RESEARCH

André Greiner-Petter

Making Presentation Math Computable

A Context-Sensitive Approach for Translating LaTeX to Computer Algebra Systems

OPEN ACCESS

Springer Vieweg

Making Presentation Math Computable

André Greiner-Petter

Making Presentation Math Computable

A Context-Sensitive Approach for Translating LaTeX to Computer Algebra Systems

Springer Vieweg

André Greiner-Petter
Berlin, Germany

ISBN 978-3-658-40472-7 ISBN 978-3-658-40473-4 (eBook)
https://doi.org/10.1007/978-3-658-40473-4

This Springer Vieweg imprint is published by the registered company Springer Fachmedien Wiesbaden GmbH, part of Springer Nature.
The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Front Matter

Contents

List of Figures

List of Tables

FRONT MATTER

Abstract

This thesis addresses the issue of translating mathematical expressions from LaTeX to the syntax of Computer Algebra Systems (CAS). Over the past decades, especially in the domain of Science, Technology, Engineering, and Mathematics (STEM), LaTeX has become the de-facto standard to typeset mathematical formulae in publications. Since scientists are generally required to publish their work, LaTeX has become an integral part of today's publishing workflow. On the other hand, modern research increasingly relies on CAS to simplify, manipulate, compute, and visualize mathematics. However, existing LaTeX import functions in CAS are limited to simple arithmetic expressions and are, therefore, insufficient for most use cases. Consequently, the workflow of experimenting and publishing in the Sciences often includes time-consuming and error-prone manual conversions between presentational LaTeX and computational CAS formats.

To address the lack of a reliable and comprehensive translation tool between LaTeX and CAS, this thesis makes the following three contributions.

First, it provides an approach to semantically enhance LaTeX expressions with sufficient semantic information for translations into CAS syntaxes. This, so called, *semantification* process analyzes the structure of the formula and its textual context to conclude semantic information. The research for this semantification process additionally contributes towards related Mathematical Information Retrieval (MathIR) tasks, such as mathematical education assistance, math recommendation and question answering systems, search engines, automatic plagiarism detection, and math type assistance systems.

Second, this thesis demonstrates the first context-aware LaTeX to CAS translation framework LaCaST. LaCaST uses the developed semantification approach to transform LaTeX expressions into an intermediate semantic LaTeX format, which is then further translated to CAS based on translation patterns. These patterns were manually crafted by mathematicians to assure accurate and reliable translations. In comparison, this thesis additionally elaborates a non-context aware neural machine translation approach trained on a mathematical library generated by Mathematica.

Third, the thesis provides a novel approach to evaluate the performance for LaTeX to CAS translations on large-scaled datasets with an automatic verification of equations in digital mathematical libraries. This evaluation approach is based on the assumption that equations in digital mathematical libraries can be computationally verified by CAS, if a translation between both systems exists. In addition, the thesis provides an in-depth manual evaluation on mathematical articles from the English Wikipedia.

The presented context-aware translation framework LaCaST increases the efficiency and reliability of translations to CAS. Via LaCaST, we strengthened the Digital Library of Mathematical Functions (DLMF) by identifying numerous of issues, from missing or wrong semantic annotations to sign errors. Further, via LaCaST, we were able to discover several issues with the commercial CAS Maple and Mathematica. The fundamental approaches to semantically enhance mathematics developed in this thesis additionally contributed towards several related MathIR tasks. For

instance, the large-scale analysis of mathematical notations and the studies on math-embeddings motivated new approaches for math plagiarism detection systems, search engines, and allow typing assistance for mathematical inputs. Finally, LaCasT translations will have a direct real-world impact, as they are scheduled to be integrated into upcoming versions of the DLMF and Wikipedia.

FRONT MATTER

Zusammenfassung

Diese Dissertation befasst sich mit der Problematik von Übersetzungen mathematischer Formeln zwischen LaTeX und Computeralgebrasystemen (CAS). Im Laufe des digitalen Zeitalters wurde LaTeX zum Quasistandard für das Schreiben mathematischer Formeln auf dem Computer, insbesondere in den Disziplinen Mathematik, Informatik, Naturwissenschaften und Technik (MINT). Da Wissenschaftler gemeinhin ihre Arbeit publizieren, ist LaTeX zu einem integralen Bestandteil moderner Forschung geworden. Gleichermaßen verlassen sich Wissenschaftler immer mehr auf die Möglichkeiten moderner CAS, um effektiv mit mathematischen Formeln zu arbeiten, zum Beispiel, indem sie diese umformen, lösen oder auch visualisieren. Die momentanen Ansätze, welche eine Übersetzung von LaTeX zu CAS erlauben, wie beispielsweise interne Import-Funktionen einiger CAS, sind jedoch häufig auf einfache arithmetische Ausdrücke beschränkt und daher nur wenig hilfreich im realen Arbeitsalltag. Infolgedessen ist die Arbeit moderner Wissenschaftler in den MINT Disziplinen häufig geprägt von zeitraubenden und fehleranfälligen manuellen Übersetzungen zwischen LaTeX und CAS.

Die vorliegende Dissertation leistet die folgenden Beiträge, um das Problem des Übersetzens von mathematischen Ausdrücken zwischen LaTeX und CAS zu lösen.

Zunächst ist LaTeX ein Format, welches lediglich die visuelle Präsentation mathematischer Ausdrücke kodiert, nicht jedoch deren semantische Informationen. Die semantischen Informationen sind jedoch notwendig für CAS, welche keine mehrdeutigen Eingaben erlauben. Daher führt die vorliegende Arbeit als ersten Schritt für eine Übersetzung eine sogenannte Semantifizierung mathematischer Ausdrücke ein. Diese Semantifizierung extrahiert semantische Informationen aus dem Kontext und den Bestandteilen der Formel, um Rückschlüsse auf ihre Bedeutung zu ziehen. Da die Semantifizierung eine klassische Aufgabe auf dem Gebiet der mathematischen Informationsgewinnung darstellt, leistet dieser Teil der Dissertation auch Beiträge zu verwandten Themengebieten. So sind die hier vorgestellten Ansätze auch nützlich für pädagogische Programme, Frage-Antwort Systeme, Suchmaschinen und die digitale Plagiatserkennung.

Als zweiten Beitrag, stellt die vorliegende Dissertation das erste kontextbezogene LaTeX zu CAS Übersetzungsprogramm vor, genannt LaCaSt. LaCaSt nutzt die zuvor eingeführte Semantifizierung, um LaTeX in ein Zwischenformat zu transformieren, welches die semantischen Informationen explizit darstellt. Dieses Format wird semantisches LaTeX genannt, da es eine technische Erweiterung von LaTeX ist. Die weitere Übersetzung zu CAS wird durch heuristische Übersetzungsmuster für mathematische Funktionen realisiert. Diese Übersetzungsmuster wurden in Zusammenarbeit mit Mathematikern definiert, um eine korrekte Übersetzung in diesem letzten Schritt zu gewährleisten. Um die Vorzüge einer kontextbezogenen Übersetzung besser zu verstehen, stellt diese Arbeit zum Vergleich auch eine Maschinenübersetzung auf neuronalen Netzen vor, welche den Kontext einer Formel nicht berücksichtigt.

Der dritte Beitrag dieser Dissertation führt eine neue Methode zur Evaluierung von mathematischen Übersetzungen ein, welche es erlaubt, auch eine große Anzahl an Übersetzungen auf ihre Korrektheit hin zu überprüfen. Diese Methode folgt dem Ansatz, dass Gleichungen

in mathematischen Bibliotheken auch nach der Übersetzung in ein CAS noch korrekt sein müssten. Ist dies nicht der Fall, ist entweder die Ausgangsgleichung, die Übersetzung, oder das CAS fehlerhaft. Hierbei ist zu beachten, dass jede Fehlerquelle einen Mehrwert für das jeweilige System darstellt. Zusätzlich zu dieser automatischen Evaluierung, erfolgt noch eine manuelle Analyse von Übersetzungen auf Basis englischer Wikipedia Artikel.

Zusammenfassend ermöglicht das kontextbezogene Übersetzungsprogramm LaCaSt eine effizientere Arbeitsweise mit CAS. Mit Hilfe dieser Übersetzungen konnten auch mehrere Probleme, wie falsche Informationen oder Vorzeichenfehler, in der Digital Library of Mathematical Functions (DLMF) sowie Fehler in den kommerziell vertriebenen CAS Maple und Mathematica automatisch aufgedeckt und behoben werden.

Die hier vorgestellte Grundlagenforschung zum semantischen Anreichern mathematischer Ausdrücke, hat zudem etliche Beiträge zu verwandten Forschungsthemen geleistet. Zum Beispiel hat die Analyse der Verteilung von mathematischen Notationen in großen Datensätzen neue Ansätze in der digitalen Plagiatserkennung ermöglicht. Des Weiteren wird zurzeit daran gearbeitet, die Übersetzungen von LaCaSt in kommende Versionen von Wikipedia und der DLMF zu integrieren.

FRONT MATTER

Acknowledgements

This thesis would not have been possible without the tremendous help and support from numerous family members, friends, colleagues, supervisors, and several international institutions. In the following, I want to take the opportunity to thank all the individuals and organizations that helped me along the way to make this work possible.

My first sincere wishes go to my prodigious doctoral advisers Bela Gipp and Akiko Aizawa. Their continuous support and counsel enabled me to realize this thesis at marvelous places and together with numerous wonderful people from all over the world. Their enduring encouragement and assistance, Bela's abiding and infectious positivity, and Akiko's steadfast and kind endorsement empowered my personal and professional life. Both of their competent and sincere guidance helped me to find my way in the intricate maze of research and career decisions and turned my often onerous time into a joyful and memorable experience.

Moreover, I am very grateful to my adviser and friend Moritz Schubotz, who supported and guided me throughout the entire time of my doctoral thesis and even beyond. Our fruitful and always engaging discussions, even when exhausting, enriched and positively affected most, if not all, of my work. It is not an exaggeration to admit that my career, including my Master's thesis and this doctoral thesis, would not have been possible and nearly as successful and joyful as it has been without his continuous and sincere support over the years. I am wholeheartedly thankful for all the years we worked together.

I further wish to gratefully acknowledge my friends, colleagues, and advisers Howard Cohl, Abdou Youssef, and Bruce Miller at the National Institute of Standards and Technology (NIST) for their valuable advice, continuous drive to perfection, and our rewarding collaborations. I thank Jürgen Gerhard at Maplesoft, who kindly provided me access and support for Maple on several occasions. I am just as thankful for the assistance and support from Norman Meuschke, who always helped me to overcome governmental and organizational hurdles, Corinna Breitinger, who never failed to refit my gibberish, and my colleagues and friends Terry Lima Ruas and Philipp Scharpf for many visionary discussions. I also thank all my collaborators and colleagues with whom I had the distinct opportunity to work together, including Takuto Asakura, Fabian Müller, Olaf Teschke, William Grosky, Marjorie McClain, Yusuke Miyao, Malte Ostendorff, Bonita Saunders, Kenichi Iwatsuki, Takuma Udagawa, Anastasia Zhukova, and Felix Hamborg. I further want to thank the students I worked with, including Avi Trost, Rajen Dey, Joon Bang, Kevin Chen, and Felix Petersen. I especially appreciate the help and assistance from people at the National Institute of Informatics (NII) to overcome governmental and daily life issues. I wish to especially thank Rie Ayuzawa, Noriko Katsu, Akiko Takenaka, and Goran Topic.

My genuine gratitude also goes to my host organizations and those that provided financial support for my research. I am thankful for the German Academic Exchange Program (DAAD) for enabling two research stays at the NII in Tokyo, the NII for providing me a wonderful work environment, the German Research Foundation (DFG) to financially support many of my projects, the NIST for hosting me as a guest researcher, and Maplesoft for offering me

an internship during my preliminary research project on the Digital Library of Mathematical Functions (DLMF). I finally thank the ACM Special Interest Group on Information Retrieval (SIGIR), the University of Konstanz, the University of Wuppertal, and Maplesoft for supporting several conference participations.

My last and most crucial gratitude goes to my family and friends, who always cheered me in good and bad times and constantly backed and supported me so that I could selfishly pursue my dreams. I am deeply grateful for my lovely parents Rolf & Regina, who have always been on my side and make all this possible behind the scenes. I am also tremendously thankful for the enduring personal support from my dear friends Kevin, Lena, Vici, Dong, Peter, Vitor, Ayuko, and uncountably more. Finally, I thank my lovely partner Aimi for brightening even the darkest times and pushing every possible obstacle aside. I dedicate this thesis to my lovely parents, my dear friends, and my enchanting girlfriend.

I went to the woods because I wanted to live deliberately. I wanted to live deep and suck out all the marrow of life. To put to rout all that was not life; and not, when I had come to die, discover that I had not lived.

Neil Perry - *Dead Poet Society*

CHAPTER 1

Introduction

Contents

This thesis addresses the issue of translating mathematical expressions from LaTeX to the syntax of Computer Algebra Systems (CAS), which is typically a time-consuming and error-prone task in the modern life of many researchers. A reliable and comprehensive translation approach requires analyzing the textual context of mathematical formulae. In turn, research advances in translating LaTeX contribute directly towards related tasks in the Mathematical Information Retrieval (MathIR) arena. In this chapter, I provide an introduction to the topic. Section 1.1 introduces my motivation and provides an overview of the problem. Section 1.2 summarizes the research gap. In Section 1.3, I define the research objective and research tasks of this thesis. Section 1.4 concludes with an outline of the thesis including an overview of the publications that contributed to the goals of this thesis and the research path that led to these publications.

1.1 Motivation & Problem

Consider a researcher is working on Jacobi polynomials and examines the existing English Wikipedia article about the topic[1]. While she might be familiar with the Digital Library of Mathematical Functions (DLMF) [98], a standard resource for Orthogonal Polynomials and Special Functions (OPSF), the equation 1.1 from the article might be new to her

$$P_n^{(\alpha,\beta)}(x) = \frac{\Gamma(\alpha+n+1)}{n!\,\Gamma(\alpha+\beta+n+1)} \sum_{m=0}^{n} \binom{n}{m} \frac{\Gamma(\alpha+\beta+n+m+1)}{\Gamma(\alpha+m+1)} \left(\frac{z-1}{2}\right)^m. \tag{1.1}$$

In order to analyze this new equation, e.g., to validate it, she wants to use CAS. CAS are powerful mathematical software tools with numerous applications [207]. Today's most widely

[1] https://en.wikipedia.org/wiki/Jacobi_polynomials [accessed 2021-10-01].
Hereafter, dates follow the ISO 8601 standard. i.e., YYYY-MM-DD.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-658-40473-4_1.

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_1

Table 1.1: Different representations of a Jacobi polynomial.

System	Representation
Rendered Version	$P_n^{(\alpha,\beta)}(\cos(a\Theta))$
Generic LaTeX	`P_n^{(\alpha, \beta)}(\cos(a\Theta))`
Semantic LaTeX	`\JacobipolyP{\alpha}{\beta}{n}@{\cos@{a\Theta}}`
Maple [36]	`JacobiP(n, alpha, beta, cos(a*Theta))`
Mathematica [393]	`JacobiP[n, \[Alpha], \[Beta], Cos[a \[CapitalTheta]]]`
SymPy [252]	`jacobi(n,Symbol('alpha'),Symbol('beta'),cos(a*Symbol('Theta')))`

used CAS include Maple [36], Mathematica [393], and MATLAB [246]. Scientists use CAS[2] to simplify, manipulate, evaluate, compute, or even visualize mathematical expressions. Thus, CAS play a crucial role in the modern era for pure and applied mathematics [8, 184, 207, 262] and even found their way into classrooms [237, 363, 365, 389, 390]. In turn, CAS are the perfect tool for the researcher in our example to examine the formula further. In order to use a CAS, she needs to translate the expression into the correct CAS syntax.

Table 1.1 illustrates the differences between computable and presentational encodings for a Jacobi polynomial. While the rendered version and the LaTeX [220] encoding only provide visual information, semantic LaTeX [403] and the CAS encodings explicitly encode the meaning, i.e., the semantics, of the formula. On the one hand, LaTeX[3] has become the de-facto standard to typeset mathematics in scientific publications [129, 248, 402], especially in the domain of Science, Technology, Engineering, and Mathematics (STEM). On the other hand, computational advances make CAS an essential asset in the modern workflow of experimenting and publishing in the Sciences. Translating expressions between LaTeX and CAS syntaxes is, therefore, a typical task in the everyday life of our hypothetical researcher. Despite this common need, no reliable translation from a presentational format, such as LaTeX, to a computable format, such as Mathematica, is available to date. The only option our hypothetical researcher has is to manually translate the expression in the specific syntax of a CAS. This process is time-consuming and often error-prone.

Problem: No reliable translation from a presentational mathematical format to a computable mathematical format exists to date.

If a translation between LaTeX and CAS is so essential, why are there no translation tools available? As is often the case in research, the reasons for this are diversified. First, there are translation approaches available. Some CAS, such as Mathematica and SymPy, allow to import LaTeX expressions. Most CAS support at least the Mathematical Markup Language (MathML), since it is the current web standard to encode mathematical formulae. With numerous tools available to transfer LaTeX to MathML [18], a translation from LaTeX to CAS syntaxes should not be a difficult task. However, none of these available translation techniques are reliable

[2] In the sequel, the acronym CAS is used interchangeably with its plural.
[3] https://www.latex-project.org/ [accessed 2021-10-01]

Table 1.2: Examples of Mathematica's LaTeX import function `ToExpression["x", TeXForm]`. Tested with Mathematica [393] v.12.1.1. The second sum in row 8 (marked with ?) is only partially correct. Since the second summand contains the summation index n, the second summand should be part of the sum.

LaTeX	Rendering	Import	Result
`\int_a^b x dx`	$\int_a^b x dx$	`Error`	✘
`\int_a^b x \mathrm{d}x`	$\int_a^b x \mathrm{d}x$	`Error`	✘
`\int_a^b x\, dx`	$\int_a^b x\, dx$	`Integrate[x, {x, a, b}]`	✔
`\int_a^b x\; dx`	$\int_a^b x\; dx$	`Error`	✘
`\int_a^b x\, \mathrm{d}x`	$\int_a^b x\, \mathrm{d}x$	`Error`	✘
`\int_a^b \frac{dx}{x}`	$\int_a^b \frac{dx}{x}$	`Error`	✘
`\sum_{n=0}^N n^2`	$\sum_{n=0}^{N} n^2$	`Sum[n^2, {n, 0, N}]`	✔
`\sum_{n=0}^N n^2 + n`	$\sum_{n=0}^{N} n^2 + n$	`Sum[n^2, {n, 0, N}] + n`	?
`{n \choose m}`	$\binom{n}{m}$	`Error`	✘
`\binom{n}{m}`	$\binom{n}{m}$	`Binomial[n, m]`	✔

and comprehensive. Table 1.2 illustrates how Mathematica, one of the major proprietary CAS, fails to import even simple formulae. Another option is SnuggleTeX [251], a LaTeX to MathML converter which also supports translations to Maxima [324]. SnuggleTeX fail to translate all expressions in Table 1.2. Alternative translations via MathML as an intermediate format perform similarly (as we will show later in Section 2.3).

While the simple cases shown in Table 1.2 could be solved with a more comprehensive and flexible parser and mapping strategy, such a solution would ignore the real challenge of translating mathematics to CAS, the ambiguity. The interpretation of the majority of mathematical expressions is *context-dependent*, i.e., the same formula may refer to different concepts in different contexts. Take the expressions $\pi(x + y)$ as an example. In number theory, the expression most likely refers to the number of primes less than or equal to $x + y$. In another context, however, it may just refer to a multiplication $\pi x + \pi y$. Without considering the context, an appropriate translation of this ambiguous expression is infeasible. Today's translation solutions, however, do not consider the context of an input. Instead, they translate the expression based on internal decisions, which are often not transparent to a user.

Table 1.3 shows the results of importing $\pi(x + y)$ to different CAS. Each CAS in Table 1.3 interprets π as a function call but does not associate it with the prime counting function (nor any other predefined function). Only SnuggleTeX translated π as the mathematical constant to Maxima syntax. However, Maxima does not contain a prime counting function. The CAS import functions consider the expression as a generic function with the name π. Mathematica surprisingly links π still with the mathematical constant which results in a peculiar behaviour for numeric evaluations. The expression `N[Pi[x+y]]` (numeric evaluation of the imported expression) is evaluated to $3.14159[x + y]$. Associating the variables x and y with numbers, say $x, y = 1$, would result in the rather odd expression $3.14159[2]$.

Table 1.3: The results of importing $\pi(x+y)$ in different CAS. For Maple, a MathML representation was used. Content MathML was not tested, since there is no content dictionary available that defines the prime counting function. SnuggleTeX translated the expression to the CAS Maxima. The two right most columns show the expected expressions in the context of the prime counting function or a multiplication. None of the CAS choose any of the two expected interpretations. Note that the prime counting function in Maple can also be written with `pi(x+y)` and requires to pre-load the extra package `NumberTheory`. Nonetheless, this function `pi(x+y)` is still different to the actual imported expression `Pi(x+y)`. Note further that Maxima does not define a prime counting function.

	Translated	**Expected Expression**	
System	**Expression**	**Number of primes**	**Multip.**
Maple [36] v.2019	`Pi(x+y)`	`PrimeCounting(x+y)`	`Pi*(x+y)`
Mathematica [393] v.12.1.1	`Pi[x+y]`	`NPrimes[x+y]`	`Pi*(x+y)`
SymPy [252] v.1.8	`pi(x+y)`	`primepi(x+y)`	`pi*(x+y)`
SnuggleTeX [251] v.1.2.2	`%pi*(x+y)`	-	`%pi*(x+y)`

Why do existing translation techniques not allow to specify a context? Mainly because it is an open research question of what this context is or needs to be. The exact information needs to perform translation to CAS syntaxes, and where to find them is unlcear [11]. Some required information is indeed encoded in the structure of the expression itself. Consider a simple fraction $\frac{1}{2}$. This expression is context-independent and can be directly translated. The expression $P_n^{(\alpha,\beta)}(x)$ in the context of OPSF is also often unambiguous for general-purpose CAS. Since Mathematica supports no other formula with this presentational structure, i.e., P followed by a subscript and superscript with paranthesis, Mathematica is able to correctly associate $P_\bullet^{(\bullet,\bullet)}(\bullet)$, where $\bullet$ are wildcards, with the function `JacobiP`. In other cases, the immediate textual context of the formula provides sufficient information to disambiguate the expression [54, 329]. Consider, an author explicitly declares $\pi(x)$ as the prime counting function right before she uses it with $\pi(x+y)$. In this case, it might be sufficient to scan the surrounding context for key phrases [183, 214, 329], like 'prime counting function' in order to map π to, for instance, `NPrimes` in Mathematica.

Often, the semantic explanations of mathematical objects in an article are scattered around in the context or absent entirely [394]. An interested reader needs to retrieve sufficient semantic explanations and correctly link them with mathematical objects in order to comprehend the meaning of a complex formula. Sometimes, an author presumes the interpretation of an expression can be considered as common knowledge and, therefore, does not require further explanations. Consider $\pi(x+y)$ refers to a multiplication between π and $(x+y)$. In general, an author may consider π (the mathematical constant) as common knowledge and does not explicitly declare its meaning. The same could be true for scientific articles, where the length is often limited. An article about prime numbers probably not explicitely declare the meaning of $\pi(x+y)$ because the author presumes the semantics are unambiguis given the overall context of the article.

In other cases, the information needs go beyond a simple text analysis. Consider $\pi(x+y)$ as a generic function that was previously defined in the article and simply has no name. An appropriate translation would require to retrieve the definition of the function from the context. But even if a function is well-known and supported by a CAS, a direct translation might be inappropriate because the definition in the CAS is not what our researcher expected [3, 13]. Legendre's incomplete elliptic integral of the first kind $F(\phi, k)$, for example, is defined with the amplitude ϕ as its first argument in the DLMF and Mathematica. In Maple, however, one needs to use the sine of the amplitude $\sin(\phi)$ for the first argument[4]. In turn, an appropriate translation to Maple might be `EllipticF(sin(phi), k)` rather than `EllipticF(phi, k)` depending on the source of the original expression. The English Wikipedia article about elliptic integrals[5] contains both versions and refers to them with $F(\phi, k)$ and $F(x; k)$ respectively. Even though both versions in Wikipedia refer to the same function, correct translations to Maple of $F(\phi, k)$ and $F(x; k)$ are not the same.

In cases of multi-valued functions, translations between different systems can become eminently more complex [83, 91, 172]. Even for simple cases, such as the arccotangent function $\operatorname{arccot}(x)$, the behavior of different CAS might be confusing. For example, since $\operatorname{arccot}(x)$ is multi-valued, there are multiple solutions of $\operatorname{arccot}(-1)$. CAS, like any general calculator too, only compute values on the principle branches and, therefore, return only a single value. The principle branches, however, are not necessarily uniformly positioned among multiple systems [84, 172]. In turn, the returned value of a multi-valued function may depends on the system, see Table 1.4. A translation of $\operatorname{arccot}(x)$ from the DLMF to `arccot(x)` in Maple would be only correct for $\Re x > 0$. Finally, CAS may also compute irrational looking expressions without objections, e.g., $\operatorname{arccot}\left(\frac{1}{0}\right)$ returns 1.5708 in MATLAB[6]. Even for field experts, it can be challenging to keep track of every property and characteristic of CAS [20, 100].

Table 1.4: Different computation results for $\operatorname{arccot}(-1)$ (inspired by [84]).

System or Source	$\operatorname{arccot}(-1)$
[276] 1st printing	$3\pi/4$
[276] 9th printing	$-\pi/4$
Maple [36] v.2020.2	$3\pi/4$
Mathematica [393] v.12.1.1	$-\pi/4$
SymPy [252] v.1.5.1	$-\pi/4$
Axiom [173] v.Aug.2014	$3\pi/4$
Reduce [151] v.5865	$3\pi/4$
MATLAB [246] v.R2021a	$-\pi/4$

Problem: Existing LaTeX to CAS converters are context-agnostic, inflexible, limited to simple expressions, and nontransparent.

In combination, all of the issues underline that an accurate manual translation to the syntax of CAS is challenging, time-consuming, error-prone, and requires deep and substantial knowledge about the target system. Especially with the increasing complexity of the translated expressions, errors during the translation process might be inevitable. Real-world scenarios often include

[4] https://www.maplesoft.com/support/help/maple/view.aspx?path=EllipticF [accessed 2021-10-01]

[5] https://en.wikipedia.org/wiki/Elliptic_integral [accessed 2021-10-01]

[6] MATLAB evaluates $\frac{1}{0}$ to infinity and the limit in positive infinity of the arccotangent function is $\frac{\pi}{2}$ (or roughly 1.5708). Yet, the interpretation of the division by zero is not wrong, since it follows the official IEEE 754 standard for floating-point arithmetic [170].

much more complicated formulae compared to the expressions in Table 1.2 or even equation (1.1). Moreover, if an error occurs, the cause of the error can be very challenging to detect and traced back to its origin. The issue of translating $\operatorname{arccot}(x)$ to Maple, for example, may remain undiscovered until a user calculates negative values. If the function is embedded into a more complex equation, even experts can lose track of potential issues. In combination with unreliable translation tools, working with CAS may even be frustrating. Mathematica, for example, is able to import our test expression (1.1) mentioned earlier without throwing an error[7]. However, investigating the imported expression reveals an incorrect translation due to an issue with factorials. To productively work with CAS, our hypothetical researcher from above needs to carefully evaluate if the automatically imported expression was correct. As a consequence, existing translation approaches are not practically useful.

In this thesis, I will focus on discovering the information needs to perform correct translations from presentational formats, here mainly LaTeX, to computational formats, here mainly CAS syntaxes. My personal motivation is to improve the workflow of researchers by providing them a reliable translation tool that offers crucial additional information about the translation process. Further, I limit the support of such a translation tool to general-purpose CAS, since many general mathematical expressions simply cannot be translated to appropriate CAS expressions for task-specific CAS (or other mathematical software, such as theorem provers). The focus on general-purpose CAS allows me to provide a broad solution to a general audience. Note further that, in this thesis, I mostly focus on the two major CAS Maple and Mathematica. However, the goal is to provide a translation tool that is easy to extend and support more CAS.

Further, the real-world applications of such a translation tool go far beyond an improved workflow with CAS. A computable formula can be automatically verified with CAS [51, 52, 2, 8, 13, 153, 184, 414, 415], translated to other semantically enhanced formats, such as OpenMath [53, 57, 119, 152, 303, 361], content MathML [59, 60, 159, 270, 318, 342] or other CAS syntaxes [110, 361], imported to theorem prover [35, 57, 152, 163, 338, 375], or embedded in interactive documents [85, 131, 150, 162, 201, 284]. Since an appropriate translation is generally context-dependent, a translator must use MathIR [141] techniques to access sufficient semantic information. Hence, advances in translating LaTeX to CAS syntaxes also contribute directly towards related MathIR tasks, including entity linking [150, 208, 212, 316, 319, 321, 322], math search engines [92, 181, 182, 203, 211, 236, 274], semantic tagging of math formulae [71, 402], recommendation systems [30, 31, 50, 319], type assistance systems [103, 106, 14, 321, 400], and even plagiarism detection platforms [253, 254, 334].

1.2 Research Gap

Existing translation approaches from presentational formats to computable formats share the same issues. Currently, these translation approaches are

1. context-independent, i.e., a translation of an expression is unique regardless of the context from where the expression came from (see the $\pi(x+y)$ example mentioned earlier);
2. nontransparent, i.e., the internal translation decisions are not communicated to the user, which makes the translation untrustworthy and errors challenging to trace or detect;

[7] If the binomial is given with the `\binom` macro rather than `\choose`.

3. inflexible, i.e., slight changes in the notation can cause the translation to fail (see the integral imports from Table 1.2); and
4. limited to simple expression due to missing mappings between function definition sources, i.e., even with semantic information, a translation often fails.

Issue 4 raises from the fact that there are semantically enhanced data formats that have been specifically developed to make expressions between CAS interchangeable, such as OpenMath [119, 303, 361] and content MathML [318, 343]. Nonetheless, most CAS do not support OpenMath natively [303] and the support for content MathML is limited to school mathematics [318]. The reason is that such translation requires a database that maps functions between different semantic sources. As discussed above, creating such a comprehensive database can be time-consuming due to slight differences between the systems (e.g., positions of branch cuts, different supported domains, etc.) [361]. Hence, for economic reasons, crafting and maintaining such a library is unreasonable. Translations between semantic enhanced formats, e.g., between CAS syntaxes, OpenMath, or content MathML, are consequentially often unreliable.

In previous research, I was focusing on the issues 2-4 by developing a rule-based LaTeX to CAS translator, called LaCaST. Originally, LaCaST performs translations from semantic LaTeX to Maple. Relying on semantic LaTeX allows LaCaST to largely ignore the ambiguity Issue 1 and focus on the other problems. For this thesis, I continued to develop LaCaST to further mitigate the *limitation* and *inflexibility* issues 3 and 4. Further, I focused on extending LaCaST to become the first context-aware translator to tackle the *context-independency* issue 1.

