Do the Math: Making Mathematics in Wikipedia Computable

André Greiner-Petter, Moritz Schubotz, Corinna Breitinger, Philipp Scharpf, Akiko Aizawa, and Bela Gipp

Abstract—Wikipedia combines the power of AI solutions and human reviewers to safeguard article quality. Quality control objectives include detecting malicious edits, fixing typos, and spotting inconsistent formatting. However, no automated quality control mechanisms currently exist for mathematical formulae. Spell checkers are widely used to highlight textual errors, yet no equivalent tool exists to detect algebraically incorrect formulae. Our paper addresses this shortcoming by making mathematical formulae computable. We present a method that (1) gathers the semantic information surrounding the context of each mathematical formulae, (2) provides access to the information in a graph-structured dependency hierarchy, and (3) performs automatic plausibility checks on equations. We evaluate the performance of our approach on 6,337 mathematical expressions contained in 104 Wikipedia articles on the topic of orthogonal polynomials and special functions. Our system, LACAST, verified 358 out of 1,516 equations as error-free. LACAST successfully translated 27% of the mathematical expressions and outperformed existing translation approaches by 16%. Additionally, LACAST achieved an F1 score of .495 for annotating mathematical expressions with relevant textual descriptions, which is a significant step towards advancing searchability, readability, and accessibility of mathematical formulae in Wikipedia. A prototype of LACAST and the semantically enhanced Wikipedia articles are available at: https://tpami.wmflabs.org.

Index Terms—Mathematical information retrieval, presentation to computation translation, mathematical objects of interest, mathematical representation transformation, computer algebra systems

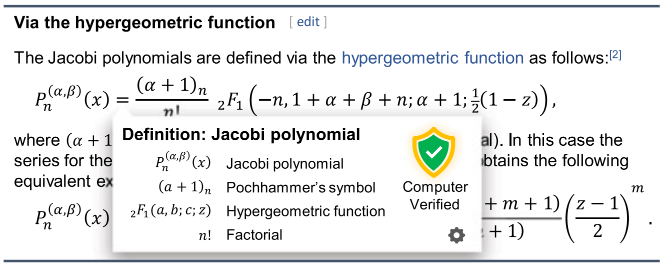
1 Introduction

Like many other knowledge base systems, Wikipedia encodes mathematical formulae in a representational format similar to LaTeX [1], [2], [3]. While this representational format is simple to comprehend by readers possessing the required mathematical training, an additional explicit knowledge of the semantics associated with each expression in a given formula, could make mathematical content in Wikipedia even more explainable, unambiguous, and most importantly, machine-readable. Additionally, making math machine-readable can allow even visually impaired individuals to receive a semantic description of the mathematical content. Finally, and crucially, moderating and curating mathematical content in a free and community-driven encyclopedia like Wikipedia, is more time-consuming and error-prone without explicit access to the semantics of a formula. Wikipedia currently uses the *Objective Revision Evaluation Service* (ORES) to predict the damaging or good faith intention of an edit using multiple independent classifiers trained on different datasets [4]. The primary motivation behind ORES was to reduce the overwhelming workload of content moderation with machine learning classification solutions. Until now, the ORES system applies no special care to mathematical content. Estimating the trustworthiness of an edit in a mathematical expression is significantly more challenging for human curators and almost infeasible for AI classification models due to the complex nature of mathematics.

This paper proposes a semantification and translation pipeline that makes the math in Wikipedia computable via Computer Algebra Systems (CAS). CAS, such as Maple [5] and Mathematica [6], are complex mathematical software tools that allow users to manipulate, simplify, plot, and evaluate mathematical expressions. Hence, translating mathematics in Wikipedia to CAS syntaxes enables automatic numeric and symbolic verification checks on complex mathematical equations [7], [8]. Integrating such verifications into the existing ORES system can significantly reduce the overload of moderating mathematical content and increasing credibility in the quality of Wikipedia articles at the same time [9]. Since such a translation is context-sensitive, we also propose a semantification approach for the mathematical content. This semantification uses unambiguous *semantic* LaTeX macros [10] from the Digital Library of Mathematical Functions (DLMF) [11] and noun phrases

- *André Greiner-Petter is with the University of Wuppertal, 42119 Wuppertal, Germany. E-mail: greinerpetter@uni-wuppertal.de.*
- *Moritz Schubotz is with the FIZ Karlsruhe — Leibniz Institute for Information Infrastructure, 10587 Berlin, Germany. E-mail: moritz.schubotz@fiz-karlsruhe.de.*
- *Corinna Breitinger and Bela Gipp are with the University of Göttingen, 37073 Göttingen, Germany. E-mail: breitinger@gipplab.org, gipp@uni-goettingen.de.*
- *Philipp Scharpf is with the University of Konstanz, 78464 Konstanz, Germany. E-mail: philipp.scharpf@uni-konstanz.de.*
- *Akiko Aizawa is with the National Institute of Informatics, Tokyo 101-8430, Japan. E-mail: aizawa@nii.ac.jp.*

Manuscript received 15 September 2021; accepted 29 June 2022. Date of publication 1 August 2022; date of current version 6 March 2023.
This work was supported in part by German Research Foundation DFG under Grant GI 1259/1 and in part by German Academic Exchange Service DAAD under Grant 57515245.
(Corresponding author: André Greiner-Petter.)
Recommended for acceptance by R. Dechter.
This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2022.3195261, provided by the authors.
Digital Object Identifier no. 10.1109/TPAMI.2022.3195261



Via the hypergeometric function [edit]

The Jacobi polynomials are defined via the hypergeometric function as follows:[2]

$$P_n^{(\alpha,\beta)}(x) = \frac{(\alpha+1)_n}{n!} {}_2F_1\left(-n, 1+\alpha+\beta+n; \alpha+1; \tfrac{1}{2}(1-z)\right),$$

where $(\alpha+1$ … al). In this case the series for the … btains the following equivalent ex…

$$P_n^{(\alpha,\beta)}(x) \quad \ldots + m + 1) \ldots + 1) \left(\frac{z-1}{2}\right)^m .$$

Definition: Jacobi polynomial

$P_n^{(\alpha,\beta)}(x)$	Jacobi polynomial
$(a+1)_n$	Pochhammer's symbol
${}_2F_1(a,b;c;z)$	Hypergeometric function
$n!$	Factorial

Computer Verified

Fig. 1. Mathematical semantic annotation in Wikipedia.

from the textual context to semantically annotate math formulae. The semantic encoding in the DLMF provides additional information about the components of a formula, the domain, constraints, links to definitions, and improves searchability and discoverability of the mathematical content [10], [12]. Our semantification approach enables the features from the DLMF for mathematics in Wikipedia. Fig. 1 provides an example vision of our semantic annotations and verification results in Wikipedia [3]. Head *et al.* [13] recently evaluated that providing readers information on the individual elements in mathematical expressions onsite [14], [15], such as shown in Fig. 1, can significantly support users of all experience levels to read and comprehend articles more efficiently.

Mathematics is not a formal language. Its interpretation heavily depends on the context, e.g., $\pi(x+y)$[1] can be interpreted as a multiplication $\pi x + \pi y$ or the number of primes less than or equal to $x+y$. CAS syntaxes, on the other hand, are unambiguous content languages. Therefore, the main challenge to enable CAS verifications for mathematical formulae in Wikipedia is a reliable translation between an ambiguous, context-dependent format and an unambiguous, context-free CAS syntax. Hence, we derive the following research question:

> What information is required to translate mathematical formulae from natural language contexts to CAS and how can this information be extracted?

In this paper, we present the first context-dependent translation from mathematical LaTeX expressions to CAS, specifically Maple and Mathematica. We show that a combination of nearby context analysis (extraction of descriptive terms) and a list of standard notations for common functions provide sufficient semantic information to outperform existing context-independent translation techniques, such as CAS internal LaTeX import functions. We achieve reliable translations in a four-step augmentation pipeline. These steps are: (1) pre-processing Wikipedia articles to enable natural language processing on it, (2) constructing an annotated mathematical dependency graph, (3) generating semantic enhancing replacement patterns, and (4) performing CAS-specific translations (see Fig. 2). In addition, we perform automatic symbolic and numeric computations on the translated expressions to verify equations from Wikipedia articles [7], [8]. We show that the system is capable of detecting potential errors in mathematical equations in Wikipedia articles. Future releases could be integrated into the ORES system to reduce vandalism and improve trust in mathematical articles in Wikipedia. We demonstrate the feasibility of the translation approach on English Wikipedia articles and provide access to an interactive demo of our *LaTeX to CAS translator* (LaCASt)[2].

For the evaluation of the translations, we focus on the sub-domain of orthogonal polynomials and special functions (OPSF). OPSF are generally well-supported by general-purpose CAS [16], which allows us to estimate the full potential of our proposed translation and verification pipeline. Since CAS syntaxes are programming languages, one has the option to add new functionality to a CAS, such as defining a new function. Defining new functions in CAS, however, can vary significantly in complexity. While translating a generic function like $f(x) := x^2$ is straightforward, defining the prime counting function from above could be very complex. If a function is explicitly declared in the CAS, we call a translation to that function *direct*. General mathematics often does not have such direct translations. For example, translating the generic function $f(x)$ is meaningless without considering the actual definition of $f(x)$. Hence, we first focus on translations of OPSF, which often have direct translations to CAS. In addition, OPSF are highly interconnected, i.e., many OPSF can be expressed (or even defined) in terms of other OPSF. One of the main tasks for our future work is to support more non-direct translations enabling our LaCASt to handle more general mathematics.

This paper is structured as follows. In Section 2, we present our pipeline and discuss each of the augmentation steps for mathematical objects. In Section 3, we evaluate our proposed pipeline on Wikipedia articles. Section 4 analyzes and discusses possible solutions for the remaining issues.

