Improving the representation and conversion of mathematical formulae by considering their textual context*

Moritz Schubotz, André Greiner-Petter, Philipp Scharpf, Norman Meuschke, Howard S. Cohl, Bela Gipp

Abstract

Mathematical formulae represent complex semantic information in a concise form. Especially in Science, Technology, Engineering, and Mathematics, mathematical formulae are crucial for communicating information, e.g., in scientific papers, and to perform computations using computer algebra systems. Enabling computers to access the information encoded in mathematical formulae requires machine-readable formats that can represent both the presentation and content, i.e., the semantics, of formulae. Exchanging such information between systems additionally requires conversion methods for mathematical representation formats.

We analyze how the semantic enrichment of formulae improves the format conversion process and show that considering the textual context of formulae reduces the error rate of such conversions. Our main contributions are: (1) providing an openly available benchmark dataset for the mathematical format conversion task consisting of a newly created test collection, an extensive, manually curated gold standard and task-specific evaluation metrics; (2) performing a quantitative evaluation of state-of-the-art tools for mathematical format conversions; (3) presenting a new approach that considers the textual context of formulae to reduce the error rate for mathematical format conversions.

Our benchmark dataset facilitates future research on mathematical format conversions as well as research on many problems in mathematical information retrieval. Because we annotated and linked all components of formulae, e.g., identifiers, operators and other entities, to Wikidata entries, the gold standard can, for instance, be used to train methods for formula concept discovery and recognition. Such methods can then be applied to improve mathematical information retrieval systems, e.g., for semantic formula search, recommendation of mathematical content, or detection of mathematical plagiarism.

* A version of this paper was published at JCDL 2018: M. Schubotz et al., "Improving the Representation and Conversion of Mathematical Formulae by Considering their Textual Context", in Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL), Fort Worth, USA, 2018.

1 Introduction

In STEM disciplines, i.e., Science, Technology, Engineering, and Mathematics, mathematical formulae are ubiquitous and crucial for communicating information in documents such as scientific papers, and to perform computations in computer algebra systems (CAS). Mathematical formulae represent complex semantic information in a concise form that is independent of natural language. These characteristics make mathematical formulae particularly interesting features to be considered by information retrieval systems.

In the context of digital libraries, major information retrieval applications for mathematical formulae include search and recommender systems as well as systems that support humans in understanding and applying mathematical formulae, e.g., by visualizing mathematical functions or providing autocompletion and error correction functionality in typesetting and CAS.

However, the extensive, context-dependent polysemy and polymorphism of mathematical notation is a major challenge to exposing the knowledge encoded in mathematical formulae to such systems. The number of mathematical concepts, e.g., mathematical structures, relations and principles, is much larger than the set of mathematical symbols available to represent these concepts. Therefore, the meaning of mathematical symbols varies in different contexts, e.g., in different documents, and potentially even in the same context. Identical mathematical formulae, even in the same document, do not necessarily represent the same mathematical concepts. Identifiers are prime examples of mathematical polysemy. For instance, while the identifier E commonly denotes energy in physics, E commonly refers to expected value in statistics.

Polymorphism of mathematical symbols is another ubiquitous phenomenon of mathematical notation. For example, whether the operator $\cdot$ denotes scalar multiplication or vector multiplication depends on the type of the elements to which the operator is applied. In contrast to programming languages, which handle polymorphism by explicitly providing type information about objects to the compiler, e.g., to check and call methods offered by the specific objects, mathematical symbols mostly denote such type information only implicitly, so that they need to be inferred from the context.

Humans account for the inherent polysemy and polymorphism of mathematical notation by defining context-dependent meanings of mathematical symbols in the text that surrounds formulae, e.g.,

for identifiers, subscripts and superscripts, brackets, and invisible operators. Without such explanations, determining the meaning of symbols is challenging, even for mathematical experts. For example, reliably determining whether $[a, b]$ represents an interval or the commutator $[a, b] = ab - ba$ in ring theory requires information on whether [ ] represent the Dirac brackets.

Enabling computers to access the full information encoded in mathematical formulae mandates machine-readable representation formats that capture both the presentation, i.e., the notational symbols and their spacial arrangement, and the content, i.e., the semantics, of mathematical formulae. Likewise, exchanging mathematical formulae between applications, e.g., CAS, requires methods to convert and semantically enrich different representation formats. The Mathematical Markup Language (MathML) allows one to encode both presentation and content information in a standardized and extensible way (see section 3).

Despite the availability of MathML, most Digital Mathematical Libraries (DML) currently exclusively use presentation languages, such as TEX and LATEX to represent mathematical content. On the other hand, CAS, such as Maple, Mathematica and SageMath,[1] typically use representation formats that include more content information about mathematical formulae to enable computations. Conversion between representation formats entails many conceptual and technical challenges, which we describe in more detail in section 2. Despite the availability of numerous conversion tools, the inherent challenges of the conversion process result in a high error rate and often lossy conversion of mathematical formulae in different representation formats.

To advance research on mathematical format conversion, we make the following contributions, which we describe in the subsequent sections:

1. We provide an openly available benchmark dataset to evaluate tools for mathematical format conversion (cf. section 3). The dataset includes:
   - a new test collection covering diverse research areas in multiple STEM disciplines;
   - an extensive, manually curated gold standard that includes annotations for both presentation and content information of mathematical formulae;
   - tools to facilitate the future extension of the gold standard by visually supporting human annotators; and
   - metrics to quantitatively evaluate the quality of mathematical format conversions.
2. We perform an extensive, quantitative evaluation of state-of-the-art tools for mathematical format conversion and provide an automated evaluation framework to support easily rerunning the evaluation in future research (cf. section 4).
3. We propose a novel approach to mathematical format conversion (cf. section 5). The approach imitates the human sense-making process for mathematical content by analyzing the textual context of formulae for information that helps link symbols in formulae to a knowledge base, in our case Wikidata, to determine the semantics of formulae.

2 Background and related work

In the following, we use the Riemann hypothesis (1) as an example to discuss typical challenges of converting different representation formats of mathematical formulae:

$$\zeta(s) = 0 \Rightarrow \Re s = \frac{1}{2} \vee \Im s = 0. \tag{1}$$

We will focus on the representation of the formula in LATEX and in the format of the CAS Mathematica. LATEX is a common language for encoding the presentation of mathematical formulae. In contrast to LATEX, Mathematica's representation focuses on making formulae computable. Hence the content must be encoded, i.e., both the structure and the semantics of mathematical formulae must be taken into consideration.

In LATEX, the Riemann hypothesis can be expressed using the following string:

```
\zeta(s) = 0 \Rightarrow \Re s
= \frac12 \lor \Im s=0
```

In Mathematica, the Riemann hypothesis can be represented as:

```
Implies[Equal[Zeta[s], 0], Or[Equal[Re[s],
Rational[1, 2]], Equal[Im[s], 0]]]
```

The conversion between these two formats is challenging due to a range of conceptual and technical differences.