1.3 Research Objective

This doctoral thesis aims to:

Research Objective

Develop and evaluate an automated context-sensitive process that makes presentational mathematical expressions computable via computer algebra systems.

Hereafter, I consider the semantic information of a mathematical expression as sufficient if a translation of the expression into the syntax of a CAS becomes feasible. To achieve the research objective, I define the following five research tasks:

Research Tasks

I Analyze the strengths and weaknesses of existing *semantification* approaches for translating mathematical expressions to computable formats.

II Develop a semantification process that will improve on the weaknesses of current approaches.

III Implement a system for the automated semantification of mathematical expressions in scientific documents.

IV Implement an extension of the system to provide translations to computer algebra systems.

V Evaluate the effectiveness of the developed semantification and translation system.

1.4 Thesis Outline

Chapter 1 provides an introduction for translating presentational mathematical expressions into computable formats. The chapter further defines the research gap for such translations and defines the research objective and tasks this thesis addresses. Finally, it outlines the structure of the thesis and briefly summarizes the main publications.

Chapter 2 provides an overview of related work by examining existing mathematical formats and translation approaches between them. This chapter focuses on **Research Task I** by analyzing the strengths and weaknesses of existing translation approaches with the main focus on the standard formats LaTeX and MathML.

Chapter 3 addresses **Research Task II** by studying the capability of math embeddings, introducing a new concept to describe the nested structure of mathematical objects, and presenting a novel context-sensitive semantification process for LaTeX expressions.

Chapter 4 presents the first context-sensitive LaTeX to CAS translator: LaCaSt. In particular, this chapter focuses on **Research Tasks III** and **IV** by implementing the previously introduced semantification process and integrates it into the rule-based semantic LaTeX to CAS translator LaCaSt. In addition, the chapter briefly summarizes a context-independent neural machine translation approach to estimate how much structural information is encoded in mathematical expressions.

Chapter 5 evaluates the new translation tool LaCaSt and, therefore, contributes mainly towards **Research Task V**. In particular, it introduces the novel evaluation concept of equation verifications to estimate the appropriateness of translated CAS expressions. Our new evaluation concept not only detects issues in the translation pipeline but is also able to identify errors in the source equation, e.g., from the DLMF or Wikipedia, and the target CAS, e.g., Maple or Mathematica. In order to maximize the number of verifiable DLMF equations via our novel evaluation technique, this chapter also introduces some heuristic extensions to the LaCaSt pipeline. Hence, this chapter partially contributes to **Research Task IV** too.

Chapter 6 concludes the thesis by summarizing contributions and their impact on the MathIR community. It further provides a brief overview of the remaining issues and future work.

An Appendix is available in the electronic supplementary material and provides additional information about certain aspects of this thesis including an extended error analysis, result tables, and a summary of bugs and issues we discovered with the help of LaCaSt in the DLMF, Maple, Mathematica, and Wikipedia.

1.4.1 Publications

Most parts of this thesis were published in international peer-reviewed conferences and journals. Table 1.5 provides an overview of the publications that are reused in this thesis. The first column identifies the chapter a publication contributed to. The venue rating was taken from the Core ranking[8] for conferences and the Scimago Journal Rank (SJR)[9] for journal articles. Each rank

[8] http://portal.core.edu.au/conf-ranks/ with the ranks: A* – flagship conference (top 5%), A – excellent conference (top 15%), B – good conference (top 27%), and C – remaining conferences [accessed 2021-10-01].

[9] https://www.scimagojr.com/ with the ranks Q1 – Q4 where Q1 refer to the best 25% of journals in the field, Q2 to the second best quarter, and so on [accessed 2021-10-01].

was retrieved for the year of publication (or year of submission, in case the paper has not been published yet). Table 1.6 similarly shows publications that partially contributed towards the goal of this thesis but are not reused within a chapter. Note that the publication [3] (in Table 1.6) was part of my Master's thesis and contributed towards this doctoral thesis as a preliminary project. The Journal publication [13] (also in Table 1.6) is an extended and (with new results) updated version of the thesis and the mentioned article [3]. The venue abbreviations in both tables are explained in the glossary. Lastly, note that the TPAMI journal [11] is reused in Chapter 4 (for the methodology) and in Chapter 5 (for the evaluation) to provide a coherent structure. My publications, talks, and submissions are separated from the general bibliography in the back matter and can be found on page 171.

Table 1.5: Overview of the primary publications in this thesis.

Ch.	Venue	Year	Type	Length	Author Position	Venue Rating	Ref.
2	SIGIR	2019	Workshop	Full	1 of 6	Core A*	[9]
	JCDL	2018	Conference	Full	2 of 6	Core A*	[18]
3	Scientometrics	2020	Journal	Full	1 of 7	SJR Q1	[15]
	WWW	2020	Conference	Full	1 of 7	Core A*	[14]
	ICMS	2020	Conference	Full	1 of 4	n/a	[10]
4	TPAMI[10]	2021	Journal	Full	1 of 6	SJR Q1	[11]
5	TACAS	2021	Conference	Full	1 of 8	Core A	[8]
	CICM	2018	Conference	Full	2 of 3	n/a	[2]
6	JCDL	2020	Conference	Poster	2 of 5	Core A*	[17]

Table 1.6: Overview of secondary publications that partially contributed to this thesis.

Year	Venue	Type	Length	Author Position	Venue Rating	Ref.
2020	CLEF	Workshop	Full	4 of 6	n/a	[16]
	EMNLP	Workshop	Full	2 of 4	Core A	[1]
2019	AJIM	Journal	Full	1 of 4	SJR Q1	[13]
2018	CICM	Conference	Short	1 of 4	n/a	[12]
2017	CICM	Conference	Full	4 of 9	n/a	[3]

1.4.2 Research Path

This section provides a brief overview of my research path that led to this thesis, i.e., it discusses the primary publications and the motivations behind them. Every publication is marked with the associated chapter and a reference. This research path is logically (not chronologically) divided into three sections: preliminary work, the semantification of LaTeX, and the evaluation of translations.

Preliminary Work I had the first contact with the problem of translating LaTeX to CAS syntaxes during my undergraduate studies in mathematics. During that time, I regularly used

[10]The methodology part of this journal is reused in Chapter 4 while the evaluation part is reused in Chapter 5.

CAS like MATLAB and SymPy for numeric simulations and for plotting results. At the same time, we were required to hand in our homework as LaTeX files. While exporting content from the CAS to LaTeX files was rather straight forward, the other way around, i.e., importing LaTeX into the CAS, required manual conversions. I decided to explore the reasons for this shortcoming in my Master's thesis. During that time, I developed the first version of a semantic LaTeX to CAS translator, which was later coined LaCaST[11]. The results from this first study were published at the Conference of Intelligent Computer Mathematics (CICM) in 2017.

"Semantic Preserving Bijective Mappings of Mathematical Formulae Between Document Preparation Systems and Computer Algebra Systems" by Howard S. Cohl, Moritz Schubotz, Abdou Youssef, **André Greiner-Petter**, Jürgen Gerhard, Bonita Saunders, Marjorie McClain, Joon Bang, and Kevin Chen. **In:** *Proceedings of the International Conference of Intelligent Computer Mathematics* (CICM), 2017.

Not Reused — [3]

This first version of LaCaST focused specifically on the CAS Maple but was designed modularly to allow later extensions to other CAS. The main limitation of LaCaST, however, was the requirement of using semantic LaTeX macros to disambiguate mathematical expressions manually. An automatic disambiguation process did not exist at the time. Moreover, only a few previous projects focused on a semantification for translating mathematical formats. Hence, I continued my research in this direction.

In the following, I will use 'we' rather than 'I' in the subsequent parts of this thesis, since none of the presented contributions would have been possible without the tremendous and fruitful discussions and help from advisors, colleagues, students, and friends.

Semantification of LaTeX As an alternative for semantic LaTeX, we closely investigated existing converters for MathML first (see Section 2.2.1). Since MathML was (and still is) the standard encoding for mathematical expressions in the web, most CAS support MathML. MathML uses two markups, presentation and content MathML. The former visualizes a formula, while the latter describes the semantic content. Hence, content MathML can disambiguate math much like semantic LaTeX. Since MathML is the official web standard and LaTeX the de-facto standard for writing math, there are numerous of converters available that translate LaTeX to MathML. As our first contribution, we developed MathMLben, a benchmark dataset for measuring the quality of MathML markup that appears in a textual context. With this benchmark, we evaluated nine state-of-the-art LaTeX to MathML converters, including Mathematica as a major CAS. We published our results in the Joint Conference on Digital Libraries (JCDL) in 2018.

"Improving the Representation and Conversion of Mathematical Formulae by Considering their Textual Context" by Moritz Schubotz, **André Greiner-Petter**, Philipp Scharpf, Norman Meuschke, Howard S. Cohl, and Bela Gipp. **In:** *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries* (JCDL), 2018.

Chapter 2 — [18]

[11] ***La****TeX to* ***CAS*** *Translator.*

We discovered that three of the nine tools were able to generate content MathML but with insufficient accuracy. None of the available tools were capable of analyzing a context for a given formula. Hence, the converters were unable to conclude the correct semantic information for most of the symbols and functions. In our study, we proposed a manual semantification approach that semantically enriches the translation process of existing converters by feeding them semantic information from the surrounding context of a formula. The enrichment process was manually illustrated via the converter LaTeXML, which allowed us to add custom semantic macros to improve the generated MathML data. In fact, we used this manual approach to create the entries of MathMLben in the first place.

Naturally, our next goal was to automatically retrieve semantic information from the context of a given formula. Around this time, word embeddings [256] began to gain interest in the MathIR community [121, 215, 242, 400, 404]. It seems that vector representations were able to capture some semantic properties of tokens in natural languages. Can we create such semantic vector representations of mathematical expressions too? Unfortunately, we discovered that the related work in this new area of interest did not discuss a crucial underlying issue with embedding mathematical expressions. In math expressions, certain symbols or entire groups of tokens are fixed, such as the red tokens in the Gamma function $\Gamma(x)$ or the Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$, while other may vary (gray). Inspired by words in natural languages, we call these fixed tokens the stem of a mathematical object or operation. Unfortunately, in mathematics, this stem is context-dependent. If π is a function, the red tokens are its stem $\pi(x+y)$. However, if π is not a function, the stem is just the symbol itself $\pi(x+y)$. If we do not know the stem of a mathematical object, how can we group them so that a trained model understands the connection between variations like $\Gamma(z)$ and $\Gamma(x)$? The answer is: we cannot. The only alternative is to use context-independent representations, e.g., we only embed the identifiers or the entire expression. Each of these approaches has advantages and disadvantages. We shared our discussion with the community at the BIRNDL Workshop at the conference on Research and Development in Information Retrieval (SIGIR) in 2019.

"Why Machines Cannot Learn Mathematics, Yet" by **André Greiner-Petter**, Terry Raus, Moritz Schubotz, Akiko Aizawa, William I. Grosky, and Bela Gipp. **In:** *Proceedings of the 4th Joint Workshop on Bibliometric-Enhanced Information Retrieval and Natural Language Processing for Digital Libraries* (BIRNDL@SIGIR), 2019.

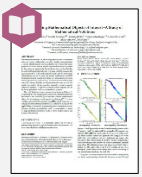
Chapter 2 — [9]

Nonetheless, context-independent math embeddings still have many valuable applications. Search engines, for example, can profit from a vector representation that represents a mathematical expression in a particular context. Such a trained model would still be unable to tell us what the expression is, but it can tell us efficiently if the expression is *semantically similar* (e.g., because the surrounding text is similar) to another expression. Further, embedding semantic LaTeX allows us to overcome the issue of unknown stems for most functions since the macro unambiguously defines the stem. Youssef and Miller [404] trained such a model on the DLMF formulae. Later, we published an extended version of our workshop paper together with Youssef and Miller in the *Scientometrics* journal.

"Math-Word Embedding in Math Search and Semantic Extraction" by **André Greiner-Petter**, Abdou Youssef, Terry Raus, Bruce R. Miller, Moritz Schubotz, Akiko Aizawa, and Bela Gipp. **In:** *Scientometrics* 125(3): 3017-3046, 2020.

Chapter 3 — [15]

Unfortunately, this sets us back to the beginning, where we need manually crafted semantic LaTeX. We started to investigate the issue of interpreting the semantics of mathematical expressions from a different perspective. As we will see later in Section 2.2.4, humans tend to visualize mathematical expressions in a tree structure, where operators, functions, or relations are parent nodes of their components. Identifiers and other terminal symbols are the leaves of these trees. The MathML tree data structure comes close to these so-called *expression trees* (see Section 2.2.4) but does not strictly follow the same idea [331]. The two aforementioned context-independent approaches to embed mathematical expressions take either the leaves or the roots of such trees. The subtrees in between are the context-dependent mathematical objects we need. Not all subtrees, however, are meaningful, and the mentioned expression trees are only theoretical interpretations. In searching for an approach to discover meaningful subexpressions, which we call Mathematical Objects of Interest (MOI), we performed the first large-scale study of mathematical notations on real-world scientific articles. In this study, we followed the assumption that every subexpression with at least one identifier can be semantically important. Hence, we split every formula into their MathML subtrees and analyzed their frequency in the corpora. Overall, we analyzed over 2.5 Billion subexpressions in 300 Million documents and showed that the frequency distribution of mathematical subexpressions is similar to words in natural language corpora. By applying known frequency-based ranking functions, such as BM25, we were also able to discover topic-relevant notations. We published these results at The Web Conference (WWW) in 2020.



"Discovering Mathematical Objects of Interest — A Study of Mathematical Notations" by **André Greiner-Petter**, Moritz Schubotz, Fabien Müller, Corinna Bretinger, Howard S. Cohl, Akiko Aizawa, and Bela Gipp. **In:** *Proceedings of the Web Conference* (WWW), 2020.

Chapter 3 — [14]

The applications that we derived from simply counting mathematical notations were surprisingly versatile. For example, with the large set of indexed math notations, we implemented the first type assistant system for math equations, developed a new faceted search engine for zbMATH, and enabled new approaches to measure potential plagiarism in equations. Besides these practical applications, it also gave us the confidence to continue focusing on subexpressions for our LaTeX semantification. Previous projects that aimed to semantically enrich mathematical expressions with information from the surrounding context primarily focused on one of the earlier mentioned extremes, i.e., the leaves or roots in expression trees [139, 214, 279, 329, 330]. Our study also revealed that the majority of unique mathematical formulae are neither single identifier nor highly complex mathematical expressions. Hence, we concluded that we should

focus on semantically enriching subexpressions (subtrees) rather than the roots or leaves. We proposed a novel context-sensitive translation approach based on semantically annotated MOI and shared our theoretical concept with the community at the International Conference on Mathematical Software (ICMS) in 2020.

"Making Presentation Math Computable: Proposing a Context Sensitive Approach for Translating LaTeX to Computer Algebra Systems" by **André Greiner-Petter**, Moritz Schubotz, Akiko Aizawa, and Bela Gipp. **In:** *Proceedings of the International Conference on Mathematical Software* (ICMS), 2020.

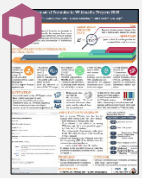
Chapter 3 — [10]

Afterward, we started to realize the proposed pipeline with a specific focus on Wikipedia. We focused on this encyclopedia for two reasons. First, Wikipedia is a free and community-driven encyclopedia and, therefore, (a) less strict on writing styles and (b) more descriptive compared to scientific articles. Second, Wikipedia can actively benefit from our contribution since additional semantic information about mathematical formulae can support users of all experience levels to read and comprehend articles more efficiently [150]. Moreover, a successful translation from a formula in Wikipedia to a CAS makes the formula computable which enables numerous of additional applications. In theory, a mathematical article could be turned into an interactive document to some degree with our translations. However, the most valuable application of a translation of formulae in Wikipedia would be the ability to check equations for their plausibility. With the help of CAS, we are able to analyze if an equation is semantically correct or suspicious. This evaluation would enable existing quality measures in Wikipedia to incorporate mathematical equations for the first time. The results from our novel context-sensitive translator including the plausibility check algorithms have been accepted for publication in the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) journal and are currently in press.

"Do the Math: Making Mathematics in Wikipedia Computable." **André Greiner-Petter**, Moritz Schubotz, Corinna Bretinger, Philipp Scharpf, Akiko Aizawa, and Bela Gipp. In press: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), 2021.

Chapter 4 & 5 — [11]

Currently, we are also actively working on extending the backbone of Wikipedia itself for presenting additional semantic information about mathematical expressions by hovering over or clicking on the formula. This new feature helps Wikipedia users to better understand the meaning of mathematical formulae by providing details on the elements of formulae. Moreover, it paves the way towards an interface to actively interact with mathematical content in Wikipedia articles. We presented our progress and discussed our plans in the poster session at the JCDL in 2020.



"Mathematical Formulae in Wikimedia Projects 2020." Moritz Schubotz, **André Greiner-Petter**, Norman Meuschke, Olaf Teschke, and Bela Gipp. **In:** Poster Session at the *ACM/IEEE Joint Conference on Digital Libraries* (JCDL), 2020.

Chapters 6 — [17]

Evaluating Digital Mathematical Libraries Alongside this main research path, we continuously improved and extended LaCasT with new features and new supported CAS. Our first goal was to verify the translated, now computable, formulae in the DLMF. The primary motivation behind this approach was to quantitatively measure the accuracy of LaCasT translations. How can we very if a translation was correct? The well-established Bilingual Evaluation Understudy (BLEU) [282] measure in natural language translations is not directly applicable for mathematical languages because an expression may contain entirely different tokens but is still equivalent to the gold standard. Since the translation is computable, however, we can take advantage of the power of CAS to verify a translation. The basic idea is that a human-verified equation in one system must remain valid in the target system. If this is not the case, only three sources of errors are possible: either the source equation, the translation, or the CAS verification was incorrect. With the assumption that equations in the DLMF and major proprietary CAS are mostly error-free, we can translate equations from the DLMF to discover issues within LaCasT. First, we focused on symbolic verifications, i.e., we used the CAS to symbolically simplify the difference between left- and right-hand side of an equation. If the simplified difference is 0, the CAS symbolically verified the equivalence of the left- and right-hand side and confirmed a correct translation via LaCasT. Additionally, we extended the verification approach to include more precise numeric evaluations. If a symbolic manipulation failed to return 0, it could also mean the CAS was unable to simplify the expression. We numerically calculate the difference on specific test values and check if the difference is below a given threshold to overcome this issue. If all test calculations are below the threshold, we consider it numerically verified. Even though this approach cannot verify equivalence, it is very effective in discovering disparity. We published the first paper with this new verification approach based on Maple at the CICM in 2018.

"Automated Symbolic and Numerical Testing of DLMF Formulae Using Computer Algebra Systems" by Howard S. Cohl, **André Greiner-Petter**, and Moritz Schubotz. **In:** *Proceedings of the International Conference on Intelligent Computer Mathematics* (CICM), 2018.

Chapter 5 — [2]

The extension of the system and the new results led us to an extended journal version of the initial LaCasT publication [3]. This extended version mostly covered parts of my Master's thesis and is not reused in this thesis. For technical details about LaCasT, see the journal publication [13]. In Appendix D available in the electronic supplementary material, we summarized all significant issues and reported bugs we discovered via LaCasT. The section also includes new issues that we

discovered during the work on the journal publication. This journal version was published in the Aslib Journal of Information Management in 2019.

"Semantic preserving bijective mappings for expressions involving special functions between computer algebra systems and document preparation systems" by **André Greiner-Petter**, Howard S. Cohl, Moritz Schubotz, and Bela Gipp. **In:** *Aslib Journal of Information Management* 71(3): 415-439, 2019.

Appendix D — [13]

It turned out that LaCASt translations on semantic LaTeX were so stable that we can use the same approach for verifying translations also to specifically search for errors in the DLMF and issues in CAS. To maximize the number of supported DLMF formulae, we implemented additional heuristics to LaCASt, such as a logic to identify the end of a sum or to correctly interpret prime notations as derivatives. Additionally, we added support for translations to Mathematica and SymPy. We extended the support for Mathematica even further to perform the same verifications in Maple also in Mathematica. The Mathematica support finally allows us to identify computational differences in two major proprietary CAS. Moreover, we extended the previously introduced symbolic and numeric evaluation pipeline with more sophisticated variable extraction algorithms, more comprehensive numeric test values, resolved substitutions, and improved constraint-awareness. All discovered issues are summarized in Appendix D available in the electronic supplementary material. We further made all translations of the DLMF formulae publicly available, including the symbolic and numeric verification results. The results of this recent study have been published at the international conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS).

"Comparative Verification of the Digital Library of Mathematical Functions and Computer Algebra Systems" by **André Greiner-Petter**, Howard S. Cohl, Abdou Youssef, Moritz Schubotz, Avi Trost, Rajen Dey, Akiko Aizawa, and Bela Gipp. **In:** *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2022.

Chapter 5 — [8]

We also applied the same verification technique to the Wikipedia articles we mentioned earlier, which enabled LaCASt to symbolically and numerically verify even complex equations in Wikipedia articles. This evaluation is also part of the TPAMI submission.

Preprints of my publicationsare available at
https://pub.agp-research.com

My Google Scholar profile is available at
https://scholar.google.com/citations?user=Mq2B9ogAAAAJ

All translations of the DLMF formulae are available at
https://lacast.wmflabs.org

A prototype of LaCaST for Wikipedia is available at
https://tpami.wmflabs.org

I don't know half of you half as well as I should like, and I like less than half of you half as well as you deserve.

Bilbo Baggins - *The Lord of the Rings*

CHAPTER 2

Mathematical Information Retrieval

Contents

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-658-40473-4_2.

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_2

Making presentational math computable implies a transformation from one mathematical representation to another. In order to frame this task, we need to introduce presentational and computable formats, and analyze available transformation tools between these formats. There is a large variety of different formats available to encode mathematical expressions, from visual formats, such as LaTeX [220] or MathML [60], to semantic enhanced encodings, such as content MathML [270], semantic LaTeX [260], SLaTeX [200], or OpenMath [19], and entire programming languages, such as CAS syntaxes [36, 128, 173, 175, 176, 177, 178, 393], theorem provers [37, 266, 287, 340, 354, 384], or mathematical packages in `C++` [168], Python [252] or Java [79]. This chapter introduces what we understand as presentational and computable formats, provides an overview of math formats, and discusses existing transformation tools between these formats.

In particular, Section 2.1 introduces presentational and computable formats. Section 2.2 provides an extensive overview of mathematical formats, their attributes, and conversion approaches between them. Since there are a large variety of conversion tools and approaches available for many different formats [39, 200, 18, 351, 406] a translation from a presentational to a computable format can be achieved in many different ways. In this thesis, we mainly focus on translations from LaTeX to CAS syntaxes. The most well-studied translation path from LaTeX to CAS syntaxes would use content MathML as an intermediate, semantically enriched format. Hence, Section 2.3 analyzes state-of-the-art LaTeX to MathML converters. Section 2.4 underlines the research gap and paves the way for the rest of the thesis by briefly discussing MathIR approaches for conversions from presentational to computable formats. Section 2.3 has been published at the JCDL [18]. The introduction of math embeddings in Section 2.2 was published as a workshop paper at the SIGIR conference [9] and later reused in an extended article for the Scientometrics journal [15].

2.1 Background and Overview

Computable encodings are interpretable formal languages in which keywords or sequences of tokens are associated with specific implemented definitions, which allows performing certain mathematical actions on these elements, such as evaluating numeric values or symbolically manipulating the elements. Computable encodings, therefore, must be semantically unambiguous. Otherwise, an interpreter is unable to associate the sequence of tokens with a unique underlying definition. This ambiguity problem is mainly solved by interpreters in two ways: either the system automatically performs disambiguation steps following a decision tree with a fixed set of internal rules, such as `x^y^z` in Mathematica, or the system refuses to parse the expression and returns an error, such as for `x^y^z` in Maple.

Computable formats are formal languages that link key words or phrases with unique implemented definitions. Computable expressions are semantically unambiguous.

Presentational formats, on the other hand, focus on controlling the visualization of mathematical formulae. They generally allow users to change spaces between tokens (e.g., `\,` and `\;` in LaTeX), support two-dimensional visualizations (e.g., $\int_a^b \frac{dx}{x}$), or render entire graphs and images. However, pure presentational formats (in contrast to enhanced semantic encodings) do not specify the meaning of an expression. Consequently, mathematical expressions in presentational for-

mats are generally semantically ambiguous, and it is the author's responsibility to disambiguate the meaning of the expression by providing additional information in the context. Digital presentational formats, such as LaTeX, are also interpretable formal languages[1]. In contrast to computable formats, presentational languages link tokens with specific visualizations rather than executable subroutines. Hence, expressions in these formats must be *unambiguous* too. Otherwise, interpreters are unable to link an expression with a unique visualization (see `x^y^z` in LaTeX). The difference to computable encodings is that expressions in presentational formats must be visually but not semantically unambiguous. For instance, LaTeX refuses to parse `x^y^z` because the rendering of `{x^y}^z` (see x^{yz}) and `x^{y^z}` (see x^{y^z}) is different. In contrast, Maple rejects `x^y^z` because there is a mathematical (and in consequence a computational) difference between $(x^y)^z$ and $x^{(y^z)}$.

Presentational formats are formal languages with a focus on visualization. Presentational expressions can be semantically but not visually ambiguous.

In this thesis, we focus on LaTeX for the presentational format and CAS syntaxes for computable formats. We choose LaTeX because it is currently the de-facto standard for writing scientific papers in the STEM disciplines [129, 402]. Several other word processors, such as the article's editor in Wikipedia[2] or Microsoft's Word [248], entirely or partially support LaTeX inputs. In addition, LaTeX is the main presentational format that is entered by hand. In contrast, MathML, due to its XML datastructure, is not a user-friendly[3] encoding and mostly automatically generated from other formats [82, 159, 18, 374]. Image formats are the result of pictures, scans, or handwritten inputs, and, therefore, less machine-readable. As a consequence, image formats of mathematical formulae are mainly converted into LaTeX or MathML in a pre-processing step [27, 39, 267, 378, 379, 406, 411]. We choose CAS syntaxes for our target computable format because CAS generally support a large variety of different use cases, from manipulations and visualizations to computations and simulations [81, 413]. Especially general-purpose CAS, such as Maple [36] and Mathematica [393], address a broad range of topics [128, 392]. In contrast, theorem provers, proof assistants, and similar software, as potential other computable formats, solely focus on automated reasoning [147, 266, 354, 384]. Hence, the computation of mathematical formulae plays a less significant role in such software.

2.2 Mathematical Formats and Their Conversions

Figure 2.1 provides an overview of different math encodings and existing conversion approaches between them. In addition to the figure, Table 2.1 provides quick access to references for specific translation directions. Figure 2.1 organizes formats by their level of semantics and the level of machine readability. This categorization is not meant to be as accurate as possible nor to be complete. Instead, the figure aims to provide a rough visualization of the most common encodings and their differences. For instance, there is no notable technical difference between

[1] Note that this interpretation of presentational formats does not include images. Since images are less machine-readable formats, they are generally first converted into interpretable formats, such as LaTeX. This conversion process is very challenging on its own [406, 411]. Hence, including images for our task would not provide any benefits but makes it unnecessarily more complicated.

[2] https://en.wikipedia.org/wiki/Help:Displaying_a_formula [accessed 2021-10-01]

[3] A little histrionically described as '*Making humans edit XML is sadistic!*' from the Django 1.7.11 documentation [118].

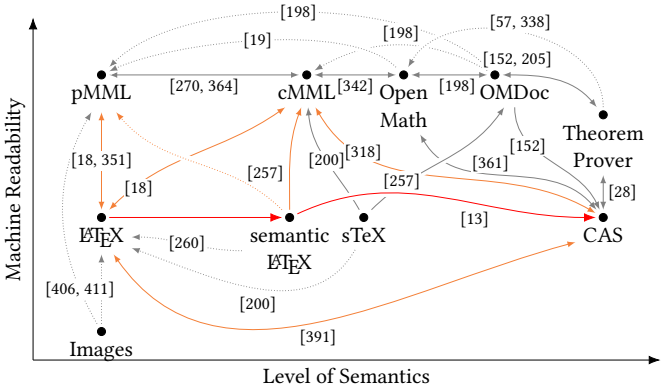


Figure 2.1: Reference map of mathematical formats and translations between them. The red path illustrates the main subject of this thesis. In Section 2.3, we focus specifically on existing translation approaches from LaTeX to MathML (orange arrows) to evaluate an alternative to the red translation path.

the levels of semantics in content MathML and OpenMath (see the paragraph about OpenMath in Section 2.2.1). Nonetheless, OpenMath defines the content dictionaries that content MathML uses to semantically annotate symbols beyond school mathematics. Hence, we could argue that content MathML encodes less semantic information without the help of OpenMath and, therefore, should be positioned more to the left. Another disparity can be found in the level of machine readability between CAS syntaxes and theorem provers. Since both formats are programming languages, any CAS or theorem prover expression requires a very specific (often proprietary) parser. Thus, a programming language is arguably never more *machine readable* than any other programming language. Nonetheless, most CAS prefer a more intuitive input format (sometimes even 2D inputs) similar to LaTeX over a machine-readable syntax [88, 128, 179] to improve their user experience. Because of these more user-friendly input formats, we positioned CAS syntaxes below theorem prover formats. Note also that math embeddings, i.e., vector representation of math tokens, are not in Figure 2.1 because the level of semantics these vectors capture is still unclear and an open research question (see Section 2.2.5). The red path in Figure 2.1 shows the new translation path that we focus on in this thesis. Dotted arrows represent translation paths that generally do not require context analysis and are, therefore, of less interest for the subject of this thesis. The orange and red arrows (and highlighted cells in Table 2.1) refer to our contributions for this thesis. The red arrows refer the main research contribution explained in the chapters 3 and 4.

2.2.1 Web Formats

Web formats are designed to display mathematical formulae and knowledge on the web. Consequently, those formats prioritize machine readability over user experience. Hence, a variety of different translation approaches to, from, or between web formats exists. Since mathematics in the web is generally embedded in HTML code, most web formats use the XML encoding

Table 2.1: Overview table of available mathematical format translations. The highlighted conversion fields refer to contributions made in this thesis. The columns and rows refer to: 'pMML' for presentation MathML, 'cMML' for content MathML, 'sem.LaTeX' for semantic LaTeX, 'Theo. Prov.' for theorem prover or proof assistants, 'Img' for images, and 'Speech' for spoken (audio) mathematical content. The group 'Comp.' refers to computable formats. In some cases, no transformation is necessary, e.g., from OMDoc to OpenMath because OMDoc uses OpenMath internally. In this (and similar) cases, we simply refer to the overview publication of the format, here [198] for OMDoc.