1.1 Related Work

Our proposed pipeline tangents several well-known tasks from MathIR, namely descriptive entity recognition for mathematical expressions [15], [17], [18], [19], [20], math tokenization [21], [22], math dependency recognition [23], [24], and automatic verification [7], [8]. Existing approaches to translate mathematical formulae from presentational languages, e.g., LaTeX or MathML, to content languages, e.g., content MathML or CAS syntax, do not analyze the context of a formula [24], [25], [26]. Hence, existing approaches to translate LaTeX to CAS syntaxes are limited to simple arithmetic expressions [26] or require manual semantic annotations [24]. Some CAS, such as Mathematica, support LaTeX imports. Those functions fall into the first category [26] and are limited to rather simple expressions. A semantic annotation, on the other hand, can be directly encoded in LaTeX via macros and allows for translations of more complex formulae. Miller *et al.* [10] developed a set of the previously mentioned semantic macros that link specific mathematical expressions with definitions in the DLMF [11]. The manually generated semantic data from the DLMF [12] was successfully translated to and evaluated by CAS with our proposed framework LaCASt [7], [16]. Therefore, our translation pipeline contains two steps: First, the semantic enhancement process towards the *semantic* LaTeX dialect used by the DLMF. Second, the translation from semantic LaTeX to

1. In the following, we use this color coding for examples to easily distinguish them from other mathematical content in this paper.

2. https://tpami.wmflabs.org [Accessed 09/15/2021]

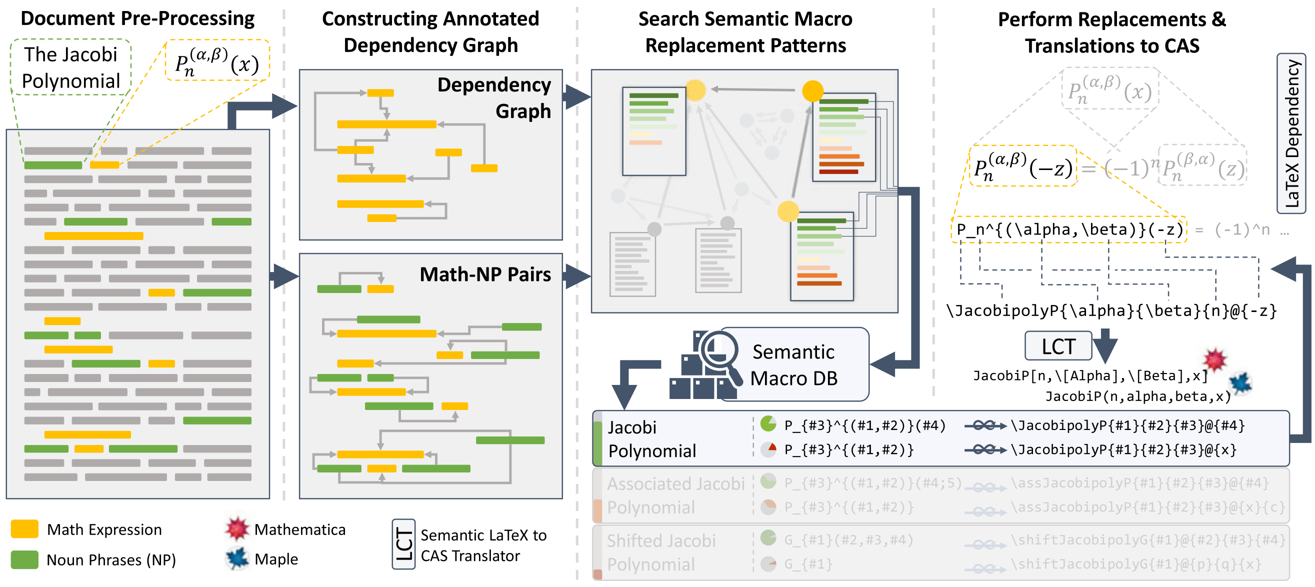

Document Pre-Processing
The Jacobi Polynomial
$P_n^{(\alpha,\beta)}(x)$
Constructing Annotated Dependency Graph
Dependency Graph
Math-NP Pairs
Search Semantic Macro Replacement Patterns
Semantic Macro DB
Perform Replacements & Translations to CAS
$P_n^{(\alpha,\beta)}(x)$
$P_n^{(\alpha,\beta)}(-z) = (-1)^n P_n^{(\beta,\alpha)}(z)$
LaTeX Dependency
P_n^{(\alpha,\beta)}(-z) = (-1)^n …
\JacobipolyP{\alpha}{\beta}{n}@{-z}
LCT
JacobiP[n,\[Alpha],\[Beta],x]
JacobiP(n,alpha,beta,x)
Jacobi Polynomial
P_{#3}^{(#1,#2)}(#4)
\JacobipolyP{#1}{#2}{#3}@{#4}
P_{#3}^{(#1,#2)}
\JacobipolyP{#1}{#2}{#3}@{x}
Associated Jacobi Polynomial
P_{#3}^{(#1,#2)}(#4;5)
\assJacobipolyP{#1}{#2}{#3}@{#4}
P_{#3}^{(#1,#2)}
\assJacobipolyP{#1}{#2}{#3}@{x}{c}
Shifted Jacobi Polynomial
G_{#1}(#2,#3,#4)
\shiftJacobipolyG{#1}@{#2}{#3}{#4}
G_{#1}
\shiftJacobipolyG{#1}@{p}{q}{x}
Math Expression
Noun Phrases (NP)
Mathematica
Maple
LCT
Semantic LaTeX to CAS Translator

Fig. 2. The workflow of our context sensitive translation from LaTeX to CAS syntax.

CAS via LACAST. In this paper, we focus on the first step. The second phase is largely covered by [7], [8], [16].

2 METHODOLOGY

First, we will introduce an abstract formalized concept for our translation approach followed by a detailed technical explanation of our system. Inspired by the pattern-matching translation approaches in compilers [27], we introduce a translation on mathematical expressions as a sequence of tree transformations. In the following, we mainly distinguish between two kinds of mathematical languages: presentational languages $\mathcal{L}_P$, such as LaTeX[3] or presentation MathML[4], and content languages $\mathcal{L}_C$, such as content MathML, OpenMath [28], or CAS syntaxes [5], [6]. Elements of these languages are often referred to as symbol layout trees for $e \in \mathcal{L}_P$ or operator trees for $e \in \mathcal{L}_C$ [29]. Then we call a context-dependent translation $\mathrm{t} : \mathcal{L}_P \times X \to \mathcal{L}_C$ with $\mathrm{t} \mapsto \mathrm{t}(e, X)$ *appropriate* if the intended semantic meaning of $e \in \mathcal{L}_P$ is the same as $\mathrm{t}(e, X) \in \mathcal{L}_C$. We further define the context X of an expression e as a set of facts from the document $\mathcal{D}$ the expression e appears in and a set of common knowledge facts $\mathcal{K}$ so that facts from the document may overwrite facts from the common knowledge set

$$X := \{f | f \in \mathcal{D} \cup \mathcal{K} \wedge (f \in \mathcal{K} \Rightarrow f \notin \mathcal{D})\}. \tag{1}$$

A fact f is a tuple $(\mathrm{MOI}, \mathrm{MC})$ of a Mathematical Object of Interest (MOI) [24] and a Mathematical Concept (MC). An MOI m refers to a meaningful mathematical object in a document and the MC uniquely defines the semantics of that MOI. In particular, from the MC of an MOI m, we derive a semantic enhanced version $\widetilde{m}$ of m so that $\widetilde{m} \in \mathcal{L}_C$. Hence, from f, we derive a graph transformation rule $r_f = m \to \widetilde{m}$ and define $g_f(e)$ as the application $e \underset{r_f}{\Rightarrow} \tilde{e}$ with $e \in \mathcal{L}_P, \tilde{e} \in \mathcal{L}_C$.

We split the translation $\mathrm{t}(e, X)$ into two steps, a semantification $\mathrm{t}_s(e, X)$ and a mapping $\mathrm{t}_m(e)$ step. The semantification $\mathrm{t}_s(e, X)$ transforms all subexpressions $\bar{e} \subseteq e$ that are not operator trees, i.e., $\bar{e} \in \mathcal{L}_P \setminus \mathcal{L}_C$, to operator tree representations $\tilde{\bar{e}} \in \mathcal{L}_C$. In the following, we presume that these subexpressions $\bar{e}$ are MOI so that we can derive $\tilde{\bar{e}}$ from a fact $f \in X$. Then we define the semantification step as the sequence of fact-based graph transformations

$$\mathrm{t}_s(e, X) := g_{f_1} \circ \cdots \circ g_{f_n}(e), \tag{2}$$

with $f_k \in X, k = 1, \ldots, n$. Again, we call a graph transformation $g(e)$ *appropriate* if the intended semantics of the expression e and its transformation $g(e)$ are the same. Further, we call $\mathrm{t}_s(e, X)$ *complete* if all subexpressions $e' \subseteq \mathrm{t}_s(e, X)$ are in $\mathcal{L}_C$ and *incomplete* otherwise. Note that graph transformations are not commutative, i.e., there could be $f_1, f_2 \in X$ so that $g_{f_1} \circ g_{f_2}(e) \neq g_{f_2} \circ g_{f_1}(e)$.