First, the grammars underlying the two representation formats differ greatly. LATEX uses the unrestricted grammar of the TEX typesetting system. The entire set of commands can be re-defined and extended at runtime, which means that TEX effectively allows its users to change every character used

[1] 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.

for the markup, including the \ character typically used to start commands. The high degree of freedom of the TEX grammar significantly complicates recognizing even the most basic tokens contained in mathematical formulae. In contrast to LATEX, CAS use a significantly more restrictive grammar consisting of a predefined set of keywords and set rules that govern the structure of expressions. For example, in Mathematica function arguments must always be enclosed in square brackets and separated by commas.

Second, the extensive differences in the grammars of the two languages are reflected in the resulting expression trees. Similar to parse trees in natural language, the syntactic rules of mathematical notation, such as operator precedence and function scope, determine a hierarchical structure for mathematical expressions that can be understood, represented, and processed as a tree. The mathematical expression trees of formulae consist of functions or operators and their arguments. We used nested square brackets to denote levels of the tree and Arabic numbers in a gray font to indicate individual tokens in the markup. For the LATEX representation of the Riemann hypothesis, the expression tree is:

$$\Big[\quad \zeta_{\mathrm{l}}^{1}\,({}_{\mathrm{l}}^{2}\,s_{\mathrm{l}}^{3}\,)_{\mathrm{l}}^{4} =_{\mathrm{l}}^{5} 0_{\mathrm{l}}^{6} \Rightarrow_{\mathrm{l}}^{7}$$

$$\Re_{\mathrm{l}}^{8}\, s_{\mathrm{l}}^{9} =_{\mathrm{l}}^{10} \Big[{}_{\vdots}^{11}\; 1_{\mathrm{l}}^{12}\, 2_{\mathrm{l}}^{13}\Big]\; {}_{\mathrm{e}}^{14} \vee_{\mathrm{l}}^{15} \Im_{\mathrm{l}}^{16}\, s_{\mathrm{l}}^{17} =_{\mathrm{l}}^{18} 0_{\mathrm{l}}^{19} \quad \Big].$$

The tree consists of 18 nodes, i.e., tokens, with a maximum depth of two (for the fraction command `\frac12`). The expression tree of the Mathematica expression consists of 16 tokens with a maximum depth of five:

$$\Big[{}_{\Rightarrow}^{20} \quad \Big[{}_{=}^{21}\Big[{}_{\zeta}^{22}\, s_{\mathrm{l}}^{23}\Big]\; 0_{\mathrm{n}}^{24}\Big]$$

$$\Big[{}_{\vee}^{25}\; \Big[{}_{=}^{26}\Big[{}_{\Re}^{27} s_{\mathrm{l}}^{28}\Big]\; \Big[{}_{\mathbb{Q}}^{29} 1_{\mathrm{n}}^{30}\, 2_{\mathrm{n}}^{31}\Big]\Big]\; \Big[{}_{=}^{32}\Big[{}_{\Im}^{33} s_{\mathrm{l}}^{34}\Big]\; 0_{\mathrm{n}}^{35}\Big]\Big]\Big].$$

The higher complexity of the Mathematica expression reflects that a CAS represents the content structure of the formula, which is deeply nested. In contrast, LATEX exclusively represents the presentational layout of the Riemann hypothesis, which is nearly linear.

For the given example of the Riemann hypothesis, finding alignments between the tokens in both representations and converting one representation into the other is possible. In fact, Mathematica and other CAS offer a direct import of TEX expressions, which we evaluate in section 4.

However, aside from technical obstacles, such as reliably determining tokens in TEX expressions, conceptual differences also prevent a successful conversion between presentation languages, such as TEX, and content languages. Even if there was only one generally accepted presentation language, e.g., a standardized TEX dialect, and only one generally accepted content language, e.g., a standardized input language for CAS, an accurate conversion between the representation formats could not be guaranteed.

The reason is that neither the presentation language nor the content language always provide all the information required to convert an expression to the respective language. This can be illustrated by the simple expression: $F(a + b) = Fa + Fb$. The inherent content ambiguity of F prevents a deterministic conversion from the presentation language to a content language. F might, for example, represent a number, a matrix, a linear function or even a symbol. Without additional information, a correct conversion to a content language is not guaranteed. On the other hand, the transformation from content language to presentation language often depends on the preferences of the author and the context. For example, authors sometimes change the presentation of a formula to focus on specific parts of the formula or to improve its readability.

Another obstacle to conversions between typical presentation languages and typical content languages, such as the formats of CAS, are the restricted set of functions and the simpler grammars that CAS offer. While TEX allows users to express the presentation of virtually all mathematical symbols, thus denoting any mathematical concept, CAS do not support all available mathematical functions or structures. A significant problem related to the discrepancy in the space of concepts expressible using presentation markup and the implementation of such concepts in CAS are branch cuts. Branch cuts are restrictions of the set of output values that CAS impose for functions that yield ambiguous, i.e., multiple mathematically permissible outputs. One example is the complex logarithm [14, eq. 4.2.1], which has an infinite set of permissible outputs resulting from the periodicity of its inverse function. To account for this circumstance, CAS typically restrict the set of permissible outputs by cutting the complex plane of permissible outputs. However, since the method of restricting the set of permissible outputs varies between systems, identical inputs can lead to drastically different results [5]. For example, multiple scientific publications address the problem of accounting for branch cuts when entering expressions in CAS, such as [7] for Maple.

Our review of obstacles to the conversion of representation formats for mathematical formulae

Listing 1: MathML representation of the Riemann hypothesis (1) (excerpt).

```
<math><semantics><mrow>...
  <mo id="5" xref="20">=</mo>
  <mn id="5" xref="21">0</mn>
  <mo id="7" xref="19">⇒</ci>...</mrow>
<annotation-xml encoding="MathML-Content">
  <apply><implies id="19" xref="7" />
  <apply><eq id="20" xref="5" />...
  <apply><csymbol id="21" xref="1"
    cd="wikidata">Q187235...
</annotation-xml></semantics></math>
```

highlights the need to store *both* presentation and content information to allow for reversible transformations. Mathematical representation formats that include presentation and content information can enable the reliable exchange of information between typesetting systems and CAS.

MathML offers standardized markup functionality for both presentation and content information. Moreover, the declarative MathML XML format is relatively easy to parse and allows for cross references between presentation language (PL) and content language (CL) elements. Listing 1 represents excerpts of the MathML markup for our example of the Riemann hypothesis (1). In this excerpt, the PL token 7 corresponds to the CL token 19, PL token 5 corresponds to CL token 20, and so forth.