From \ To	pMML	cMML	OpenMath	OMDoc	LaTeX	sem.LaTeX	sTeX	Theo. Prov.	CAS	Vector	Img	Speech	
pMML	/	[364]			[61]				[391]	[86]	[358]	[349]	
cMML	[300]	/	[342]	[198]				[391]	[318]	[242]	[257]		Web
OpenMath	[59]	[342]	/	[198]	[61]			[57]	[303]				
OMDoc	[198]	[198]	[198]	/	[198]		[198]	[152]	[152]				
LaTeX	[18]	[159]	[257]	[198]	/	[11]	[195]		[11]	[15]	[358]	[249]	
sem.LaTeX	[257]	[18]	[257]		[257]	/			[13]	[404]	[257]		TeX
sTeX	[257]	[257]	[257]	[195]	[198]		/				[257]		
Theo. Prov.			[205]	[205]	[167]			[62]	[338]				
CAS	[391]	[391]			[391]	[13]		[338]	[361]		[391]		Comp.
Vector					[400]					/			
Img	[406]				[406]					[406]	/		
Speech	[386]				[386]				[387]				
	Web				TeX			Comp.					

structure. Thus, web formats are often described as verbose and rarely edited or created by hand. On the other hand, the XML structure simplifies the inter-connectivity between web formats, e.g., via XSL Transformations (XSLT) [362]. There are three main formats used in the web: the current web standard MathML, the pure semantic encoding OpenMath, and the semantic document encoding OMDoc. Note that many websites still use image formats to display math. We will discuss image formats in Section 2.2.4.

2.2.1.1 MathML

For the web, the Mathematical Markup Language (MathML) [60] is the current official recommendation from the World Wide Web Consortium (W3C) and even an official standard since 2015 [169] for HTML5. MathML is defined via two different markups: the *presentation*[4] and

[4] https://www.w3.org/TR/MathML3/chapter3.html [accessed 2021-10-01]

the *content*[5] markup. MathML containing only presentation markup elements is, therefore, also called presentation MathML or, in case it only contains content markup elements, content MathML, respectively. Both markups can be used together side by side for a single expression in so-called parallel markup [202, 259, 270]. If elements in the presentation markup are linked back and forth with elements from the content markup, the encoding is also called cross-referenced MathML.

Content MathML, in contrast to presentation MathML, aims to encode the meaning, i.e., the semantics, of mathematical expressions. Content MathML addresses the issues of ambiguous presentational encodings by providing a standard representation of the content of mathematics. The encoding comes with a large number of predefined functions, e.g., for $\sin$ and $\log$, intending to cover most of K-14[6] mathematics. For formulae beyond school mathematics, content MathML use so-called Content Dictionaries (DCs) [204] (see the OpenMath paragraph for more details about CDs). Listing 2.1 shows presentation and content MathML encodings for the Legendre polynomial $P_n(x)$. Note that the presentation MathML encoding contains an operator (`<mo>` for *mathematical operator*) between P_n and (x) which contains the invisible character *function application* (unicode character `U+2061`). Nowadays, content MathML is often used in digital libraries to improve the performance of math search engines with accessible semantic information [345, 347, 348, 381].

Since MathML is the web standard, there are numerous tools available that convert other encodings from and to MathML. Most common conversions include translations between presentation and content MathML [139, 270, 364], from [159, 257, 267, 335, 374] and to[7] LaTeX, OpenMath [59, 342, 343], CAS [318], PDF [27, 267], images [406], and audio encodings (mainly in the math to speech research field) [67, 349, 387]. The W3C officially lists 42 converters and other softare tools that generate MathML on their wiki[8]. In addition, the official *interoperability report*[9] of MathML provides a comprehensive overview of software that supports MathML and show official statements from implementors. Due to its XML format, most conversion tools use XSLT [362] to transform MathML into either other XML encodings or string representations [59, 61]. This translation approach can be described as rule-based, because in XSLT, we define a set of transformation rules for XML subtrees.

Most of the converters to MathML do not support content MathML. Translations from presentational formats to content MathML face a wide range of ambiguity issues [159, 259, 374]. For example, the `<mo>` element in Listing 2.1 regularly contains the *invisible times* symbol (unicode character `U+2062`) rather than *function application* because most conversion tools interpret P_n not as a function. For content MathML, even more disambiguation steps are required to link P with the Legendre polynomial correctly. For such disambiguation, a combination of semantification and XSLT rules are used to perform translations to content MathML [139, 270, 364]. Nghiem et al. [270] proposes a machine translation approach to generate content MathML from presentation MathML but does not consider textual descriptions from the surrounding context of a formula. Likewise, Toloaca and Kohlhase [364] uses patterns of notation definitions

[5] https://www.w3.org/TR/MathML3/chapter4.html [accessed 2021-10-01]

[6] Kindergarten to early college.

[7] Two well-known projects for translations from MathML to LaTeX use XSL transformations: `web-xslt` https://github.com/davidcarlisle/web-xslt/tree/main/pmml2tex and `mml2tex` https://github.com/transpect/mml2tex [accessed 2021-10-01].

[8] https://www.w3.org/wiki/Math_Tools [accessed 2021-10-01]

[9] https://www.w3.org/Math/iandi/mml3-impl-interop20090520.html [accessed 2021-10-01]

to find a content MathML expression that matches the presentation MathML parse tree. Grigore et al. [139], on the other hand, generates a local context of five nouns prior to the expression first to conclude symbol declarations from OpenMath CDs. Besides Grigore et al. [139], other existing approaches for translations to content MathML only consider the semantics within the given formula itself or in formulae in the same document [159, 259, 374] but ignore the textual context surrounding a formula. For example, these tools follow the assumption that a P with subscript followed by an expression in parenthesis should be interpreted as the Legendre polynomial. However, many expressions cannot be disambiguated without considering the textual context, such as the $\pi(x+y)$ example from the introduction.

Most CAS support MathML either directly or via external software packages [318, 343]. However, to the best of our knowledge, no CAS currently consider the CD in content MathML correctly. Hence, these import and export functions in CAS are generally limited to school mathematics. It should be noted that the CDs are considered by CAS but only in OpenMath, e.g., via the transport protocol *Symbolic Computation Software Composability Protocol* (SCSCP) [361]. Since this protocol was developed to enable inter-CAS communication, we explain this project more in detail in Section 2.2.3.

In summary, a reliable generation of content MathML requires a semantic enhanced source formula, e.g., in CAS syntaxes [318, 343], theorem prover formats [152], or OpenMath [59, 342]. Otherwise, translations tend to generate inaccurate MathML. In Section 2.3, we will examine existing LaTeX to MathML converters more in detail to investigate the practicality of using MathML as an intermediate format for translations from LaTeX to CAS encodings.

2.2.1.2 OpenMath

The OpenMath Society (originally OpenMath Consortium [19]) defines another standard encoding called OpenMath [53]. The OpenMath standard aims to focus exclusively on the semantics of mathematics and, therefore, going a step further compared to MathML [204], which aims to cover both the presentation and the content information in a single format. Originally, OpenMath was invented during a series of workshops starting in 1993, mainly from researchers in the computer algebra community, to easily exchange mathematical expressions between CAS and other systems [19, 89]. MathML, originally developed with the same goal, was first released in 1998[10]. Both formats are very similar to each other [204] and one may ask for the purpose of two different formats for more or less the same tasks [82, 114]. Discussions about the necessity of both formats raise from time to time even decades later [25, 204]. However, OpenMath and MathML have been and are still developed alongside each other rather than competing with one another due to a large overlap of people working on both formats [204]. To summarize the coexistence today: MathML provides rendered visualizations for OpenMath, while the Content Dictionaries (CDs) from OpenMath add semantics to MathML[11].

The OpenMath Society maintains a set of standard CDs. A CD is a set of declarations (i.e., definitions, notations, constraints, etc.) for mathematical symbols, functions, operators, and other mathematical concepts. The idea behind the publicly maintained CDs by the OpenMath

[10] https://www.w3.org/TR/1998/REC-xml-19980210 [accessed 2021-10-01]

[11] A more detailed discussion about the history of both formats can be found at https://openmath.org/projects/esprit/final/node6.htm, https://openmath.org/om-mml/ [both accessed 2021-10-01], and [198, pp. 5].

Presentational MathML

```
<mrow>
  <msub>
    <mi>P</mi>
    <mi>n</mi>
  </msub>
  <mo>
    <!-- Invisible
    Funct. Appl.
    Unicode U+2061 -->
  </mo>
  <mrow>
    <mo>(</mo>
    <mi>x</mi>
    <mo>)</mo>
  </mrow>
</mrow>
```

Content MathML

```
<apply>
  <csymbol definitionURL="http://www.
      openmath.org/cd/orthpoly1.ocd"
      encoding="OpenMath">legendreP
  </csymbol>
  <ci>n</ci><ci>x</ci>
</apply>
```

OpenMath

```
<OMOBJ><OMA>
  <OMS name="legendreP" cd="orthpoly1"/>
  <OMV name="n"/>
  <OMV name="x"/>
</OMA></OMOBJ>
```

Listing 2.1: The Legendre polynomial in two MathML encodings and in OpenMath.

Society is to provide a ground truth for math declarations so that the used symbols become interchangeable among different parties. However, everybody can create new custom CDs which might be integrated into the existing standard set maintained by the OpenMath Society [90]. M. Schubotz [327], for example, proposed a concept for a CD that uses on the knowledge base Wikidata. More recently, B. Miller [258] created a content dictionary specifically for the functions in the DLMF.

Listing 2.1 compares both MathML markups with OpenMath. While the tree structures of content MathML and OpenMath cannot directly be compared with mathematical expression trees [331] (see also Section 2.2.4), the XML tree structure of both formats is unique. Both formats rely on the CD entry of the Legendre polynomial in `orthpoly1`[12]. Since the CD is from OpenMath, the OpenMath encoding does not require the entire `url`. The CD entry further specifies that the Legendre polynomial has two arguments. Hence, the following two siblings in the tree structure are considered to be the arguments. OpenMath specifically annotate them as `OMV` (for variable objects). Alternatively to the `orthpoly1` CD by OpenMath, one can also use Schubotz's [327] Wikidata CD to annotate P with the Wikidata item Q215405 or Miller's [258] DLMF CD to link P to §18.3 of the DLMF [98, (18.3)].

As previously mentioned, both formats (content MathML and OpenMath) are rather similar to each other [56, 343]. Hence, there are several ways to transform mathematical expressions between both formats [343], e.g., via XSLT [59, 342]. This transformation is possible without information retrieval techniques since both formats encode the same level of semantic information via CDs. Even though the primary goal for OpenMath was to provide a format that allows communication between mathematical software [19], most CAS do not support OpenMath directly. Instead, an independent project of research institutions funded by the European Union was launched to improve the *symbolic computation infrastructure in Europe*. The main

[12] https://openmath.org/cd/orthpoly1.html#legendreP [accessed 2021-10-01]

result of this project was the SCSCP protocol for inter-CAS communication via OpenMath. We will discuss the SCSCP protocol and the project more in detail in Section 2.2.3. Several CAS, including Maple [243] and Mathematica [44], implemented endpoints for the SCSCP protocol. Hence, via this new protocol, CAS support OpenMath to some degree. Apart from the protocol solution, there are some research projects available that use OpenMath as an interface to and between CAS and theorem prover formats [57, 152, 303, 338, 343].

2.2.1.3 OMDoc

Sometimes, it might be worthwhile to annotate the context of mathematical expressions with additional information explicitly. For example, an equation might be part of a theorem that has not been proven yet. Hence, that particular equation and its context should not be confused with a definition. Since this meta-information about mathematical expressions is organized on a document level, Kohlhase [198, 199] introduced another format, the Open Mathematical Document (OMDoc), to semantically describe entire mathematical documents. While formats like OpenMath or MathML encode the semantics of single expressions, which Kohlhase describes as the *microscopic* level, OMDoc aims for the *macroscopic*, i.e., the document level. This format can be especially useful for interactive documents [80, 85, 131, 150, 162, 201] and theorem prover [38, 146, 163, 340] which generally rely more on the meta information from a document level. Single math expressions in OMDoc are still encoded as OpenMath for the semantics and MathML for the visualization. In turn, this thesis focuses more on the formats that directly encode mathematical expressions rather than a *macroscopic* level encoding. Nonetheless, it should be noticed that a translation to a CAS might be different depending on the scope of an equation, e.g., an equation symbol in a definition differs from an equation symbol in an example. Heras et al. [152], for example, used OMDoc to interface CAS and theorem prover. Hence, the OMDoc format might be worth supporting once the translation reaches a level of reliability and comprehensiveness that the semantics on the document level matter (see the future work section 6.3).

2.2.2 Word Processor Formats

The previously explained formats of mathematics are beneficial for web applications and exchanging mathematical knowledge between systems. However, the underlying verbose XML data structure makes manual maintenance of these formats too cumbersome. In turn, MathML and OpenMath, considering a specific size, are almost always computer-generated. The actual source of the data, something a human manually typed, uses a different format, such as LaTeX, visual template editors, or image formats. In the following, we introduce formats and methods used to type mathematics in word processors manually.

2.2.2.1 LaTeX

LaTeX is currently the de-facto standard for writing scientific papers in the STEM disciplines [129, 220, 402] and has even been described as *the lingua franca of the scientific world* [220]. Numerous other word processors entirely or partially support LaTeX inputs. LaTeX was developed by Leslie Lamport and extended the TeX system with some valuable macros that make working with TeX easier [220]. TeX was developed by Donald E. Knuth [189, p.559] in 1977. Knuth was dissatisfied with the typography of his book, *The Art of Computer Programming* [189, pp. 5, 6, and 24] and created TeX to overcome the hurdles of consistently and reliably typesetting mathematical

formulae for printing. Today, there is no significant difference between LaTeX and TeX in terms of mathematical expressions. Hence, we continue using LaTeX as the modern successor and refer to TeX only to underline technical differences or to describe the underlying base for other TeX-like encodings. LaTeX provides an intuitive syntax for mathematics that is similar to the way a person would write the math by hand, e.g., by using the underscore to set a sequence of tokens in subscript.

LaTeX is an interpretable language that requires a parser. Theoretically, the flexibility of LaTeX (and especially the underlying TeX implementation) makes parsing LaTeX really challenging [187]. For example, TeX allows to redefine every literal at runtime, making TeX (and therefore LaTeX too) to a context-sensitive formal language. However, in practice, most LaTeX literals are generally not redefined. Instead, it is common to extend LaTeX with additional commands rather than redefining existing logic. Especially in mathematical expressions, several projects simply presume that LaTeX is parsable with a context-free grammar, which makes parsing mathematical expressions in LaTeX a lot simpler [71, 402].

Since LaTeX is the standard to typeset mathematics, there are numerous of translation tools to the webstandard MathML available [133, 135, 159, 257, 267, 335, 374] (see also MathML explanation in Section 2.2.1). In the next Section 2.3, we will focus more closely on translations between LaTeX and MathML. LaTeX is also a standard target encoding for Optical Character Recognition (OCR) techniques [406, 411], which retrieve mathematical expressions from images or PDF files (see Section 2.2.4). LaTeX focus solely on the representation of math (similar to presentation MathML). Additionally, recent studies try to explore the capabilities of trained vector representations of LaTeX expressions [121, 15, 215, 360, 400, 404] to explore new similarity measure and search engines [404], classification approaches [404], and even automatically generating new LaTeX expressions [400]. Nonetheless, the effectiveness of capturing the semantic information with these methods is controversial [9].

LaTeX to CAS converters Most relevant for our task are existing translation approaches directly from LaTeX to CAS sytanxes. These translators can be categorized in two groups: (1) CAS internal import functions and (2) external programs for specific or multiple CAS. Mathematica [391] and SymPy [357] are two CAS with the ability to import LaTeX expressions directly. SymPy's import function was ported from the external `latex2sympy`[13] project. Examples of external tools are SnuggleTeX [251] and our in-house translator LaCaST [3, 13]. SnuggleTeX is a LaTeX to MathML converter with the experimental feature to perform translations to the CAS Maxima [324]. LaCaST is the predecessor project of this thesis and focused on translating semantic LaTeX from the DLMF to the CAS Maple.

All of these converters are rule-based translators, i.e., they perform translations on hard-coded pre-defined conversion rules. SnuggleTeX support translations to Maxima since version 1.1.0 [251]. The tool allows users to manually predefine translation rules, such as interpreting e as the mathematical constant, Γ as the Gamma function, or f as a general function. SnuggleTeX is no longer actively maintained and mostly fail to translate general expressions. The developers themselves declare the translation to Maxima as experimental and limited[14]. SymPy, in contrast,

[13] The project is therefore no longer actively developed but still available on GitHub: https://github.com/augustt198/latex2sympy [accessed 2021-10-01]

[14] https://www2.ph.ed.ac.uk/snuggletex/documentation/semantic-enrichment.html [accessed 2021-10-01]

is actively maintained and provide a more sophisticated import function for LaTeX expressions. SymPy's import function parses a given LaTeX expression via ANTLR[15] and traverses through the parse tree to convert each token (and subtree) into the SymPy syntax. SymPy uses a set of heuristics that mostly cover standard notations, including `\sin`. Additionally, it uses pattern matching approaches to identify typical mathematical concepts, such as the derivative notation in $\frac{d}{dx}\sin(x)$. Similarly, LaCASt first parses the input expression with the Part-of-Math (POM) tagger [402] and performs translations by traversing through the parse tree. The POM tagger tags tokens with additional information from external lexicon files. LaCASt manipulates these lexicon files to tag tokens with their appropriate translation patterns. LaCASt takes the translation patterns attached to a single token and fills them with with the following and preceding nodes in the parse tree to perform a translation. Within this thesis, we will extend LaCASt further with pattern matching techniques and human-inspired heuristics to perform more general formulae, including the derivative notation example, sums, products, and other operators. A more detailed discussion about the first version of LaCASt is available in [13].

While SymPy and SnuggleTeX are open source and allows interested readers to analyze the internal implementation details, we can only speculate about the solutions in proprietary software, such as Mathematica. As we saw in Table 1.2 (and later in Chapter 4), Mathematica seems to follow a pattern recognition approach to link known notations, such as $P_n^{(\alpha,\beta)}(x)$, to their internal counterparts, such as `JacobiP[n, \[Alpha], \[Beta], x]`. Since Mathematica (nor does any other CAS or mentioned converter) analyze the textual context of a formula, importing ambiguous notations generally fail. Since the internal logic (and therefore the underlying patterns) is hidden, it is difficult to estimate the accuracy and power of Mathematica's LaTeX import function. As an alternative to Mathematica itself, one can use WolframAlpha[16] [309]. WolframAlpha is described as a knowledge or answer engine. Technically, WolframAlpha is a web interface which uses Mathematica as backbone for computations. WolframAlpha performs numerous of pre-processing and interpretation steps to allow users to generate scientific information without inputting specific Mathematica syntax [64, 383].

Table 2.2 compares the converters on our introduction examples (see Table 1.2). The table contains also LaCASt first version (published in 2017 [3]) for comparison. We observe that WolframAlpha clearly performs best on this simple general inputs. The reason is that WolframAlpha focus on a broad, less scientific audience which allows the system to make several assumptions. On more topic specific inputs, such as $P_n^{(\alpha,\beta)}(\cos(a\Theta))$, it fails. This is further underlined by the fact that Mathematica itself has no trouble interpreting $P_n^{(\alpha,\beta)}(\cos(a\Theta))$. This indicates that both systems are optimized for their expected user groups. On these simple cases, SymPy also performs better compared to Mathematica. However, SymPy's size and support of special functions is not comparable with Mathematica and therefore falls behind Mathematica on a more scientific dataset, such as the DLMF.

A more sophisticated evaluation on 100 randomly selected DLMF formulae revealed that Mathematica can be considered the current state-of-the-art for translating LaTeX to CAS. Nonetheless, it only translated 11 cases correctly compared to 7 successful translations by SymPy and 22 by LaCASt. The full benchmark is available in Table E.1 in Appendix E.1 available in the electronic supplementary material.

[15] ANother Tool for Language Recognition (ANTLR): https://www.antlr.org/index.html [accessed 2021-10-01]

[16] Often stylized with Wolfram|Alpha

Table 2.2: LaTeX to CAS translation comparison between Mathematica's (MM) and SymPy's (SP) import functions, SnuggleTeX (ST) translation to Maxima, WolframAlpha (WA) interpretation of LaTeX inputs, and the first version of LaCaST (LCT_1)

LaTeX	Rendering	MM	SP	ST	WA	LCT_1
`\int_a^b x dx`	$\int_a^b x dx$	✘	✔	✘	✔	✘
`\int_a^b x \mathrm{d}x`	$\int_a^b x \mathrm{d}x$	✘	✘	✘	✔	✘
`\int_a^b x\, dx`	$\int_a^b x\, dx$	✔	✔	✘	✔	✘
`\int_a^b x\; dx`	$\int_a^b x\; dx$	✘	✔	✘	✔	✘
`\int_a^b x\, \mathrm{d}x`	$\int_a^b x\, \mathrm{d}x$	✘	✘	✘	✔	✘
`\int_a^b \frac{dx}{x}`	$\int_a^b \frac{dx}{x}$	✘	✔	✘	✔	✘
`\sum_{n=0}^N n^2`	$\sum_{n=0}^{N} n^2$	✔	✔	✔	✔	✔
`\sum_{n=0}^N n^2 + n`	$\sum_{n=0}^{N} n^2 + n$	?	?	✘	?	?
`{n \choose m}`	$\binom{n}{m}$	✘	✘	✘	✔	✘
`\binom{n}{m}`	$\binom{n}{m}$	✔	✔	✔	✔	✔
`P_n^{(\alpha,\beta)}(\cos(a\Theta))`	$P_n^{(\alpha,\beta)}(\cos(a\Theta))$	✔	✘	✘	✘	✔
`\cos(a\Theta)`	$\cos(a\Theta)$	✔	✔	✔	✔	✔
`\frac{d}{dx} \sin(x)`	$\frac{d}{dx}\sin(x)$	✘	✔	✘	✔	✘

Since LaTeX can be easily extended with new content via macros, some projects try to semantically enhance LaTeX with unambiguous commands. The two most comprehensive projects are semantic LaTeX and sTeX.

2.2.2.2 Semantic/Content LaTeX

The Jacobi polynomial in LaTeX and semantic LaTeX

```
1 P_n^{(\alpha , \beta)}(x)           % Generic LaTeX
2 \JacobipolyP{n}{\alpha}{\beta}@{x} % Semantic LaTeX
```

Listing 2.2: The Jacobi polynomial in LaTeX (line 1) and semantic LaTeX (line 2).

Semantic LaTeX (also known as content LaTeX) was developed by Bruce Miller [260] at the National Institute of Standards and Technology (NIST) to semantically enhance the equations in the DLMF [403]. Essentially, semantic LaTeX is a set of custom LaTeX macros which are linked to unique definitions in the DLMF. Consider for example the Jacobi polynomial in Listing 2.2. The general LaTeX expression does not contain any information linked to the Jacobi polynomial. However, semantic LaTeX replaces the general expression with a new macro `\JacobipolyP` which is linked to the DLMF [98, (18.3#T1.t1.r2)][17]. In addition, all variable arguments (parame-

[17] Hereafter, we refer to specific equations in the DLMF by their labels. The label can be added to the base URL of the DLMF. For example, the sine function is defined at `4.14.E1`, which can be reached via https://dlmf.nist.gov/4.14.E1 [accessed 2021-10-01].

ters and variables) are separated and ordered following the function command. This separation is essential to disambiguate notations. For example, the sine function is sometimes written without parenthesis, such as $\sin x$, resulting in ambiguous semantic notations, such as in $\sin x + y$. The semantic LaTeX macros allow to visualize this expression but encode it unambiguously via `\sin@@{x+y}` (which is rendered as $\sin x + y$). Originally, the semantic LaTeX helped to develop a reliable search engine for the DLMF [260]. Nowadays, the macros are also in use in other projects and have been even extended for the Digital Repository of Mathematical Formulae (DRMF) [77, 78], an outgrowth of the DLMF.

Semantic LaTeX will play a crucial role in the rest of this thesis because it allows us to stick with the easily maintainable syntax of LaTeX but semantically elevates the information of math expressions to a level that can be exploited for translations towards CAS [3, 8, 13]. The main reason is that the semantic LaTeX macros mostly cover OPSF from the DLMF. OPSF are a set of functions and polynomials which are generally considered as important, such as the trigonometric functions (also categorized as elementary functions), the Beta function, or orthogonal polynomials. Most OPSF have more or less well-established names and standard notations. The DLMF (i.e., especially the original book [276]) is considered a standard reference for OPSF [381]. General-purpose CAS, such as Mathematica and Maple, focus also on the comprehensive support of OPSF [381]. Hence, semantic LaTeX macros play a crucial role for translations from LaTeX to CAS syntaxes. Since CAS syntaxes are programming languages, CAS can be extended with new code. However, translating new math formulae to CAS can become arbitrarily complex. Consider the prime counting function would be not supported by Mathematica. In this case, $\pi(x + y)$ cannot be translated to a simple mathematical formula in the syntax of Mathematica but would require entire new subroutines. Therefore, a comprehensive, viable, and reliable translator from LaTeX to the syntax of CAS should maximize its support for OPSF in order to be useful.

Definition 2.1 provides a brief definition for the elements of a semantic macro. While the semantic source of the DLMF is publicly available [403], the actual definitions, i.e., the LaTeX style files, of the macros, are still private[18]. B. Miller provided access to the definitions of the macros for this thesis. Later in this thesis, we will rely on additional meta-information given for each semantic macro. This includes default parameters and variables, a short textual description, and links to the DLMF CD [258]. Further information is not explicitly given in the macro definition files. For example, function constraints, domains, branch cut positions, singularities, and other properties are only given in the DLMF.

As previously mentioned, we[19] developed LaCASt for translating semantic LaTeX DLMF formulae to CAS [3, 13]. The first version did not contain any disambiguation steps or pattern matching approaches to deduce the intended meaning of an expression. Instead, if fully relied on the semantic LaTeX macros to perform translations to Maple. For example, sums or products were not supported directly but required the semantically enhanced macros from the DRMF [77, 78]. The source of LaCASt is not yet publicly available[20] due to the dependency to the POM tagger [402] and the semantic LaTeX macros [260, 403] but accessible via open API endpoints[21].

[18] As of 2021-10-1.

[19] The first version of LaCASt was the subject of my Master's thesis and laid the foundation for a reliable translation from semantic LaTeX to multiple CAS.

[20] As of 2021-10-01.

[21] The API contains a Swagger UI and is reachable at https://vmext-demo.formulasearchengine.com [accessed 2021-10-01]. LaCASt is available under `math/translation` path (in the math controller). The experimental

Definition 2.1: The elements of a semantic macro

A semantic LaTeX macro is a LaTeX macro with a unique name followed by a number of arguments. Certain elements of the following arguments are optional but the order remains the same. While a caret and primes are interchangeable, each order would have a different meaning, as it can be seen in the example below.

A semantic macro and its arguments:

`\macro`	The unique semantic macro name with a backslash
`[optPar]`	An optional parameter in square brackets
`{par}`	Parameters in curly brackets
`'` or `^`	Optional prime symbols or a caret for power notations
`@`	A number of `@` symbols to control the visualization of the macro
`{var}`	Variables in curly brackets

Examples:

`\sin@{x}` $\longrightarrow \sin(x)$
`\sin@@{x}` $\longrightarrow \sin x$
`\BesselJ{\nu}''^2@{z}` $\longrightarrow J_n''^2(z)$
`\BesselJ{\nu}^2''@{z}` $\longrightarrow (J_n^2)''(z)$
`\genhyperF{2}{1}@{a,b}{c}{z}` $\longrightarrow {}_2F_1(a,b;c;z)$
`\genhyperF{2}{1}@@{a,b}{c}{z}` $\longrightarrow {}_2F_1\left(\begin{smallmatrix}a,b\\c\end{smallmatrix};z\right)$
`\genhyperF{2}{1}@@@{a,b}{c}{z}` $\longrightarrow {}_2F_1(z)$

Apart from LaCaST, LaTeXML [257] is another tool that supports semantic LaTeX and provides conversions to LaTeX, MathML, and a variety of image formats. LaTeXML was also developed by B. Miller with the original goal to support the development of DLMF [133]. LaTeXML is a general LaTeX to XML converter. However, in order to support the development of the DLMF, LaTeXML is able to fully load semantic LaTeX definition files to convert semantic LaTeX into semantically appropriate content MathML. With this ability, LaTeXML is generally capable of converting other LaTeX encodings too, such as the following sTeX.

2.2.2.3 sTeX

sTeX refers to semantic TeX and should not be confused with B. Miller's semantic LaTeX. sTeX was developed around 2008 [194, 195, 200] with the goal to semantically annotate LaTeX documents with semantic macros. Specifically, sTeX should serve as a source format to generate the semantic document format OMDoc. While the underlying motivation and technical solution of sTeX and semantic LaTeX are very similar, there are some core differences between both formats. Semantic LaTeX was developed specifically for the DLMF and, therefore, provide semantic macros for OPSF. In particular, a semantic macro in the DLMF represents a specific unique function. In turn, sTeX aim to cover general mathematical notations and provide a logic to semantically annotate general functions and symbols. Consider the aforementioned example $\pi(x+y)$. If π is referring to the prime counting function, we can resolve the ambiguity with semantic LaTeX via `\nprimes@{x+y}` since the semantic macro `\nprimes` is referring to that function.

flag performs pattern matching approaches described later in this thesis. The label allows to specify a DLMF equation label to perform specific assumptions (e.g., that i is an index and not the imaginary unit).

In ꝜTEX, an author can use modules and IDs to define the function and set the notation via `\symdef{\pi}[1]{\prefix{\pi}{#1}}`. While this makes the interpretation of $\pi(x+y)$ unambiguous, an underlying definition is still missing. Hence, ꝜTEX provides the option to link symbols with their definitions in the document. This definition linking underlines the original motivation and connection to the semantic document format OMDoc.

Since ꝜTEX is not limited to specific domains, we could define any notation we want in our semantic document. On the other hand, this generalizability of ꝜTEX makes the format more verbose and somehow similar to a programming language. In ꝜTEX, we need to define and declare symbols explicitly. In addition, a defined new symbol still needs to be manually linked to an underlying definition. In semantic LaTeX, the macro itself is linked to the appropriate definition in the DLMF. ꝜTEX provide access to predefined sets of macros that aim to cover K-14 mathematics [195].

In conclusion, ꝜTEX is flexible but verbose. The format is useful when it comes to annotating a general mathematical document semantically. However, the strength of ꝜTEX, for example, the ability to define any symbol with specific semantics, is generally not very important for translations to CAS. CAS have a fixed set of supported functions and often try to mimic common notation styles, e.g., one does not need to define $-$ as a unary postfix operator in -2. In turn, a translation from LaTeX to CAS faces the issue of identifying the name of the functions involved, its arguments, and the appropriate mappings to counterparts in CAS syntax. Semantic LaTeX, on the other hand, provides a syntax that makes it easy to solve these issues. The name of the function is directly encoded in the name of the macro, the arguments are explicitly declared and distinguishable (by curly brackets), and a mapping to an appropriate counterpart in the CAS can be more easily found due to the large overlap of functions in the DLMF and supported functions in CAS.

