11] is able to

[1] https://bugs.chromium.org/p/chromium/issues/detail?id=152430#c43.

generate presentation and content MathML using LaTeX input. Instead of processing the input directly to MathML, LaTeXML uses a customized XML format, and creates MathML in a post-processing step. The Speech Rule Engine (SRE) implemented in MathJax [2] is able to translate presentation MathML expressions to speech. Translations of the SRE use a set of rules that are applied to a nonstandard tree representation of the original MathML input. Computer Algebra Systems (CAS) allow for complex computations of mathematical expressions. These systems usually create their own internal data format, but typically allow for MathML export. For example, the CAS MAPLE[2] parses inputs to a directed acyclic graph (DAG) presentation [1, Chap. 2], whereas Mathematica uses a custom tree representation [20, Chap. 33].

There are other tools that internally use MathML in combination with customized implementations for parsing MathML. Examples such tools include MathMLCan [5], a canonicalization tool for MathML data; the Visualization of Mathematical Expression Trees (VMEXT) [19] for MathML data; and dependency graphs [7] for analyzing relationships between math expressions.

3 MathTools

We provide MathTools, a collection of tools for convenient handling of MathML data and an open Java API to access useful services related to MathML data. MathTools allows one to parse, manipulate, and analyze fully descriptive MathML data conveniently. We haven chosen Java to realize MathTools, because the majority of related work is also implemented in Java. Java is also the most frequently used programming language. The source of this project is publicly available on GitHub[3] and on maven-central under an Apache 2 license.

The project is organized in multiple modules, of which the **core** module provides the main functionality. Figure 1 illustrates possible workflows for processing the input and shows the interaction of different modules. The **core** module is designed for parsing, loading, storing, and manipulating presentation and content MathML data. The **core** also provides helper functions for easy access to specific information. Examples of supported operations include splitting presentation and content trees; accessing original TeX data if available; and extracting all of the identifiers, e.g., `mi` or `ci` elements from the data. The **core** module also allows one to 'clean' MathML of unwanted features. Such features include cross-references or entire content subtrees. This functionality is particularly helpful for obtaining simple MathML for testing and learning purposes.

[2] The mention of specific products, trademarks, or brand names is for purposes of identification only. Such mention is not to be interpreted in any way as an endorsement or certification of such products or brands by the National Institute of Standards and Technology, nor does it imply that the products so identified are necessarily the best available for the purpose. All trademarks mentioned herein belong to their respective owners.

[3] https://github.com/ag-gipp/MathMLTools.

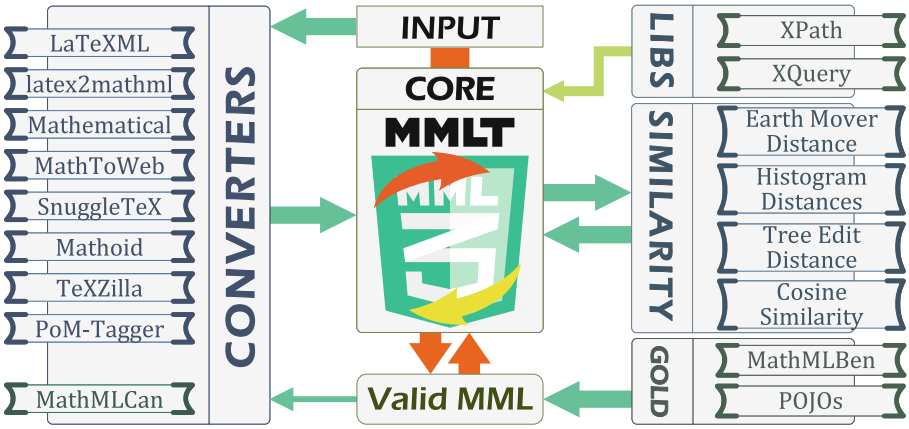

Fig. 1. The pipeline schema of MathTools and the modules. The orange arrow indicates the general workflow for processing a MathML input to a valid MathML output. The converters also allows for mathematical LaTeX input, while the MathML feature requires valid MathML input. The gold module provides valid MathML without the core module. Distances and similarities can be calculated using the similarity module. Single elements or subtrees from valid MathML can be accessed through the core module. (Color figure online)

We equipped MathTools with several extra modules, which we consider useful for typical use cases related to processing MathML data. MathTools contains the following modules.