The mapping step $\mathrm{t}_m(e)$ is a sequence of applications on graph transformation rules that replace a node (or subtree) with the codomain-specific syntax version of the node (or subtree). Hence, the mapping step is a context-independent translation $\mathrm{t}_m : \mathcal{L}_{C_1} \to \mathcal{L}_{C_2}$ with $\mathcal{L}_{C_1}, \mathcal{L}_{C_2} \subset \mathcal{L}_C$ and a fixed rule set $\mathcal{R}_{C_2}^{C_1}$ so that $r_k = \mathcal{L}_{C_1} \to \mathcal{L}_{C_2}$ for $r_k \in \mathcal{R}_{C_2}^{C_1}, k = 1, \ldots, n$. Then we define

$$\mathrm{t}_m(e) := g_{r_1} \circ \cdots \circ g_{r_n}(e). \tag{3}$$

Note that $\mathrm{t}_m(e)$ ignores subexpressions $\bar{e} \subseteq e$ that are not in $\mathcal{L}_C$. For CAS languages $\mathcal{L}_M \subset \mathcal{L}_C$, certain subtrees of an expression $\tilde{e} \subseteq e \in \mathcal{L}_P$ are operator trees in the target language, $\tilde{e} \in \mathcal{L}_M$. Hence, we call $\mathrm{t}_m(e)$ complete, if all $e' \subset e$ with $e' \in \mathcal{L}_{C_1} \setminus \mathcal{L}_{C_2}$ were transformed to $\mathcal{L}_{C_2}$. Note that a complete $\mathrm{t}_m(e)$ is not necessarily appropriate because such an $e \in \mathcal{L}_P \cap \mathcal{L}_C$ could have a different semantic meaning in $\mathcal{L}_P$ and $\mathcal{L}_C$ (see the π example from the introduction). For a given target CAS language $\mathcal{L}_M \subset \mathcal{L}_C$, a set of rules $\mathcal{R}_M^C$, and a context X, we define the two step translation process as

$$\mathrm{t}(e, X) := \mathrm{t}_m(\mathrm{t}_s(e, X)). \tag{4}$$

We call $\mathrm{t}(e, X)$ complete if $\mathrm{t}_s(e, X)$ and $\mathrm{t}_m(e)$ are complete and appropriate.

3. https://www.latex-project.org/ [Accessed 06/29/2021]
4. https://www.w3.org/TR/MathML3/ [Accessed 06/29/2021]

Splitting the translation $\mathrm{t}(e, X)$ into these two steps has the advantage of modularity. Considering an appropriate and complete semantification, we can translate an expression e to any context language $\mathcal{L}_M \subset \mathcal{L}_C$ by using a different set of rules $\mathcal{R}_M^C$ for $\mathrm{t}_m(e)$. In previous research, we developed LACAST [16], [30] as an implementation of $\mathrm{t}_m(e)$ between the content languages *semantic* LaTeX [12] (the semantic enhanced LaTeX used in the DLMF) and the CAS syntaxes of Maple and Mathematica. Technically, semantic LaTeX is simply normal LaTeX, where specific subexpressions are replaced by semantic enhanced macros. In this paper, we extend LACAST to identify the subexpressions that can be replaced with these semantic LaTeX macros. This semantification is our first translation step $\mathrm{t}_s(e, X)$. The results $\mathrm{t}_s(e, X)$ are in semantic LaTeX which is in $\mathcal{L}_C$. For the second step (the mapping), we rely on the original LACAST implementation (from semantic LaTeX to CAS syntaxes) for $\mathrm{t}_m(e)$ and presume that $\mathrm{t}_m(e)$ is complete and appropriate [7], [8].

To perform a complete and appropriate semantification, we need to solve three remaining issues. First, how can we derive sufficiently many facts from a document $f \in \mathcal{D}$ so that the transformation rules r_f are appropriate and the semantification $\mathrm{t}_s(e, X)$ is appropriate and complete? Second, since the transformation rules are not commutative, a different order of facts may result in an inappropriate semantification $\mathrm{t}_s(e, X)$. Hence, we need to develop a fact-ranking $\mathrm{rk}(f)$ so that the sequence of transformations is performed in an appropriate order. Third, how can we determine if a translation was appropriate and complete? There is no general solution available to determine the intended semantic information of an expression $e \in \mathcal{L}_P$. In turn, it is probably impossible to certainly determine if a translation is appropriate for general expressions. Therefore, we propose different evaluation approaches that allow automatically verifying the appropriateness and completeness of a translation. We performed the same evaluation approaches on the manually annotated semantic LaTeX sources of the DLMF and successfully identified errors in the DLMF and the two CAS Maple and Mathematica [7], [8]. Hence, we presume the same technique is appropriate to detect errors in Wikipedia too. In addition to these verification evaluations, we perform a manual evaluation on a smaller test set for a qualitative analysis.

The number of facts (transformation rules) that we derive from a document $\mathcal{D}$ is critical. A low number of transformation rules may result in an incomplete translation. On the other hand, too many transformation rules may increase the number of false positives and result in an inappropriate transformation. To solve this issue, we propose a dependency graph of mathematical expressions containing the MOI of a document as nodes. A dependency in this graph describes the subexpression relationship between two MOI. We further annotate each MOI with textual descriptions from the surrounding context. We interpret these descriptions as references to the mathematical concepts MC that defines the MOI and rank each description according to distance and heuristic measures. Since MOI are often compositions of other MOI, the dependencies allow us to derive relevant facts for an expression e from the subexpressions $e' \subseteq e$. To derive a semantically enhanced version $\widetilde{m}$ for an MOI m, we use the semantic macros from the DLMF. Each semantic macro is a semantically enhanced version $\widetilde{m}$ of a standard representational m. To derive relevant semantic macros, i.e., transformation rules, we search for the semantic macro's description that matches the MC of the facts. In turn, we have a large number of ranked facts with the same MOI m and a ranked list of transformation rules $r_1, \ldots, r_n$ for each fact f. The rankings allow us to control the number and order of the graph transformation $g_{f_r}(e)$ in $\mathrm{t}_s(e, X)$. In turn, the annotated dependency graph should solve the mentioned issues one and two. The pipeline is visualized in Fig. 2. The rest of this section explains the pipeline in more detail. The third issue, i.e., determining the appropriateness and completeness of a translation is discussed in Section 3.

Example. Consider the example from the introduction $\pi(x+y)$ in a document $\mathcal{D}$ that describes $\pi(x)$ as the prime counting function. Hence, we derive the fact $f = (\pi(x), \text{prime counting function}) \in \mathcal{D}$. In our dependency graph, $\pi(x+y)$ depends on $\pi(x)$. Hence, we derive the same fact f for $\pi(x+y)$. Based on this fact, we find a function in the DLMF described as 'the number of primes not exceeding x' which uses the semantic macro `\nprimes@{x}` and the presentation $\pi(x)$. Hence, we derive the transformation rule

$$r_f = \texttt{\textbackslash pi}(v_1) \to \texttt{\textbackslash nprimes@}\{v_1\}, \tag{5}$$

where v_1 is a wildcard for variables. For simplicity reasons, this example only derived a single transformation rule r_f rather than an entire set of ranked rules and facts as described above. LACAST defines a translation rule $r_1 \in \mathcal{R}_{\text{Mathematica}}^C$ for this function to `PrimePi [x]` and a rule $r_2 \in \mathcal{R}_{\text{Maple}}^C$ to `pi (x)` in Maple[5], respectively. Hence, the translation to Mathematica would be performed via r_1 as

$$\begin{aligned}
\mathrm{t}(\texttt{\textbackslash pi}(\mathrm{x}+\mathrm{y}), X) &= \mathrm{t}_m(\mathrm{t}_s(\texttt{\textbackslash pi}(\mathrm{x}+\mathrm{y}), X)) && (6)\\
&= g_{r_1}(g_f(\texttt{\textbackslash pi}(\mathrm{x}+\mathrm{y}))) && (7)\\
&= g_{r_1}(\texttt{\textbackslash nprimes@}\{\mathrm{x}+\mathrm{y}\}) && (8)\\
&= \texttt{PrimePi}[\mathrm{x}+\mathrm{y}]. && (9)
\end{aligned}$$

Note that the subexpression $x+y$ was not transformed by $g_f(e)$ nor by $g_{r_1}(e)$, because $x+y \in \mathcal{L}_M \cap \mathcal{L}_P$. This translation is complete and appropriate.

2.1 Document Pre-Processing

For extracting the facts from a document $\mathcal{D}$, we need to identify all MOI and MC. In previous research [15], we have shown that noun phrases can represent definiens of identifiers. Hence, we presume noun phrases are good candidates for MCs too. To properly extract noun phrases, we use CoreNLP [31] as our Part-of-Speech (POS) tagger [32]. Since CoreNLP is unable to parse mathematics, we replace all math by placeholders first. In a previous project [19], we proposed a Mathematical Language Processor (MLP) that replaces mathematical expressions with placeholders. Occasionally, this approach yields wrong annotations. For example, CoreNLP may tag *factorial* or *polynomial* as adjectives when a math token follows, even in cases where they are clearly naming mathematical objects[6]. However, the MLP approach works reasonably well in most cases.

5. Maple requires to pre-load the *NumberTheory* package.
6. For example, 'The Jacobi polynomial `MATH_1` is an orthogonal polynomial.' Both 'polynomial' tokens in this sentence are tagged as `JJ` (Adjective) with CoreNLP version 4.4.0.

Since Wikipedia articles are written in Wikitext, we use Sweble [33] to parse an article, replace MOI with placeholders, remove visual templates, and generate a plain text version of an article. Wikipedia officially recommends encoding in-line mathematics via templates that do not use LaTeX encoding. In addition, since Wikipedia is community-driven, many mathematical expressions are not properly annotated as such. This makes it challenging to detect all MOI in a given document. For example, the Jacobi polynomial article[7] contains several formulae that do not use the `math` template nor the `<math>` tag (for LaTeX), such as the single identifier 'x' and the UTF-8 character sequences $\epsilon < 0$, $[\epsilon, \{\{\mathrm{pi}\}\} - \epsilon]$, and $0 \leq \phi \leq 4\{\{\mathrm{pi}\}\}$. As an approach to detect such erroneous math, we consider sequences of symbols with specific Unicode properties as math. This includes the properties `Sm` for math symbols, `Sk` for symbol modifier, `Ps`, `Pe`, `Pd`, and `Po` for several forms of punctuation and brackets, and `Greek` for Greek letters. In addition, single letters in italic, e.g., 'x', are interpreted as math as well, which was already successfully used by MLP. Via MLP we also replace UTF-8 characters by their TeX equivalent. In the end, the erroneous UTF-8 encoded sequence $0 \leq \phi \leq 4\{\{\mathrm{pi}\}\}$ is replaced by `0 \leq \phi \leq 4pi` and considered as a single MOI. Using this approach, we detect 27 math-tags, 11 math-templates (including one `numblk`), and 13 in-line mathematics with erroneous annotations in the Jacobi polynomials article. The in-line math contains six single italic letters and seven complex sequences. In one case, the erroneous math was given in parentheses and the closing parenthesis was falsely identified as part of the math expression. Every other detection was correct. In the future, more in-depth studies can be applied to improve the accuracy of in-line math detection in Wikitext [34], [35].