As previously mentioned, LaTeXML [257] is able to load TeX definition files and support conversion to XML encodings. Hence, LaTeXML can transform ꝜTEX expressions to content MathML[200]. The ability to link ꝜTEX symbols with their definitions in a document or external source further makes it to a source for generating entire semantic enhanced OMDoc documents [195]. ꝜTEX could be also used as an alternative to semantic LaTeX for translations to CAS. However, due to the natural overlap of functions in the DLMF and CAS, at some point in the development of a translation process on ꝜTEX, we would create semantic enhanced macros for OPSF similar to the existing semantic LaTeX macros. Hence, using ꝜTEX in comparison to semantic LaTeX has no direct advantages to perform translations towards CAS. The higher flexibility of ꝜTEX makes it a good candidate for translations beyond OPSF.

2.2.2.4 Template Editors

Since LaTeX is an interpretable language with over ten thousand mathematical symbols alone [280], learning LaTeX syntax is often simply too time-consuming and complex for many users. To provide an easier access to rendered mathematics, especially in so-called *what you see is what you get* (WYSIWYG) editors, such as Microsoft's Office programs[22] or Wikipedia's visual article editor[23], template editors become the norm. Template editors provide visual templates

[22] https://support.microsoft.com/en-us/office/equation-editor-6eac7d71-3c74-437b-80d3-c7dea24fdf3f [accessed 2021-10-01]

[23] The wikipedia's article about formula editors (`https://en.wikipedia.org/wiki/Formula_editor` [accessed 2021-10-01]

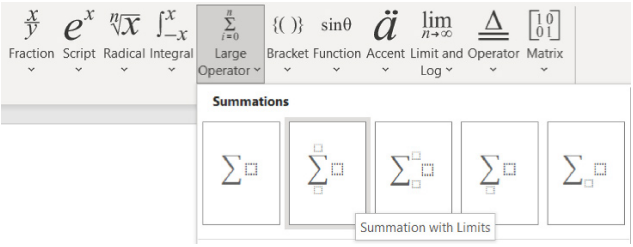


Figure 2.2: The math template editor of Microsoft's Word [395].

of standard mathematical notations so that the user only needs to fill in the remaining spaces. Figure 2.2 shows the template editor of Microsoft's Word [395] for a snippet of the templates for sums. Modern graphic interfaces of CAS also often contain such template editors to improve the user experience further. In comparison to LaTeX, template editors are generally easier to use but limited to the offered templates. Hence, for more complex expressions, template editors are often described as confining [273]. Template editors do not introduce a new math format. The editors only provide a different input method but encode the mathematical formulae in system-specific formats, such as MathML in Microsoft's Word or Maple syntax in Maple.

2.2.3 Computable Formats

So far, we have covered the major formats that focus on the presentation of mathematical expressions and on formats that capture the semantics. Even though formats like content MathML, OpenMath, and the semantic LaTeX extensions can resolve the ambiguity of math formulae, they are not computable formats, i.e., we cannot perform actual calculations and computations on them. The syntax of a computable format is a formal language in which every word is linked to specific subroutines. Much like programming languages, computable formats are semantically unambiguous and interpretable. In turn, computable formats are generally part of a larger software package that ships an interpreter to parse inputs and an engine that performs the computations. In the following, we briefly discuss CAS and theorem prover formats as examples of computable formats. We will not specifically focus on math packages for specific programming languages, such as C++ [168], Python [252] or Java [79]. Most CAS and theorem provers, however, internally rely on those *lower-level* packages to some degree.

2.2.3.1 Computer Algebra Systems

A CAS is a mathematical software that can perform a variety of mathematical operations on math inputs, such as symbolic manipulations, numeric calculations, plotting and visualization, simplification, and many more [76, 81, 128, 413]. With the increasing power of computers, CAS became a crucial part of the modern scientific world [32, 262, 352, 356] and are widely used for mathematical problem solving [49, 51, 127, 216, 414], simulations [46, 142, 166, 265, 294], symbolic manipulations [115, 325], and even for teaching students from schools to universities [158, 237, 244, 350, 363, 365, 389, 390]. Due to their complexity, CAS are often large and expensive proprietary software packages [36, 164, 393]. However, there are several well-known open

source options available [42], such as SymPy [252], Axiom[24] [176], and Reduce[25] [151]. Many CAS focus on specific domains or mathematical tasks, such as Cadabra [289, 290, 291] (tensor field theory), FORM [372] (particle physics), GAP [177] (group theory and combinatorics), PARI/GP [283] (number theory), or MATLAB [164] (primarily for numeric computation). In contrast, general-purpose CAS, including Mathematica [393], Maple [36], Axiom [176], SymPy [178, 252], Maxima [264, 324], or Reduce [151], aim to provide a large set of tools and algorithms that are beneficial for many mathematical applications. Therefore, general-purpose CAS support a large number of OPSF, since these functions and polynomials are used in a large variety of different scientific fields, from pure and applied mathematics to physics and engineering. Therefore, we primarily focus on translations to general-purpose CAS in this thesis rather than to domain-specific CAS.

The input formats of general-purpose CAS are often multi-paradigm programming languages [88], i.e., they combine multiple standard programming features, such as functional, mathematical, and procedural approaches. Major CAS generally use their own input language, such as the *Wolfram Language* in Mathematica [392]. Like any programming language, the input format must be unambiguous to the underlying parser of the CAS so that every keyword is uniquely linked to subroutines in the CAS engine. This link to a subroutine makes the expression computable. In contrast, the semantic LaTeX macros are linked to theoretical mathematical concepts defined in the DLMF but not with specific implementations. Hence, a translation to a CAS syntax requires to link mathematical notations, e.g., $\Gamma(z)$, that refer to specific mathematical concepts, e.g., the Gamma function, to the correct sequence of keywords in the CAS, e.g., `GAMMA(z)` in Maple.

Since computable languages naturally encode the highest level of semantic information in their expressions, a translation towards other systems that encode less semantic information is possible with a comprehensive list of simple mapping rules. Many CAS therefore provide a variety of different output formats, from LaTeX to MathML (including content MathML) and images. Translations between CAS or other mathematical software, such as theorem prover, require more sophisticated mappings due to system-specific implementations [110]. From 2006 to 2011, a joint research project funded by the European Union with over 3 Million Euro launched intending to improve the symbolic computation infrastructure for Europe[26]. The result of the *SCIEnce project* was the *Symbolic Computation Software Composability Protocol* (SCSCP) [119, 361], which uses the OpenMath encoding to transfer mathematical expressions. Using the SCSCP, interfaces for GAP [206], KANT [120], Maple [243], MuPAD [155], Mathematica [44], and Macaulay2 [311] were implemented.

Note that there are solutions available that do not require any translation between LaTeX and CAS. For example, the CAS syntax of Cadabra [291] is a subset of TeX itself. Similarly, SageTeX[27] is a LaTeX package that allows authors to enter SageMath [317] expressions into LaTeX documents, turning the document into an interactive document [201] to some degree. SageMath is a general-purpose CAS that relies on existing solutions for domain-specific tasks, such as GAP [177] for group theory or PARI/GP [283] for number theory problems. These solutions do not require

[24] Open source since 2001 (first released in 1965).
[25] Open source since 2008 (first released in 1963).
[26] EU FP6 project 026133: https://cordis.europa.eu/project/id/26133/ [accessed 2021-10-01]
[27] https://doc.sagemath.org/html/en/tutorial/sagetex.html [accessed 2021-10-01]

translations since the input must be provided in the syntax of the CAS. Hence, a translation must be performed manually or via external tools.

In the introduction, we mentioned potential issues of CAS with multi-valued functions. Multi-valued functions map values from a domain to multiple values in a codomain and frequently appear in the complex analysis of elementary and special functions [8]. Prominent examples are the inverse trigonometric functions, the complex logarithm, or the square root. All modern CAS[28] compute multi-valued functions on their principle branches which makes these functions effectively single-valued (e.g., a calculator always returns 2 for $\sqrt{4}$ rather than ± 2 or just -2). The correct properties of multi-valued functions on the complex plane may no longer be valid by their counterpart functions on CAS, e.g., $(1/z)^w = 1/(z^w)$ for $z, w \in \mathbb{C}$ and $z \neq 0$ is no longer valid within CAS. The positioning and handling of branch cuts in CAS is often discussed in scientific articles and generally prominantly noticed in CAS handbooks [83, 84, 91, 108, 171, 172]. However, especially in more complex scenarios, it is easy to lose track of branch cut positioning and evaluate expressions on incorrect values. We provide a more complex example and a more detailed explanation of branch cuts in Appendix A available in the electronic supplementary material. To the best of our knowledge, no available translation tool from, to, or between CAS (including the SCSCP solutions) consider branch cut positions.

2.2.3.2 Theorem Prover

The idea of automated reasoning and deduction systems is as old as computers [147]. With the power of computers and a strict axiomatic approach as in *Principia Mathematica* [385], computers can perform automatic reasoning steps to discover and proof new mathematical theorems. Up until today, automated theorem proving and verifying is an extensive research area with an ever-growing interest [266, 354, 384]. There are numerous theorem provers and proof assistants systems available, such as HOL Light [146], HOLF [340], or Isabelle [287]. However, focusing on the deduction, the encoding of theorem provers generally goes beyond mathematical expressions. The syntax provides specific options for assumptions, links between multiple concepts, and logical steps. An example of a proof by Isabelle, which clearly visualizes the different notation of theorem provers and CAS, is given in Appendix C available in the electronic supplementary material.

Nonetheless, theorem prover formats are computable formats with specific mathematical applications. Hence, there is a genuine interest in transferring findings and solutions from one system to the other. There are some translation approaches between theorem prover and CAS available, from direct translations [28, 148] to translations over OpenMath [57, 338] and OMDoc [152]. Theorem provers are generally unable to *compute* a single mathematical formula in the sense of numeric computations or symbolic manipulations. Hence, we do not choose theorem provers as the target computable format for our desired translation process.

2.2.4 Images and Tree Representations

In the following, we briefly discuss formats with the specific visualization focus: images and tree representations. Especially older literature is often only available in digital scans, and many copies of publications do not provide access to the original LaTeX source. Images can be con-

[28] The authors are not aware of any example of a CAS which treats multi-valued functions without adopting principal branches.

sidered as the purest presentational format of mathematical expressions. Tree representations of math expressions, on the other hand, are more theoretical concepts to visualize the logical or presentational structure of math. Tree representations are primarily used for explanation purposes to underline or visualize an idea or concept. Parse trees, as a generated specific tree format of mathematical string inputs, on the other hand, play a crucial role in almost every mathematical software tool. Often, digital mathematical formats try to mimic the logical tree structure of math expressions. This is also one of the reasons why the web formats (MathML and OpenMath) use XML to encode mathematical content.

Symbolic Layout, Operator, Parse, and Expression Trees Mathematical expressions are often represented in tree structures. For example, MathML itself is an XML tree data structure. Moreover, mathematicians often have a logical but theoretical tree representation of a formula in mind in which numbers and identifiers are terminal symbols (leaves) and children of math operators, functions, and relations [192, 331]. These so-called *expression trees* are more or less a theoretical structure and are mainly used to visualize logical correlations and connections in mathematical expressions. Schubotz et al. [331] attempted to automate the visualization process of expression trees based on cross-referenced MathML data which resulted in VMEXT, a visualization tool for MathML. Figure 2.3 shows a possible expression tree visualization for the Jacobi polynomial definition in terms of the hypergeometric function.

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

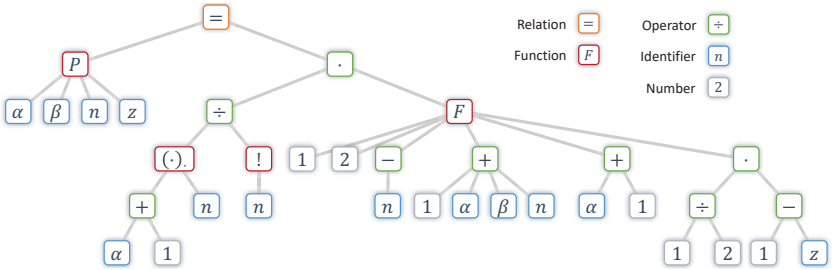


Figure 2.3: An expression tree representation of the explicit Jacobi polynomial definition in terms of the hypergeometric function.

For visualization and education purposes, these tree representations can be beneficial. However, generating these trees requires a deep understanding of the logical structure of the expression. In addition, there is no exact definition available for expression trees. Hence, the exact visualization is often up for discussions, e.g., whether parameters are children similar to variables or part of the function node itself [9]. A missing standard definition makes expression trees unreliable and, therefore, less practical for a mathematical encoding.

Parse Trees Parse trees are generated tree representations of source expressions (strings). These trees are generated by a parser that follows a strict set of rules, e.g., a context-free grammar [101, 188, 298]. Mathematical LaTeX (as a subset of TeX) considering a couple simplifications

(e.g., no re-defined standard literals and macros) can also be described in a context-free grammar [402] even though TEX itself is Turing complete [133, 135, 187]. The POM tagger [402], for example, parses mathematical LATEX following a context-free grammar. Similarly, Chien and Cheng [71] build a custom context-free grammar parser for their semantic tokenization of mathematical LATEX expressions. LATEXML follows the more sophisticated TEX-like digestion methods [187] to parse entire TEX files [133, 135]. CAS inputs are parsed internally for further processing [138, 392]. Maple's internal parser also generates a parse tree in which equivalent nodes are merged together for more efficient memory usage (mathematically speaking, this data structure is no longer a valid tree but instead a directed, acyclic graph, or simply DAG) [3, 13].

In contrast to theoretical tree representations, such as the mentioned expression trees, parse trees are crucial for many applications because a tree data format is more easy to process due to their structural logic [93, 242, 286, 406]. While string sequences of commands may contain ambiguities, tree data structures are unique and provide easy access to single logical nodes, groups of nodes, and their dependencies. Hence, parsing a mathematical input (such as in CAS inputs or LATEX expressions) is typically the first step in any processing pipeline. Later in this thesis, we will also take advantage of tree representations by defining a translation between math formats as graph transformations on their tree representations. To generate a tree representation of mathematical LATEX formats, we can either build a custom parser [71] or rely on existing parsers, such as LATEXML [257] or the POM tagger [402]. Parse trees (and other custom generated tree formats that are generated by analyzing a given input) can also be categorized into symbol layout trees (for presentational formats) and operator trees (for content/semantic formats) [406]. For example, parsing LATEX may result in a symbol layout tree that describes the visual structure of formulae while parsing semantic LATEX (or CAS inputs) may result in operator trees which describe the logical mathematical structure of the input.

Images From pixel graphics (e.g., JPEG or PNG) to vector graphics (e.g., Scalable Vector Graphics (SVG)) and document formats (e.g., PDF), mathematical expression can appear in a variety of different image formats. The two-dimensional structure of mathematics makes drawing mathematical formulae on a sheet of paper or touch screens the most intuitive input method for mathematics. In addition, with rising digitization, scans of old scientific articles are no longer the only source of math images. Handwriting systems are more and more adopted in offices and educational institutions [411]. In 2016, Wikipedia switched from non-scalable PNG images to vector graphics for visualizing mathematics [17] (see Appendix B available in the electronic supplementary material, for a more sophisticated overview of the history of math formulae in Wikipedia).

However, image formats are not directly interpretable and are, therefore, less machine-readable. Hence, the first step of analyzing mathematics in images is always converting into a more machine-readable, digital format. The majority of conversion approaches, including handwriting recognition and Optical Character Recognition (OCR), focus on translations to MathML or LATEX [373, 406, 411]. Hence, for our task (translating presentational formats to computable formats), starting with image formats is not practically useful.

Nonetheless, one particular issue in math OCR is also of interest for our translation task: detection of inline mathematics. In image formats, detecting inline mathematics is difficult because formulae may blend into texts [74, 125, 126, 230, 398]. Even a detection of italic fonts

can be a challenging task [66, 112, 113, 233]. A variable can easily be confused with words, such as the Latin letter 'a.' A similar issue raises in other formats, including LaTeX documents and Wikipedia articles when an author does not correctly annotate mathematical formulae. In Wikipedia, for example, single identifiers in a text are often put in italic font rather than in mathematical environments. The capability of using UTF-8 encodings incites Wikipedia editors to put inline mathematics into the text directly, even when special characters are involved. For example, the mathematical expression $0 \leq \phi \leq 4\pi$ in the English Wikipedia article about Jacobi polynomials[29] is a sequence of UTF-8 characters and thus challenging to identify as mathematics for MathIR parser. Nevertheless, identifying all mathematical expressions in a document might be necessary for more reliable translations towards computable formats. For example, the mentioned relation of ϕ defines the domain of the Wigner d-matrix and is of interest for automatic evaluations (see Chapter 5).

2.2.5 Math Embeddings

Word embedding techniques has received significant attention over the last years in the Natural Language Processing (NLP) community, especially after the publication of word2vec [256]. Therefore, more and more projects try to adapt this knowledge for solving tasks in the MathIR arena [121, 15, 141, 215, 353, 360, 400, 404]. These projects try to embed math expressions into natural languages to create a vector representation of the formula. A vector representation is the data format with the highest machine readability among all other representations of mathematical formula. The math embeddings successfully enabled a new approach to measure the similarity between math expressions, which is especially useful for math search, classification, and similar tasks [121, 215, 400, 404].

Considering the equation embedding techniques in [215], we devise three main types of mathematical embedding: *Mathematical Expressions as Single Tokens*, *Stream of Tokens*, and *Semantic Groups of Tokens*. In the following we briefly explain each type on an example expression containing the inequality for Van der Waerden numbers

$$W(2,k) > 2^k/k^\varepsilon. \tag{2.1}$$

This expression is the first entry in the the MathML benchmark [18] we are going to explain in detail in Section 2.3.

Mathematical Expressions as Single Tokens So called equation embeddings (EqEmb) were introduced by Krstovski and Blei [215] and use an entire mathematical expression as one token. In a one-token representation, the inner structure of the mathematical expression is not considered. For example, $W(r,k)$ is represented as one single token t_1. Any other expression, such as $W(2,k)$ in the context, is an entirely independent token t_2. Therefore, this approach does not learn any connections between $W(2,k)$ and $W(r,k)$. However, [215] has shown promising results for comparing mathematical expressions with this approach.

Stream of Tokens As an alternative to embedding mathematical expressions as a single token, one can also represent an expression through a sequence of its inner elements. For example, considering only the identifiers in Equation (2.1), it would generate W, k, and ε as a sequence/stream of tokens. This approach has the advantage of learning all mathematical tokens.

[29] https://en.wikipedia.org/wiki/Jacobi_polynomials#Applications [accessed 2021-10-01]

However, this method also has some drawbacks. Complex mathematical expressions may lead to long chains of elements, which can be especially problematic when the window size of the training model is too small. Naturally, there are approaches to reduce the length of chains. Gao et al. [121] use a continuous bag of words (CBOW) approach and embed all mathematical symbols, including identifiers and operands, such as $+$, $-$ or variations of equalities $=$. Krstovski and Blei [215] also evaluated the stream of tokens approach but do not cut out symbols. They trained their model on the entire sequence of tokens that the LaTeX tokenizer generates. Considering Equation (2.1), it would result in a stream of 13 tokens. They use a long short-term memory (LSTM) architecture to overcome the limiting window size and further limit chain lengths to $20 - 150$ tokens. Usually, in word embedding, such behaviour is not preferred since it increases the noise in the data.

We [15] also use this stream of tokens approach to train our model on the DLMF without any filters. Thus, Equation (2.1) generates all 13 tokens. Later in Section 3.1, we show another model trained on the arXiv collection, which uses a stream of mathematical identifiers and cut out all other expressions, i.e., in case of (2.1), we embed W, k, and ε. We presume this approach is more appropriate to learn connections between identifiers and their definiens. We will see later that both of our models trained on math embedding are able to detect similarities between mathematical objects, but does not perform well on detecting connections to word descriptors.

Semantic Groups of Tokens The third approach of embedding mathematics is only theoretical. Current MathIR and Machine Learning (ML) approaches would benefit from a basic structural knowledge of mathematical expressions, such that variations of function calls (e.g., $W(r, k)$ and $W(2, k)$) can be recognized as the same function. Instead of defining a unified standard, current techniques use their ad-hoc interpretations of structural connections. We assume that an embedding technique would benefit from a system that can detect the parts of interest in mathematical expressions before any training process. However, such a system still does not exist. Later in Section 3.2, we will introduce a new concept to interpret logical groups of mathematical objects that may enable a semantic embedding in the future.

It is important to mention that it remains unclear to what degree math semantic information can be embedded in a vector representation [9]. Since there is no answer to this question, we have not included math embeddings (i.e., vector representations of formulae) to Figure 2.1. Nonetheless, a vector representation can be decoded into a CAS syntax representation again to perform a ML based translation [296]. We will elaborate on such an approach more in Chapter 4.

2.3 From Presentation to Content Languages

We introduced several different formats for encoding mathematical formulae digitally and provided an overview of several existing conversion tools between these formats. Considering Figure 2.1, the goal of this thesis, i.e., making presentational math computable, requires to convert mathematical formats from the most left of the figure to the most right. We have chosen LaTeX as the source format and general-purpose CAS syntaxes for the target formats. Considering the merit of communicating knowledge in sciences, it comes to no surprise that there are numerous of translation tools and theoretical approaches available to convert math formulae between multiple formats, including our goal translation from LaTeX to CAS syntaxes.

Since MathML is the web standard which is supported by several CAS at least partially [57, 110, 303, 338] (or OpenMath respectively), a translation from LaTeX to CAS could be performed over MathML (preferably content MathML). In this section, we analyze state-of-the-art LaTeX to MathML converters to study the applicability of using MathML as an intermediate format for translations from LaTeX to CAS syntaxes. This section was previously published [18].

2.3.1 Background

In the following, we use the Riemann hypothesis (2.2) as an example to explain typical challenges of converting different representation formats of mathematical formulae:

$$\zeta(s) = 0 \Rightarrow \Re s = \frac{1}{2} \vee \Im s = 0. \tag{2.2}$$

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:

Riemann hypothesis in LaTeX

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

In Mathematica, the Riemann hypothesis can be represented as:

Riemann hypothesis in Mathematica

```
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 greatly differ. LaTeX uses the unrestricted grammar of the TeX typesetting system. The entire set of commands can be redefined and extended at runtime, which means that TeX effectively allows its users to change every character used for the markup, including the \ character typically used to start commands. The large degree of freedom of the TeX grammar significantly complicates recognizing even the most basic tokens contained in mathematical formulae. In difference 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:

Representation tree of Riemann hypothesis in LaTeX

$$\left[\zeta_{\mathrm{l}}^{1}\ (_{\mathrm{l}}^{2}\ s_{\mathrm{l}}^{3}\ )_{\mathrm{l}}^{4} =_{\mathrm{l}}^{5} 0_{\mathrm{l}}^{6} \Rightarrow_{\mathrm{l}}^{7} \Re_{\mathrm{l}}^{8}\ s_{\mathrm{l}}^{9} =_{\mathrm{l}}^{10} \left[\begin{smallmatrix}11\\ \vdots\end{smallmatrix}\ 1_{\mathrm{l}}^{12}\ 2_{\mathrm{l}}^{13}\right] \vee_{\mathrm{l}}^{14} \Im_{\mathrm{l}}^{15}\ s_{\mathrm{l}}^{16} =_{\mathrm{l}}^{17} 0_{\mathrm{l}}^{18}\right].$$

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:

Representation tree of Riemann hypothesis in Mathematica

$$\left[\begin{smallmatrix}19\\ \Rightarrow\end{smallmatrix}\ \left[\begin{smallmatrix}20\\ =\end{smallmatrix}\ \left[\begin{smallmatrix}21\\ \zeta\end{smallmatrix}\ s_{\mathrm{l}}^{22}\right]\ 0_{\mathrm{n}}^{23}\right]\ \left[\begin{smallmatrix}24\\ \vee\end{smallmatrix}\ \left[\begin{smallmatrix}25\\ =\end{smallmatrix}\ \left[\begin{smallmatrix}26\\ \Re\end{smallmatrix}\ s_{\mathrm{l}}^{27}\right]\ \left[\begin{smallmatrix}28\\ \mathbb{Q}\end{smallmatrix}\ 1_{\mathrm{n}}^{29}\ 2_{\mathrm{n}}^{30}\right]\right]\ \left[\begin{smallmatrix}31\\ =\end{smallmatrix}\ \left[\begin{smallmatrix}32\\ \Im\end{smallmatrix}\ s_{\mathrm{l}}^{33}\right]\ 0_{\mathrm{n}}^{34}\right]\right].$$

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 almost 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 2.3.3.

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 provides all required information 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 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 of 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 [98, (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 [3]. For example, multiple scientific publications address the problem of accounting for branch cuts when entering expressions in CAS, such as [109] for Maple.

Our review of obstacles to the conversion of representation formats for mathematical formulae 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 2.3 represents excerpts of the MathML markup for our example of the Riemann hypothesis (2.2). In this excerpt, the PL token 7 corresponds to the CL token 19, PL token 5 corresponds to CL token 20, and so forth.

Riemann hypothesis in MathML