Combined presentation and content formats, such as MathML, significantly improve the access to mathematical knowledge for users of digital libraries. For example, including content information of formulae can advance search and recommendation systems for mathematical content. The quality of these *mathematical information retrieval systems* crucially depends on the accuracy of the computed document-query and document-document similarities. Considering the content information of mathematical formulae can improve these computations by:

1. Enabling the consideration of mathematical equivalence as a similarity feature. Instead of exclusively analyzing presentation information as indexed, e.g., by considering the overlap in presentational tokens, content information allows modifying the query and the indexed information. For example, it would become possible to recognize that the expressions $a(\frac{b}{c} + \frac{d}{c})$ and $\frac{a(b+d)}{c}$ have a distance of zero.
2. Allowing the association of mathematical tokens with mathematical concepts. For example, linking identifiers, such as E, m, and c, to energy, mass, and speed of light, could enable searching for all formulae that combine all or a subset of the concepts.
3. Enabling the analysis of structural similarity. The availability of content information would enable the application of measures, such as derivatives of the tree edit distance, to discover structural similarity, e.g., using λ-calculus. This functionality could increase the capabilities of *math-based plagiarism detection systems* when it comes to identifying obfuscated instances of reused mathematical formulae [10].

Content information could also enable interactive support functions for consumers and producers of mathematical content. For example, readers of mathematical documents could be offered interactive computations and visualizations of formulae to accelerate the understanding of STEM documents. Authors of mathematical documents could benefit from automated editing suggestions, such as auto-completion, reference suggestions, and sanity checks, e.g., type and definiteness checking, similar to the functionality of word processors for natural language texts.

Related work

A variety of tools exists to convert format representations of mathematical formulae. However, to our knowledge, Kohlhase et al. [26] presented the only study evaluating the conversion quality of tools. Many of the tools evaluated in that study are no longer available or out of date. Watt [27] presents a strategy to preserve formula semantics in TEX to MathML conversions. His approach relies on encoding the semantics in custom TEX macros rather than to expand the macros. Padovani [15] discusses the roles of MathML and TEX elements for managing large repositories of mathematical knowledge. Nghiem et al. [13] used statistical machine translation to convert presentation to content language. However, they do not consider the textual context of formulae. We will present detailed descriptions and evaluation results for specific conversion approaches in section 4.

Youssef [28] addressed the semantic enrichment of mathematical formulae in presentation language. He developed an automated tagger that parses LATEX formulae and annotates recognized tokens very similarly to part-of-speech (POS) taggers for natural language. Their tagger currently uses a predefined, context-independent dictionary to identify and annotate formula components. Schubotz et al. [19, 20]

proposed an approach to semantically enrich formulae by analyzing their textual context for the definitions of identifiers.

With their 'math in the middle' approach, Dehaye et al. [6] envision an entirely different approach to exchanging machine readable mathematical expressions. In their vision, independent and enclosed virtual research environments use a standardized format for mathematics to transfer mathematical expressions and numerical results between different systems.

For an extensive review of format conversion and retrieval approaches for mathematical formulae, refer to [18, Chapter 2].

3 Benchmarking MathML

This section presents MathMLben — a benchmark dataset for measuring the quality of MathML markup of mathematical formulae appearing in a textual context. MathMLben is an improvement of the gold standard provided by Schubotz et al. [24]. The dataset considers recent discussions of the International Mathematical Knowledge of Trust (`imkt.org`) working group, in particular the idea of a 'Semantic Capture Language' [9], which makes the gold standard more robust and easily accessible. MathMLben:

- allows comparisons to prior works;
- covers a wide range of research areas in STEM literature;
- provides references to manually annotated and corrected MathML items that are compliant with the MathML standard;
- is easy to modify and extend, i.e., by external collaborators;
- includes default distance measures; and
- facilitates the development of converters and tools.

In section 3.1, we present the test collection included in MathMLben. In section 3.2, we present the encoding guidelines for the human assessors and describe the tools we developed to support assessors in creating the gold standard dataset. In section 3.3, we describe the similarity measures used to assess markup quality.

3.1 Collection

Our test collection contains 305 formulae (more precisely, mathematical expressions ranging from individual symbols to complex multi-line formulae) and the documents in which they appear.

Expressions 1 to 100 correspond to the search targets used for the 'National Institute of Informatics Testbeds and Community for Information access Research Project' (NTCIR) 11 Math Wikipedia Task [24]. This list of formulae has been used for formula search and content enrichment tasks by at least 7 different research institutions. The formulae were randomly sampled from Wikipedia and include expressions with incorrect presentation markup.

Expressions 101 to 200 are random samples taken from the NIST Digital Library of Mathematical Functions (DLMF) [14]. The DLMF website contains 9,897 labeled formulae created from semantic LaTeX source files [3, 4]. In contrast to the examples from Wikipedia, all these formulae are from the mathematics research field and exhibit high quality presentation markup. The formulae were curated by renowned mathematicians and the editorial board keeps improving the quality of the markup of the formulae.[2] Sometimes, a labeled formula contains multiple equations. In such cases, we randomly chose one of the equations.

Expressions 201 to 305 were chosen from the queries of the NTCIR arXiv and NTCIR-12 Wikipedia datasets. 70% of these queries originate from the arXiv [1] and 30% from a Wikipedia dump.

All data are openly available for research purposes and can be obtained from `mathmlben.wmflabs.org`.[3]

3.2 Gold standard

We provide explicit markup with universal, context-independent symbols in content MathML. Since the symbols from the default content dictionary (CD) of MathML[4] alone were insufficient to cover the range of semantics in our collection, we added the Wikidata content dictionary [17]. As a result, we could refer to all Wikidata items as symbols in a content tree. This approach has several advantages. Descriptions and labels are available in many languages. Some symbols even have external identifiers, e.g., from the Wolfram Functions Site, or from StackExchange topics. All symbols are linked to Wikipedia articles, which offer extensive human-readable descriptions. Finally, symbols have relations to other Wikidata items, which opens a range of new research opportunities, e.g., for improving the taxonomic distance measure [25].

Our Wikidata-enhanced, yet standard-compliant, MathML markup facilitates the manual creation of content markup. To further support human assessors in creating content annotations, we extended the VMEXT visualization tool [21] to develop a visual support tool for creating and editing the MathMLben gold standard.

[2] `dlmf.nist.gov/about/staff`
[3] Visit `mathmlben.wmflabs.org/about` for a user guide.
[4] `www.openmath.org/cd`

Table 1: Special content symbols added to LaTeXML for the creation of the gold standard.

No	rendering	meaning	example ID
1	$[x, y]$	commutator	91
2	$x_{\tilde{z}}^{y}$	tensor	43, 208, 226
3	$x^{\dagger}$	adjoint	224, 277
4	x'	transformation	20
5	x°	degree	20
6	$x^{(dim)}$	contraction	225

For each formula, we saved the source document written in different dialects of LaTeX and converted it into content MathML with parallel markup using LaTeXML [11, 8]. LaTeXML is a Perl program that converts LaTeX documents to XML and HTML. We chose LaTeXML because it is the only tool that supports our semantic macro set. We manually annotated our dataset, generated the MathML representation, manually corrected errors in the MathML, and linked the identifiers to Wikidata concept entries whenever possible. Alternatively, one could initially generate MathML using a CAS and then manually improve the markup.