2.2 Annotated Dependency Graph Construction

Retrieving the correct noun phrase (i.e., MC) that correctly describes a single MOI is most likely infeasible. Instead, we will retrieve multiple noun phrases for each MOI and try to rank them accordingly. In the following, we construct a mathematical dependency graph for Wikipedia articles in order to retrieve as many relevant noun phrases for an MOI as possible. As we have discussed in an earlier project [23], there are multiple valid options to construct a dependency graph. We decided to use the Part-of-Math (POM) tagger [22] to generate parse trees from LaTeX expressions to build a dependency graph. The POM tagger lets us establish dependencies by comparing annotated, semantic parse trees. Since the POM tagger aims to disambiguate mathematical expressions in the future, the accuracy of our new dependency graph directly scales with an increasing amount of semantic information available to the POM tagger. In addition, the more the POM tagger is able to disambiguate expressions, the more subexpressions $\bar{e} \subseteq e \in \mathcal{L}_P$ are already in our target language $\bar{e} \in \mathcal{L}_M$. Our translator LaCASt also relies on the parse tree of the POM tagger [16], [30]. Technically, this allows us to feed LaCASt directly with additional semantic information via manipulating the parse tree from the POM tagger. For example, consider the expression $a(b+c)$. In general, LaCASt would interpret the expression as a multiplication between a and $(b+c)$, as most conversion tools would [26]. However, we can easily tag the first token a as a function in the parse tree and thereby change the translation accordingly without further programmatic changes. In the following, we only work on the parse tree of the POM tagger, which can be considered as part of $\mathcal{L}_P$.

To establish dependencies between MOI, we introduce the concept of a mathematical stem (similar to 'word stems' in natural languages) that describes the static part of a function that does not change, e.g., the red tokens in $\Gamma(x)$ or $P_n^{(\alpha,\beta)}(x)$. Mathematical functions often have a unique identifier as part of the stem that represents the function, such as $\Gamma(x)$ or $P_n^{(\alpha,\beta)}(x)$. The identification of a stem of an MOI, however, is already context-dependent. As our introduction example of $\pi(x+y)$ shows, the location of the stem depends on the identification of $\pi(x+y)$ as the prime counting function. At this point in our pipeline, we lack sufficient semantic information about the MOI to identify the stem. On the other hand, a basic logic is necessary to avoid erroneous MOI dependencies. We apply the following heuristic for an MOI dependency: (i) at least one identifier must match in the same position in both MOI and (ii) this identifier is not embraced by parenthesis. Now, we replace every identifier in an MOI m_1 by a wildcard that matches a sequence of tokens or entire subtrees. If this pattern matches another MOI m_2 and the match obeys our heuristics (i) and (ii), we say m_2 depends on m_1 and define a directed edge from m_1 to m_2 in the graph. With the second heuristic, we avoid a dependency between $\Gamma(x)$ and $\pi(x)$ (since x fulfill the first heuristic but not the second). In the future, it would be worthwhile to study more heuristics on MOI to identify the stem via machine learning algorithms. A more comprehensive heuristic analysis is desirable, since not every function has a unique identifier in the stem, e.g., the Pochhammer's symbol $(x)_n$. Examples of dependencies between MOI can be found in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3195261, and on our demo page.

In addition to the new concept for addressing math stems, we also changed our approach for definition detection. Previously [23], we presumed that every equation symbol declares a definition for the left-hand side expression. This would have a significant impact on the translation to CAS. Further, definitions must be translated differently compared to normal equations. Currently, there is no reliable approach available to distinguish an equation from a definition. Existing approaches try to classify entire textual sections in a document as definitions [20], [36], [37], [38] but not a single formula. We will elaborate more on this matter in Section 3.3. For now, we only consider an equation symbol as a definition if it is explicitly declared as such via $:=$.

For annotating MOIs with textual descriptions, we first used a support vector machine [18] and later applied distance metrics [15], [19], [39] between single identifiers and textual descriptions. We were able to reach an F1 score of .36 for annotating single identifiers with textual descriptions. Since we are working on more complex, less overloaded [24], MOI expressions now, we can presume an improvement if we apply the same approach again. Hence, we used our latest improvements [39] and applied some changes to annotate MOI rather than single identifiers with textual descriptions from the surrounding context. Originally, we considered only

7. https://en.wikipedia.org/wiki/Jacobi_polynomials [Accessed: 7/6/2021]

TABLE 1
Mappings and Likelihoods for the Semantic LaTeX Macro of the General Hypergeometric Function in the DLMF

Prob.	Semantic Macro	LaTeX	Rendered Form
19.7%	`\genhyperF{par1}{par2}@{var1}{var2}{var3}`	`{}_{par1}F_{par2}(var1; var2; var3)`	${}_2F_1(a,b;c;z)$
80.3%	`\genhyperF{par1}{par2}@@{var1}{var2}{var3}`	`{}_{par1}F_{par2}({var1 \atop var2};var3)`	${}_2F_1\left({a,b \atop c};z\right)$
0.0%	`\genhyperF{par1}{par2}@@@{a_1,\dots,a_p}{b_1,\dots,b_q}{var3}`	`{}_{par1}F_{par2}(var3)`	${}_2F_1(z)$
0.0%	`\genhyperF{par1}{par2}@{a_1,\dots,a_p}{b_1,\dots,b_q}{z}`	`{}_{par1}F_{par2}`	${}_2F_1$

nouns, noun sequences, adjectives followed by nouns, and Wikipedia links as candidates of definiens (now MC) [15]. However, in the field of OPSF, such descriptions are generally insufficient. Hence, we include connective possessive endings and prepositions between noun phrases (see supplementary materials, available online, for further details).

Originally [15], we scored an identifier-definiens pair based on (1) the distance between the current identifier and its first occurrence in the document, (2) the distance (shortest path in the parse tree) between the definiens and the identifier, and (3) the distribution of the definiens in the sentence. We adopt this scoring technique for MOI and MC with slight adjustments. For condition (2), we declare the first noun in an MC as the representative token in the natural language parse tree. Therefore, (2) uses the shortest path between an MOI and the representative token in the parse tree. For condition (1), we need to identify the locations of MOIs throughout an entire document. Our dependency graph allows us to track the location of an MOI in the document. Hence, (1) calculates the distance of an MOI and its first occurrence isolated or as a dependent of another MOI in the document. In addition, we set the score to 1 if a combination of MOI and noun phrases match the patterns `NP MOI` or `MOI (is|are) DT? NP`. These basic patterns have been proven to be very effective in previous experiments for extracting descriptions of mathematical expressions [18], [19], [23], [39]. We denote the final score of a fact f, i.e., of an MOI and MC pair, with $s_{\mathrm{MLP}}(\mathrm{MOI}, \mathrm{MC})$.

2.3 Semantic Macro Replacement Patterns

Now, we derive a rule r_f for a fact f so that the MOI $m \in \mathcal{L}_P$ can be replaced by a semantic enhanced version $\widetilde{m} \in \mathcal{L}_C$ of it. The main issue is that we are still unable to identify the stems of a formula. Consider we have the MOI $P_n^{(\alpha,\beta)}(z)$ identified as *Jacobi polynomial*. How do we know the stem of a Jacobi polynomial and that n, α, β, and z are parameters and variables? For an appropriate translation, we even need to identify the right order of these arguments. There are two approaches, (i) we identify the definition of the formula in the article or (ii) we lookup a standard notation. The first approach works because with the definition, we can deduce the stem of a function by identifying which identifiers of the function are reused in the definition. For example, in Fig. 1, we see that n, α, β, and z appear in the definition of the Jacobi polynomial but not P. Hence, we can conclude that the stem of the Jacobi polynomial must be $P_n^{(\alpha,\beta)}(x)$. There are two remaining issues with this approach. First, what if a definition does not exist in the same article? This happens relatively often for OPSF, since OPSF are well established with more or less standard notation styles. Second, as previously pointed out, we cannot distinguish definitions from normal equations yet. As long as there is no reliable approach to identify definitions, approach (i) is infeasible. As a workaround, we focus on approach (ii) and leave (i) for future work.

In order to get standard notations and derive patterns of them, we use the semantic macros in the DLMF [10], [12]. A semantic macro is a semantically enhanced LaTeX expression that unambiguously describes the content of the expression. Hence, we can interpret a semantic macro as an unambiguous operator subtree $\widetilde{m} \in \mathcal{L}_C$. The rendered version of the macro (i.e., the *normal* LaTeX version) is in a presentational format $m \in \mathcal{L}_P$. Hence, we can derive a fact-based rule $r_f = m \to \widetilde{m}$ by finding the appropriate semantic macro for a given mathematical description (the MC in a fact f). The DLMF defines more than 600 different semantic macros for OPSF. A single semantic macro may produce multiple rendered forms, e.g., by omitting the parentheses around the argument in $\sin x$. This allows for fine controlling the visualization of the formulae. Table 1 contains the four different versions for the general hypergeometric function (controlled by the number of `@`s). The last version (without variables and no `@` symbol) is a special case, which never appears in the DLMF. However, every semantic macro is also syntactically valid without arguments. Note also that not every version visualizes all information that is encoded in a semantic macro. For example, `\genhyperF{2}{1}@@@{a,b}{c}{z}` omits the variables a, b, and c. Table 1 also shows the LaTeX for each version of the macro. By replacing the arguments with wildcards, we generate a LaTeX pattern m that defines a rule $m \to \widetilde{m}$. If the LaTeX omits information, we fill the missing slots of $\widetilde{m}$ with the default arguments denoted in the definitions of the semantic macros. For example, the default arguments for the general hypergeometric function are p and q for the parameters and $a_1, \ldots, a_p$, $b_1, \ldots, b_q$, and z for the variables. Hence, the last version in Table 1 fills up the slots for the variables with these default arguments (given in gray). In addition, the default arguments from the DLMF definitions also tell us if the argument can be a list, i.e., it may contain commas. Hence, we allow the two wildcards for the first two variables `var1` and `var2` to match sequences with commas while the other wildcards are more restrictive and reject sequences with commas.