```
<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</csymbol>...
</annotation-xml></semantics></math>
```

Listing 2.3: MathML representation of the Riemann hypothesis (2.2) (excerpt).

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 [253].

Content information could furthermore 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 suggestion, and sanity checks, e.g., type and definiteness checking, similar to the functionality of word processors for natural language texts.

2.3.1.1 Related Work

A variety of tools exist to convert format representations of mathematical formulae. However, to our knowledge, Stamerjohanns et al. [351] presented the only study that evaluated the conversion quality of tools. Unfortunately, many of the tools evaluated by Stamerjohanns et al. are no longer available or out of date. Watt 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 [380]. Padovani discusses the roles of MathML and TEX elements for managing large repositories of mathematical knowledge [278]. Nghiem et al. used statistical machine translation to convert presentation to content language [271]. However, they do not consider the textual context of formulae. We will present detailed descriptions and evaluation results for specific conversion approaches in Section 2.3.3.

Youssef addressed the semantic enrichment of mathematical formulae in presentation language. They developed an automated tagger that parses LATEX formulae and annotates recognized tokens very similarly to Part-of-Speech (POS) taggers for natural language [402]. Their tagger currently uses a predefined, context-independent dictionary to identify and annotate formula components. Schubotz et al. proposed an approach to semantically enrich formulae by analyzing their textual context for the definitions of identifiers [329, 330].

With their 'math in the middle approach', Dehaye et al. 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 avoid computions and transfers between different systems. [94].

For an extensive review of format conversion and retrieval approaches for mathematical formulae, refer to [326, Chapter 2].

2.3.2 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. [329]. The dataset considers recent discussions of the International Mathematical Knowledge of Trust[30] working group, in particular the idea of a 'Semantic Capture Language' [165], 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 2.3.2.1, we present the test collection included in MathMLben. In Section 2.3.2.2, 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 2.3.2.3, we describe the similarity measures used to assess the markup quality.

2.3.2.1 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 [329]. 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 DLMF [98]. The DLMF website contains 9,897 labeled formulae created from semantic LaTeX source files [77, 78]. 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 formulae's markup[31]. 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 [22] and 30% from a Wikipedia dump.

[30] http://imkt.org/ [accessed 2021-08-03]
[31] http://dlmf.nist.gov/about/staff [accessed 2021-08-03]

All data is openly available for research purposes and can be obtained from: https://mathmlben.wmflabs.org[32].

2.3.2.2 Gold Standard

We provide explicit markup with universal, context-independent symbols in content MathML. Since the symbols from the default content dictionary of MathML[33] alone were insufficient to cover the range of semantics in our collection, we added the Wikidata content dictionary [328]. 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 stack-exchange 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 [336].

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 [331] to develop a visual support tool for creating and editing the *MathMLben* gold standard.

Table 2.3: Special content symbols added to LaTeXML for the creation of the gold standard.

No.	Rendering	Meaning	Example IDs
1	$[x, y]$	commutator	91
2	x_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 [135, 257]. 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 decision 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[34]. Table 2.3 lists the newly created macros. Hereafter, we explain our decisions and give examples of formulae in our dataset that were affected by the decisions.

[32] Visit https://mathmlben.wmflabs.org/about for a user guide [accessed 2021-08-03].
[33] http://www.openmath.org/cd [accessed 2021-08-03]
[34] http://dlmf.nist.gov/latexml/manual/customization/customization.latexml.html#SS1.SSS0.Px1 [accessed 2021-08-03]

- 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 2.3);
- 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 2.3);
- 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

$$E = mc^2, \tag{$\star$}$$

  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 2.3) 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 [Gold ID 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 [] (omitting the index `_\text{DB}`) and an anticommutator by defining new macros (see table 2.3). 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 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 [54].

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 and exaggerated. A borderline example in regard to this problem is Δx [GoldID 280]. Formulae of this form could be annotated as `\operatorname{}`, `\identifiername{}` or more generally as `\expressionname{}`. We interpret Δ as a difference applied to a variable, and render the expression as a function call.

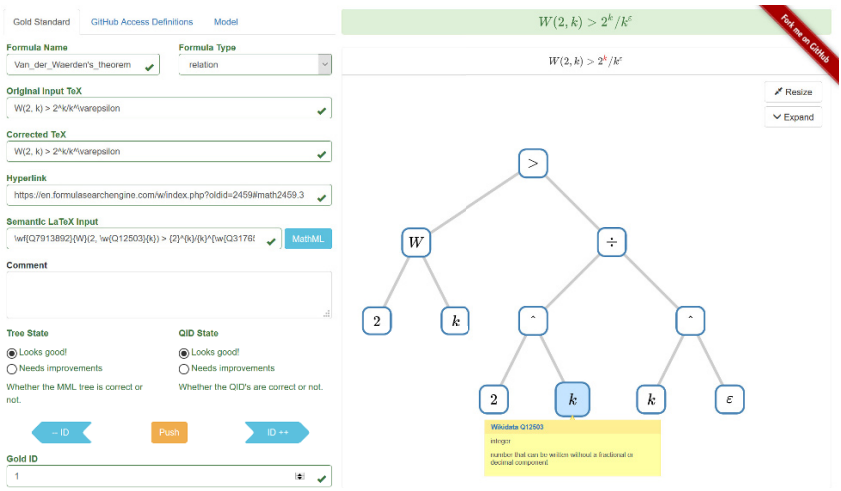


Figure 2.4: Graphical User Interface (GUI) to support the creation of our gold standard. The interface provides several TEX input fields (left) and a mathematical expression tree rendered by the VMEXT visualization tool (right).

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: `https://mathmlben.wmflabs.org/`. The GUI provides the following information for each Gold ID entry.

- **Formula Name:** the name of the formula (optional)
- **Formula Type:** either *definition*, *equation*, *relation* or *General Formula* (if none of the previous names fit)
- **Original Input TEX:** the LATEX expression extracted from the source
- **Corrected TEX:** the manually corrected LATEX expression
- **Hyperlink:** the hyperlink to the position of the formula in the source
- **Semantic LATEX Input:** the manually created semantic version of the corrected LATEX field. This entry is used to generate our MathML with Wikidata annotations.

- **Preview of Corrected LaTeX:** a preview of the corrected LaTeX input field rendered as an SVG image in real time using Mathoid [335], a service to generate SVGs and MathML from LaTeX input. It is shown in the top right corner of the GUI.
- **VMEXT Preview:** the VMEXT field renders the expression tree based on the content MathML. The symbol in each node is associated with the symbol in the cross referenced presentation MathML.

Figure 2.4 shows the GUI that allows to manually modify 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 [260] and our developed extensions. The following list contains some example 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 to interpret semantic macros, generates content MathML that can be controlled with macros, and is easily extensible 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 2.3, we created macros to semantically enrich identifiers, operators, and other mathematical concepts by linking them to their Wikidata items. As shown in Figure 2.4, the annotations are visualized using yellow info boxes appearing on mouse over. The boxes show the Wikidata QID, the name, and the description (if available) of the linked concept.

Aside from 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, `%\n`, 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 (also LaTeXML) try to include these formating improvements into the MathML markup. For our

gold standard, we focus on the pure semantic information and forgo formating 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.

2.3.2.3 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 [336], 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 [288]. 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 'costs' for performing these edits using the stock implementation of RTED are $c = 3i + r$. However, the actual difference is an equivalence, which we think should be assigned a cost of $e < 3i + r$. We set $e < r < i$.

2.3.3 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 (cf. Section 2.3.2). 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`.

2.3.3.1 Tool Selection

We compiled a list of available conversion tools from the W3C[35] wiki, from *GitHub*, and from questions about automated conversion of mathematical LATEX to MathML on *Stack Overflow*. We selected the following converters:

[35] https://www.w3.org/wiki/Math_Tools [accessed 2021-08-03]

- LaTeXML: can convert generic and semantically annotated LaTeX expressions to XML/HTML/MathML. The tool is written in Perl [257] and is actively maintained. LaTeXML was specifically developed to generate the DLMF web page and can therefore parse entire TeX documents. LaTeXML also supports conversions to content MathML.
- LaTeX2MathML: is a small python project and is able to generate presentation MathML from generic LaTeX expressions [245].
- Mathoid: is a service developed using Node.js, PhantomJS and MathJax (a javascript display engine for mathematics) to generate SVGs and MathML from LaTeX input. Mathoid is currently used to render mathematical formulae on Wikipedia [335].
- SnuggleTeX: is an open-source Java library developed at the University of Edinburgh [251]. The tool allows to convert simple LaTeX expression to XHTML and presentation MathML.
- MathToWeb: is an open-source Java-based web application that generates presentation MathML from LaTeX expressions[36].
- TeXZilla: is a javascript web application for LaTeX to MathML conversion capable of handling Unicode characters[37].
- Mathematical: is 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[38].
- CAS: we included Mathematica, which is capable of parsing LaTeX expressions.
- Part-of-Math (POM) Tagger: is a grammar-based LaTeX parser that tags recognized tokens with information from a dictionary [402]. The POM tagger is currently under development. In this paper, we use the first version. In [3], this version was used to provide translations LaTeX to the CAS Maple. In its current state, the 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.

2.3.3.2 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 format instead of the originally extracted string expressions, see 2.3.2.2.

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

2.3.3.3 Results

Figure 2.5 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

[36] https://www.mathtowebonline.com [accessed 2021-08-03]
[37] https://fred-wang.github.io/TeXZilla [accessed 2021-08-03]
[38] https://github.com/gjtorikian/mathematical [accessed 2021-08-03]

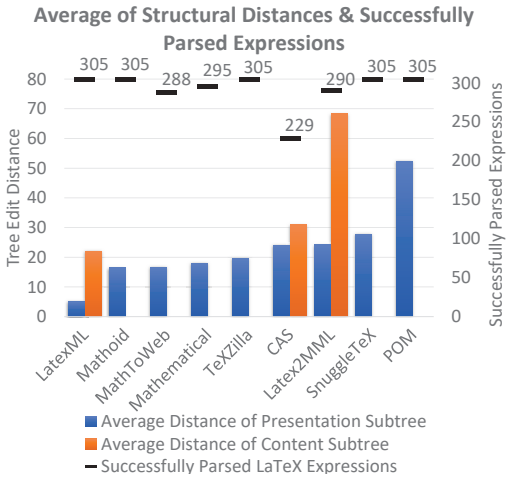


Figure 2.5: Overview of the structural tree edit distances (using $r = 0,\ i = d = 1$) between the MathML trees generated by the conversion tools and the gold standard MathML trees.

calculate the structural tree edit distances, we used the RTED [288] algorithm with costs of $i = 1$ for inserting, $d = 1$ for deleting and $r = 0$ for renaming nodes. Furthermore, the Figure shows the total number of successful transformations for the 305 expressions (black ticks). Note that we also 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, while the content trees are identical.

Figure 2.6 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 LaTeX2MathML, 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.

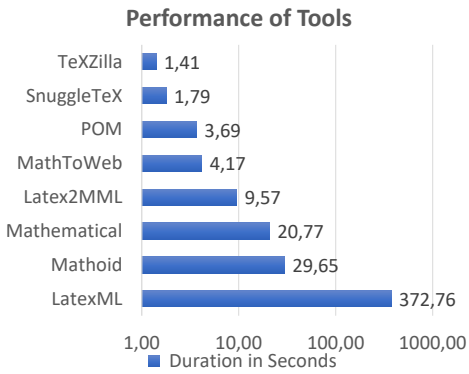


Figure 2.6: Time in seconds required by each tool to parse the 305 gold standard LaTeX expressions in logarithmic scale.

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.

2.3.4 Summary of MathML Converters

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 comparably low error rate, we chose to extend the LaTeXML output with semantic enhancements.

2.4 Mathematical Information Retrieval for LaTeX Translations

In the following, we will briefly discuss related work in the Mathematical Information Retrieval (MathIR) arena in order to find existing practical approaches for a translation from presentational to computable formats. MathIR is the research area that aims to retrieve additional (generally semantic) information about mathematical content [141]. In turn, the task of translating mathematical presentational formats to computable formats is part of this research area

since it requires a context-dependent semantification[39], i.e., the semantic enhancement or enrichment of mathematical objects with additional information. One of the most well-studied tasks in MathIR[40] is searching for relevant mathematical expressions or content [21, 22, 241, 346, 405, 408]. However, successful solutions in this area focus on similarity measures and do not necessarily require a deep understanding of the meaning and content of a formula. Likewise, other tasks in MathIR, such as entity linking, use similarity measures to retrieve connections between entities rather than semantic relatedness [208, 319, 321]. Thus, many related work in MathIR is not particularly beneficial for translating presentational encodings to computable formats. One of the reasons for this research gap is presumably a semantic version of the *chicken or the egg* causality dilemma. On the one hand, semantically enriching mathematical objects in an expression require identifying the meaningful objects. On the other hand, identifying those meaningful objects requires semantic information about those objects. In other words, if we want to annotate $P_n^{(\alpha,\beta)}(x)$ with *Jacobi polynomial* in our use case equation (1.1), we need to know that $P_n^{(\alpha,\beta)}(x)$ refers to the Jacobi polynomial.

Figure 2.7 illustrates this issue by splitting a math expression into four layers of mathematical objects. The identifier layer contains all identifiers (which may include general symbols and numbers too). The arithmetic layer contains arithmetic structures that combine tokens from the identifier layer to mathematical terms. This layer may include logic terms, sets, and other mathematical concepts with specific notations. The function layer combines elements from the lower layers to entire function calls. The top expression layer contains entire expressions in documents which are often a composition of elements in the previous layers. The difference of elements in the function and arithmetic layer is the ambiguity of the notations. Elements in the arithmetic layer generally do not need to be mapped to specific keywords in CAS because they are often semantically unique. In contrast, elements in the function layer are potentially ambiguous. However, a clear distinction between both layers is not always necessary and may even confuse in other MathIR related scenarios. For our task, the distinction is beneficial because elements in the function layer must be mapped to specific keywords in the CAS syntax, while elements in the arithmetic layer can be mostly ignored.

Existing MathIR tasks focus on semantically enhancing either the expression [208, 209, 215], arithmetic [93, 242, 339], or the identifier [121, 279, 329, 330, 339, 400] layer, missing the important function layer entirely. An algorithm needs to understand the involved functions to identify objects in the function layer. This dilemma is usually avoided in MathIR tasks since objects in the other layers can be extracted primarily context-independently. The meaning of arithmetic operators usually does not change (e.g., $+$, $-$, or $/$) and math identifiers can often be presumed to be Latin or Greek letters. The function layer, however, contains the most crucial objects for the translation task. Identifiers generally represent mutable objects, such as variables or parameters, and do not require specific mapping rules. Similarly, arithmetic operations are natively supported by most mathematical software. Finally, objects in the expression layers are often too abstract (because they are compositions of multiple objects) and cannot be mapped as a whole to a single logic procedure in a computable format.

There are approaches available that try to semantically enrich elements in the function layer. However, most of these semantic enrichment approaches focus solely on mathematical expressions themselves and do not analyze textual information [159, 259, 270, 339, 364, 374].

[39] Also often called *semantic enrichment.*

[40] For an extensive review of retrieval approaches for mathematical formulae, see also [326, Chapter 2].

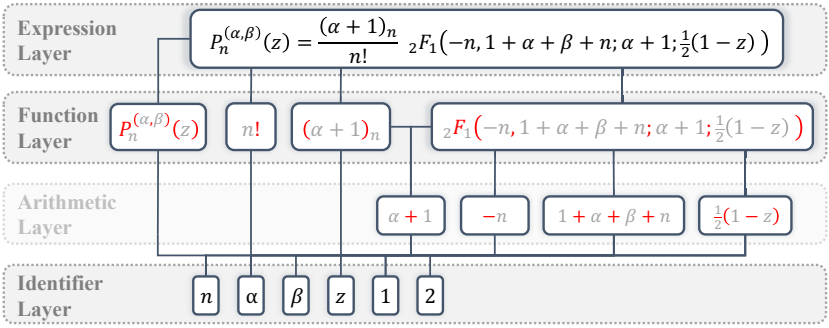


Figure 2.7: Four different layers of math objects in a single mathematical expression. The red highlights in the function and arithmetic layer refer to the fixed structure (or stem) of the function or operator. Gray tokens are mutable. Elements in the arithmetic layer are generally understood without further mappings and are mostly context-independent while elements in the function layer must be mapped to specific procedures in CAS and require disambiguation. However a strict distinction is not always required and might be even confusing. For example, $n!$ is mostly understood by CAS and context-independent but can (and sometimes should) be mapped to the specific factorial procedure making it more to an element of the function layer.

Approaches that take the textual context of a formula into account, on the other hand, do not semantically enrich objects in the function layer. Instead, they focus on other specific applications, including math embeddings with the goal of a semantic vector representation [121, 215, 360, 400, 404], entity linking [208, 212, 316, 321], math word problem solving [285, 409], semantic annotation [183, 214, 279, 329, 330], and context-aware math search engines [93, 122, 124, 145, 210, 211, 232, 273, 314, 315, 366]. Regarding translating mathematical expressions from a lower level of semantics to a higher level, relevant literature is limited. The main relevant related literature for our task include semantic tagging [71, 402], annotations [139, 183, 214, 279, 329, 330], and term disambiguations [339]. In the following, we distinguish semantic tagging (the task of precisely tagging math objects with a pre-defined set of semantic tags) and semantic annotation (the task of adding any number of relevant descriptions to math objects).

Semantic Tagging and Term Disambiguation Semantic tagging of mathematical tokens has rarely been studied in the past and has not reached a well-established reliability level yet. To the best of our knowledge, only Chien et al. [71] (2015) and Youssef [402] (2017) addressed the issue for semantic tokenization of math formulae. Youssef [402] created the POM tagger, which tags tokens in the LaTeX parse tree with additional information from a manually crafted lexicon. The POM tagger is still a work in progress and does not perform disambiguation steps yet. In the future, it is planned to reduce the number of possible tags for a token by analyzing the textual context and eliminating false tags. Ideally, the extracted context information results in a single, unique tag for each token. However, no update of the POM tagger, including the disambiguation steps, has been published so far. Recently, however, Shan and Youssef [339] presented several machine learning approaches as the first step towards disambiguation of mathematical terms. They trained different models on the semantic DLMF dataset and successfully disambiguated

prime notations with an $F1$ score of 0.83. However, if the models only adapted the relatively strict DLMF notation style for primes or if they are also able to disambiguate other real-world data has not been discussed.

Chien et al. [71] proposed a probabilistic model on entire document collections to conclude semantic tags of mathematical tokens. They focused on tagging single identifiers (i.e., no groups of tokens). They constituted that the *consistency property* and *user habits* are critical aspects for successful tag disambiguation. With *user habits*, the authors referred to the different education levels and expertise of users so that a model can predict the preferred notation for specific semantics. The *consistency property* refers to the assumption that the meaning of a single term does not change within a certain context, e.g., a document. Recent efforts on annotating mathematical symbols by Asakura et al. [1], however, indicate that the scope of consistent tags could be significantly smaller than an entire document or a document collection. The semantics of frequently used symbols, such as x or t, may even change within single paragraphs. Another interesting counterexample is the connection between Euler numbers and Euler polynomials [98, (24.2.9)] in

$$E_n = 2^n \, E_n\left(\frac{1}{2}\right). \tag{2.3}$$

While clearly connected, the first E refers to the Euler number but the second E refers to Euler polynomials. This underlines that under special circumstances, even within the scope of a single equation, an identifier may refer to two different mathematical concepts. Chien et al. reported a maximum accuracy of 0.94.

Semantic Annotation Task While the task of semantic annotation has been studied more comprehensively, none of these existing approaches tried to convert the source expressions into a computable format [139, 183, 214, 279, 329, 330]. Grigore et al. [139], Nghiem et al. [269], Pagel et al. [279], Schubotz et al. [329, 330], and Kristianto et al. [214] analyze nouns or noun phrases in the surrounding context of a formula to semantically annotate an entire expression or parts of an expression. Only Grigore et al. [139] tried to use this information to perform a translation to a semantically enhanced format, here content MathML. The authors deduced a CD entry for a math symbol by calculating the similarity of the nouns surrounding the symbol and the textual description (or more precisely: the cluster of nouns in that description) of the CD entry. They measured the similarity with distributional properties from WordNet [261]. The other approaches either use the gained semantic information to improve search engines [214, 269] or enable entity linking [279, 329, 330]. While other semantification approaches exist that elevate source presentational formats to a semantically enriched format [245, 251, 257, 270, 271, 364, 391], none of them take the textual context into account. Some of them, however, perform disambiguation steps by considering other mathematical expressions in the same document (again presuming a semantic consistency of math notation within a single document as proposed by Chien et al. [71]) [270, 271]. None of the previous work considered the possibility of an identifier that has multiple meanings within a single formula, as shown in equation (2.3).

Summary In summary, semantic enriching approaches avoid the essential function layer [159, 259, 270, 364, 374], ignore the textual context surrounding a formula [71, 245, 251, 257, 270, 271, 296, 364, 391], or does not use the extracted information for a translation towards a semantic enhanced format [183, 214, 279, 329, 330, 402]. Nonetheless, the related work underlines the benefits of analyzing the textual context of a formula. More importantly, the research has

shown that even simple noun phrase extraction provide viable information for numerous of applications [139, 183, 214, 279, 329, 330]. This motivated us to apply these promising approaches for our semantification pipeline too.

Regarding the final translations towards computable formats, our comprehensive analysis of LaTeX to MathML conversion tools in the previous section revealed that we probably gain no benefits from translating LaTeX to MathML in an intermediate step. While many CAS provide import functions for MathML, there is no substantial support for OpenMath CDs. Another option would be OpenMath, since the SCSCP protocol uses OpenMath for inter-CAS communications. However, the SCSCP is relatively complex for our task and difficult to extend for new CAS if we do not have access to the internal libraries. Additionally, there are no translation tools from LaTeX to OpenMath even though LaTeXML can be exploited to realize rule-based translations.

In a previous research project, we developed LaCaSt, a semantic LaTeX to CAS translator, specifically for the DLMF [3, 13]. The goal of LaCaSt was to translate DLMF formulae, given in semantic LaTeX, to the CAS Maple. The semantic LaTeX macros reduced the ambiguity in mathematical expressions and enabled LaCaSt to focus on other translation issues, such as definition disparity between the DLMF and Maple. Hence, we already established a reliable and expandable translation pipeline from semantic LaTeX to Maple. As a consequence, we focus our efforts on the more promising semantification of LaTeX to semantic LaTeX rather than from LaTeX to content MathML in this thesis[41].

[41] Since the original development of LaCaSt was part of my Master's thesis, the content of the associated early publications [3, 13] is not reused in this thesis. For more details about LaCaSt, see [13].

You must strive to find your own voice because the longer you wait to begin, the less likely you are going to find it at all.

John Keating - *Dead Poet Society*

CHAPTER 3

Semantification of Mathematical LaTeX

Contents

In this chapter, we will focus on the research task **II**, i.e., we develop a new semantification process that addresses the issues of existing approaches outlined in the previous chapter. We identified two main issues with existing MathIR approaches for disambiguation and semantification of LaTeX expressions. First, many semantification approaches solely focus on single tokens, such as identifiers, or the entire mathematical expression but miss to enrich the essential subexpressions between both extremes semantically. Second, existing translation approaches lack context sensitivity and disambiguate expressions by following an internal (often hidden)

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_3

context-agnostic decision process. This chapter addresses these issues within three parts. First, we elaborate on the capabilities of word embedding techniques to semantically enrich mathematical expressions. Second, we study the frequency distribution of mathematical subexpressions in scientific corpora to understand the variety and complexity of subexpressions better. Third, we briefly outline a context-sensitive translation pipeline based on the gained knowledge from the first two parts.

The primary goal of this chapter is to develop a context-sensitive LaTeX to CAS translation pipeline. Unfortunately, it is not clear where we can find sufficient semantic information in the context to perform reliable translations. We can expect a certain amount of inclusive information in the given expression itself [54, 71, 394]. Additionally, related work has proven that noun phrases in the nearby textual context (such as the leading or following sentences of a formula) can successfully disambiguate math formulae [139, 209, 213, 329]. However, many functions are not necessarily declared in the surrounding context because the author presumes the interpretation is unambiguous. Wolska and Grigore [394] have shown that only around 70% of mathematical identifiers are explicitly declared in the surrounding context. In this case, the location of the information that disambiguates the expression may vary greatly depending on many factors, such as the expected education level of the target audience of the article, the given references in the document, or even the author's preferred notation style. One possible solution for exploiting this source of semantic information is to build a common knowledge database for mathematical expressions.

As a first attempt to automatically build such a common knowledge database that stores the standard, i.e., most common, meanings of mathematical symbols, we explore the capabilities of machine learning algorithms in the first part of this chapter. Specifically, we use word embeddings to train common co-occurrences of mathematical and natural language tokens. We will show that this approach is not as successful as we hoped for our knowledge extraction task but enables new approaches for mathematical search engines. Further, the results will once again underline the issues with the interpretation of nested mathematical objects. Word embeddings for mathematical tokens are mainly unable to properly train the connections with defining expressions in the context because they still ignore the function layer of mathematical expressions. In the following, we focused our studies on mathematical subexpressions.

As a thought experiment, consider mathematical expressions are like entire sentences in natural languages rather than single words. Following this analogy, entire math terms are analog to words, and the notation of mathematical expressions certainly follow a specific grammar [54]. However, our *mathematical sentences* have one distinct difference compared to natural language sentences. The grammar of mathematical expressions is built around a nested structure in contrast to the sequential order of words. For example, a math term representing a variable is a placeholder and can be replaced with arbitrarily complex and deeply nested subexpressions without violating any grammatical rules. This nested structure makes the semantic tokenization of mathematical expressions to a complex and eventually context-dependent task [71, 402]. In order to review our analogy, we perform the most extensive notation analysis of mathematical subexpressions (since those are the potential words) on two real-world scientific datasets. We discovered that the frequency distributions of mathematical objects obey Zipf's law, similar to words in natural language corpora. In turn, we can use frequency-based retrieval functions to distinguish important or informative mathematical objects from stop-word-like structures. We coin these essential and informative objects Mathematical Objects of Interest (MOI). The

success of this new interpretation finally motivated us to move away from the established MathIR techniques that focus on single identifiers or entire math expressions to meaningful subexpressions. Hence, we conclude this chapter with an abstract context-sensitive translation approach that finally attributes to the nested grammar of mathematical formulae and is based on the new concept of MOI.

In summary, this chapter is organized as follows. In Section 3.1, we explore the capabilities of word embeddings to discover common co-occurrences of natural language tokens and math tokens in large scientific datasets. In Section 3.2, we introduce the new concept of MOI and perform the first extensive frequency distribution study of mathematical notations in two large scientific corpora. Section 3.3 concludes the findings of the previous sections by introducing a novel context-sensitive translation approach from LaTeX to CAS expressions. Section 3.1 was published as an article in the Scientometrics journal [15]. Section 3.2 was published as full paper at the WWW conference [14]. Excerpts of Section 3.3 have been published at the ICMS conference in a full paper [10].

3.1 Semantification via Math-Word Embeddings

Mathematics is capable of explaining complicated concepts and relations in a compact, precise, and accurate way. Learning this idiom takes time and is often difficult, even to humans. The general applicability of mathematics allows a certain level of ambiguity in its expressions. Short explanations or mathematical expressions are often used to mitigate the ambiguity problem, that serve as a context to the reader. Along with context-dependency, inherent issues of linguistics (e.g., ambiguity, non-formality) make it even more challenging for computers to understand mathematical expressions. Nevertheless, a system capable of automatically capturing the semantics of mathematical expressions would be suitable for improving several applications, from search engines to recommendation systems. Word embedding [33, 34, 43, 65, 73, 217, 222, 239, 250, 255, 272, 293, 295] has made it possible to apply deep learning in NLP with great effect. That is because embedding represents individual words with numerical vectors that capture contextual and relational semantics of the words. Such representation enables inputting words and sentences to a Neural Network (NN) in numerical form. This allows the training of NNs and using them as predictive models for various NLP tasks and applications, such as semantic role modeling [149, 412], word-sense disambiguation [160, 305], sentence classification [186], sentiment analysis [344], coreference resolution [223, 388], named entity recognition [72], reading comprehension [75], question answering [234], natural language inference [69, 137], and machine translation [97]. The performance of word embedding in NLP tasks has been measured and shown to deliver fairly high accuracy [256, 293, 295].

As math text consists of natural text as well as math expressions that exhibit linear and contextual correlation characteristics that are very similar to those of natural sentences, word embedding applies to math text much as it does to natural text. Accordingly, it is worthwhile to explore the use and effectiveness of word embedding in Mathematical Language Processing (MLP), Mathematical Knowledge Management (MKM), and MathIR. Still, math expressions and math writing styles are different from natural text to the point that NLP techniques have to undergo significant adaptations and modifications to work well in math contexts.

While some efforts have started to apply word embedding to MLP, such as equation embedding [121, 9, 215, 400, 404], there is a healthy skepticism about the use of ML and Deep Learning

(DL) in MLP and MKM, on the basis that much work is still required to prove the effectiveness of DL in MLP. To learn how to adapt and apply DL in the MLP/MKM/MathIR context is not an easy task. Most applications of DL in MLP/MKM/MathIR rest on the effectiveness of word/math-term embedding (henceforth *math embedding*) because the latter is the most basic foundation in language DL. Therefore, it behooves us to start to look at the effectiveness of math embedding in basic tasks, such as term similarity, analogy, information retrieval, and basic math search, to learn more about their extension and limitations. More importantly, we need to learn how to refine and evolve math embedding to become accurate enough for more severe applications, such as knowledge extraction. That is the primary objective of this section.

To that effect, there is a fundamental need for datasets and benchmarks, preferably standard ones, to allow researchers to measure the performance of various math embedding techniques, and applications based on them, in an objective and statistically significant way, and to measure improvements and comparative progress. Such resources are abundant in the natural language domain but scarce in the MLP domain. Developing some of such datasets and benchmarks will hopefully form the nucleus for further development by the community to facilitate research and speed up progress in this vital area of research.

While the task of creating such resources for DL applications in MLP can be long and demanding, the examination of math embedding should not wait but should proceed right away, even if in an exploratory manner. Early evaluations of math embedding should ascertain its value for MLP/MKM/MathIR and inform the process and trajectory of creating the corpora and benchmarks. Admittedly, until adequate datasets and benchmarks become available for MLP, we have to resort to less systematic performance evaluation and rely on performing preliminary tests on the limited resources available. The DLMF [98] and arXiv.org preprint archive[1] are good resources to start our exploratory embedding efforts. The DLMF offers high quality, and the authors are familiar with its structure and content (which aids in crafting some of the tests). As for the arXiv collection, its large volume of mostly math articles makes it an option worth to investigate as well.

In this section, we provide an exploratory investigation of the effectiveness and use of word embedding in MLP and MKM through different perspectives. First, we train word2vec models on the DLMF and arXiv with slightly different approaches for embedding math. Since the DLMF is primarily a handbook of mathematical equations, it does not provide extensive textual content. We will show that the DLMF trained model is appropriate to discover mathematical term similarities and term analogies, and to generate query expansions. We hypothesize that the arXiv trained models are beneficial to extract definiens, i.e., textual descriptive phrases for math terms. We examine the possible reasons why the word embedding models, trained on the arXiv dataset, does not present valuable results for this task. Besides, we discuss some of the reasons that we believe thwart the progress in MathIR in the direction of machine learning. In summary, we focus on five tasks (i) term similarity, (ii) math analogies, (iii) concept modeling, (iv) query expansion, and (v) knowledge extraction. In the context of this thesis, we are mostly interested in the latter, i.e., knowledge extractions, and will solely focus on these experiments and results. For the tasks (i-iv), see [15].

[1] https://arxiv.org/ [accessed 2019-09-01]

3.1.1 Foundations and Related Work

Understanding mathematical expressions essentially mean comprehending the semantic value of its internal components, which can be accomplished by linking its elements with their corresponding mathematical definitions. Current MathIR approaches [213, 329, 330] try to extract textual descriptors of the parts that compose mathematical equations. Intuitively, there are questions that arise from this scenario, such as (i) how to determine the parts which have their own descriptors, and (ii) how to identify correct descriptors over others.

Answers to (i) are more concerned in choosing the correct definitions for which parts of a mathematical expression are considered as one mathematical object [197, 18, 402]. Current definition-languages, such as the content MathML 3.0[2] specification, are often imprecise[3]. For example, content MathML 3.0 uses 'csymbol' elements for functions and specifies them as expressions that *refer to a specific, mathematically-defined concept with an external definition*[4]. However, in case of the Van der Waerden number, for instance, it is not clear whether W or the sequence $W(r, k)$ should be declared as a 'csymbol'. Another example involves content identifiers, which MathML specifies as *mathematical variables that have properties, but no fixed value*[5]. While content identifiers are allowed to have complex rendered structures (e.g., β_i^2), it is not permitted to enclose identifiers within other identifiers. Let us consider α_i, where α is a vector and α_i its i-th element. In this case, α_i should be considered as a composition of three content identifiers, each one carrying its own individualized semantic information, namely the vector α, the element α_i of the vector, and the index i. However, with the current specification, the definition of these identifiers would not be canonical. One possible workaround to represent such expressions with content MathML is to use a structure of four nodes, interpreting α_i as a function via a 'csymbol' (one parent 'apply' node with the three children *vector-selector*, α, and i). However, ML algorithms and MathIR approaches would benefit from more precise definitions and a unified answer for (i). Most of the related work relies on these relatively vague definitions and in the analysis of content identifiers, focusing their efforts on (ii).

Questions (i), (ii), and other pragmatic issues are already in discussion in a bigger context, as data production continues to rise and digital repositories seem to be the future for any archive structure. A prominent example is the National Research Council's effort to establish what they call the Digital Mathematical Library (DML)[6], a project under the International Mathematical Union. The goal of this project is to take advantage of new technologies and help to solve the inability to search, relate, and aggregate information about mathematical expressions in documents over the web.

The advances most relevant to our work are the recent developments in *word embedding* [43, 65, 73, 256, 293, 295, 313]. Word embedding takes as input a text collection and generates a numerical feature vector (typically with 100 or 300 dimensions) for each word in the collection. This vector captures latent semantics of a word from the contexts of its occurrences in the

[2] https://www.w3.org/TR/MathML3/ [accessed 2019-09-01]

[3] Note that OpenMath is another specification designed to encode semantics of mathematics. However, content MathML is an encoding of OpenMath and inherent problems of content MathML also apply to OpenMath (see https://www.openmath.org/om-mml/ [accessed 2019-09-01]).

[4] https://www.w3.org/TR/\gls{mathml}3/chapter4.html#contm.csymbol [accessed 2019-09-01]

[5] https://www.w3.org/TR/\gls{mathml}3/chapter4.html#contm.ci [accessed 2019-09-01]

[6] https://www.nap.edu/read/18619 [accessed 2019-09-01]

collection; in particular, words that often co-occur nearby tend to have similar feature vectors (where similarity is measured by the cosine similarity, the Euclidean distance, etc.).

Recently, more and more projects try to adapt these word embedding techniques to learn patterns of the correlations between context and mathematics. In the work of Gao et al. [121], they embed single symbols and train a model that can discover similarities between mathematical symbols. Similarly to this approach, Krstovski and Blei [215] uses a variation of word embedding to represent complex mathematical expressions as single unit tokens for IR. In 2019, Yusanaga and Lafferty [400] explore an embedding technique based on recurrent neural networks to improve topic models by considering mathematical expressions. They state their approach outperforms topic models that do not consider mathematics in text and report a topic coherence improvement of 0.012 over the LDA[7] baseline. Equation embedding, as in [121, 215, 400], present promising results for identifying similar equations and contextual descriptive keywords. In the following, we will explore in more detail different techniques of word embedding.

3.1.1.1 Word Embedding

In this section, we apply *word2vec* [256] on the DLMF [98] and on the collection of arXiv documents for generating embedding vectors for various math symbols and terms. The word2vec technique computes real-valued vectors for words in a document using two main approaches: skip-gram and continuous bag-of-words (CBOW). Both produce a fixed-length n-dimensional vector representation for each word in a corpus. In the skip-gram training model, one tries to predict the context of a given the word, while CBOW predicts a target word given its context. In word2vec, context is defined as the adjacent neighboring words in a defined range, called a sliding window. The main idea is that the numerical vectors representing similar words should have close values if the words have similar context, often illustrated by the *king-queen relationship*.

King-Queen Relationship of Word-Embedding Vectors

The king-queen relationship describes the similarity (in terms of the cosine distance between the vectors) of:

$$\vec{v}_{\text{king}} - \vec{v}_{\text{man}} \approx \vec{v}_{\text{queen}} - \vec{v}_{\text{woman}}, \tag{3.1}$$

where $\vec{v}_t$ represents the vector for the token t.

Extending word2vec's approaches, Le and Mikolov [222] propose *Paragraph Vectors*, a framework that learns continuous distributed vector representations for any size of text segments (e.g., sentences, paragraphs, documents). This technique alleviates the inability of word2vec to embed documents as one single entity. This technique also comes in two distinct variations: *Distributed Memory* and *Distributed Bag-of-Words*, which are analogous to the skip-gram and CBOW training models, respectively.

Other approaches also produce word embedding given a training corpus as input, such as fastText [43], ELMo [295], and GloVe [293]. The choice for word2vec for our experiments is justified because of its implementation ease, training speed using modest computing resources,

[7] Latent Dirichlet Allocation

general applicability, and robustness in several NLP tasks [160, 161, 229, 238, 302, 312]. Additionally, in fastText they propose to learn word representations as a sum of the n-grams of its constituent characters (sub-words). The sub-word structure would incorporate a certain noise[8] to our experiments. In ELMo, they compute their word vectors as the average of their characters representations, which are obtained through a two-layer bidirectional language model (biLM). This would bring even more granularity than fastText, as they consider each character in a word as having their own n-dimensional vector representation. Another factor that prevents us from using ELMo, for now, is its expensive training process[9]. Closer to the word2vec technique, GloVe [293] is also considered, but its co-occurrence matrix would escalate the memory usage, making its training for arXiv not possible at the moment. We also examine the recently published Universal Sentence Encoder [65] from Google, but their implementation does not allow one to use a new training corpus, only to access its pre-calculated vectors based on words. We also considered BERT [96] with its recent advances of Transformer-based architectures in NLP as an alternative to *word2vec*. However, incorporating BERT and other Transformer-based architectures would require a significant restructuring of the core idea of our work. BERT is pre-trained in two general tasks that are not directly transferable to mathematics embeddings: *Masked Language Modelling* and *Next Sentence Prediction*. Since this work is an exploratory investigation of the potential of word embedding techniques in MLP and MKM, we gave preference to tools that could be applied directly. Nonetheless, since some of our results are promising, we plan to include Transformer-based systems, such as BERT [96], XLNet [399], RoBERTa [235], and Transformers-XL [87], in future work.

The overall performance of word embedding algorithms has shown superior results in many different NLP tasks, such as machine translation [256], relation similarity [161], word sense disambiguation [55], word similarity [268, 312], and topic categorization [301]. In the same direction, we also explore how well mathematical tokens can be embedded according to their semantic information. However, mathematical formulae are highly ambiguous and, if not properly processed, their representation is jeopardized.

To investigate the situations described in Sections 3.1.1.1 and 2.2.5 we applied word2vec on two different scenarios, one focusing on MathIR (DLMF) and the other on semantic knowledge extraction (arXiv), i.e., identifying definiens for math objects. To summarize our decisions, for the DLMF and arXiv, we choose the stream of token embedding technique, i.e., each inner token is represented as a single n-dimensional vector in the embedding model. For the DLMF, we embed all inner tokens, while for the arXiv, we only embed the identifiers. In this thesis, we are more interested in applying math embeddings to semantic extraction task. The MathIR task is described in [15, Section 3].

3.1.2 Semantic Knowledge Extraction

Extracting definiens of mathematical objects from a textual context is a common task in MathIR [214, 279, 329, 330, 405] that often provides a gold dataset for its evaluation. Since the DLMF does not provide extensive textual information for its mathematical expressions, we considered an alternative scenario in our analysis, one in which we trained a second word2vec model on a much larger corpus composed of articles/papers from the arXiv collection. In this section, we compare our findings against the approach by Schubotz et al. [330]. We apply varia-

[8] Noise means, the data consists of many uninteresting tokens that affect the trained model negatively.
[9] https://github.com/allenai/bilm-tf [accessed 2019-09-01]

tions of a word2vec [256] and paragraph vectors [222] implementation to extract mathematical relations from the arXMLiv 2018 [132] dataset (i.e., an HTML collection of the arXiv.org preprint archive[10]), which is used as our training corpus. We also consider the subsets that do not report errors during the document conversion (i.e., *no_problem* and *warning*) which represent 70% of archive.org. We make the code, regarding our experiments, publicly available[11].

3.1.2.1 Evaluation of Math-Embedding-Based Knowledge Extraction

As a pre-processing step, we represent mathematical expressions using the MathML[12] notation. First, we replace all mathematical expressions with the identifiers sequence it contains, i.e., $W(2,k)$ is replaced by '$W\ k$'. We also add the prefix 'math-' to all identifier tokens to distinguish between textual and mathematical terms later. Second, we remove all common English stopwords from the training corpus. Finally, we train a word2vec model (skip-gram) using the following hyperparameters[13]: vector size of 300 dimensions, a window size of 15, minimum word count of 10, and a negative sampling of $1E-5$. We justify the hyperparameter used in our experiments based on previous publications using similar models [63, 221, 222, 255, 312].

In the following, distances between vectors are calculated via the cosine distance. The trained model was able to partially incorporate semantics of mathematical identifiers. For instance, the closest 27 vectors to the mathematical identifier f are mathematical identifiers themselves and the fourth closest noun vector to f is '*function*'. We observe that the results of the model trained on arXiv are comparable with our previous experiments on the DLMF.

Previously, we used the semantic relations between embedding vectors to search for relevant terms in the model. Hereafter, we will refer to this algebraic property as *semantic distance* to a given term with respect to a given relation, i.e., two related vectors. For example, to answer the query/question: What is to 'complex' as x is to 'real', one has to find the closest *semantic vectors* to 'complex' with respect to the relation between x and 'real', i.e., finding $\vec{v}$ in

$$\vec{v} - \vec{v}_{\text{complex}} \approx \vec{v}_x - \vec{v}_{\text{real}}.$$

Instead of asking for mathematical expressions, we will now reword the query to ask for specific words. For example, to retrieve the meaning of f from the model, we can ask for: What is to f as 'variable' is to x? Or in other words, what is semantically close to f with respect to the relation between 'variable' and x? Table 3.1 shows the top 10 semantically closest results to f with respect to the relations between $\vec{v}_{\text{variable}}$ and $\vec{v}_x$, $\vec{v}_{\text{variable}}$ and $\vec{v}_y$, and $\vec{v}_{\text{variable}}$ and $\vec{v}_a$.

From Table 3.1, we can observe a similar behaviour. Later, we will explore that mathematical vectors build a cluster in the trained model, i.e., that the vectors of $\vec{v}_f$, $\vec{v}_x$, and $\vec{v}_y$ are close to each other with respect to the cosine similarity. This cluster, and the fact that we did not use stemming and lemmatization for preprocessing, explains that the top hit to the queries is always 'variables'. To refine the order of the extracted answers, we calculated the cosine similarity between $\vec{v}_f$ and the vectors for the extracted words directly. Table 3.2 shows the cosine distances between $\vec{v}_f$ and the extracted words from the query: *Term* is to f what 'variable' is to a.

[10] https://arxiv.org/ [accessed 2019-09-01]
[11] https://github.com/ag-gipp/math2vec [accessed 2019-09-01]
[12] The source TEX file has to use mathematical environments for its expressions.
[13] Non mentioned hyperparameters are used with their default values as described in the Gensim API [307]

Table 3.1: Analogies of the form: Find the *Term* where *Term* is a word that is to X what Y is to Z.

Top-10 best *Terms* and their cosine similarities where					
***Term* is to f what 'variable' is to x**		***Term* is to f what 'variable' is to y**		***Term* is to f what 'variable' is to a**	
variables	0.7655	variables	0.7481	variables	0.7600
independent	0.7411	**function**	0.7249	**function**	0.7154
appropriate	0.7279	given	0.7103	appropriate	0.6925
means	0.7250	means	0.7083	independent	0.6789
ie	0.7234	ie	0.7067	instead	0.6784
instead	0.7233	independent	0.7030	defined	0.6729
namely	0.7139	thus	0.6925	namely	0.6719
function	0.7131	instead	0.6922	continuous	0.6707
following	0.7117	appropriate	0.6891	depends	0.6629
depends	0.7095	defined	0.6889	represents	0.6623

Asking for the meaning of f is a very generic question. Thus, we performed a detailed evaluation on the first 100 entries[14] of the MathMLben benchmark [18]. We evaluated the average of the *semantic distances* with respect to the relations between $\vec{v}_{\text{variable}}$ and $\vec{v}_{\text{x}}$, $\vec{v}_{\text{variable}}$ and $\vec{v}_{\text{a}}$, and $\vec{v}_{\text{function}}$ and $\vec{v}_{\text{f}}$. We have chosen to test on these relations because we believe that these relations are the most general and still applicable, e.g., seen in Table 3.2. In addition, we consider only results with a cosine similarity equal to or greater than 0.70 to maintain a minimum quality in our experiments. The overall results were poor, with a precision of $p = .0023$ and a recall of $r = .052$. Despite the weak results, an investigation of some specific examples showed interesting characteristics; for example, for the identifier W, the four semantically closest results were *functions*, *variables*, *form*, and the mathematical identifier q. The poor performance illustrates that there might be underlying issues with our approach. However, as mentioned before, mathematical notation is highly flexible and content-dependent. Hence, in the next section, we explore a technique that rearranges the hits according to the actual close context of the mathematical expression.

3.1.2.2 Improvement by Considering the Context

We also investigate how a different word embedding technique would affect our experiments. To do so, we trained a Distributed Bag-of-Words of Paragraph Vectors (DBOW-PV) [222] model. We trained this DBOW-PV in the same corpus as our word2vec model (with the same preprocessing steps) with the following configuration: 400 dimensions, a window size of 25, and minimum count of 10 words. Schubotz et al. [330] analyze all occurrences of mathematical identifiers and consider the entire article at once. We believe this prevents the algorithm from finding the right descriptor in the text, since later or prior occurrences of an identifier might appear in a different context, and potentially introduce different meanings. Instead of using the entire document, we consider the algorithm by Schubotz et al. [330] only in the input paragraph and

[14] Same entries used in [330]

Table 3.2: The cosine distances of f regarding to the hits in Table 3.1.

Terms from Table 3.1 to f	Cosine distances between the
function	0.8138
defined	0.7932
independent	0.7323
namely	0.7214
depends	0.7022
represents	0.6983
instead	0.6837
appropriate	0.6698
continuous	0.6203
variables	0.5638

similar paragraphs given by our DBOW-PV model. Unfortunately, the obtained variance within the paragraphs brings a high number of false positives to the list of candidates, which affects the performance of the original approach negatively.

As a second approach for improving our system, we considered a given textual context to reorder extracted words according to their cosine similarities to the given context. For example, consider the sentence: 'Let $f(x, y)$ be a continuous function where x and y are arbitrary values.'. We ask for the meaning of f concerning this given context sentence. The top-k closest words to f in the word2vec model only represent the distance over the entire corpus, in this case, arXiv, but not regarding a given context. To address this issue, we retrieved similar paragraphs to our context example via the DBOW-PV model and computed the weighted average distance between all top-k words, that are similar to f and the retrieved sentences. We expected that the word describing f in our example sentence would also present a higher cosine similarity to the context itself. Table 3.3 shows the top-10 closest words (i.e., we filtered out other math tokens) and their cosine similarity to f in the left column. The right column shows the average cosine similarities of the extracted words to the context example sentence we used and its retrieved similar sentences.

As Table 3.3 illustrates, this context-sensitive approach was not beneficial; in fact it undermined our model. According to the fact that the identifier should be closer to the given context sentence rather than to the related sentences retrieved from the DBOW-PV model, we also explored the use of weighted average. However, the weighted average did not improve the results of the normal average. Other hyperparameters for the word embedding models were also tested in an attempt to tune our system. However, we could not determine any drastic changes regarding the measured performances.

3.1.2.3 Visualizing Our Model

Figure 3.1 illustrates four t-SNE[154] plots of our word2vec model. Since t-SNE plots may produce misleading structures [382], we plot four t-SNE plots with different perplexity values.

Table 3.3: We are looking for descriptive terms for f in a given context '*Let* $f(x, y)$ *be a continuous function where* x *and* y *are arbitrary values*'. To achieve this, we retrieved close vectors to f and computed their distances to the given context sentence. To bring variety to the context, we used our DBOW-PV model to retrieve related sentences to the given context and computed the average distance of the words to these related sentences.

Top-10 closes words (no math symbols) to f and their cosine similarities.		**After reordering the hits according to their distances to the context vector.**	
given	0.8162	case	0.8568
case	0.7960	corresponding	0.8562
corresponding	0.7957	note	0.8451
function	0.7900	thus	0.8414
note	0.7803	obtain	0.8413
thus	0.7726	ie	0.8335
obtain	0.7712	since	0.8250
value	0.7682	**function**	0.8086
ie	0.7656	value	0.8015
since	0.7583	given	0.7096

Other parameters were set to their default values according to the t-SNE python package. We colored word tokens in blue and math tokens in red. The plots illustrate, though not surprisingly, that math tokens are clustered together. However, a certain subset of math tokens appear isolated from other math tokens. By attaching the content to some of the vectors, we can see that math tokens, such as *and* (an *and* within math mode) and *im* (most likely referring to imaginary numbers) are part of a second cluster of math tokens. The plot is similar to the visualized model presented in [121], even though they use a different word embedding technique. Hence, the general structure within math word2vec models seems to be insensitive to the embedding technique of formulae used. Compared to [121], we provide a model with richer details that reveal some dense clusters, e.g., numbers (bottom right plot at $(11, 8)$) or equation labels (bottom right plot at $(-14, 0)$).

Based on the presented results, one can still argue that more settings should be explored (e.g., different parameters and embedding techniques) for the embedding phase, different pre-processing steps (e.g., stemming and lemmatization) should be adopted, and post-processing techniques (e.g., boosting terms of interest based on a knowledge database such as OntoMathPro [104, 105]) should also be investigated. This presumably solves some minor problems, such as removing the inaccurate first hit in Table 3.1. Nevertheless, the overall results would not surpass the ones in [330], which reports a precision score of $p = 0.48$. On the grounds that mathematics is highly customizable, many of the defined relations between mathematical concepts and their descriptors are only valid in a local scope. Let us consider an author that notates his algorithm using the symbol π. The author's specific use of π does not change its general use, but it affects the meaning in the scope of the article. Current ML approaches only learn patterns of most frequently used combinations, e.g., between f and 'function', as seen in Table 3.1.

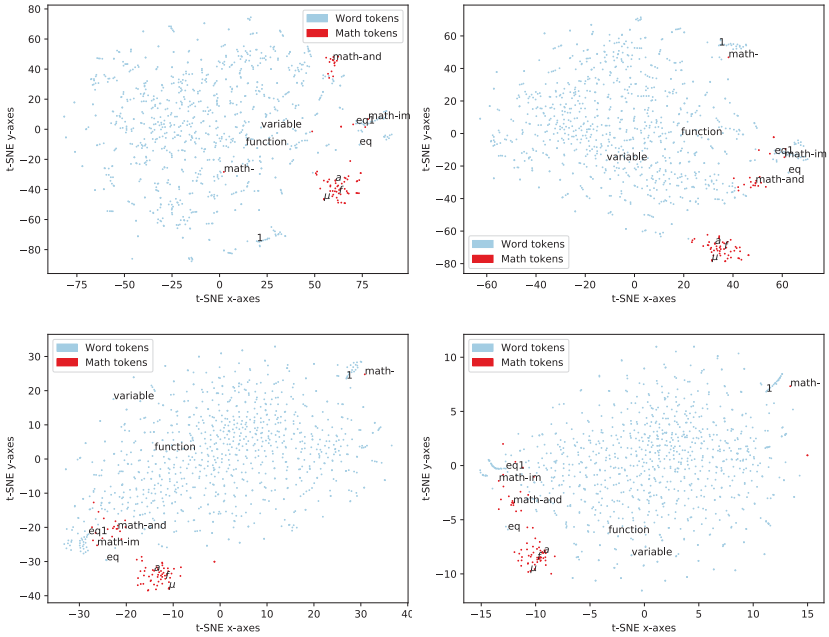


Figure 3.1: t-SNE plot of top-1000 closest vectors of the identifier f with perplexity values 5 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) and the default values of the t-SNE python package for other settings.

Even though math notations can change, such as π in the example above, one could assume the existence of a common ground for most notations. The low performance of our experiments compared to the results in [330] seem to confirm that math notations change regularly in real-world documents, i.e., are tied to a specific context. If a common ground exists, for math notations, it must be marginally small, at least in the 100 test cases from [18].

3.1.3 On Overcoming the Issues of Knowledge Extraction Approaches

We assume the low performance regarding our knowledge extraction experiments are caused by fundamental issues that should be discussed before more efforts are made to train ML algorithms for extracting knowledge of math expressions. In the following, we discuss some reasons that we believe can help ML algorithms to understand mathematics better.

It is reported that 70% of mathematical symbols are explicitly declared in the context [394]. Only four reasons justify an explicit declaration in the context: (a) a new mathematical symbol is defined, (b) a known notation is changed, (c) used symbols are present in other contexts and require specifications to be correctly interpreted, or (d) authors' declarations are redundant (e.g., for improving readability). We assume (d) is a rare scenario compared to the other ones (a-c), except in educational literature. Current math-embedding techniques can learn semantic

connections only in that 70%, where the definiens is available. Besides (d), the algorithm would learn either rare notations (in case of (a)) or ambiguous notations (in cases (b-c)). The flexibility that mathematical documents allow to (re)define used mathematical notations further corroborates the complexity of learning mathematics.

Learning algorithms would benefit from literature focused on (a) and (d), instead of (b) and (c). Similar to students who start to learn mathematics, ML algorithms have to consider the structure of the content they learn. It is hard to learn mathematics only considering arXiv documents without prior or complementary knowledge. Usually, these documents represent state-of-the-art findings containing new and unusual notations and lack of extensive explanations (e.g., due to page limitations). In contrast, educational books carefully and extensively explain new concepts. We assume better results can be obtained if ML algorithms are trained in multiple stages, first on educational literature, then on datasets of advanced math articles. A basic model trained in educational literature should capture standard relations between mathematical concepts and descriptors. This model should also be able to capture patterns independently of how new or unusual the notations are present in the literature. In 2014, Matsuzaki et al. [247] presented some promising results to answer mathematical questions from Japanese university entrance exams automatically. While the approach involves many manual adjustments and analysis, the promising results illustrate the different levels of knowledge that is still required for understanding arXiv documents vs. university entrance level exams. A well-structured digital mathematical library that distinguishes the different levels of sophistication in articles (e.g. introductions vs. state-of-the-art publications) would also benefit mathematical machine learning tasks.

The lack of references and applications that provide a solid semantic structure of natural language for mathematical identifiers make the disambiguation process of the latter even more challenging. In natural texts, one can try to infer the most suitable word sense for a word based on the lemma[15] itself, the adjacent words, dictionaries, and thesauri to name a few. However, in the mathematical arena, the scarcity of resources and the flexibility of redefining their identifiers make this issue much harder. The context text preceding or following the mathematical equation is essential for its understanding. This context can be considered in a long or short distance away from the equation, which aggravates the problem. Thus, a comprehensive annotated dataset that addresses these needs of structural knowledge would enable further progress in MathIR with the help of ML algorithms.

Another primary source of complexity is the inherent ambiguity present in any language, especially in mathematics. A typical workaround in linguistics for such ambiguous notations is to consider the use of lexical databases (e.g., WordNet [116, 261]) to identify the most suitable word senses for a given word. These databases allow embeddings algorithms to train a vector for each semantic meaning for every token. For example, *Java* could have multiple vectors in a single model according to its different meanings of the word, e.g., the island in the south of Indonesia, the programming language or the coffee beans. However, mathematics lacks such systems, which makes its adoption not feasible at the moment. Youssef [402] proposes the use of tags, similarly to the PoS tags in linguistics, but for tagging mathematical TEX tokens, bringing more information to the tokens considered. As a result, a lexicon containing several meanings for a large set of mathematical symbols is developed. OntoMathPro [104, 105] aims for generating a comprehensive ontology of mathematical knowledge and, therefore, also contain

[15] canonical form, dictionary form, or citation form of a set of words

information about the different meanings of mathematical tokens. Such dictionaries might enable the disambiguation approaches in linguistics to be used in mathematical embedding in the near future.

Another issue in recent publications is the lack of standards and the scarcity of benchmarks to properly evaluate MathIR algorithms. Krstovski and Blei [215], and Yasunaga and Lafferty [400] provide an interesting perspective on the problem of mathematic embeddings. Their experiments are focused on math-analogies. Our findings on Section 3.2 corroborate with the math-analogies results, as our experiments have comparable results in a controlled environment. However, because of a missing well-established benchmark, we, as well the mentioned publications, are only able to provide incipient results. Existing datasets are often created for and, therefore, limited to specific tasks. For example, the NTCIR math tasks [21, 22, 405] or the upcoming ARQMath[16] task, provide datasets that are specifically designed to tackle problems of mathematical search engines. The secondary task of ARQMath actually search for math-analogies. In general, a proper, common standard for interpreting semantic structures of mathematics (see for example the mentioned problems with α_i in Section 2) would be beneficial for several tasks in MathIR, such as semantic knowledge extraction.

3.1.4 The Future of Math Embeddings

As we explored through this section, our preliminary results stress the urgent need for creating extensive math-specific benchmarks for testing math embedding techniques on math-specific tasks. To appreciate more the magnitude and dimensions of creating such benchmarks, it is instructive to look at some of those developed for NLP whose tasks can beneficially inform and guide corresponding tasks in MLP. The NLP benchmarks include one for natural language inference [47], one for machine comprehension [306], one for semantic role modeling [281], and one for language modeling [68], to name a few. With such benchmarks, which are often *de facto* standards for the corresponding NLP tasks, the NLP research community has been able to (1) measure the performance of new techniques up to statistical significance, and (2) track progress in various NLP techniques, including deep learning for NLP, by quickly comparing the performance of new techniques to others and to the state-of-the-art.

While our exploratory studies regarding our term similarities, analogies, and query expansions need extensive future experimentation for statistically significant validation on large datasets and benchmarks, they show some of the promise and limitations of word embedding in math (MLP) applications. Especially its applicability for our desired knowledge extraction process is highly questionable. One of the main issues we encountered for embedding mathematics is the inability to model the nested semantic structure of mathematical expressions. In the following, we will further explore properties of mathematical subexpressions by analyzing their frequency distributions in large datasets.

3.2 Semantification with Mathematical Objects of Interest

As discussed before, math expressions often contain meaningful and important subexpressions. MathIR [141] applications could benefit from an approach that lies between the extremes of

[16] https://www.cs.rit.edu/~dprl/ARQMath/ [accessed 2020-02-01]

examining only individual symbols or considering an entire equation as one entity. Consider for example, the explicit definition for Jacobi polynomials [98, (18.5.7)]

The Explicit Definition of Jacobi Polynomials

$$P_n^{(\alpha,\beta)}(x) = \frac{\Gamma(\alpha+n+1)}{n!\,\Gamma(\alpha+\beta+n+1)} \sum_{m=0}^{n} \binom{n}{m} \frac{\Gamma(\alpha+\beta+n+m+1)}{\Gamma(\alpha+m+1)} \left(\frac{x-1}{2}\right)^m \tag{3.2}$$

The *interesting* components in this equation are $P_n^{(\alpha,\beta)}(x)$ on the left-hand side, and the appearance of the gamma function $\Gamma(s)$ on the right-hand side, implying a direct relationship between Jacobi polynomials and the gamma function. Considering the entire expression as a single object misses this important relationship. On the other hand, focusing on single symbols can result in the misleading interpretation of Γ as a variable and $\Gamma(\alpha+n+1)$ as a multiplication between Γ and $(\alpha+n+1)$. A system capable of identifying the important components, such as $P_n^{(\alpha,\beta)}(x)$ or $\Gamma(\alpha+n+1)$, is therefore desirable. Hereafter, we define these components as Mathematical Objects of Interest (MOI) [9].

The *importance* of math objects is a somewhat imprecise description and thus difficult to measure. Currently, not much effort has been made in identifying meaningful subexpressions. Kristianto et al. [214] introduced dependency graphs between formulae. With this approach, they were able to build dependency graphs of mathematical expressions, but only if the expressions appeared as single expressions in the context. For example, if $\Gamma(\alpha+n+1)$ appears as a stand-alone expression in the context, the algorithm will declare a dependency with Equation (3.2). However, it is more likely that different forms, such as $\Gamma(s)$, appear in the context. Since this expression does not match any subexpression in Equation (3.2), the approach cannot establish a connection with $\Gamma(s)$. Kohlhase et al. studied in [191, 193, 196] another approach to identify essential components in formulae. They performed eye-tracking studies to identify important areas in rendered mathematical formulae. While this is an interesting approach that allows one to learn more about the insights of human behaviors of reading and understanding math, it is inaccessible for extensive studies.

This section presents the first extensive frequency distribution study of mathematical equations in two large scientific corpora, the e-Print archive arXiv.org (hereafter referred to as arXiv[17]) and the international reviewing service for pure and applied mathematics zbMATH[18]. We will show that math expressions, similar to words in natural language corpora, also obey Zipf's law [297], and therefore follows a *Zipfian* distribution. Related research projects observed a relation to Zipf's law for single math symbols [71, 329]. In the context of quantitative linguistics, Zipf's law states that given a text corpus, the frequency of any word is inversely proportional to its rank in the frequency table. Motivated by the similarity to linguistic properties, we will present a novel approach for ranking formulae by their relevance via a customized version of the ranking function BM25 [310]. We will present results that can be easily embedded in other systems in order to distinguish between common and uncommon notations within formulae. Our results lay a foundation for future research projects in MathIR.

[17] https://arxiv.org/ [accessed 2019-09-01]
[18] https://zbmath.org [accessed 2019-09-01]

Fundamental knowledge on frequency distributions of math formulae is beneficial for numerous applications in MathIR, ranging from educational purposes [341] to math recommendation systems [50], search engines [92, 274], and even automatic plagiarism detection systems [253, 254, 334]. For example, students can search for the conventions to write certain quantities in formulae; document preparation systems can integrate an auto-completion or auto-correction service for math inputs; search or recommendation engines can adjust their ranking scores with respect to standard notations; and plagiarism detection systems can estimate whether two identical formulae indicate potential plagiarism or are just using the conventional notations in a particular subject area. To exemplify the applicability of our findings, we present a textual search approach to retrieve mathematical formulae. Further, we will extend zbMATH's faceted search by providing facets of mathematical formulae according to a given textual search query. Lastly, we present a simple auto-completion system for math inputs as a contribution towards advancing mathematical recommendation systems. Further, we show that the results provide useful insights for plagiarism detection algorithms. We provide access to the source code, the results, and extended versions of all of the figures appearing in this paper at https://github.com/ag-gipp/FormulaCloudData.

3.2.1 Related Work

Today, mathematical search engines index formulae in a database. Much effort has been undertaken to make this process as efficient as possible in terms of precision and runtime performance [92, 181, 231, 236, 407]. The generated databases naturally contain the information required to examine the distributions of the indexed mathematical formulae. Yet, no in-depth studies of these distributions have been undertaken. Instead, math search engines focus on other aspects, such as devising novel similarity measures and improving runtime efficiency. This is because the goal of math search engines is to retrieve relevant (i.e., similar) formulae which correspond to a given search query that partially [211, 231, 274] or exclusively [92, 181, 182] contains formulae. However, for a fundamental study of distributions of mathematical expressions, no similarity measures nor efficient lookup or indexing is required. Thus, we use the general-purpose query language XQuery and employ the BaseX[19] implementation. BaseX is a free open-source XML database engine, which is fully compatible with the latest XQuery standard [140, 396]. Since our implementations rely on XQuery, we are able to switch to any other database which allows for processing via XQuery.

3.2.2 Data Preparation

LaTeX is the de facto standard for the preparation of academic manuscripts in the fields of mathematics and physics [129]. Since LaTeX allows for advanced customizations and even computations, it is challenging to process. For this reason, LaTeX expressions are unsuitable for an extensive distribution analysis of mathematical notations. For mathematical expressions on the web, the XML formatted MathML[20] is the current standard, as specified by the World Wide Web Consortium (W3C). The tree structure and the fixed standard, i.e., MathML tags, cannot be changed, thus making this data format reliable. Several available tools are able to convert from LaTeX to MathML [18] and various databases are able to index XML data. Thus, for this study,

[19] http://basex.org/ [accessed 2019-09-01]; We used BaseX 9.2 for our experiments.
[20] https://www.w3.org/TR/MathML3/ [accessed 2019-09-01]

we have chosen to focus on MathML. In the following, we investigate the databases arXMLiv (08/2018) [132] and zbMATH[21] [333].

The arXMLiv dataset (≈1.2 million documents) contains HTML5 versions of the documents from the e-Print archive arXiv.org. The HTML5 documents were generated from the TEX sources via LATEXML [257]. LATEXML converted all mathematical expressions into MathML with parallel markup, i.e., presentation and content MathML. In this study we only consider the subsets *no-problem* and *warning*, which generated no errors during the conversion process. Nonetheless, the MathML data generated still contains some errors or falsely annotated math. For example, we discovered several instances of affiliation and footnotes, SVG[22] and other unknown tags, encoded in MathML. Regarding the footnotes, we presumed that authors falsely used mathematical environments for generating footnote or affiliation marks. We used the TEX string, provided as an attribute in the MathML data, to filter out expressions that match the string '`{}^{*}`', where '`*`' indicates any possible expression. In addition, we filtered out SVG and other unknown tags. We assume that these expressions were generated by mistake due to limitations of LATEXML. The final arXiv dataset consisted of 841,008 documents which contained at least one mathematical formula. The dataset contained a total of 294,151,288 mathematical expressions.

In addition to arXiv, we investigated zbMATH, an international reviewing service for pure and applied mathematics which contains abstracts and reviews of articles, hereafter uniformly called abstracts, mainly from the domains of pure and applied mathematics. The abstracts in zbMATH are formatted in TEX [333]. To be able to compare arXiv and zbMATH, we manually generated MathML via LATEXML for each mathematical formula in zbMATH and performed the same filters as used for the arXiv documents. The zbMATH dataset contained 2,813,451 abstracts, of which 1,349,297 contained at least one formula. In total, the dataset contained 11,747,860 formulae. Even though the total number of formulae is smaller compared to arXiv, we hypothesize that math formulae in abstracts are particularly meaningful.

3.2.2.1 Data Wrangling

Since we focused on the frequency distributions of visual expressions, we only considered pMML. Rather than normalizing the pMML data, e.g., via MathMLCan [117], which would also change the tree structure and visual core elements in pMML, we only eliminated the attributes. These attributes are used for minor visual changes, e.g., stretched parentheses or inline limits of sums and integrals. Thus, for this first study, we preserved the core structure of the pMML data, which might provide insightful statistics for the MathML community to further cultivate the standard. After extracting all MathML expressions, filtering out falsely annotated math and SVG tags, and eliminating unnecessary attributes and annotations, the datasets required 83GB of disk space for arXiv and 6GB for zbMATH, respectively.

In the following, we indexed the data via BaseX. The indexed datasets required a disk space of 143.9GB in total (140GB for arXiv and 3.9GB for zbMATH). Due to the limitations[23] of databases in BaseX, it was necessary to split our datasets into smaller subsets. We split the datasets

[21] https://zbmath.org/ [accessed 2019-09-01]

[22] Scalable Vector Graphics

[23] A detailed overview of the limitations of BaseX databases can be found at http://docs.basex.org/wiki/Statistics [accessed 2019-09-01].

according to the 20 major article categories of arXiv[24] and classifications of zbMATH. To increase performance, we use BaseX in a server-client environment. We experienced performance issues in BaseX when multiple clients repeatedly requested data from the same server in short intervals. We determined that the best workaround for this issue was to launch BaseX servers for each database, i.e., each category/classification.

Mathematical expressions often consist of multiple meaningful subexpressions, which we defined as MOIs. However, without further investigation of the context, it is impossible to determine meaningful subexpressions. As a consequence, every equation is a potential MOI on its own and potentially consists of multiple other MOIs. For an extensive frequency distributional analysis, we aim to discover all possible mathematical objects. Hence, we split every formula into its components. Since MathML is an XML data format (essentially a tree-structured format), we define subexpressions of equations as subtrees of its MathML format.

$P_n^{(\alpha,\beta)}(x)$