Since there is no generally accepted definition of expression trees, we made several design decisions to create semantic representations of the formulae in our dataset using MathML trees. In some cases, we created new macros to be able to create a MathML tree for our purposes using LaTeXML.[5] Table 1 lists the newly created macros. Hereafter, we explain our decisions and give examples of formulae in our dataset that were affected by the decisions.

- did not assign Wikidata items to basic mathematical identifiers and functions like `factorial`, `\log`, `\exp`, `\times`, `\pi`. Instead, we left these annotations to the DLMF LaTeX macros, because they represent the mathematical concept by linking to the definition in the DLMF and LaTeXML creates valid and accurate content MathML for these macros [GoldID 3, 11, 19, ...];
- split up indices and labels of elements as child nodes of the element. For example, we represent `i` as a child node of `p` in `p_i` [GoldID 29, 36, 43, ...];
- create a special macro to represent tensors, such as for $T_{\alpha\beta}$ [GoldID 43], to represent upper and lower indices as child nodes (see table 1);
- create a macro for dimensions of tensor contractions [GoldID 225], e.g., to distinguish the three dimensional contraction of the metric tensor in $g^{(3)}$ from a power function (see table 1);
- chose one subexpression randomly if the original expression contained lists of expressions [GoldID 278];
- remove equation labels, as they are not part of the formula itself. For example, in
$$E = mc^2, \qquad (\star)$$
the $(\star)$ is the ignored label;
- remove operations applied to entire equations, e.g., applying the modulus. In such cases, we interpreted the modulus as a constraint of the equation [GoldID 177];
- use additional macros (see table 1) to interpret complex conjugations, transformation signs, and degree-symbols as functional operations (identifier is a child node of the operation symbol), e.g., `*` or `\dagger` for complex conjugations [GoldID 224, 277], `S'` for transformations [GoldID 20], `30^\circ` for thirty degrees [GoldID 30];
- for formulae with multiple cases, render each case as a separate branch [GoldID 49];
- render variables that are part of separate branches in bracket notation. We implemented the Dirac Bracket commutator [ ] (we omitted the index `_\text{DB}`) and an anticommutator { } by defining new macros (see table 1). Thus, there is a distinction between a (ring) commutator
```
[a,b] = ab - ba
```
and an anticommutator
```
{a,b} = ab + ba,
```
without further annotation of Dirac or Poisson brackets [GoldID 91];
- use the command `\operatorname{}` for multi-character identifiers or operators [GoldID 22]. This markup is necessary because most of the LaTeX parsers, including LaTeXML, interpret multi-character expressions as multiplications of the characters. In general, this interpretation is correct, since it is inconvenient to use multi-character identifiers [2].

Some of these design decisions are debatable. For example, introducing a new macro, such as `\identifiername{}`, to distinguish between multi-character identifiers and operators might be advantageous to our approach. However, introducing many highly specialized macros is likely not a viable approach. A borderline example of this problem is Δx [GoldID 280]. Formulae of this form could be annotated as `\operatorname{}`, `\identifiername{}`

[5] `dlmf.nist.gov/LaTeXML/manual/customization/customization.latexml.html#SS1.SSS0.Px1`

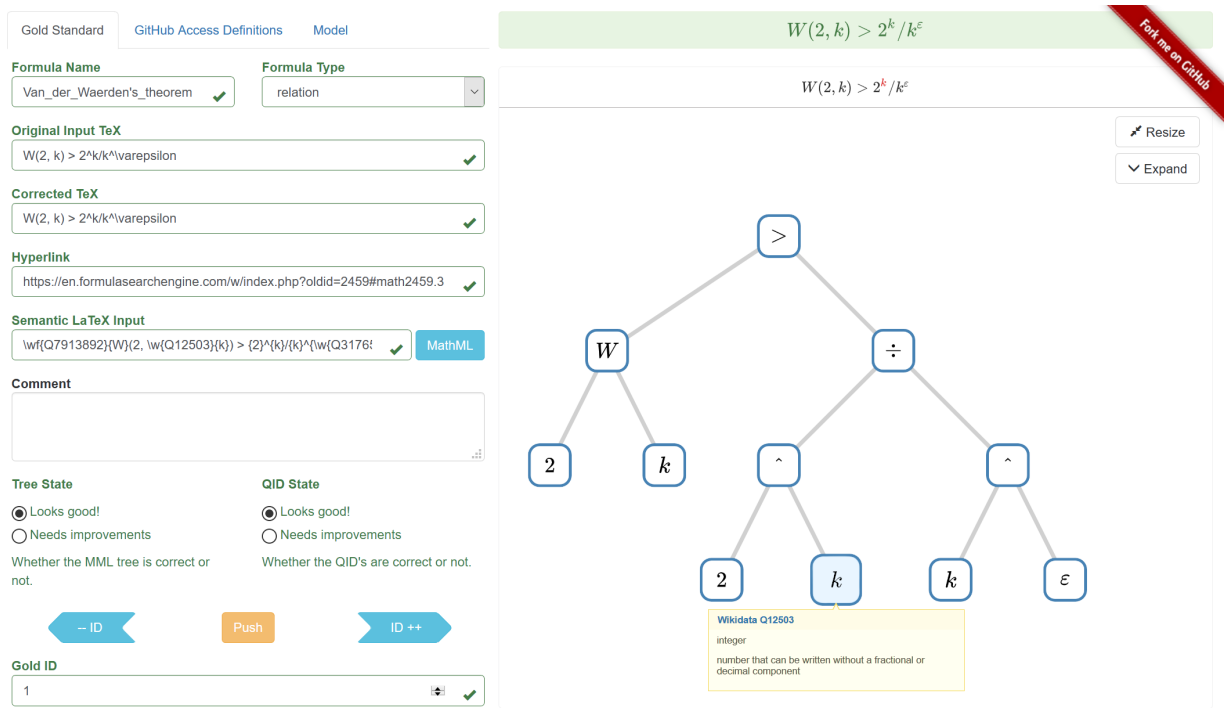

Figure 1: Graphical user interface to support the creation of our gold standard. The interface provides several TEX input fields (left) and a mathematical expression tree rendered by the VMEXT visualization tool (right).

or more generally as `\expressionname{}`. We interpret Δ as a difference applied to a variable, and render the expression as a function call.

Similar cases of overfeeding the dataset with highly specialized macros are bracket notations. For example, the bracket (Dirac) notation, e.g., [GoldID 209], is mainly used in quantum physics. The angle brackets for the Dirac notation, $\langle$ and $\rangle$, and a vertical bar | is already interpreted correctly as "latexml — quantum-operator-product". However, a more precise distinction between a twofold scalar product, e.g., $\langle a|b\rangle$, and a threefold expectation value, e.g., $\langle a|A|a\rangle$, might become necessary in some scenarios to distinguish between matrix elements and a scalar product.

We developed a Web application to create and cultivate the gold standard entries, which is available at `mathmlben.wmflabs.org`. The Graphical User Interface (GUI) provides the following information for each GoldID entry.