Since every semantic macro in the DLMF has a description, we can retrieve semantic macros and also the replacement rule r_f, by using the annotations in the dependency graph as search queries. Currently, every fact has an MLP score $s_{\mathrm{MLP}}(f)$. But for each fact, we may retrieve multiple replacement patterns depending on how well the noun phrase (the MC) matches semantic macro description in the DLMF. To solve this issue, we develop a cumulated ranking for each fact $\mathrm{rk}(f)$. The first part of the ranking is the MLP score $s_{\mathrm{MLP}}(f)$ that ranks the pair of MOI and description MC. Second, we index all DLMF replacement patterns in an Elasticsearch (ES)[8] database to search for a semantic macro for a given description. ES uses the BM25 score to retrieve relevant semantic

8. https://github.com/elastic/elasticsearch [09/01/2021]

macros for a given query. Hence, the second component of the ranking function is the ES score (normalized over all retrieved hits) for a retrieved semantic macro $\widetilde{m}$ and the given description MC: $s_{\mathrm{ES}}(f)$. Lastly, every semantic macro $\widetilde{m}$ has multiple rendered forms, of which some are more frequently used than others in the DLMF, see the probability in Table 1. Hence, we score a rule $r_f = m \to \widetilde{m}$ based on its likelihood of use in the DLMF. We counted the different versions of each semantic macro in the DLMF to calculate the likelihood of use. The last two replacement patterns in the Table (the ones omitting information) never appear in the DLMF and have a probability of 0%. We denote this score as $s_{\mathrm{DLMF}}(r_f)$. The ranking for a fact $\mathrm{rk}(f)$ is simply the average over the three components $s_{\mathrm{MLP}}(f)$, $s_{\mathrm{ES}}(f)$, and $s_{\mathrm{DLMF}}(r_f)$.

Since LACAST was specifically developed for the semantics of the DLMF, it is not aware of general mathematical notation conventions. We fixed this issue by defining rules as part of the common knowledge $\mathcal{K}$ set of facts. We rank facts from $\mathcal{K}$ higher compared to facts from the article $\mathcal{A}$ to perform common knowledge pre-processing transformations prior to the facts derived from the article. Note that we do not presume that the following rules are always true. However, in the context of OPSF, we achieved better results by activating them by default and, if applicable, deactivating them for certain scenarios. This includes that π is always interpreted as the constant, e is Euler's number if e is followed by a superscript (power) at least once in the expression, i is the imaginary unit if it does not appear in a subscript (index), γ is the Euler-Mascheroni constant if the terms *Mascheroni* or *Euler* exists in any $f \in \mathcal{A}$. Note that these heuristics are consistent in an equation, i.e., i is never both an index and the imaginary unit within one equation. Further, we add rules for derivative notations, such as $\frac{\mathrm{d}y}{\mathrm{d}x}$ where y is optional and d can be followed by a superscript with a numeric value. In addition, LACAST presumes `\diff{.}` (e.g., for $\mathrm{d}x$) after integrals indicating the end of the argument of an integral. Hence, we search for d or d followed by a letter after integrals to replace it with `\diff{.}` (see [8] for a more detailed discussion on this approach). Finally, a letter preceding parenthesis is tagged as a function in the parse tree, if the expression in parenthesis contains commas or semicolons or it does not contain arithmetic symbols, such as $+$ or $-$. Note that once a symbol is identified as a function following this rule, it is tagged as such everywhere, regardless of the local situation. For example, in $f(x+\pi) = f(x)$ we would tag f as a function even though the first part $f(x+\pi)$ violates the mentioned rule. As previously mentioned, this changes the translation from `f * (x + Pi)` in Mathematica to `f[x + Pi]`. We provide a detailed step-by-step example of the translation pipeline and an interactive demo at: https://tpami.wmflabs.org.

3 Evaluation

Resulting from our motivation - improving Wikipedia articles - we choose Wikipedia for our test dataset. More specifically, we considered every English Wikipedia article that references to the DLMF via the `{{dlmf}}` template[9]. This should limit the domain to OPSF problems that we are currently examining. The English Wikipedia contains 104 such pages, of which only one page did not contain any formula (Spheroidal wave function)[10]. For the entire dataset (the remaining 103 Wikipedia pages), we detected 6,337 formulae in total (including the previously mentioned erroneous math).

One of our initial three issues still remains unsolved: how can we determine if a translation was appropriate and complete when we do not know the intended semantic meaning of an expression $e \in \mathcal{L}_P$? In natural languages, the BLEU score [40] is widely used to judge the quality of a translation. The effectiveness of the BLEU score, however, is questionable when it comes to math translations due to the complexity and high interconnectedness of mathematical formulae. Consider, a translation of the arccotangent function $\mathrm{arccot}(x)$ was performed to $\arctan(1/(x))$ in Maple. This translation is correct and even preferred in certain situations to avoid issues with so-called branch cuts (see [16, Section 3.2]). Previously, we developed a new approach that relies on automatic verification checks with CAS [7], [8] to verify a translation. This approach is very powerful for large datasets. However, it requires a large and precise amount of semantic data about the involved formulae, including constraints, domains, the position of branch cuts, and other information to reach high accuracy. In turn, we perform this automatic verification on the entire 103 Wikipedia pages but additionally created a benchmark dataset with 95 entries for qualitative analysis. To avoid issues like with the BLEU score, we manually evaluated each translation of the 95 test cases.

3.1 Symbolic and Numeric Testing

The automatic verification approach makes the assumption that a correct equation in the domain must remain valid in the codomain after a translation. If the equation is incorrect after a translation, we conclude a translation error. Previously [7], [8], we examined two approaches to verify an equation in a CAS. The first approach tries to symbolically simplify the difference of the left- and right-hand sides of an equation to zero. If the simplification returned zero, the equation was symbolically verified by the CAS. Symbolic simplifications of CAS, however, are rather limited and may even fail on simple equations. The second approach overcomes this issue by numerically calculating the difference between the left- and right-hand sides of an equation on specific numeric test values. If the difference is zero (or below a given threshold due to machine accuracy) for every test calculation, the equivalence of an equation was numerically verified. Clearly, the numeric evaluation approach cannot prove equivalence. However, it can prove disparity and therefore detect an error due to the translation.

We found that the translations by LACAST [16] were so reliable that the combination of symbolic and numeric evaluations was able to detect errors in the domain library (i.e., the DLMF) and the codomain systems (i.e., the CAS Maple and Mathematica) [7], [8]. Unfortunately, the number of false positives, i.e., correct equations that were not verified symbolically nor numerically, is relatively high. The main reason is unconsidered semantic information, such as constraints for specific variables or the position of branch cuts. Unconsidered semantic information causes the system to test equivalence on invalid conditions, such as invalid values,

9. Templates in Wikitext are placeholders for repetitive information which get resolved by Wikitext parsers. The DLMF-template, for example, adds the external reference for the DLMF to the article.

10. Retrieved from https://en.wikipedia.org/wiki/Special:WhatLinksHere by searching for *Template:Dlmf* [accessed 01/01/2021]

and therefore yields inequalities between the left- and right-hand sides of an equation even though the source equation and the translation were correct. Nonetheless, the symbolic and numeric evaluation approach proofs to be very useful also for our translation system. It allows us to quantitatively evaluate a large number of expressions in Wikipedia. In addition, it enables continuous integration testing for mathematics in Wikipedia article revisions. For example, an equation previously verified by the system that fails after a revision could indicate a poisoned revision of the article. This automatic plausibility check might be a jump start for the ORES system to better maintain the quality of mathematical documents [9]. For changes in math equations, ORES could trigger a plausibility check through our translation and verification pipeline and adjust the score of good faith of damaging an edit accordingly. A more comprehensive explanation about the automatic evaluation pipeline is given in the supplementary materials, available online.

3.2 Benchmark Testing

To compensate for the relatively low number of verifiable equations in Wikipedia with the symbolic and numeric evaluation approach, we crafted a benchmark test dataset to qualitatively evaluate the translations. This benchmark includes a single equation (the formulae must contain a top-level equality symbol[11], no `\text`, and no `\color` macros) randomly picked from each Wikipedia article from our dataset. For eight articles, no such equation was detected. Hence, the benchmark contains 95 test expressions. For each formula, we marked the extracted descriptive terms as irrelevant (0), relevant (1), or highly relevant (2), and manually translated the expressions to semantic LaTeX and to Maple and Mathematica. If the formula contained a function for which no appropriate semantic macro exists, the semantic LaTeX equals the generic (original) LaTeX of this function. In 18 cases, even the human annotator was unable to appropriately translate the expressions to the CAS, which underlines the difficulty of the task. The main reason for a manual translation failure was missing information (the necessary information for an appropriate translation was not given in the article) or it contained elements for which an appropriate translation was not possible, such as contour integrals, approximations, or indefinite lists of arguments with dots (e.g., $a_1, \ldots, a_n$). Note that the domain of orthogonal polynomials and special functions is a well-supported domain for many general-purpose CAS, like Maple and Mathematica. Hence, in other domains, such as in group, number, or tensor field theory, we can expect a significant drop of human-translatable expressions[12]. Since Mathematica is able to import LaTeX expressions, we use this import function as a baseline for our translations to Mathematica. We provide full access to the benchmark via our demo website and attached an overview to the supplementary materials, available online.