```
1  <math><mrow>
2    <msubsup>
3      <mi>P</mi>
4      <mi>n</mi>
5      <mrow>
6        <mo>(</mo>
7        <mi>α</mi>
8        <mo>,</mo>
9        <mi>β</mi>
10       <mo>)</mo>
11       <mo></mo>
12     </mrow>
13   </msubsup>
14   <mo></mo>
15   <mrow>
16     <mo>(</mo>
17     <mi>x</mi>
18     <mo>)</mo>
19   </mrow>
20 </mrow></math>
```

Listing 3.1: MathML representation of $P_n^{(\alpha,\beta)}(x)$.

Listing 3.1 illustrates a Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$ in pMML. The `<mo>` element on line 14 contains the *invisible times* UTF-8 character. By definition, the `<math>` element is the root element of MathML expressions. Since we cut off all other elements besides pMML nodes, each `<math>` element has one and only one child element[25]. Thus, we define the child element of the `<math>` element as the root of the expression. Starting from this root element, we explore all subexpressions. For this study, we presume that every meaningful mathematical object (i.e., MOI) must contain at least one identifier.

Hence, we only study subtrees which contain at least one `<mi>` node. Identifiers, in the sense of MathML, are '*symbolic names or arbitrary text*' [26], e.g., single Latin or Greek letters. Identifiers do not contain special characters (other than Greek letters) or numbers. As a consequence, arithmetic expressions, such as $(1+2)^2$, or sequences of special characters and numbers, such as $\{1,2,...\} \cap \{-1\}$, will not appear in our distributional analysis. However, if a sequence or arithmetic expression consists of an identifier somewhere in the pMML tree (such as in $\{1,2,...\} \cap A$), the entire expression will be recognized. The Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$, therefore consists of the following subexpressions: $P_n^{(\alpha,\beta)}$, (α,β), (x), and the single identifiers P, n, α, β, and x. The entire expression is also a mathematical object. Hence, we take entire expressions with an identifier into account for our analysis. In the following, the set of subexpressions will be understood to include the expression itself.

For our experiments, we also generated a string representation of the MathML data. The string is generated recursively by applying one of two rules for each node: (i) if the current node is a leaf, the node-tag and the content will be merged by a colon, e.g., `<mi>x</mi>` will be converted

[24] The arXiv categories *astro-ph* (astro physics), *cond-mat* (condensed matter), and *math* (mathematics) were still too large for a single database. Thus, we split those categories into two equally sized parts.

[25] Sequences are always nested in an `<mrow>` element.

[26] https://www.w3.org/TR/MathML3/chapter3.html [accessed 2019-09-01]

to `mi:x`; (ii) otherwise the node-tag wraps parentheses around its content and separates the children by a comma, e.g.,