- **Formula Name:** name of the formula (optional)
- **Formula Type:** one of *definition*, *equation*, *relation* or *General Formula* (if none of the previous names fit)
- **Original Input TEX:** the LATEX expression as extracted from the source
- **Corrected TEX:** the manually corrected LATEX expression
- **Hyperlink:** hyperlink to the position of the formula in the source
- **Semantic LATEX Input:** manually created semantic version of the corrected LATEX field. This entry is used to generate our MathML with Wikidata annotations.
- **Preview of Corrected LATEX:** preview of the corrected LATEX input field rendered as SVG in real time using Mathoid [23], a service to generate SVG and MathML from LATEX input. It is shown in the top right corner of the GUI.
- **VMEXT Preview:** rendering of the expression tree based on the content MathML. The symbol in each node is associated with the symbol in the cross-referenced presentation markup.

Figure 1 shows the GUI for manual modification of the different formats of a formula. While the other fields are intended to provide additional information, the pipeline to create and cultivate a gold standard entry starts with the semantic LATEX input field. LATEXML will generate content MathML based on this input and VMEXT will render the generated content MathML afterwards. We control the output by using the DLMF LATEX macros [12] and

our developed extensions. The following list contains some examples of the DLMF LaTeX macros.

- `\EulerGamma@{z}`: $\Gamma(z)$: gamma function,
- `\BesselJ{\nu}@{z}`: $J_\nu(z)$: Bessel function of the first kind,
- `\LegendreQ[\mu]{\nu}@{z}`: $Q_\nu^\mu(z)$: associated Legendre function of the second kind,
- `\JacobiP{\alpha}{\beta}{n}@{x}`: $P_n^{(\alpha,\beta)}(x)$: Jacobi polynomial.

The DLMF web pages, which we use as one of the sources for our dataset, were generated from semantically enriched LaTeX sources using LaTeXML. Since LaTeXML is capable of interpreting semantic macros, generates content MathML that can be controlled with macros, and is easily extended by new macros, we also used LaTeXML to generate our gold standard. While the DLMF is a compendium for special functions, we need to annotate every identifier in the formula with semantic information. Therefore, we extended the set of semantic macros.

In addition to the special symbols listed in Table 1, we created macros to semantically enrich identifiers, operators, and other mathematical concepts by linking them to their Wikidata items. As shown in Figure 1, the annotations are visualized using yellow (grayscaled in print) info boxes appearing on mouseover. The boxes show the Wikidata QID, the name, and the description (if available) of the linked concept.

In addition to naming, classifying, and semantically annotating each formula, we performed three other tasks:

- correcting the LaTeX string extracted from the sources;
- checking and correcting the MathML generated by LaTeXML;
- visualizing the MathML using VMEXT.

Most of the extracted formulae contained concepts to improve human readability of the source code, such as commented line breaks (`%`⟨*newline*⟩), in long mathematical expressions, or special macros to improve the displayed version of the formula, e.g., spacing macros, delimiters, and scale settings, such as `\!`, `\,` or `\>`. Since they are part of the expression, all of the tested tools (including LaTeXML) try to include these formatting improvements into the MathML markup. For our gold standard, we focus on the pure semantic information and forgo formatting improvements related to displaying the formula. The corrected TeX field shows the cleaned mathematical LaTeX expression.

Using the corrected TeX field and the semantic macros, we were able to adjust the MathML output using LaTeXML and verify it by checking the visualization from VMEXT.

3.3 Evaluation metrics

To quantify the conversion quality of individual tools, we computed the similarity of each tool's output and the manually created gold standard. To define the similarity measures for this comparison, we built upon our previous work [25], in which we defined and evaluated four similarity measures: taxonomic distance, data type hierarchy level, match depth, and query coverage.

The measures taxonomic distance and data type hierarchy level require the availability of a hierarchical ordering of mathematical functions and objects. For our use case, we derived this hierarchical ordering from the MathML content dictionary. The measures assign a higher similarity score if matching formula elements belong to the same taxonomic class. The match depth measure operates under the assumption that matching elements, which are more deeply nested in a formula's content tree, i.e., farther away from the root node, are less significant for the overall similarity of the formula, hence are assigned a lower weight. The query coverage measure performs a simple 'bag of tokens' comparison between two formulae and assigns a higher score the more tokens the two formulae share.

In addition to these similarity measures, we also included the tree edit distance. For this purpose, we adapted the robust tree edit distance (RTED) implementation for Java [16]. We modified RTED to accept any valid XML input and added math-specific 'shortcuts', i.e., rewrite rules that generate lower distance scores than arbitrary rewrites. For example, rewriting $\frac{a}{b}$ to ab^{-1} causes a significant difference in the expression tree: Three nodes $(\wedge, -, 1)$ are inserted and one node is renamed $\div \to \cdot$. The 'cost' for performing these edits using the stock implementation of RTED is $c = 3i + r$. However, the actual difference is an equivalence, which we think should be assigned a cost of $e < 3i + r$. We set $e < r < i$.

4 Evaluation of context-agnostic conversion tools

This section presents the results of evaluating existing, context-agnostic conversion tools for mathematical formulae using our benchmark dataset MathMLben (see section 3). We compare the distances between the presentation MathML and the content MathML tree of a formula yielded by each tool to the respective trees of formulae in the gold standard. We use the tree edit distance with customized weights and math-specific shortcuts. The goal of shortcuts is

eliminating notational-inherent degrees of freedom, e.g., additional PL elements or layout blocks, such as `mrow` or `mfenced`.

4.1 Tool selection

We compiled a list of available conversion tools from the W3C[6] wiki, from *GitHub*, and from questions about automated conversion of mathematical LaTeX to MathML on *Stack Overflow*. We selected the following converters:

- LaTeXML: supports converting generic and semantically annotated LaTeX expressions to XML/HTML/MathML. The tool is written in Perl [11] and is actively maintained. LaTeXML was specifically developed to generate the DLMF web page and can therefore parse entire TeX documents. Notably, LaTeXML supports conversions to content MathML.
- LaTeX2MathML(a.k.a. LaTeX2MML): a small Python project to generate presentation or content MathML from generic LaTeX expressions.[7]
- Mathoid: a service using Node.js, PhantomJS and MathJax (a JavaScript display engine for mathematics) to generate SVG and MathML from LaTeX input. Mathoid is currently used to render mathematical formulae on Wikipedia [23].
- SnuggleTeX: an open-source Java library developed at the University of Edinburgh.[8] The tool can convert simple LaTeX expressions to XHTML and presentation MathML.
- MathToWeb: an open-source Java-based web application that generates presentation MathML from LaTeX expressions.[9]
- TeXZilla: a JavaScript web application for LaTeX to MathML conversion capable of handling Unicode characters.[10]
- Mathematical: an application written in C and wrapped in Ruby to provide a fast translation from LaTeX expressions to the image formats SVG and PNG. The tool also provides translations to presentation MathML.[11]
- CAS: we included a prominent CAS capable of parsing LaTeX expressions.
- Part-of-Math (POM) Tagger: a grammar-based LaTeX parser that tags recognized tokens with information from a dictionary [28]. The POM tagger is currently under development. In this paper, we use the first version. In [5], this version was used to provide translations LaTeX to the CAS Maple. In its current state, this program offers no export to MathML. We developed an XML exporter to be able to compare the tree provided by the POM tagger with the MathML trees in the gold standard.