Gold: is part of the MathMLBen project [17], whose purpose is to provide a comfortable interface to MathML gold standards within Java. The module uses Plain Old Java Objects (POJOs) for representing entries of the gold standard. The **gold** module can be used to implement benchmarks that use MathML.

Converters: is an outgrowth of the MathMLBen project and provides fast access to state-of-the-art conversion tools. We implemented a Java interface for each supported tool and the MathMLCan [5] that allows canocalization of the tool's outputs. The module allows one to: (1) embed uniform MathML data into translation workflows; (2) conveniently change the translation engines; and (3) add new conversion tools to the API.

Libraries: is a collection of reliable XPath and XQuery expressions for accessing elements in MathML data. Due to the XML structure of MathML documents, MathTools uses data query languages to access specific elements. The **library** module can be used to reliably query data from MathML documents independently of programming languages.

Similarity: implements several distance and similarity measures for MathML documents. Comparing MathML data is a common task for many mathematical information retrieval systems, e.g., for mathematical search engines and for plagiarism detection systems that analyze mathematical expressions, as well as for

many evaluation and benchmark scenarios. All measures included in the module, except the tree edit distance, produce histograms for MathML elements using the names of the elements and their accumulated frequency of occurrence. Frequency histograms of MathML elements have been successfully applied as part of plagiarism detection methods to determine the similarity of mathematical expressions, see [9,10] for a more detailed explanation. We implemented the following similarity measures.

- **Histogram Distance:** calculates the absolute and relative differences of histograms formed from MathML data. A small histogram distance indicates semantic similarity between MathML expressions.
- **Tree Edit Distance:** calculates the number of changes that must be performed to transform one MathML tree into another. The user can control the weights for insertions, deletions, and renaming of elements. We use an implementation of RTED [13] to perform these calculations. Tree edit distances are helpful for detecting structural changes of MathML trees.
- **Earth Mover Distance (EMD):** is originally a distance measure for comparing probability distributions. The measure models a cost function for changing one distribution to another. In our case, the probability distribution is the histogram of the MathML data. Our calculations are performed using a Java implementation of [14]. EMD is widely used in multimedia information retrieval and for pattern recognition algorithms.
- **Cosine Similarity:** is a distance measure between two non-zero vectors. In our case, the vectors are represented by the histogram of the MathML data.

Note that EMD and Cosine Similarity can be used for entire documents. In this case, the histograms are an accumulation of all MathML expressions in the document. In this scenario, EMD and Cosine Similarity metrics obtain semantic similarities between documents.

4 Conclusion and Future Work

The MathTools project is a unified toolkit to facilitate working with MathML. By using MathTools, other researchers will no longer need to develop their own data representations, nor will they have to deal with the full complexity of XML. The developed tools are actively being used in our own projects, such as MathMLBen [17], VMEXT [19], the Mathematical Language Processing (MLP) Project [12], and HyPlag [6,8,10] for mathematical plagiarism detection. Our hope is that the tools presented in this paper will help others to realize MathML related projects in the future.

Our goal is to actively maintain and extend MathTools. Plans for current and future work include enlarging the libraries of XPath and XQuery expressions and providing transformation tools for MathML trees. Furthermore, we are currently working on semantic enhancements of MathML through the use of WikiData (as proposed in the MathMLBen project). This semantic enhanced MathML data

can be used to calculate semantic distances through the calculation of distances between WikiData links. We plan to maintain the current API for the foreseeable future.

Acknowledgements. We would like to thank Felix Hamborg, Vincent Stange, Jimmy Li, Telmo Menezes, and Michael Kramer for contributing to the MathTools project. We are also indebted to Akiko Aizawa for her advice and for hosting the first two authors as visiting researchers at the National Institute of Informatics (NII) in Tokyo. This work was supported by the FITWeltweit program of the German Academic Exchange Service (DAAD) as well as by the German Research Foundation (DFG) through grant no. GI 1259/1.