```
<mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
```
(3.3)

will be converted to

```
mrow(mo:(,mi:x,mo:)).
```
(3.4)

Furthermore, the special UTF-8 characters for invisible times (U+2062) and function application (U+2061) are replaced by `ivt` and `fa`, respectively. For example, the gamma function with argument $x+1$, $\Gamma(x+1)$ would be represented by

```
mrow(mi:Γ,mo:ivt,mrow(mo:(,mrow(mi:x,mo:+,mn:1),mo:))).
```
(3.5)

Between Γ and $(x+1)$, there would most likely be the special character for *invisible times* rather than for *function application*, because LaTeXML is not able to parse Γ as a function. Note that this string conversion is a bijective mapping. The string representation reduces the verbose XML format to a more concise presentation. Thus, an equivalence check between two expressions is more efficient.

3.2.2.2 Complexity of Math

Mathematical expressions can become complex and lengthy. The tree structure of MathML allows us to introduce a measure that reflects the complexity of mathematical expressions. More complex expressions usually consist of more extensively nested subtrees in the MathML data. Thus, we define the complexity of a mathematical expression by the maximum depth of the MathML tree. In XML the content of a node and its attributes are commonly interpreted as children of the node. Thus, we define the depth of a single node as 1 rather than 0, i.e., single identifiers, such as `<mi>P</mi>`, have a complexity of 1. The Jacobi polynomial from Listing 3.1 has a complexity of 4.

We perform the extraction of subexpressions from MathML in BaseX. The algorithm for the extraction process is written in XQuery. The algorithm traverses recursively downwards from the root to the leaves. In each iteration, it checks whether there is an identifier, i.e., `<mi>` element, among the descendants of the current node. If there is no such element, the subtree will be ignored. It seems counterintuitive to start from the root and check if an identifier is among the descendants rather than starting at each identifier and traversing upwards to the root. If an XQuery requests a node in BaseX, BaseX loads the entire subtree of the requested node into the cache (up to a specified size). If the algorithm traverses upwards through the MathML tree, the XQuery will trigger database requests in every iteration. Hence, the downwards implementation performs better, since there is only one database request for every expression rather than for every subexpression.

Since we only minimize the pMML data rather than normalizing it, two identically rendered expressions may have different complexities. For instance,