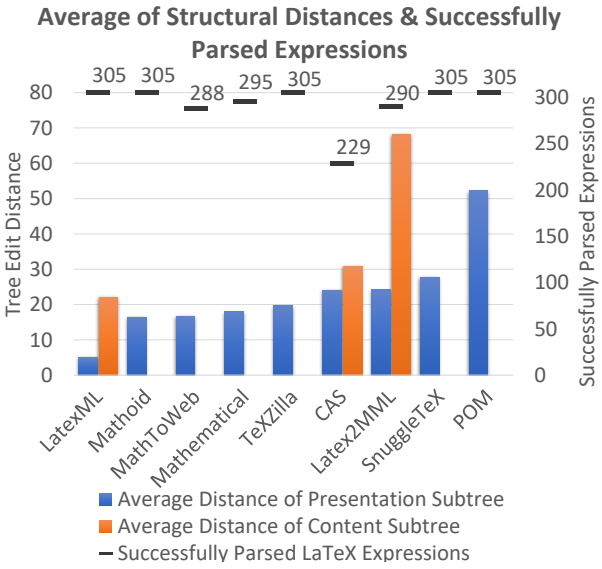

Figure 2: Overview of the structural tree edit distances (using $r = 0$, $i = d = 1$) between the MathML trees generated by the conversion tools and the gold standard MathML trees.

4.2 Testing framework

We developed a Java-based framework that calls the programs to parse the corrected TeX input data from the gold standard to presentation MathML, and, if applicable, to content MathML. In case of the POM tagger, we parsed the input string to a general XML document. We used the corrected TeX input instead of the originally extracted string expressio (see section 3.2).

Executing the testing framework requires the manual installation of the tested tools. The POM tagger is not yet publicly available.

4.3 Results

Figure 2 shows the averaged structural tree edit distances between the presentation trees (blue) and content trees (orange) of the generated MathML files and the gold standard. To calculate the structural tree edit distances, we used the RTED [16] algorithm with costs of $i = 1$ for inserting, $d = 1$ for deleting and $r = 0$ for renaming nodes. Furthermore, the

[6] `www.w3.org/wiki/Math_Tools`
[7] `github.com/Code-ReaQtor/latex2mathml`
[8] `www2.ph.ed.ac.uk/snuggletex/documentation/overview-and-features.html`
[9] `www.mathtowebonline.com`
[10] `fred-wang.github.io/TeXZilla`
[11] `github.com/gjtorikian/mathematical`

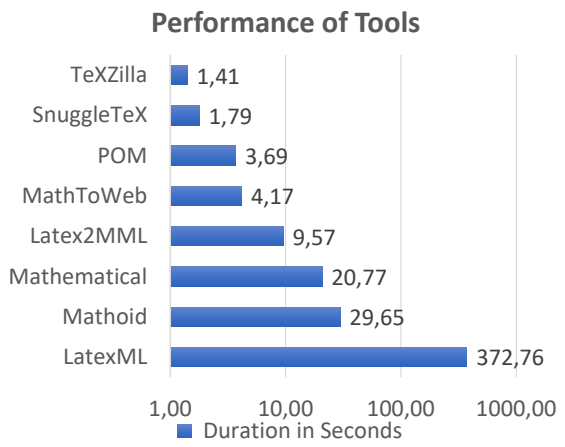

Figure 3: Time in seconds required by each tool to parse the 305 gold standard LaTeX expressions in logarithmic scale.

figure shows the total number of successful transformations for the 305 expressions (black ticks).

We consider differences of the presentation tree to the gold standard as deficits, because the mapping from LaTeX expressions to rendered expressions is unique (as long as the same preambles are used). A larger number indicates that more elements of an expression were misinterpreted by the parser. However, certain differences between presentation trees might be tolerable, e.g., reordering commutative expressions, while differences between content trees are more critical.

Also note that improving content trees may not necessarily improve presentation trees and vice versa. In case of $f(x + y)$, the content tree will change depending whether f represents a variable or a function, while the presentation tree will be identical in both cases. In contrast, $\frac{a}{b}$, a/b, and a/b have different presentation trees but identical content trees.

Figure 3 illustrates the runtime performance of the tools. We excluded the CAS from the runtime performance tests, because the system is not primarily intended for parsing LaTeX expressions, but for performing complex computations. Therefore, runtime comparisons between a CAS and conversion tools would not be representative. We measured the times required to transform all 305 expressions in the gold standard and write the transformed MathML to the storage cache. Note that the native code of LaTeX2MML, Mathematical and LaTeXML were called from the Java Virtual Machine (JVM) and Mathoid was called through local web-requests, which increased the runtime of these tools. The figure is scaled logarithmically. We would like to emphasize that LaTeXML is designed to translate sets of LaTeX documents instead of single mathematical expressions. Most of the other tools are lightweight engines.

In this benchmark, we focused on the structural tree distances rather than on distances in semantics. While our gold standard provides the information necessary to compare the extracted semantic information, we will focus on this problem in future work (see section 6).

5 Towards a context-sensitive approach

In this section, we present our new approach that combines textual features, i.e., semantic information from the surrounding text, with the converters to improve the outcome. Figure 4 illustrates the process of creating the gold standard, evaluating conversions, and how we plan to improve the converters with tree refinements (outside the MathMLben box). Our improvement approach includes three phases.

1. In the first phase, the Mathematical Language Processing (MLP) approach [19] extracts semantic information from the textual context by providing identifier-definiens[12] pairs.
2. The MLP annotations self-assess their reliability by annotating each identifier-definiens pair with its probabilities. Often, the methods do not find highly ranked semantic information. In such cases, we combine the results from the MLP with a dictionary-based method. In particular, we use the dictionaries from the POM tagger [28] that associate context-free semantics with the presentation tree. Since the dictionary entries are not ranked, we use them to drop unmentioned identifier-definiens pairs and choose the highest rank of the remaining pairs.
3. Based on the chosen semantic information, we redefine the content tree by reordering the nodes and subtrees.

Currently, the implementation is too immature to release it as a semantic annotation package. Instead, we discuss the method using the following selected examples that represent typical classes of disambiguation problems:

- Invisible operator disambiguation for the times vs. apply special case.
- Parameter vs. label disambiguation for subscripts.
- Einstein notation discovery.
- Multi-character operator discovery.

[12] In a definition, the *definiendum* is the expression to be defined and *definiens* is the phrase that defines the definiendum. Identifier-definiens pairs are candidates for an Identifier-definition. See [19] for a more detailed explanation.