References

1. Bernardin, L., et al.: Maple 2016 Programming Guide. Maplesoft, a division of Waterloo Maple Inc. (2016). ISBN 978-1-926902-46-3
2. Cervone, D., Krautzberger, P., Sorge, V.: Towards universal rendering in MathJax. In: Proceedings of the W4A 2016. ACM Press (2016). https://doi.org/10.1145/2899475.2899494
3. Cohl, H.S., et al.: Growing the digital repository of mathematical formulae with generic LaTeX sources. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V. (eds.) CICM 2015. LNCS (LNAI), vol. 9150, pp. 280–287. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20615-8_18
4. Olver, F.W.J., et al. (eds.): NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/. Release 1.0.19 of 22 June 2018
5. Formánek, D., et al.: Normalization of digital mathematics library content. In: Davenport, J., et al. (eds.) Proceeding of OpenMath/MathUI/CICM-WiP, vol. 921, Bremen, 9–13 July 2012
6. Gipp, B., et al.: Web-based demonstration of semantic similarity detection using citation pattern visualization for a cross language plagiarism case. In: ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems, vol. 2, Lisbon, 27–30 April 2014. https://doi.org/10.5220/0004985406770683
7. Kristianto, G.Y., Topic, G., Aizawa, A.: Utilizing dependency relationships between math expressions in math IR. Inf. Retr. J. **20**(2) (2017). https://doi.org/10.1007/s10791-017-9296-8
8. Meuschke, N., Gipp, B., Breitinger, C.: CitePlag: a citation-based plagiarism detection system prototype. In: Proceedings of the 5th International Plagiarism Conference, Newcastle upon Tyne (2012)
9. Meuschke, N., et al.: Analyzing mathematical content to detect academic plagiarism. In: Lim, E., et al. (eds.) Proceedings of the ACM CIKM. ACM (2017). https://doi.org/10.1145/3132847.3133144
10. Meuschke, N., et al.: HyPlag: a hybrid approach to academic plagiarism detection. In: Proceedings of the SIGIR, Ann Arbor (2018)
11. Miller, B.R.: `LaTeXML`: A LaTeX to XML/HTML/MathML Converter. http://dlmf.nist.gov/LaTeXML/. Accessed June 2018
12. Pagel, R., Schubotz, M.: Mathematical language processing project. In: England, M., et al. (eds.) Proceedings of the MathUI/OpenMath/ThEdu/CICM-WiP, vol. 1186 (2014)

13. Pawlik, M., Augsten, N.: RTED: a robust algorithm for the tree edit distance. In: CoRR abs/1201.0230 (2012). arXiv: 1201.0230
14. Pele, O., Werman, M.: Fast and robust earth mover's distances. In: IEEE 12th ICCV, Kyoto. IEEE Computer Society, 27 September–4 October 2009. https://doi.org/10.1109/ICCV.2009.5459199
15. Schubotz, M.: Augmenting mathematical formulae for more effective querying & efficient presentation. Ph.D. thesis, TU, Berlin (2017). https://doi.org/10.14279/depositonce-6034. ISBN 978-3-7450-6208-3
16. Schubotz, M., Krämer, L., Meuschke, N., Hamborg, F., Gipp, B.: Evaluating and improving the extraction of mathematical identifier definitions. In: Jones, G., et al. (eds.) CLEF 2017. LNCS, vol. 10456, pp. 82–94. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65813-1_7
17. Schubotz, M., et al.: Improving the representation and conversion of mathematical formulae by considering their textual context. In: Proceedings of the ACM/IEEE-CS JCDL, Fort Worth, June 2018. https://doi.org/10.1145/3197026.3197058
18. Schubotz, M., et al.: Semantification of identifiers in mathematics for better math information retrieval. In: Proceedings of the 39th International ACM SIGIR, Pisa. ACM (2016). https://doi.org/10.1145/2911451.2911503. ISBN 978-1-4503-4069-4
19. Schubotz, M., Meuschke, N., Hepp, T., Cohl, H.S., Gipp, B.: VMEXT: a visualization tool for mathematical expression trees. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 340–355. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_24
20. Wolfram, S.: An Elementary Introduction to the Wolfram Language, 2nd edn. Wolfram Media (2017). ISBN 978-1944183059