3.3 Results

First, we evaluated the 6,337 detected formulae with our automatic evaluation via Maple and Mathematica. Table 3 a shows an overview of this evaluation. With our translation pipeline, we were able to translate 72.6% of mathematical expressions into Maple and 73.8% into Mathematica syntax. From these translations, around 40% were symbolically and numerically evaluated (the rest was filtered due to missing equation symbols, illegal characters, etc.). We were able to symbolically verify 11% (Maple) and 15% (Mathematica), and numerically verify 18% (Maple) and 24% (Mathematica). In comparison, the same tests on the manually annotated semantic dataset of DLMF equations [12] reached a success rate of 26% for symbolic and 43% for numeric evaluations [8]. Since the DLMF is a manually annotated semantic dataset that provides exclusive access to constraints, substitutions, and other relevant information, we achieve very promising results with our context-sensitive pipeline. To test a theoretical continuous integration pipeline for the ORES system in Wikipedia articles, we also analyzed edits in math equations that have been reverted again. The Bessel function contains such an edit on the equation

$$J_n(x) = \frac{1}{\pi}\int_0^\pi \cos\left(n\tau - x\sin\tau\right) d\tau. \tag{10}$$

Here, the edit[13] changed $J_n(x)$ to $J_ZWE(x)$. Our pipeline was able to symbolically and numerically verify the original expression but failed on the revision. The ORES system could profit from this result and adjust the score according to the automatic verification via CAS.

3.3.1 Descriptive Term Extractions

Previously, we presumed that our update of the description retrieval approach to MOI would yield better results. In order to check the ranking of retrieved facts, we evaluate the descriptive terms extractions and compare the results with our previously reported F1 scores in [39]. We analyze the performance for a different number of retrieved descriptions and different depths. Here, the depth refers to the maximum depth of in-going dependencies in the dependency graph to retrieve relevant descriptions. A depth value of zero does not retrieve additional terms from the in-going dependencies but only the noun phrases that are directly annotated to the formula itself. The results for relevance 1 or higher are given in Table 2 a and for relevance 2 in Table 2 b. Since we need to retrieve a high number of relevant facts to achieve a complete translation, we are more interested in retrieving any relevant fact rather than a single but precise description. Hence, the performance for relevance 1 is more appropriate for our task. For a better comparison with our previous pipeline [39], we also analyze the performance only on highly relevant descriptions (relevance 2). As expected, for relevant noun phrases, we outperform the reported F1 score (.35). For highly relevant entries only, our updated MOI pipeline achieves similar results with an F1 score of .385.

3.3.2 Semantification

Since we split our translation pipeline into two steps, semantification and mapping, we evaluate the semantification transformations first. To do this, we use our benchmark dataset

11. This excludes equality symbols of deeper levels in the parse tree, e.g., the equality symbols in sums are not considered as such.

12. Note that there are numerous specialized CAS that would cover the mentioned domains too, such as GAP [41], PARI/GP [42], or Cadabra [43].

13. https://en.wikipedia.org/w/index. php?diff=991994767&oldid=991251002&title=Bessel_function&type= revision [Accessed 06/23/2021]

TABLE 2
Performance of Description Extractions via MLP for Low (2a) and High (2b) Relevance and the Performance of Translatons from LaTeX to semantic LaTeX (2c)

(a) Relevance 1 or higher.

		Description Extraction				
D	**N**	**TP**	**FP**	**Prec**	**Rec**	**F1**
0	1	59	32	.648	.184	.286
0	3	136	95	.589	.424	.493
0	**6**	**155**	**150**	**.508**	**.483**	**.495**
0	15	167	190	.468	.520	.493
1	1	123	211	.368	.383	.376
1	3	179	602	.229	.558	.325
1	6	210	1107	.159	.654	.256
2	1	122	210	.367	.379	.373
2	3	179	600	.230	.556	.325

(b) Relevance 2.

		Description Extraction				
D	**N**	**TP**	**FP**	**Prec**	**Rec**	**F1**
0	1	41	59	.451	.210	.287
0	**3**	**82**	**149**	**.355**	**.421**	**.385**
0	6	90	215	.295	.462	.360
0	15	95	262	.266	.487	.344
1	1	82	252	.246	.421	.310
1	3	106	675	.139	.544	.217
1	6	124	1193	.094	.636	.164
2	1	56	227	.198	.287	.234
2	3	88	661	.117	.451	.186

(c) ✓ matches the benchmark entry.

Semantic LaTeX Comparison				
Method	**D**	**N**	✓	✘
`base`	-	-	.16	.84
`ck`	-	-	.29	.71
`full`	0	3	.36	.64
	0	6	.40	.60
	0	15	.40	.60
	1	3	.43	.57
	1	**6**	**.48**	**.52**
	1	15	.45	.55
	1	20	.44	.56

In all tables, **D** *refers to the depth (following ingoing dependencies) in the dependency graph,* **N** *is the maximum number of facts and r_f for the same MOI,* **TP** *are true positives, and* **FP** *are false positives. The methods* `base` *refers to no transformations $\mathrm{t}(e, X) = e$,* `ck` *where $X = \mathcal{K}$, and* `full` *use the full proposed pipeline.*

and perform tree comparisons of our generated transformed tree $\mathrm{t}_s(e, X)$ and the semantically enhanced tree using semantic macros. The number of facts we take into account affects the performance. Fewer facts and the transformation might be not complete, i.e., there are still subtrees in e that should be already in $\mathcal{L}_C$. Too many facts increase the risk of false positives, that yield wrong transformations. In order to estimate how many facts we need to retrieve to achieve a complete transformation, we evaluated the comparison on different depths D and limit the number of facts with the same MOI, i.e., we only consider the top-ranked facts f for an MOI according to $\mathrm{s}_{\mathrm{MLP}}(f)$. In addition, we limit the number of retrieved rules r_f per MC. We observed that an equal limit of retrieved MC per MOI and r_f per MC performed best. Consider we set the limit N to five, we would retrieve a maximum of 25 facts (five r_f for each of the five MC for a single MOI). Typically, the number of retrieved facts f is below this limit because similar MC yield similar r_f. In addition, we found that considering replacement patterns with a likelihood of 0% (i.e., the rendered version of this macro never appears in the DLMF), harms performance drastically. This is because semantic macros without any arguments regularly match single letters, for example, Γ representing the gamma function with the argument (z) being omitted. Hence, we decided to consider only replacement patterns that exist in the DLMF, i.e., $\mathrm{s}_{\mathrm{DLMF}}(r_f) > 0$.

Since certain subtrees $\tilde{e} \subseteq e \in \mathcal{L}_P$ can be already operator trees, i.e., $\tilde{e} \in \mathcal{L}_C$, we calculate a baseline (`straight`) that does not perform any transformations, i.e., $e = \mathrm{t}(e, X)$. The baseline achieves a success rate of 16%. To estimate the impact of our manually defined set of common knowledge facts $\mathcal{K}$, we also evaluated the transformations for $X = \mathcal{K}$ and achieve a success rate of 29% which is already significantly better compared to the baseline. The full pipeline, as described above, achieves a success rate of 48%. Table 2 c compares the performance. The table shows that depth 1 outperforms depth 0, which intuitively contradicts the F1 scores in Table 2 a. This underlines the necessity of the dependency graph. We further examine a drop in the success rate for larger N. This is attributable to the fact that $g_f(e)$ is not commutative and large N retrieve too many false positive facts f with high ranks. We reach the best success rate for depth 1 and $\mathrm{N} = 6$. Increasing the depth further only has a marginal impact because, at depth 2, most expressions are already single identifiers, which do not provide significant information for the translation process.

3.3.3 *Translations From LaTeX to CAS*

Mathematica's ability to import TeX expressions will serve as a baseline. While Mathematica does not allow to enter a textual context, it does recognize structural information in the expression. For example, the Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$ is correctly imported as `JacobiP[n,\[Alpha],\[Beta], x]` because no other supported function in Mathematica is linked with this presentation. Table 3 b compares the performance. The methods LaCASt`_base, ck, full` are the same as in Table 2 c, but now refer to translations to Mathematica, rather than semantic LaTeX. LaCASt`_full` uses the optimal setting as shown in Table 2 c. We consider a translation a *match* (✓) if the returned value by Mathematica equals the returned value by the benchmark. The internal process of Mathematica ensures that the translation is normalized.

We observe that without further improvements, LaCASt already outperforms Mathematica's internal import function. Activating the general replacement rules further improved performance. Our full context-aware pipeline achieves the best results. The relatively high ratio of invalid translations for LaCASt`_full` is owed to the fact that semantic macros without an appropriate translation to Mathematica result in an error during the translation process. The errors ensure that LaCASt only performs translations for semantic LaTeX if a translation is unambiguous and possible for the containing functions [16]. Note that we were not able to appropriately translate 18 expressions (indicated by the human performance in Table 3 b) as discussed before.

4 Error Analysis & Discussion

In this section, we briefly summarize the main causes of errors in our translation pipeline. A more extensive analysis can be found in the supplementary materials, available online, and on our demo page at: https://tpami.wmflabs.org. In the following, we may refer to specific benchmark entries with the associated ID. Since the benchmark contains randomly picked formulae from the articles, it also contains entries that might not have been properly annotated with math templates or math-tags in the Wikitext. Four entries in the benchmark (28, 43, 78, and 85) were wrongly detected by our engine and

TABLE 3
The Symbolic and Numeric Evaluations (Left) and the Benchmark Evaluation for Translations to Mathematica (Right)

(a) Auto evaluation on all $6,337$ expressions from the dataset with the number of translated expressions (T), the number of started test evaluations (Started), the success rates (Success), and the success rates on the DLMF dataset for comparison (DLMF).