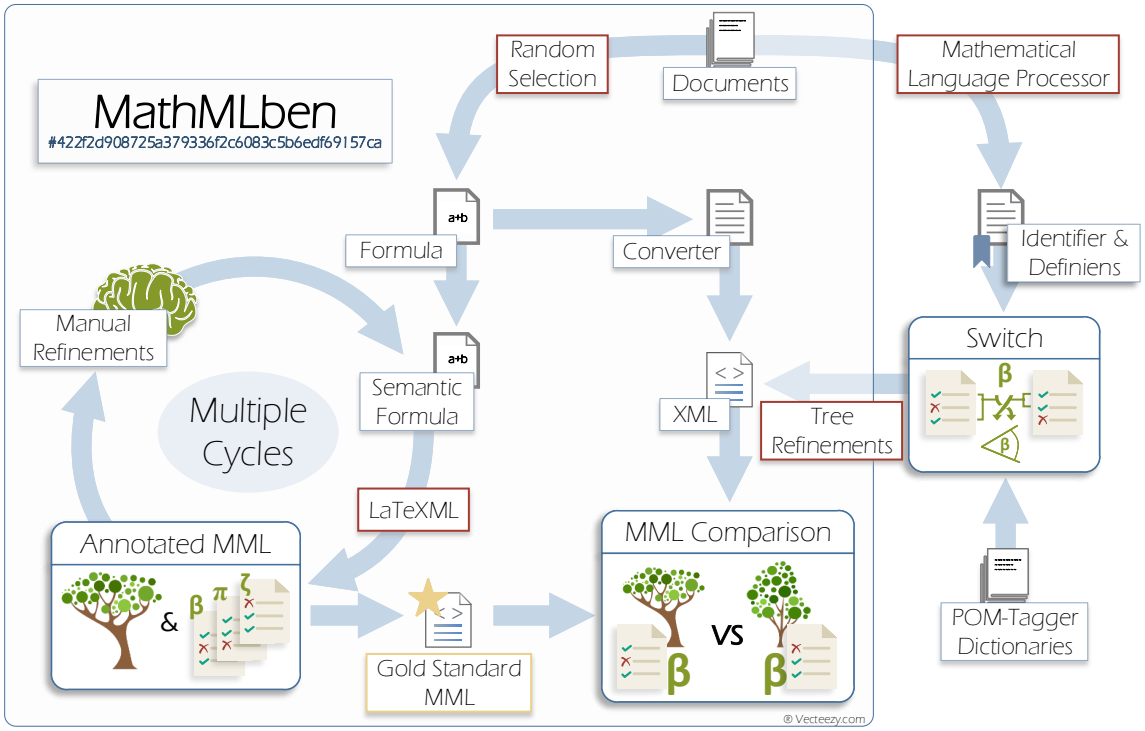

Figure 4: Mathematical language processing is the task of mapping textual descriptions to components of mathematical formulae (Part-of-Math tagging).

Learning special notations like the examples above is subject to future work. However, we deem it reasonable to start with these examples, since our manual investigation of the tree edit distances showed that such cases represented major reasons for errors in the content MathML tree.

Previously, the MLP software was limited to extracting information about identifiers, not general mathematical symbols. Moreover, the software was optimized for the Wikipedia dataset. We thus expanded the software for this study to enable parsing pure XHTML input as provided by the NTCIR tasks and the DLMF website. Achieving this goal required realizing a component for symbol identification. We chose the strategy of considering every simple expression that is not an identifier as a candidate for a symbol.

For our first experiments we tried to improve the output by LaTeXML, since LaTeXML performs best in our tests and it was able to generate content MathML. Moreover, with the newly developed semantic macros, we are able to optimize MathML in a pre-processing step by enhancing the input LaTeX expression. Consequently, we do not need to develop complex post-processing algorithms to manipulate content MathML.

As part of this study, we created a custom style sheet that fixes the following problems: (1) use of the power symbols for superscript characters unless Einstein notation was discovered, (2) interpretation of subscript indices as parameters, unless they are in text mode; for text mode, the ensemble of main symbol and subscript will be regarded as an identifier, (3) symbols that are considered as a 'function' are applied to the following identifier, rather than being multiplied with the identifier.

First experiments using these refinement techniques have proven to be very effective. We have chosen a small set of tem functions for performing the refinements and to show the potential of the techniques. Of those 10 cases, with simple regular expression matching, our MLP approach found four cases, where the highest ranked identifier-definiens pair was 'function' for at least one identifier in the formula. In these four cases, the distances of the content trees decreased to zero with all previously explained refinements enabled.

While this is just a first indication for the suitability of our approach, it shows that the long chain of processing steps shows promise. Therefore, we are actively working on the presented improvements and plan to focus on the task of learning how to generate

mappings from the input PL encoding to CL encoding without general rules for branch selection as we applied them so far.

6 Conclusion and future work

We make available the first benchmark dataset to evaluate the conversion of mathematical formulae between presentation and content formats. During the encoding process for our MathML-based gold standard, we presented the conceptual and technical issues that conversion tools for this task must address. Using the newly created benchmark dataset, we evaluated popular context-agnostic LaTeX-to-MathML converters. We found that many converters simply do not support the conversion from presentation to content format, and those that did often yielded mathematically incorrect content representations even for basic input data. These results underscore the need for future research on mathematical format conversions.

Of the tools we tested, LaTeXML yielded the best conversion results, was easy to configure, and highly extensible. However, these benefits come at the price of a slow conversion speed. Due to its comparatively low error rate, we chose to extend the LaTeXML output with semantic enhancements.

Unfortunately, we failed to develop an automated method to learn special notation. However, we could show that the application of special selection rules improves the quality of the content tree, i.e., allows choosing the most suitable tree from a selection of candidates. While the implementation of a few selection rules fixes nearly all issues we encountered in our test documents, the long tail of rules shows the limitations of a rule-based approach.

Future work

We will focus our future research on methods for automated notation detection, because we consider this approach as better suited and better scalable than implementing complex systems of selection rules. We will extract the considered notational features from the textual context of formulae and use them to extend our previously proposed approach of constructing identifier name spaces [19] towards constructing notational name spaces. We will check the integrity of such formed notational name spaces with methods comparable to those proposed in our previous publication [22] where we used physical units as a sanity check, if semantic annotation in the domain of physics are correct.

Acknowledgments. We would like to thank Abdou Youssef for sharing his Part-of-Math tagger with us and for offering valuable advice. We are also indebted to Akiko Aizawa for her advice and for hosting us as visiting researchers in her lab at the National Institute of Informatics (NII) in Tokyo. Furthermore, we thank Wikimedia Labs for providing cloud computing facilities and hosting our gold standard dataset. This work was supported by the FITWeltweit program of the German Academic Exchange Service (DAAD) as well as the German Research Foundation (DFG, grant GI-1259-1).