Symbol Evaluation

	T	Started	Success	DLMF
Maple	$4,601$	$1,747$	.113	.264
Mathematica	$4,678$	$1,692$	.158	.262

Numeric Evaluation

	T	Started	Success	DLMF
Maple	$4,601$	$1,627$	.181	.433
Mathematica	$4,678$	$1,516$	.236	.429

(b) Performance comparison for translating LaTeX to Mathematica. A translation was successful (**ST**) if it was syntactically verified by Mathematica. ✓ refers to matches with the benchmark and ✘ to mismatches. The methods are explained in Section 3.3.3.

LaTeX Translations to Mathematica

Method	ST	FT	✓	✘
`MM_import`	57 (.60)	38 (.40)	9 (.09)	48 (.51)
LACAST`_straight`	55 (.58)	40 (.42)	11 (.12)	44 (.46)
LACAST`_rules`	62 (**.65**)	33 (**.35**)	19 (.20)	43 (.45)
LACAST`_full`	53 (.56)	42 (.44)	**26** (**.27**)	**27** (**.26**)
`Theory_def`	-	-	+18 (.19)	-18 (.19)
`Theory_ck`	-	-	+3 (.03)	-3 (.03)
Human	95 (1.0)	0 (.00)	77 (.81)	18 (.19)

contained only parts of the entire formula. In the benchmark, we manually corrected these entries. Aside from the wrong identification, we identified other failure reasons for a translation to semantic LaTeX or CAS. In the following, we discuss the main reasons and possible solutions to avoid them, in order of their impact on translation performance.

Definitions. Recognizing an equation as a definition would have a great impact on performance. As a test, we manually annotated every definition in the benchmark by replacing the equal sign $=$ with the unambiguous notation $:=$ and extended LACAST to recognize such combination as a definition of the left-hand side[14]. This resulted in 18 more correct translations (e.g., 66, 68, and 75) and increased the performance from .28 to .47. The accuracy for this manual improvement is given as `Theory_def` in Table 3 b.

The dependency graph may provide beneficial information towards a definition recognition system for equations. However, rather than assuming that every equation symbol indicates a definition [23], we propose a more selective approach. Considering one part of an equation (including multi-equations) as an extra MOI would establish additional dependencies in the dependency graph, such as a connection between $x = \mathrm{sn}(u, k)$ and $F(x; k) = u$. A combination with recent advances of definition recognition in NLP [20], [36], [37], [38] may then allow us to detect x as the defining element. The already established dependency between x and $F(x; k) = u$ can finally be used to resolve the substitution. Hence, for future research, we will elaborate on the possibility of integrating existing NLP techniques for definition recognition [36], [37] into our dependency graph concept.

Missing Information. Another problem that causes translations to fail is missing facts. For example, the gamma function seems to be considered common knowledge in most articles on OPSF because it is often not specifically declared by name in the context (e.g., 19 or 31). To test the impact of considering the gamma function as common knowledge, we added a rule r_f to $\mathcal{K}$ and attached a low rank to it. The low rank ensures the pattern for the gamma function will be applied late in the list of transformations. This indeed improved performance slightly, enabling a successful translation of three more benchmark entries (`Theory_ck` in Table 3 b). This naive approach, emphasizes the importance of knowing the domain knowledge for specific articles. In combination with article classifications [17], we could activate different common knowledge sets depending on the specific domain.

Non-Matching Replacement Patterns. An issue we would more regularly faced in domains other than OPSF is non-standard notations. As previously mentioned, without definition detection, we would not be able to derive transformation rules if the MOI is not given in a standard notation, such as $p(a, b, n, z)$ for the Jacobi polynomial. This already happens for slight changes that are not covered by the DLMF. For six entries, for instance, we were unable to appropriately replace hypergeometric functions because they used the matrix and array environments in their arguments, while the DLMF (as shown in Table 1) only uses `\atop` for the same visualization. Consequently, none of our replacement patterns matched even though we correctly identified the expressions as hypergeometric functions. A possible solution to this kind of minor representational changes might be to add more possible presentational variants m for a semantic macro $\widetilde{m}$. Previously [24], we presented a search engine for MOI that allows searching for common notations for a given textual query. Searching for Jacobi polynomials in arXiv.org shows that different variants of $P_n^{(\alpha,\beta)}(x)$ are highly related or even equivalently used, such as p, H, or R rather than P. There were also a couple of other minor issues we identified during the evaluation, such as synonyms for function names, derivative notations, or non-existent translations for semantic macros. Non-existent translation patterns for semantic macros are also the main reason why our LaTeX to semantic LaTeX translator performed significantly better than the translations to Mathematica. We provide more information on these cases on our demo page.

Implementing the aforementioned improvements will increase the score from .26 (26 out of 95) to .495 (47 out of 95) for translations from LaTeX to Mathematica. We achieved these results based on several heuristics, such as the primary identifier rules or the general replacement patterns, which indicates that we may improve results even further with ML algorithms. However, a missing properly annotated dataset and no appropriate error functions made it difficult to achieve promising results with ML on mathematical translation tasks in the past [44], [45]. Our translation pipeline based on LACAST paves the way towards a baseline that can be used to train ML models in the future. Hence, we will focus on a hybrid approach of rule-based translations via LACAST on the one hand, and ML-based information extraction on the other hand, to further push the limits of our translation pipeline.

14. The DLMF did not use this notation, hence LACAST was not capable of translating $:=$ in the first place.

5 Conclusion & Future Work

We presented LACAST, the first context-sensitive translation pipeline for mathematical expressions to the syntax of two major Computer Algebra Systems (CAS), Maple and Mathematica. We demonstrated that the information we need to translate is given as noun phrases in the textual context surrounding a mathematical formula and common knowledge databases that define notation conventions. We successfully extracted the crucial noun phrases via part-of-speech tagging. Further, we have shown that CAS can automatically verify the translated expressions by performing symbolic and numeric computations. In an evaluation with 104 Wikipedia articles in the domain of orthogonal polynomials and special functions, we verified 358 formulae using our approach. We identified one malicious edit with this technique, which was reverted by the community three days later. We have shown that LACAST correctly translates about 27% of mathematical formulae compared to 9% with existing approaches and a 81% human baseline. Further, we demonstrated a potential successful translation rate of 46% if LACAST can identify definitions correctly and 49% with a more comprehensive common knowledge database.

Our translation pipeline has several practical applications for a knowledge database like Wikipedia, such as improving the readability [3] and user experience [13], and enabling entity linking [3], [17] and automatic quality checks via CAS [7], [8]. In turn, we plan to integrate [46] our evaluation engine into the existing ORES system to classify changes in complex mathematical equations as potentially damaging or good faith. In addition, the system provides access to different semantic formats of a formula, such as multiple CAS syntaxes and semantic LaTeX [10]. As shown in the DLMF, the semantic encoding of a formula can improve search results for mathematical expressions significantly. Hence, we also plan to add the semantic information from our mathematical dependency graph to Wikipedia's math formulae to improve search results [3].

In future work, we aim to mitigate the issues outlined in Section 3, primarily focusing our efforts on definition recognitions for mathematical equations. Advances on this matter will enable the support for translations beyond OPSF. In particular, we plan to analyze the effectiveness of associating equations with their nearby context classification [20], [36], [37], [38], assuming a defining equation is usually embedded in a definition context. Apart from expanding the support beyond OPSF, we further focus on improving the verification accuracy of the symbolic and numeric evaluation pipeline. In contrast to the evaluations on the DLMF, our evaluation pipeline currently disregards constraints in Wikipedia. While most constraints in the DLMF directly annotate specific equations, Wikipedia contains constraints in the surrounding context of the formula. We plan to identify constraints with new pattern matches and distance metrics, by assuming that constraints are often short equations (and relations) or set definitions and appear shortly after or before the formula they are applied to. While we made math in Wikipedia computable, the encyclopedia does not take advantage of this new feature yet. In future work, we will develop an AI [46] (as an extension to the existing ORES system) that makes use of this novel capability.

Acknowledgments

We thank Abdou Youssef who provided us access to the Part-of-Math tagger, Howard Cohl for his expertise in regard of special functions and the DLMF, and Jürgen Gerhard for the support from Maplesoft.