References

[1] A. Aizawa, M. Kohlhase, et al. NTCIR-11 math-2 task overview. In *Proc. 11th NTCIR Conf. on Evaluation of Information Access Technologies, Tokyo, Japan*, 2014.

[2] F. Cajori. *A History of Mathematical Notations*, vol. 1. Courier Corporation, 1928.

[3] H. S. Cohl, M. A. McClain, et al. Digital repository of mathematical formulae. In *Conference on Intelligent Computer Mathematics (CICM), Coimbra, Portugal*, pp. 419–422, 2014. doi:10.1007/978-3-319-08434-3_30

[4] H. S. Cohl, M. Schubotz, et al. Growing the digital repository of mathematical formulae with generic sources. In M. Kerber, J. Carette, et al., eds., *CICM, Washington, DC, USA*, vol. 9150, pp. 280–287, 2015. doi:10.1007/978-3-319-20615-8_18

[5] H. S. Cohl, M. Schubotz, et al. Semantic preserving bijective mappings of mathematical formulae between document preparation systems and computer algebra systems. In *CICM, Edinburgh, UK*, 2017. doi:10.1007/978-3-319-62075-6_9

[6] P. Dehaye, M. Iancu, et al. Interoperability in the OpenDreamKit project: The math-in-the-middle approach. In M. Kohlhase, M. Johansson, et al., eds., *CICM, Bialystok, Poland*, vol. 9791, pp. 117–131, 2016. doi:10.1007/978-3-319-42547-4_9

[7] M. England, E. S. Cheb-Terrab, et al. Branch cuts in Maple 17. *ACM Comm. Comp. Algebra* 48(1/2):24–27, 2014. doi:10.1145/2644288.2644293

[8] D. Ginev, H. Stamerjohanns, and M. Kohlhase. The LaTeXml daemon: Editable math on the collaborative web. In *LWA 2011, Magdeburg, Germany*, pp. 255–256, 2011.

[9] P. D. F. Ion and S. M. Watt. The global digital mathematical library and the international mathematical knowledge trust. In *CICM, Edinburgh, UK*, vol. 10383, pp. 56–69, 2017.

[10] N. Meuschke, M. Schubotz, et al. Analyzing mathematical content to detect academic plagiarism. In *Proc. CIKM*, 2017.

[11] B. Miller. LaTeXML: A LaTeX to XML converter. http://dlmf.nist.gov/LaTeXML/

[12] B. R. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. *Ann. Math. Artif. Intell.* 38(1-3):121–136, 2003. doi:10.1023/A:1022967814992

[13] M.-Q. Nghiem, G. Yoko, et al. Automatic approach to understanding mathematical expressions using mathml parallel markup corpora. In *26th Annu. Conf. Jap. Society for Artificial Intell.*, 2012.

[14] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.0.17 of 2017-12-22. F. W. J. Olver et al., eds.

[15] L. Padovani. On the roles of LaTeX and MathML in encoding and processing mathematical expressions. In A. Asperti, B. Buchberger, and J. H. Davenport, eds., *Mathematical Knowledge Management (MKM), Bertinoro, Italy*, vol. 2594, pp. 66–79, 2003. doi:10.1007/3-540-36469-2_6

[16] M. Pawlik and N. Augsten. RTED: A robust algorithm for the tree edit distance. *CoRR* abs/1201.0230, 2012. http://arxiv.org/abs/1201.0230

[17] M. Schubotz. Implicit content dictionaries in the NIST digital repository of mathematical formulae. Talk presented at the OpenMath workshop CICM, 2016. http://cicm-conference.org/2016/cicm.php?event=&menu=talks#03

[18] M. Schubotz. *Augmenting Mathematical Formulae for More Effective Querying & Efficient Presentation*. PhD thesis, TU Berlin, Germany, 2017. http://d-nb.info/1135201722

[19] M. Schubotz, A. Grigorev, et al. Semantification of identifiers in mathematics for better math information retrieval. In R. Perego, F. Sebastiani, et al., eds., *SIGIR, Pisa, Italy*, pp. 135–144, 2016. doi:10.1145/2911451.2911503

[20] M. Schubotz, L. Krämer, et al. Evaluating and improving the extraction of mathematical identifier definitions. In G. J. F. Jones, S. Lawless, et al., eds., *Conference and Labs of the Evaluation Forum (CLEF), Dublin, Ireland*, vol. 10456, pp. 82–94, 2017. doi:10.1007/978-3-319-65813-1_7

[21] M. Schubotz, N. Meuschke, et al. VMEXT: A visualization tool for mathematical expression trees. In *CICM, Edinburgh, UK*, pp. 340–355, 2017. doi:10.1007/978-3-319-62075-6_24

[22] M. Schubotz, D. Veenhuis, and H. S. Cohl. Getting the units right. In A. Kohlhase, P. Libbrecht, et al., eds., *Workshop and Work in Progress Papers at CICM 2016, Bialystok, Poland*, vol. 1785, pp. 146–156, 2016. http://ceur-ws.org/Vol-1785/W45.pdf

[23] M. Schubotz and G. Wicke. Mathoid: Robust, scalable, fast and accessible math rendering for Wikipedia. In *CICM, Coimbra, Portugal*, pp. 224–235, 2014. doi:10.1007/978-3-319-08434-3_17

[24] M. Schubotz, A. Youssef, et al. Challenges of mathematical information retrieval in the NTCIR-11 math Wikipedia task. In R. A. Baeza-Yates, M. Lalmas, et al., eds., *Special Interest Group on Information Retrieval (SIGIR), Santiago, Chile*, pp. 951–954, 2015. doi:10.1145/2766462.2767787

[25] M. Schubotz, A. Youssef, et al. Evaluation of similarity-measure factors for formulae based on the NTCIR-11 math task. In N. Kando, H. Joho, and K. Kishida, eds., *11th NTCIR, Tokyo, Japan*, 2014. http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/04-NTCIR11-MATH-SchubotzM.pdf

[26] H. Stamerjohanns, D. Ginev, et al. MathML-aware article conversion from LaTeX. In *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada*, pp. 109–120, 2009. http://eudml.org/doc/220017

[27] S. M. Watt. Exploiting implicit mathematical semantics in conversion between TeX and MathML. *Proc. Internet Accessible Math. Commun.*, 2002.

[28] A. Youssef. Part-of-math tagging and applications. In *CICM, Edinburgh, UK*, pp. 356–374, 2017. doi:10.1007/978-3-319-62075-6_25

◇ Moritz Schubotz
André Greiner-Petter
Philipp Scharpf
Norman Meuschke
Universitätsstraße 10
78464 Konstanz, Germany
moritz.schubotz , andre.greiner-petter , philipp.scharpf , norman.meuschke (at) uni-konstanz.de

◇ Howard S. Cohl
National Institute of Standards and Technology
Mission Viejo, CA 92694, U.S.A
howard.cohl (at) nist dot gov

◇ Bela Gipp
Universitätsstraße 10
78464 Konstanz, Germany
bela.gipp (at) uni-konstanz dot de