References

[1] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila, "NTCIR-12 MathIR task overview," in *Proc. NTCIR Conf. Eval. Inf. Access Technol.*, 2016, pp. 299–308.
[2] X. Hu, L. Gao, X. Lin, Z. Tang, X. Lin, and J. B. Baker, "WikiMirs: A mathematical information retrieval system for Wikipedia," in *Proc. IEEE/ACM-CS Joint Conf. Digit. Libraries*, 2013, pp. 11–20.
[3] M. Schubotz, A. Greiner-Petter, N. Meuschke, O. Teschke, and B. Gipp, "Mathematical formulae in Wikimedia projects 2020," in *Proc. IEEE/ACM Joint Conf. Digit. Libraries*, 2020, pp. 447–448.
[4] A. Halfaker and R. S. Geiger, "ORES: Lowering barriers with participatory machine learning in Wikipedia," *Proc. ACM Hum.-Comput. Interact.*, vol. 4, pp. 148:1–148:37, 2020.
[5] L. Bernardin *et al.*, *Maple Programming Guide*. Waterloo, ON, Canada: Maplesoft, 2011.
[6] S. Wolfram, *The Mathematica Book*, 5th ed. Champaign, IL, USA: Wolfram Media, 2003.
[7] H. S. Cohl, A. Greiner-Petter, and M. Schubotz, "Automated symbolic and numerical testing of DLMF formulae using computer algebra systems," in *Proc. Int. Conf. Intell. Comput. Math.*, 2018, pp. 39–52.
[8] A. Greiner-Petter *et al.*, "Comparative verification of the digital library of mathematical functions and computer algebra systems," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2022, pp. 87–105.
[9] N. TeBlunthuis, "Measuring Wikipedia article quality in one dimension by extending ORES with ordinal regression," 2021, *arXiv:2108.10684*.
[10] B. R. Miller and A. Youssef, "Technical aspects of the digital library of mathematical functions," *Ann. Math. Artif. Intell.*, vol. 38, no. 1–3, pp. 121–136, 2003.
[11] *NIST Digital Library of Mathematical Functions*," F. W. J. Olver *et al.*, Eds., Gaithersburg, Md. USA: NIST, 2020. [Online]. Available: http://dlmf.nist.gov/
[12] A. Youssef and B. R. Miller, "A contextual and labeled math-dataset derived from NIST's DLMF," in *Proc. Int. Conf. Intell. Comput. Math.*, 2020, pp. 324–330.
[13] A. Head *et al.*, "Augmenting scientific papers with just-in-time, position-sensitive definitions of terms and symbols," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2021, pp. 413:1–413:18.
[14] M. Wolska and M. Grigore, "Symbol declerations in mathematical writing - a corpus study," in *Proc. Towards Digit. Math. Library DML Workshop*, 2010, pp. 119–127. [Online]. Available: https://dml.cz/handle/10338.dmlcz/702580
[15] M. Schubotz *et al.*, "Semantification of identifiers in mathematics for better math information retrieval," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 135–144.
[16] A. Greiner-Petter, M. Schubotz, H. S. Cohl, and B. Gipp, "Semantic preserving bijective mappings for expressions involving special functions in computer algebra systems and document preparation systems," *Aslib J. Inf. Manage.*, vol. 71, no. 3, pp. 415–439, 2019.
[17] P. Scharpf, M. Schubotz, H. S. Cohl, and B. Gipp, "Towards formula concept discovery and recognition," in *Proc. 4th Joint Workshop Bibliometric-Enhanced Inf. Retrieval Natural Lang. Process. Digit. Libraries 42nd ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 108–115. [Online]. Available: http://ceur-ws.org/Vol-2414/paper11.pdf
[18] G. Y. Kristianto, G. Topic, and A. Aizawa, "Extracting textual descriptions of mathematical expressions in scientific papers," *D-Lib Mag.*, vol. 20, no. 11/12, 2014, Art. no. 9.
[19] R. Pagel and M. Schubotz, "Mathematical language processing project," in *Proc. Int. Conf. Intell. Comput. Math.*, 2014. [Online]. Available: http://ceur-ws.org/Vol-1186/paper-23.pdf
[20] D. Kang, A. Head, R. Sidhu, K. Lo, D. S. Weld, and M. A. Hearst, "Document-level definition detection in scholarly documents: Existing models, error analyses, and future directions," in *Proc. 1st Workshop Scholarly Document Process.*, 2020, pp. 196–206.
[21] P.-Y. Chien and P.-J. Cheng, "Semantic tagging of mathematical expressions," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 195–204.
[22] A. Youssef, "Part-of-math tagging and applications," in *Proc. Int. Conf. Intell. Comput. Math.*, 2017, pp. 356–374.

[23] G. Y. Kristianto, G. Topic, and A. Aizawa, "Utilizing dependency relationships between math expressions in math IR," *Inf. Retrieval J.*, vol. 20, no. 2, pp. 132–167, 2017.

[24] A. Greiner-Petter *et al.*, "Discovering mathematical objects of interest - A study of mathematical notations," in *Proc. Int. Conf. World Wide Web*, 2020, pp. 1445–1456.

[25] M.-Q. Nghiem, G. Y. Kristianto, and A. Aizawa, "Using MathML parallel markup corpora for semantic enrichment of mathematical expressions," *IEICE Trans. Inf. Syst.*, vol. 96-D, no. 8, pp. 1707–1715, 2013.

[26] M. Schubotz, A. Greiner-Petter, P. Scharpf, N. Meuschke, H. S. Cohl, and B. Gipp, "Improving the representation and conversion of mathematical formulae by considering their textual context," in *Proc. 18th ACM/IEEE Joint Conf. Digit. Libraries*, 2018, pp. 233–242.

[27] P.-E. Moreau, C. Ringeissen, and M. Vittek, "A pattern matching compiler for multiple target languages," in *Proc. Int. Conf. Compiler Construction*, 2003, pp. 61–76.

[28] M. Kohlhase and F. Rabe, "Semantics of OpenMath and MathML3," *Math. Comput. Sci.*, vol. 6, no. 3, pp. 235–260, 2012.

[29] K. Davila and R. Zanibbi, "Layout and semantics: Combining representations for mathematical formula search," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 1165–1168.

[30] H. S. Cohl *et al.*, "Semantic preserving bijective mappings of mathematical formulae between document preparation systems and computer algebra systems," in *Proc. Int. Conf. Intell. Comput. Math.*, 2017, pp. 115–131.

[31] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations*, 2014, pp. 55–60.

[32] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2003, pp. 173–180.

[33] H. Dohrn and D. Riehle, "Design and implementation of the Sweble Wikitext parser: Unlocking the structured data of Wikipedia," in *Proc. Int. Symp. Wikis Open Collaboration*, 2011, pp. 72–81.

[34] L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang, "A deep learning-based formula detection method for PDF documents," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 553–558.

[35] X. Wang, Z. Wang, and J.-C. Liu, "Bigram label regularization to reduce over-segmentation on inline math expression detection," in *Proc. Int. Conf. Document Anal. Recognit.*, 2019, pp. 387–392.

[36] L. E. Anke and S. Schockaert, "Syntactically aware neural architectures for definition extraction," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 378–385.

[37] D. Ginev and B. R. Miller, "Scientific statement classification over arXiv.org," in *Proc. 12th Lang. Resour. Eval. Conf.*, 2020, pp. 1219–1226. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.153/

[38] N. Vanetik, M. Litvak, S. Shevchuk, and L. Reznik, "Automated discovery of mathematical definitions in text," in *Proc. 12th Lang. Resour. Eval. Conf.*, 2020, pp. 2086–2094. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.256/

[39] M. Schubotz, L. Krämer, N. Meuschke, F. Hamborg, and B. Gipp, "Evaluating and improving the extraction of mathematical identifier definitions," in *Proc. Int. Conf. Cross-Lang. Eval. Forum Eur. Lang.*, 2017, pp. 82–94.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[41] *GAP – Groups, Algorithms, and Programming, v.4.11.0*, The GAP Group, 2020. [Online]. Available: https://www.gap-system.org

[42] *PARI/GP v.2.11.2*, The PARI Group, Univ. Bordeaux, 2019. [Online]. Available: http://pari.math.u-bordeaux.fr/

[43] K. Peeters, "Cadabra: A field-theory motivated symbolic computer algebra system," *Comput. Phys. Commun.*, vol. 176, no. 8, pp. 550–558, 2007.

[44] A. Greiner-Petter *et al.*, "Math-word embedding in math search and semantic extraction," *Scientometrics*, vol. 125, no. 3, pp. 3017–3046, 2020.

[45] T. Asakura, A. Greiner-Petter, A. Aizawa, and Y. Miyao, "Towards grounding of formulae," in *Proc. 1st Workshop Scholarly Document Process.*, 2020, pp. 138–147.

[46] Z. Ye, X. Yuan, S. Gaur, A. Halfaker, J. Forlizzi, and H. Zhu, "Wikipedia ORES explorer: Visualizing trade-offs for designing applications with machine learning API," in *Proc. Designing Interactive Syst. Conf.*, 2021, pp. 1554–1565.

André Greiner-Petter received the BSc and MSc degrees in mathematics with a focus on algorithmic and discrete mathematics from TU Berlin, Berlin, Germany, in 2014 and 2017, respectively. He received the PhD degree in computer science from the University of Wuppertal, Germany, in 2022. He has been a guest researcher at the National Institute of Standards and Technology (NIST) in the US and is a postdoctorate fellow of the National Institute of Informatics (NII), Tokyo, Japan.

Moritz Schubotz received the PhD degree from TU Berlin, in 2017 and has been postdoctoral fellow with the University of Konstanz, the NII in Tokyo, and the University of Wuppertal, before joining zbMATH. He is a senior researcher for mathematical information retrieval and decentralized open science with zbMATH. He maintains the support for mathematical formulae in Wikipedia and is an off-site collaborator with NIST.

Corinna Breitinger received the BSc degree from the University of California, Berkeley, in 2011, and the MSc degree in media and web technology from Linnaeus University, Sweden, in 2016. She is currently working toward the PhD degree in information science with the University of Konstanz, Germany. She also worked in the Silicon Valley as an associate researcher for a tech-startup, and completed a six-month research appointment with the NII, in Japan.

Philipp Scharpf received the BSc degree in physics from the University of Zurich, Switzerland, and the MSc degree from the University of Konstanz, Germany with a focus on general relativity. Currently, he is working toward the PhD degree in computer and information sciences with the University of Konstanz, Germany. During his research career, he has been a guest researcher with the University of Cambridge, United Kingdom.

Akiko Aizawa received the graduate degree from the Department of Electronics, University of Tokyo, in 1985, and the doctoral degree in electrical engineering, in 1990. At present, she is a professor with the NII, an adjunct professor with the Department of Computer Science, University of Tokyo, and also a professor with the Department of Informatics, Graduate University for Advanced Studies (Sokendai).

Bela Gipp is currently a full professor of scientific information analytics with the University of Göttingen, Germany. He held professorships with the Universities of Konstanz and Wuppertal. He completed his postdoctoral research with the University of California, Berkeley, CA, USA, and the NII, Tokyo, Japan. His research interests include intersection of information science and data science, with a primary focus on information retrieval and natural language processing.