```
<mrow><mi>x</mi></mrow>
```
(3.6)

consists of two distinct subexpressions, but both of them are displayed the same. Another problem often appears for arrays or similar visually complicated structures. The extracted expressions are not necessarily logical subexpressions. We will consider applying more advanced embedding techniques such as special tokenizers [231], symbol layout trees [92, 407], and a MathML normalization via MathMLCan [117] in future research to overcome these issues.

3.2.3 Frequency Distributions of Mathematical Formulae

By splitting each formula into subexpressions, we generated longer documents and a bias towards low complexities. Note that, hereafter, we only refer to the mathematical content of documents. Thus, the length of a document refers to the number of math formulae - here the number of subexpressions - in the document. After splitting expressions into subexpressions, arXiv consists of 2.5B and zbMATH of 61M expressions, which raised the average document length to 2,982.87 for arXiv and 45.47 for zbMATH, respectively.

For calculating frequency distributions, we merged two subexpressions if their string representations were identical. Remember, the string representation is unique for each MathML tree. After merging, arXiv consisted of 350,206,974 unique mathematical subexpressions with a maximum complexity of 218 and an average complexity of 5.01. For high complexities over 70, the formulae show some erroneous structures that might be generated from LaTeXML by mistake. For example, the expression with the highest complexity is a long sequence of a polynomial starting with '$P_4(t_1, t_3, t_7, t_{11}) =$' followed by 690 summands. The complexity is caused by a high number of unnecessarily deeply nested `<mrow>` nodes. The highest complexity with a minimum document frequency of two is 39, which is a continued fraction. Since continued fractions are nested fractions, they naturally have a large complexity. One of the most complex expressions (complexity 20) with a minimum document frequency of three was the formula

$$\left(\sum_{j_1=1}^{n}\left(\sum_{j_2=1}^{n}\left(\cdots\left(\sum_{j_m=1}^{n}\left|T\left(e_{j_1},\ldots,e_{j_m}\right)\right|^{q_m}\right)^{\frac{q_{m-1}}{q_m}}\cdots\right)^{\frac{q_2}{q_3}}\right)^{\frac{q_1}{q_2}}\right)^{\frac{1}{q_1}} \leq C^{\mathbb{K}}_{m,p,\mathbf{q}}\,\|T\|\,. \quad (3.7)$$

In contrast, zbMATH only consisted of 8,450,496 unique expressions with a maximum complexity of 26 and an average complexity of 3.89. One of the most complex expressions in zbMATH with a minimum document frequency of three was

$$M_p(r,f) = \left(\frac{1}{2\pi}\int_0^{2\pi}\left|f\left(re^{i\theta}\right)\right|^p d\theta\right)^{1/p}. \quad (3.8)$$

As we expected, reviews and abstracts in zbMATH were generally shorter and consisted of less complex mathematical formulae. The dataset also appeared to contain fewer erroneous expressions, since expressions of complexity 25 are still readable and meaningful.

Figure 3.2 shows the ratio of unique subexpressions for each complexity in both datasets. The figure illustrates that both datasets share a peak at complexity four. Compared to zbMATH, the arXiv expressions are slightly more evenly distributed over the different levels of complexities. Interestingly, complexities one and two are not dominant in either of the two datasets. Single identifiers only make up 0.03% in arXiv and 0.12% in zbMATH, which is comparable to expressions of complexity 19 and 14, respectively. This finding illustrates the problem of capturing semantic meanings for single identifiers rather than for more complex expressions [330]. It also substantiates that entire expressions, if too complex, are not suitable either for capturing the semantic meanings [214]. Instead, a middle ground is desirable, since the most unique expressions in both datasets have a complexity between 3 and 5. Table 3.4 summarizes the statistics of the examined datasets.

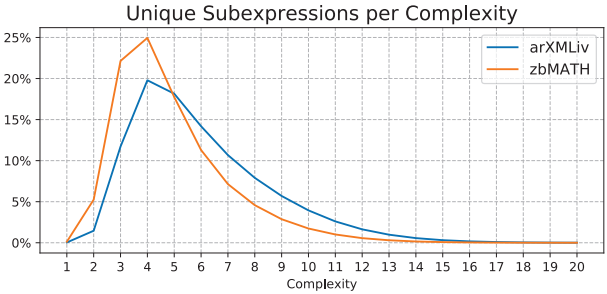


Figure 3.2: Unique subexpressions for each complexity in arXiv and zbMATH.

Table 3.4: Dataset overview. Average Document Length is defined as the average number of subexpressions per document.

Category	arXiv	zbMATH
Documents	841,008	1,349,297
Formulae	294,151,288	11,747,860
Subexpressions	2,508,620,512	61,355,307
Unique Subexpressions	350,206,974	8,450,496
Average Document Length	2,982.87	45.47
Average Complexity	5.01	3.89
Maximum Complexity	218	26

3.2.3.1 Zipf's Law

In linguistics, it is well known that word distributions follow Zipf's Law [297], i.e., the r-th most frequent word has a frequency that scales to

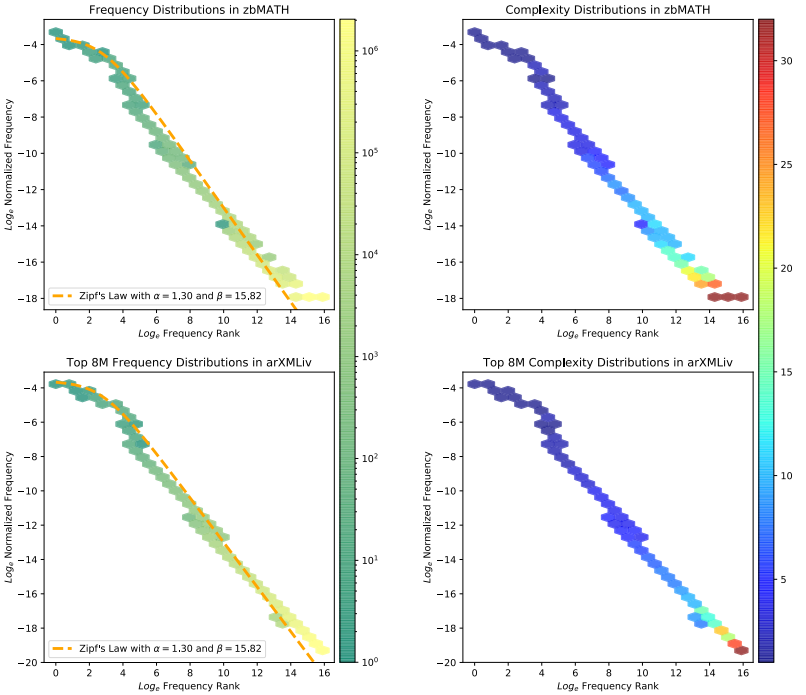
$$f(r) \propto \frac{1}{r^{\alpha}} \tag{3.9}$$

with $\alpha \approx 1$. A better approximation can be applied by a shifted distribution

$$f(r) \propto \frac{1}{(r+\beta)^{\alpha}}, \tag{3.10}$$

where $\alpha \approx 1$ and $\beta \approx 2.7$. In a study on Zipf's law, Piantadosi [297] illustrated that not only words in natural language corpora follow this law surprisingly accurately, but also many other human-created sets. For instance, in programming languages, in biological systems, and even in music. Since mathematical communication has derived as the result of centuries of research, it would not be surprising if mathematical notations would also follow Zipf's law. The primary conclusion of the law illustrates that there are some very common tokens against a large number of symbols which are not used frequently. Based on this assumption, we can postulate that a score based on frequencies might be able to measure the peculiarity of a token. The infamous TF-IDF ranking functions and their derivatives [23, 310] have performed well in linguistics for

many years and are still widely used in retrieval systems [30]. However, since we split every expression into its subexpressions, we generated an anomalous bias towards shorter, i.e., less complex, formulae. Hence, distributions of subexpressions may not obey Zipf's law.



(a) Frequency Distributions

(b) Complexity Distributions

Figure 3.3: Each figure illustrates the relationship between the frequency ranks (x-axis) and the normalized frequency (y-axis) in zbMATH (top) and arXiv (bottom). For arXiv, only the first 8 million entries are plotted to be comparable with zbMATH ($\approx$ 8.5 million entries). Subfigure (a) shades the hexagonal bins from green to yellow using a logarithmic scale according to the number of math expressions that fall into a bin. The dashed orange line represents Zipf's distribution (3.10). The values for α and β are provided in the plots. Subfigure (b) shades the bins from blue to red according to the maximum complexity in each bin.

Figure 3.3 visualizes a comparison between Zipf's law and the frequency distributions of mathematical subexpressions in arXiv and zbMATH. The dashed orange line visualizes the power law (3.10). The plots demonstrate that the distributions in both datasets obey this power law. Interestingly, there is not much difference in the distributions between both datasets. Both distributions seem to follow the same power law, with $\alpha = 1.3$ and $\beta = 15.82$. Moreover, we can observe that the developed complexity measure seems to be appropriate, since the complexity distributions for formulae are similar to the distributions for the length of words [297]. In other

words, more complex formulae, as well as long words in natural languages, are generally more specialized and thus appear less frequent throughout the corpus. Note that colors of the bins for complexities fluctuate for rare expressions because the color represents the maximum rather than the average complexity in each bin.

3.2.3.2 Analyzing and Comparing Frequencies

Figure 3.4 shows in detail the most frequently used mathematical expressions in arXiv for the complexities 1 to 7. The orange dashed line visible in all graphs represents the normal Zipf's law distribution from Equation (3.9). We explore the total frequency values without any normalization. Thus, Equation (3.9) was multiplied by the highest frequency for each complexity level to fit the distribution. The plots in Figure 3.4 demonstrate that even though the parameter α varies between 0.35 and 0.62, the distributions in each complexity class also obey Zipf's law.

The plots for each complexity class contain some interesting fluctuations. We can spot a set of five single identifiers that are most frequently used throughout arXiv: n, i, x, t, and k. Even though the distributions follow Zipf's law accurately, we can explore that these five identifiers are proportionally more frequently used than other identifiers and clearly separate themselves above the rest (notice the large gap from k to a). All of the five identifiers are known to be used in a large variety of scenarios. Surprisingly, one might expect that common pairs of identifiers would share comparable frequencies in the plots. However, typical pairs, such as x and y, or α and β, possess a large discrepancy.

The plot of complexity two also reveals that two expressions are proportionally more often used than others: (x) and (t). These two expressions appear more than three times as often in the corpus than any other expression of the same complexity. On the other hand, the quantitative difference between (x) and (t) is negligible. We may assume that arXiv's primary domain, physics, causes the quantitative disparity between (x), (t), and the other tokens. The primary domain of the dataset becomes more clearly visible for higher complexities, such as $SU(2)$ (C3[27]) or kms^{-1} (C4).

Another surprising property of arXiv is that symmetry groups, such as $SU(2)$, appear to play an essential role in the majority of articles on arXiv, see $SU(2)$ (C3), $SU(2)_L$ (C4), and $SU(2) \times SU(2)$ (C5), among others. The plots of higher complexities[28], made this even more noticeable. Given a complexity of six, for example, the most frequently used expression was $SU(2)_L \times SU(2)_R$, and for a complexity of seven it was $SU(3) \times SU(2) \times U(1)$. Given a complexity of eight, ten out of the top-12 expressions were from symmetry group calculations.

It is also worthwhile to compare expressions among different levels of complexities. For instance, (x) and (t) appeared almost six million times in the corpus, but $f(x)$ (at position three in C3) was the only expression which contained one of these most common expressions. Note that subexpressions of variations, such as (x_0), (t_0), or $(t - t')$, do not match the expression of complexity two. This may imply that (x), and especially (t), appear in many different scenarios. Further, we can examine that even though (x) is a part of $f(x)$ in only approximately 3% of all cases, it is still the most likely combination. These results are especially useful for recommendation systems that make use of math as input. Moreover, plagiarism detection

[27] We refer to a given complexity n with Cn, i.e., C3 refers to complexity 3.

[28] More plots showing higher complexities are available at https://github.com/ag-gipp/FormulaCloudData [accessed 2021-10-01]

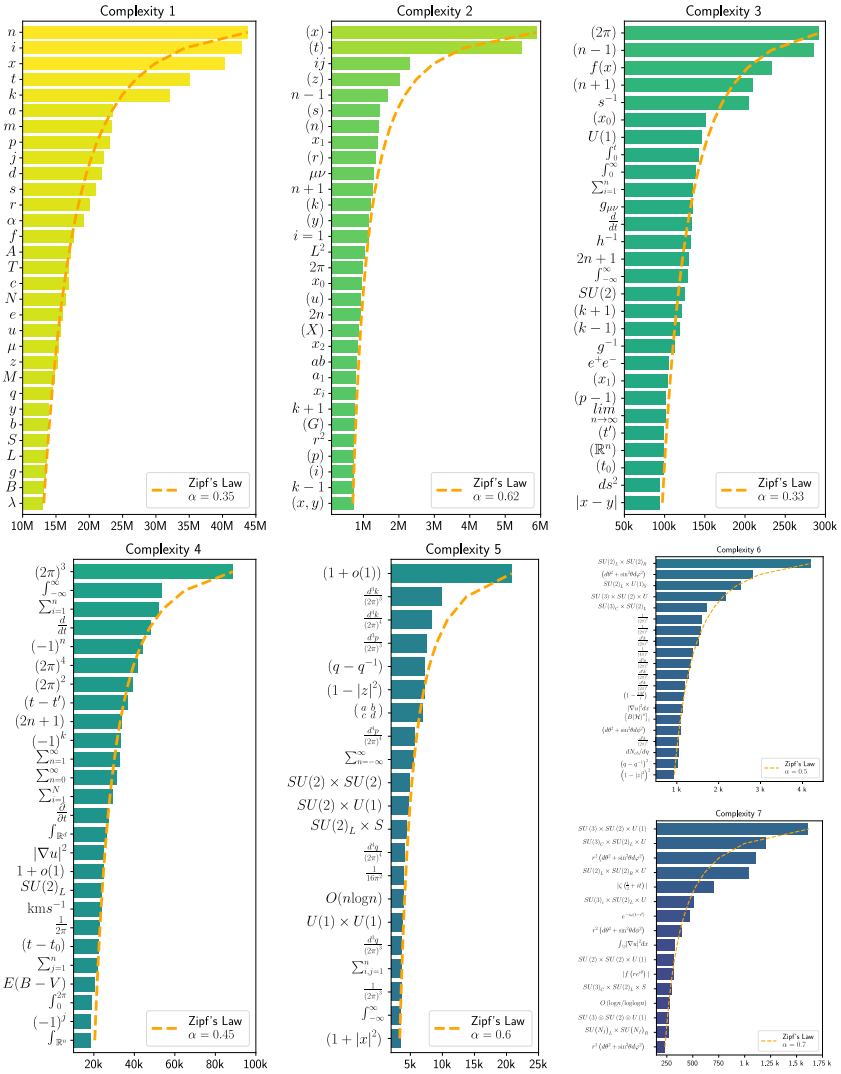


Figure 3.4: Overview of the most frequent mathematical expressions in arXiv for complexities 1-7. The color gradient from yellow to blue represents the frequency in the dataset. Zipf's law (3.9) is represented by a dashed orange line.

systems may also benefit from such a knowledge base. For instance, it might be evident that $f(x)$ is a very common expression, but for automatic systems that work on a large scale, it is not clear whether duplicate occurrences of $f(x)$ or $\Xi(x)$ should be scored differently, e.g., in the case of plagiarism detection.

Figure 3.4 shows only the most frequently occurring expressions in arXiv. Since we already explored a bias towards physics formulae in arXiv, it is worth comparing the expressions present within both datasets. Figure 3.5 compares the 25-top expressions for the complexities one to six. In zbMATH, we discovered that computer science and graph theory appeared as popular topics, see for example $G = (V, E)$ (in C3 at position 20) and the Bachmann-Landau notations in $O(\log n)$, $O(n^2)$, and $O(n^3)$ (C4 positions 2, 3, and 19).

From Figure 3.5, we can also deduce useful information for MathIR tasks which focus on semantic information. Current semantic extraction tools [330] or LaTeX parsers [18] still have difficulties distinguishing *multiplications* from *function calls*. For example as mentioned before, LaTeXML [257] adds an *invisible times* character between $f(x)$ rather than a *function application*. Investigating the most frequently used terms in zbMATH in Table 3.5 reveals that u is most likely considered to be a function in the dataset: $u(t)$ (rank 8), $u(x)$ (rank 13), u_{xx} (rank 16), $u(0)$ (rank 17), $|\nabla u|$ (rank 22). Manual investigations of extended lists reveal even more hits: $u_0(x)$ (rank 30), $-\Delta u$ (rank 32), and $u(x, t)$ (rank 33). Since all eight terms are among the most frequent 35 entries in zbMATH, it implies that u can most likely be considered to imply a function in zbMATH. Of course, this does not imply that u must always be a function in zbMATH (see $f(u)$ on rank 14 in C3), but this allows us to exploit probabilities for improving MathIR performance. For instance, if not stated otherwise, u could be interpreted as a function by default, which could help increase the precision of the aforementioned tools.

Figure 3.5 also demonstrates that our two datasets diverge for increasing complexities. Hence, we can assume that frequencies of less complex formulae are more topic-independent. Conversely, the more complex a math formula is, the more context-specific it is. In the following, we will further investigate this assumption by applying TF-IDF rankings on the distributions.

3.2.4 Relevance Ranking for Formulae

Zipf's law encourages the idea of scoring the relevance of words according to their number of occurrences in the corpus and in the documents. The family of BM25 ranking functions based on TF-IDF scores are still widely used in several retrieval systems [30, 310]. Since we demonstrated that mathematical formulae (and their subexpressions) obey Zipf's law in large scientific corpora, it appears intuitive to also use TF-IDF rankings, such as a variant of BM25, to calculate their relevance.

Okapi BM25

In its original form [310], *Okapi BM25* was calculated as follows

$$\mathrm{bm25}(t,d) := \frac{(k+1)\,\mathrm{IDF}(t)\,\mathrm{TF}(t,d)}{\mathrm{TF}(t,d) + k\left(1 - b + \frac{b|d|}{\mathrm{AVG}_{\mathrm{DL}}}\right)}. \tag{3.11}$$

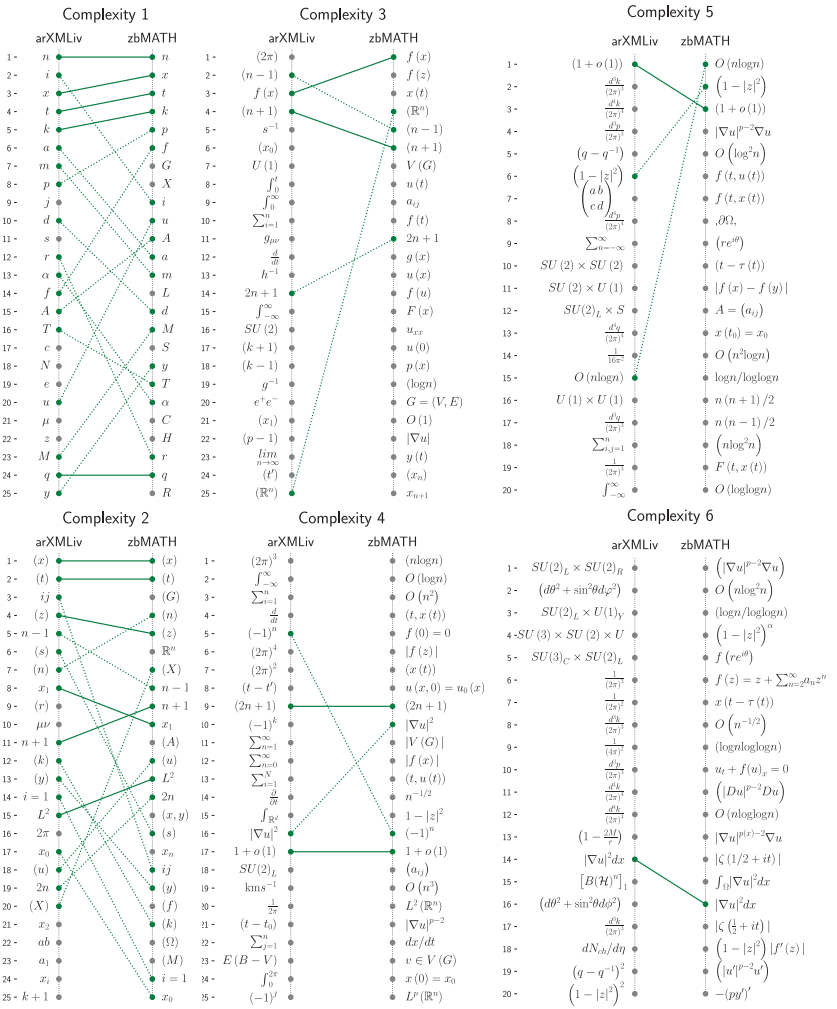


Figure 3.5: The top-20 and 25 most frequent expressions in arXiv (left) and zbMATH (right) for complexities 1-6. A line between both sets indicates a matching set. Bold lines indicate that the matches share a similar rank (distance of 0 or 1).

Here, $\mathrm{TF}\,(t, d)$ is the term frequency of t in the document d, $|d|$ the length of the document d (in our case, the number of subexpressions), $\mathrm{AVG}_{\mathrm{DL}}$ the average length of the documents in the corpus (see Table 3.4), and $\mathrm{IDF}\,(t)$ is the inverse document frequency of t, defined as

$$\mathrm{IDF}(t) := \log \frac{N - n(t) + \frac{1}{2}}{n(t) + \frac{1}{2}}, \tag{3.12}$$

where N is the number of documents in the corpus and $n(t)$ the number of documents which contain the term t. By adding $\frac{1}{2}$, we avoid $\log 0$ and division by 0. The parameters k and b are free, with b controlling the influence of the normalized document length and k controlling the influence of the term frequency on the final score. For our experiments, we chose the standard value $k = 1.2$ and a high impact factor of the normalized document length via $b = 0.95$.

As a result of our subexpression extraction algorithm, we generated a bias towards low complexities. Moreover, longer documents generally consist of more complex expressions. As demonstrated in Section 3.2.2.1, a document that only consists of the single expression $P_n^{(\alpha,\beta)}(x)$, i.e., the document had a length of one, would generate eight subexpressions, i.e., it results in a document length of eight. Thus, we modify the BM25 score in Equation (3.11) to emphasize higher complexities and longer documents. First, the average document length is divided by the average complexity AVG_C in the corpus that is used (see Table 3.4), and we calculate the reciprocal of the document length normalization to emphasize longer documents.

Moreover, in the scope of a single document, we want to emphasize expressions that do not appear frequently in this document, but are the most frequent among their level of complexity. Thus, less complex expressions are ranked more highly if the document overall is not very complex. To achieve this weighting, we normalize the term frequency of an expression t according to its complexity $c(t)$ and introduce an inverse term frequency according to all expressions in the document. We define the inverse term frequency as

$$\mathrm{ITF}(t, d) := \log \frac{|d| - \mathrm{TF}(t, d) + \frac{1}{2}}{\mathrm{TF}(t, d) + \frac{1}{2}}. \tag{3.13}$$

Definition of the importance score of a formula in a document

Finally, we define the score $\mathrm{s}(t, d)$ of a term t in a document d as

$$\mathrm{s}(t, d) := \frac{(k+1)\,\mathrm{IDF}(t)\,\mathrm{ITF}(t, d)\,\mathrm{TF}(t, d)}{\max\limits_{t' \in d|_{c(t)}} \mathrm{TF}(t', d) + k\left(1 - b + \frac{b\,\mathrm{AVG}_{\mathrm{DL}}}{|d|\,\mathrm{AVG}_C}\right)}. \tag{3.14}$$

The TF-IDF ranking functions and the introduced $\mathrm{s}(t, d)$ are used to retrieve relevant documents for a given search query. However, we want to retrieve relevant subexpressions over a set of documents.

Definition of the Mathematical BM25

Thus, we define the score of a formula (mBM25) over a set of documents as the maximum score over all documents

$$\mathrm{mBM25}(t,d) := \max_{d\in D} s(t,d), \tag{3.15}$$

where D is a set of documents.

We used *Apache Flink* [157] to count the expressions and process the calculations. Thus, our implemented system scales well for large corpora.

Table 3.6 shows the top-7 scored expressions, where D is the entire zbMATH dataset. The retrieved expressions can be considered as meaningful and real-world examples of MOIs, since most expressions are known for specific mathematical concepts, such as $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, which refers to the Galois group of $\overline{\mathbb{Q}}$ over $\mathbb{Q}$, or $L^2(\mathbb{R}^2)$, which refers to the L^2-space (also known as *Lebesgue space*) over $\mathbb{R}^2$. However, a more topic-specific retrieval algorithm is desirable. To achieve this goal, we (i) retrieved a topic-specific subset of documents $D_q \subset D$ for a given textual search query q, and (ii) calculated the scores of all expressions in the retrieved documents. To generate D_q, we indexed the text sources of the documents from arXiv and zbMATH via Elasticsearch (ES)[29] and performed the pre-processing steps: filtering stop words, stemming, and ASCII-folding[30]. Table 3.5 summarizes the settings we used to retrieve MOIs from a topic-specific subset of documents D_q. We also set a minimum hit frequency according to the number of retrieved documents an expression appears in. This requirement filters out uncommon notations.

Table 3.5: Settings for the retrieval experiments.

	arXiv	zbMATH
Retrieved Doc.	40	200
Min. Hit Freq.	7	7
Min. DF	50	10
Max. DF	10k	10k

Figure 3.6 shows the results for five search queries. We asked a domain expert from the NIST to annotate the results as related (shown as green dots in Figure 3.6) or non-related (red dots). We found that the results range from good performances (e.g., for the Riemann zeta function) to bad performances (e.g., beta function). For instance, the results for the Riemann zeta function are surprisingly accurate, since we could discover that parts of Riemann's hypothesis[31] were ranked highly throughout the results (e.g., $\zeta(\frac{1}{2}+it)$). On the other hand, for the beta function, we retrieved only a few related hits, of which only one had a strong connection to the beta function $B(x,y)$. We observed that the results were quite sensitive to the chosen settings (see Table 3.5). For instance, according to the beta function, the minimum hit frequency has a strong effect on the results, since many expressions are shared among multiple documents. For arXiv, the expressions $B(\alpha,\beta)$ and $B(x,y)$ only appear in one document of the retrieved 40. However, decreasing the minimum hit frequency would increase noise in the results.

[29] https://github.com/elastic/elasticsearch [accessed 2019-09-01]. We used version 7.0.0

[30] This means that non-ASCII characters are replaced by their ASCII counterparts or will be ignored if no such counterpart exists.

[31] Riemann proposed that the real part of every non-trivial zero of the Riemann zeta function is $1/2$. If this hypothesis is correct, all the non-trivial zeros lie on the critical line consisting of the complex numbers $1/2+it$.

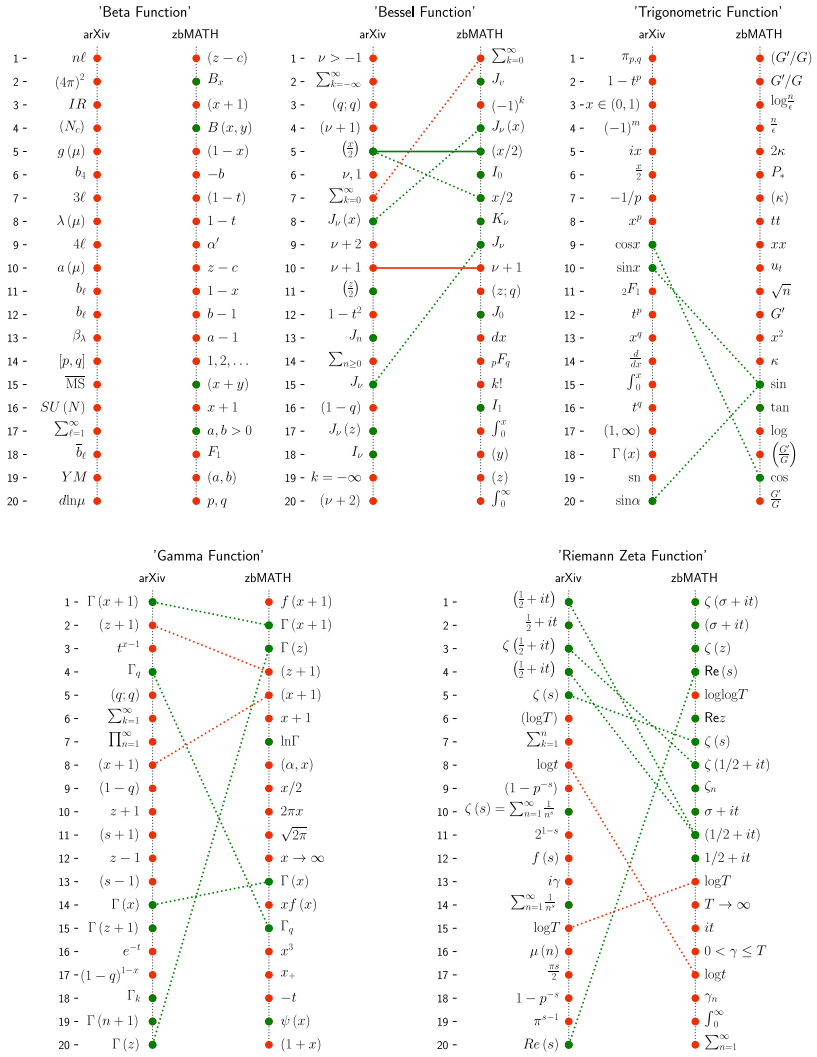


Figure 3.6: Top-20 ranked expressions retrieved from a topic-specific subset of documents D_q. The search query q is given above the plots. Retrieved formulae are annotated by a domain expert with green dots for relevant and red dots for non-relevant hits. A line is drawn if a hit appears in both result sets. The line is colored in green when the hit was marked as relevant.

Table 3.6: Top $\mathrm{s}(t, D)$ scores, where D is the set of all zbMATH documents with a minimum document frequency of 200, maximum document frequency of 500k, and a minimum complexity of 3.

C3		C4		C5	
114.84	$(n!)$	129.44	$i, j = 1, \ldots, n$	119.21	$\mathrm{Gal}\left(\overline{\mathbb{Q}}/\mathbb{Q}\right)$
108.85	ϕ^{-1}	108.52	x_{ij}	112.55	$\lvert f(z)\rvert^p$
100.19	z^{n-1}	108.50	$\dot{x} = A(t)x$	110.52	$\left(1 + \lvert x\rvert^2\right)$
100.06	(c_n)	106.66	$\lvert x - x_0\rvert$	109.19	$\lvert f(x)\rvert^p$
100.05	$B(G)$	105.52	S^{2n+1}	106.22	$\lvert\nabla u\rvert^2 dx$
99.87	$\log_2 n$	104.91	$L^2\left(\mathbb{R}^2\right)$	102.86	$n(n-1)/2$
99.65	$\xi\,(x)$	103.70	$\dot{x} = Ax + Bu$	101.40	$O(n^{-1})$

C6		C7	
110.83	$(1 + \lvert z\rvert^2)^\alpha$	98.72	$\mathrm{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
105.69	$f\left(re^{i\theta}\right)$		–
94.14	$f(z) = z + \sum_{n=2}^{\infty} a_n z^n$		–
92.33	$\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$		–
87.27	$(\log n/\log\log n)$		–
78.54	$O\left(n\log^2 n\right)$		–
	–		–

Even though we asked a domain expert to annotate the results as relevant or not, there is still plenty of room for discussion. For instance, $(x + y)$ (rank 15 in zbMATH, 'Beta Function') is the argument of the gamma function $\Gamma(x + y)$ that appears in the definition of the beta function [98, (5.12.1)] $B(x, y) := \Gamma(x)\Gamma(y)/\Gamma(x + y)$. However, this relation is weak at best, and thus might be considered as not related. Other examples are $\mathrm{Re}z$ and $\mathrm{Re}(s)$, which play a crucial role in the scenario of the Riemann hypothesis (all non-trivial zeroes have $\mathrm{Re}(s) = \frac{1}{2}$). Again, this connection is not obvious, and these expressions are often used in multiple scenarios. Thus, the domain expert did not mark the expressions as being related.

Considering the differences in the documents, it is promising to have observed a relatively high number of shared hits in the results. Further, we were able to retrieve some surprisingly good insights from the results, such as extracting the full definition of the Riemann zeta function [98, (25.2.1)] $\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}$. Even though a high number of shared hits seem to substantiate the reliability of the system, there were several aspects that affected the outcome negatively, from the exact definition of the search queries to retrieve documents via ES, to the number of retrieved documents, the minimum hit frequency, and the parameters in mBM25.

3.2.5 Applications

The presented results are beneficial for a variety of use-cases. In the following, we will demonstrate and discuss several of the applications that we propose.

Extension of zbMATH's Search Engine Formula search engines are often counterintuitive when compared to textual search, since the user must know how the system operates to enter a search query properly (e.g., does the system supports LaTeX inputs?). Additionally, mathematical concepts can be difficult to capture using only mathematical expressions. Consider, for example, someone who wants to search for mathematical expressions that are related to eigenvalues. A textual search query would only retrieve entire documents that require further investigation to find related expressions. A mathematical search engine, on the other hand, is impractical since it is not clear what would be a fitting search query (e.g., $Av = \lambda v$?). Moreover, formula and textual search systems for scientific corpora are separated from each other. Thus, a textual search engine capable of retrieving mathematical formulae can be beneficial. Also, many search engines allow for narrowing down relevant hits by suggesting filters based on the retrieved results. This technique is known as faceted search. The zbMATH search engine also provides faceted search, e.g., by authors, or year. Adding facets for mathematical expressions allows users to narrow down the results more precisely to arrive at specific documents.

Our proposed system for extracting relevant expressions from scientific corpora via mBM25 scores can be used to search for formulae even with textual search queries, and to add more filters for faceted search implementations. Table 3.7 shows two examples of such an extension for zbMATH's search engine. Searching for 'Riemann Zeta Function' and 'Eigenvalue' retrieved 4,739 and 25,248 documents from zbMATH, respectively. Table 3.7 shows the most frequently used mathematical expressions in the set of retrieved documents. It also shows the reordered formulae according to a default TF-IDF score (with normalized term frequencies) and our proposed mBM25 score. The results can be used to add filters for faceted search, e.g., show only the documents which contain $u \in W_0^{1,p}(\Omega)$. Additionally, the search system now provides more intuitive textual inputs even for retrieving mathematical formulae. The retrieved formulae are also interesting by themselves, since they provide insightful information on the retrieved publications. As already explored with our custom document search system in Figure 3.6, the Riemann hypothesis is also prominent in these retrieved documents.

The differences between TF-IDF and mBM25 ranking illustrates the problem of an extensive evaluation of our system. From a broader perspective, the hit $Ax = \lambda Bx$ is highly correlated with the input query 'Eigenvalue'. On the other hand, the raw frequencies revealed a prominent role of $\operatorname{div}(|\nabla u|^{p-2}\nabla u)$. Therefore, the top results of the mBM25 ranking can also be considered as relevant.

Math Notation Analysis A faceted search system allows us to analyze mathematical notations in more detail. For instance, we can retrieve documents from a specific time period. This allows one to study the evolution of mathematical notation over time [54], or for identifying trends in specific fields. Also, we can analyze standard notations for specific authors since it is often assumed that authors prefer a specific notation style which may vary from the standard notation in a field.

Table 3.7: The top-5 frequent mathematical expressions in the result set of zbMATH for the search queries 'Riemann Zeta Function' (top) and 'Eigenvalue' (bottom) grouped by their complexities (left) and the hits reordered according to their relevance scores (right). The TF-IDF score was calculated with normalized term frequencies.

Riemann Zeta Function

C1		C2		C3		C4	
15,051	n	4,663	(s)	1,456	$\zeta(s)$	349	$(\frac{1}{2}+it)$
11,709	s	2,460	(x)	340	$\sigma+it$	232	$(1/2+it)$
9,768	x	2,163	(n)	310	$\sum_{n=1}^{\infty}$	195	$(\sigma+it)$
8,913	k	1,485	(t)	275	$(\log T)$	136	$\frac{1}{2}+it$
8,634	T	1,415	it	264	$1/2+it$	97	$s=\sigma+it$

C5		C6		TF-IDF	mBM25
203	$\zeta(\frac{1}{2}+it)$	105	$\lvert\zeta(1/2+it)\rvert$	$\zeta(s)$	$\zeta\,(1/2+it)$
166	$\zeta(1/2+it)$	88	$\left\lvert\zeta(\frac{1}{2}+it)\right\rvert$	$\zeta(1/2+it)$	$(1/2+it)$
124	$\zeta(\sigma+it)$	81	$\lvert\zeta(\sigma+it)\rvert$	$(1/2+it)$	$(\frac{1}{2}+it)$
54	$\zeta(1+it)$	32	$\lvert\zeta(1+it)\rvert$	$\frac{1}{2}+it$	$\zeta\,(\frac{1}{2}+it)$
44	$\zeta(2n+1)$	22	$\lvert\zeta(+it)\rvert$	$(\frac{1}{2}+it)$	$(\sigma+it)$

Eigenvalue

C1		C2		C3		C4	
45,488	n	12,515	(x)	686	$-\Delta u$	218	$\lvert\nabla u\rvert^{p-2}$
43,090	x	6,598	(t)	555	$(n-1)$	218	$-\Delta_p u$
37,434	λ	4,377	λ_1	521	$\lvert\nabla u\rvert$	133	$W_0^{1,p}(\Omega)$
35,302	u	2,787	(Ω)	512	a_{ij}	127	$\lvert\nabla u\rvert^2$
22,460	t	2,725	$\mathbb{R}^n$	495	$u(x)$	97	(a_{ij})

C5		C6		TF-IDF	mBM25
139	$\lvert\nabla u\rvert^{p-2}\nabla u$	137	$\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$	$Ax=\lambda Bx$	$-\operatorname{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
68	$-d^2/dx^2$	35	$-(py')'$	$-\Delta p$	$\operatorname{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
51	$A=(a_{ij})$	26	$(\lvert u'\rvert^{p-2}u')$	$P(\lambda)$	$p=\frac{N+2}{N-2}$
46	$-\frac{d^2}{dx^2}$	18	$(\phi_p(u'))'$	λ_{k+1}	$\left(\phi_p\left(u'\right)\right)'$
45	$u\in W_0^{1,p}(\Omega)$	18	$\int_\Omega \lvert\nabla u\rvert^2\,dx$	$\lambda_1>0$	$\lambda\in(0,\lambda^*)$

Table 3.8: Suggestions to complete '$E = m$' and '$E = \{m, c\}$' (the right-hand side contains m and c) with term and document frequency based on the distributions of formulae in arXiv.

Auto-completion for '$E = m$'			Suggestions for '$E = \{m, c\}$'		
Sug. Expression	TF	DF	Sug. Expression	TF	DF
$E = mc^2$	558	376	$E = mc^2$	558	376
$E = m\cosh\theta$	23	23	$E = \gamma mc^2$	39	38
$E = mv_0$	7	7	$E = \gamma m_e c^2$	41	36
$E = m/\sqrt{1-\dot{q}^2}$	12	6	$E = m\cosh\theta$	23	23
$E = m/\sqrt{1-\beta^2}$	10	6	$E = -mc^2$	35	17
$E = mc^2\gamma$	6	6	$E = \sqrt{m^2c^4 + p^2c^2}$	10	8

Math Recommendation Systems The frequency distributions of formulae can be used to realize effective math recommendation tasks, such as type hinting or error-corrections. These approaches require long training on large datasets, but may still generate meaningless results, such as $G_i = \{(x, y) \in \mathbb{R}^n : x_i = x_i\}$ [400]. We propose a simpler system which takes advantage of our frequency distributions. We retrieve entries from our result database, which contain all unique expressions and their frequencies. We implemented a simple prototype that retrieves the entries via pattern matching. Table 3.8 shows two examples. The left side of the table shows suggested autocompleted expressions for the query '$E = m$'. The right side shows suggestions for '$E =$', where the right-hand side of the equation should contain m and c in any order. A combination using more advanced retrieval techniques, such as similarity measures based on symbol layout trees [92, 407], would enlarge the number of suggestions. This kind of autocomplete and error-correction type-hinting system would be beneficial for various use-cases, e.g., in educational software or for search engines as a pre-processing step of the input.

Plagiarism Detection Systems As previously mentioned, plagiarism detection systems would benefit from a system capable of distinguishing conventional from uncommon notations [253, 254, 334]. The approaches described by Meuschke et al. [254] outperform existing approaches by considering frequency distributions of single identifiers (expressions of complexity one). Considering that single identifiers make up only 0.03% of all unique expressions in arXiv, we presume that better performance can be achieved by considering more complex expressions. The conferred string representation also provides a simple format to embed complex expressions in existing learning algorithms.

Expressions with high complexities that are shared among multiple documents may provide further hints to investigate potential plagiarisms. For instance, the most complex expression that was shared among three documents in arXiv was Equation (3.7). A complex expression being identical in multiple documents could indicate a higher likelihood of plagiarism. Further investigation revealed that similar expressions, e.g., with infinite sums, are frequently used among a larger set of documents. Thus, the expression seems to be a part of a standard notation that is commonly shared, rather than a good candidate for plagiarism detection. Resulting from manual investigations, we could identify the equation as part of a concept called *generalized Hardy-Littlewood inequality* and Equation (3.7) appears in the three documents [24, 292, 304]. All

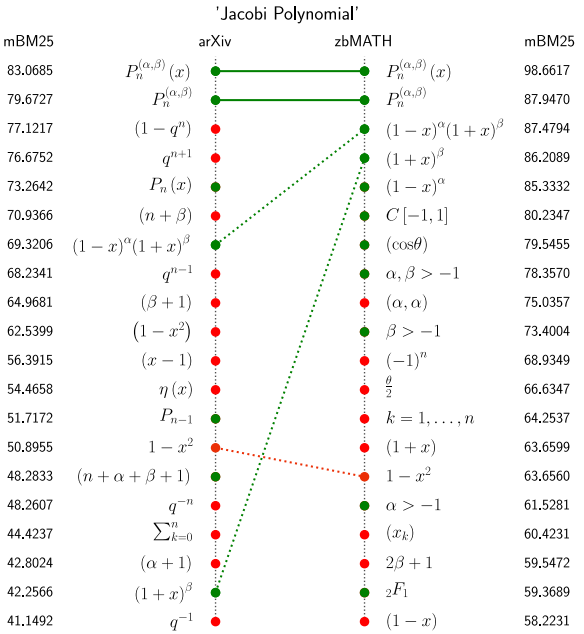


Figure 3.7: The top ranked expression for '*Jacobi polynomial*' in arXiv and zbMATH. For arXiv, 30 documents were retrieved with a minimum hit frequency of 7.

three documents shared one author in common. Thus, this case also demonstrates a correlation between complex mathematical notations and authorship.

Semantic Taggers and Extraction Systems We previously mentioned that semantic extraction systems [214, 329, 330] and semantic math taggers [71, 402] have difficulties in extracting the essential components (MOIs) from complex expressions. Considering the definition of the Jacobi polynomial in Equation (3.2), it would be beneficial to extract the groups of tokens that belong together, such as $P_n^{(\alpha,\beta)}(x)$ or $\Gamma(\alpha+m+1)$. With our proposed search engine for retrieving MOIs, we are able to facilitate semantic extraction systems and semantic math taggers. Imagine such a system being capable of identifying the term 'Jacobi polynomial' from the textual context. Figure 3.7 shows the top relevant hits for the search query 'Jacobi polynomial' retrieved from zbMATH and arXiv. The results contain several relevant and related expressions, such as the constraints $\alpha,\beta>-1$ and the weight function for the Jacobi polynomial $(1-x)^\alpha(1+x)^\beta$, which are essential properties of this orthogonal polynomial. Based on these retrieved MOIs, the extraction systems can adjust its retrieved math elements to improve precision, and semantic taggers or a tokenizer could re-organize parse trees to more closely resemble expression trees.

3.2.6 Outlook

In this first study, we preserved the core structure of the MathML data which provided insightful information for the MathML community. However, this makes it difficult to properly merge formulae. In future studies, we will normalize the MathML data via MathMLCan [117]. In addition to this normalization, we will include wildcards for investigating distributions of formula patterns rather than exact expressions. This will allow us to study connections between math objects, e.g., between $\Gamma(z)$ and $\Gamma(x+1)$. This would further improve our recommendation system and would allow for the identification of regions for parameters and variables in complex expressions.

3.3 Semantification with Textual Context Analysis

The results of our math embedding experiments and the introduction of MOI motivates us to develop a context-sensitive LaTeX to CAS translation approach around the MOI concept. In this section, we briefly discuss our novel approach to perform context-sensitive translations from LaTeX to CAS, which concludes research task **II**. We focus on three main sources of semantic information to disambiguate mathematical expressions sufficiently for such translations:

1. the inclusive structural information in the expression itself;
2. the textual context surrounding the expression; and
3. a common knowledge database.

The first source is what most existing translators rely on by concluding the semantics from a given structure. The second source is rather broad. The necessary information can be given in the sentences before and after an equation, somewhere in the same article, or even through references (e.g., hyperlinks in Wikipedia articles or citations in scientific publications). In this thesis, we will focus on the textual context in a single document, i.e., we do not analyze references or deep links to other articles yet. The last source can be considered a backup option. If we cannot retrieve information from the context of a formula, the semantic meaning of a formula might be considered common knowledge, such as π referring to the mathematical constant.

We extract knowledge from each of the three sources with different approaches. For the inclusive structural information, we rely on the semantic LaTeX macros developed by Miller [260] for the DLMF that define standard notation patterns for numerous OPSF. To analyze the textual context of a formula, we rely on the approach proposed by Schubotz et al. [330], who extracted noun phrases to enrich identifiers semantically. As a backup common knowledge database, we use the POM tagger developed by Youssef [402] that relies on manually crafted lexicon files with several common knowledge annotations for mathematical tokens.

3.3.1 Semantification, Translation & Evaluation Pipeline

Figure 3.8 illustrates the pipeline of the proposed system to convert generic LaTeX expressions to CAS. The figure contains numbered badges that represent the different steps in the system. Steps 1-4 represent the conversion pipeline, while steps 5-7 are different ways to evaluate the system.

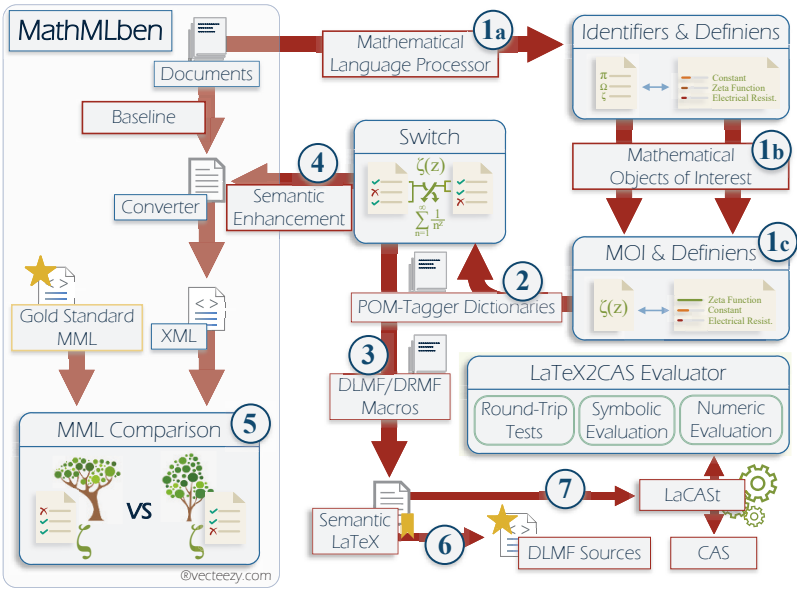


Figure 3.8: Pipeline of the proposed context-sensitive conversion process. The pipeline consists of four semantification steps (1-4) and three evaluation approaches (5-7).

The conversion pipeline starts with *mathosphere*[32] (step 1a). Mathosphere is the Java framework developed by Schubotz et al. [279, 329, 330] in a sequence of publications to semantically enrich mathematical identifier with defining phrases from the textual context. First, we will modify mathosphere so that it extracts MOI-definiens pairs rather than single identifiers (step 1b). For this purpose, we propose the following significant simplification: an isolated mathematical expression in a textual context is considered essential and informative. Hence, *isolated formulae* are defined as MOI. Moreover, mathosphere scores identifier-definiens pairs in regard of their first appearance in a document (since the first declaration of a symbol often remains valid throughout the rest of the document [394]). We adopt this scoring for MOI with a matching algorithm that allows us to identify MOI within other MOI in the same document (step 1c).

Step 2 is currently optional and combines the results from the MOI-definiens extraction process with the common knowledge database of the POM tagger. The information can then be used to feed existing LaTeX to MathML converters with additional semantic information. In Chapter 2, we created a MathML benchmark, called MathMLben, to evaluate such converters. We have also shown that, for example, LaTeXML can adopt additional semantic information via given semantic macros. Hence, via step 4 (and subsequently step 5) we can evaluate our semantification so far with the help of existing converters. The steps 2, 4, and 5 are not subject of this thesis but part of upcoming projects.

[32] https://github.com/ag-gipp/mathosphere [accessed 03-24-2020]

Besides this optional evaluation over MathMLben, we continue our main translation path. Once we extracted the MOI-definiens pairs, we replace the generic LaTeX expressions by their semantic counterparts (step (3)). We do so by indexing semantic LaTeX macros so that we can search for them by textual queries. Afterward, we are able to retrieve semantic LaTeX macros by the previously extracted definiens. Finally, we create replacement patterns so that the generic LaTeX expression can be replaced with the semantic enriched semantic macros from the DLMF. The result should be semantic LaTeX, which enables another evaluation method. Consider we perform this pipeline on the DLMF, we can compare the generated semantic LaTeX with the original, manually crafted semantic LaTeX source in the DLMF to validate its correctness (step (6)). Unfortunately, the entire pipeline focuses on the textual context. The DLMF does not provide sophisticated textual information because semantic information is available via special infoboxes, through hyperlinks, or in tables and graphs. A more comprehensive evaluation approach can be enabled by further translating the expressions to the syntax of CAS via LaCaST as we have shown in previous projects [2] (step (7)), namely symbolic and numeric evaluations. Moreover, this evaluation is most desired since it evaluates the entire proposed translation pipeline, from the semantification via mathosphere and the semantic LaTeX macros, and the final translation via LaCaST. The next chapter will aim to realize this proposed pipeline. The steps (1) and (3) are discussed in Chapter 4. The step (7) is subject of Chapter 5. Step (6) has not been realized due to the reduced amount of textual context within the DLMF. Steps (2), (4), and (5) are subject of future work.

The first time someone calls you a horse you punch him on the nose, the second time someone calls you a horse you call him a jerk but the third time someone calls you a horse, well then perhaps it's time to go shopping for a saddle.

Shlomo - *Lucky Number Slevin*

CHAPTER 4

From LaTeX to Computer Algebra Systems

Contents

This chapter addresses research tasks **III** and **IV**, i.e., implementing a system for automated semantification and translation of mathematical expressions to CAS syntax. In the previous chapter, we laid the foundation for a novel context-sensitive semantification approach that extracts the semantic information from a textual context and semantically enriches a formula with semantic LaTeX macros. In this chapter, we realize this proposed semantification approach on 104 English Wikipedia articles with 6,337 mathematical expressions. However, before we continue with this main track, we first apply a novel context-agnostic machine translation approach for translations from LaTeX to Mathematica.

Previously, we have evaluated that rule-based translators are rather limited. Mostly because the rules are carefully selected and manually crafted. This manual approach makes it difficult to estimate the level of semantics that can be concluded directly from an expression (due to its structure, notation style, or the including symbols). Finding patterns in large data is a classic task for ML solutions. Hence, we will first elaborate the effectiveness of a machine translation approach in Section 4.1. We will see that the machine translation approach is very effective in adopting the notation style generated by Mathematica's LaTeX exports but fails to generalize the

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-658-40473-4_4.

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_4

trained patterns on real world scenarios or other libraries. A qualitative evaluation on the DLMF of the same model underlines the inappropriateness of the approach for a general translator. Nonetheless, the model still outperforms Mathematica's internal LaTeX import function.

The machine translation approach presented in Section 4.1 partially contains excerpts of our[1] upcoming submission to the ACL Conference 2023.The Section 4.2 has been accepted for publication in the upcoming issue of the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) [11] journal. In order to provide a coherent story line, Section 4.2 only presents the first half of the TPAMI submission. The second half, the evaluation and discussion sections, subsequently continues in Chapter 5.

4.1 Context-Agnostic Neural Machine Translation

Mathematical formulae are generally longer compared to natural language sentences. 98% of the sentences in the Stanford Natural Language Inference (SNLI) entailment task, for example, contain less than 25 words [48]. In contrast, the average number of Mathematica tokens in the Mathematical Functions Site (MFS) dataset is 173. Short and long expressions are relatively rare but have a wider range compared to natural language sentences, e.g., 2.25% contain less than 25 tokens and 2.1% contain more than $1{,}024$ tokens. Meanwhile, a vocabulary of such a mathematical language only contains $1k$ tokens compared to $60k$ tokens for a news classification model [410][2].

The most common neural machine translation models are sequence-to-sequence recurrent neural networks [355], tree-structured recursive neural networks [136], transformer sequence-to-sequence networks [371], and convolutional sequence-to-sequence networks [130]. For natural language translation tasks, transformer networks are known to outperform the others [130, 277, 371]. In this Section, we use convolutional sequence-to-sequence networks [130] since they perform better on our mathematical language. In regard of related work, only a few approaches for mathematical language translations exists [95, 219, 275, 296, 373, 375, 376, 379].

4.1.1 Training Datasets & Preprocessing

We used two datasets for our experiments: the Mathematical Functions Site (MFS)[3] and parts of the DLMF. For the MFS, we fetched all formulea in Mathematica's `InputForm`[4] and exported every expression with Mathematica's internal `TeXForm`[5] export function. This process generated $307{,}409$ expression pairs in LaTeX and Mathematica notation. We do the same for the DLMF dataset but use LaTeXML for the conversion from semantic LaTeX to LaTeX. From the DLMF, we generated $11{,}605$ pairs in LaTeX and semantic LaTeX notation. Note that LaTeXML and Mathematica's `TeXForm` are rule-based translators. Hence, the generated data is limited to the abilities of the methods we used. Finally, we parsed the data in binary trees in postfix notation with the help of a custom rule-based tokenizer for LaTeX and Mathematica expressions.

[1] The translator is a project by Felix Petersen and was supervised by Moritz Schubotz and assisted by me. In particular, I evaluated the model on the DLMF dataset and helped to devise the final paper for publication. The section has been mostly rewritten and shortened to the main findings for avoiding conflicts.

[2] Even though more recent discussions argue that such large vocabularies are often not required and can be significantly reduced in size without a dramatic decrease in the model's performance [70].

[3] http://functions.wolfram.com/ [accessed 2021-09-20]

[4] https://reference.wolfram.com/language/ref/InputForm.html [accessed 2021-09-20]

[5] https://reference.wolfram.com/language/ref/TeXForm.html [accessed 2021-09-20]

4.1.2 Methodology

Besides our final convolutional sequence-to-sequence model [130], we also experimented with Long-Short-Term-Memory (LSTM) recurrent networks [369], recurrent, recursive, and transformer neural networks [130, 277, 371], and *LightConv* [397] as an alternative to the classic convolutional sequence-to-sequence models [130]. However, our model outperformed all other approaches. In the following, we list the hyperparameters and additional design choices that performed best for our experiments.

We use

- **Learning Rate, Gradient Clipping, Dropout, and Loss:** a learning rate of 0.25, applied gradient clipping on gradients greater than 0.1, and a dropout rate of 0.2, and a label smoothed cross-entropy for the loss;
- **State/Embedding Size(s):** a single state size of 512 tokens;
- **Number of Layers:** 11 layers;
- **Batch Size:** $48\,000$ tokens per batch (which is equivalent to a batch size of about 400 formulae); and
- **Kernel Size:** 3.

Since the MFS dataset contains more than 10^4 multi-digit numbers (in contrast to less than 10^3 non-numerical tags), these numbers cannot be interpreted as conventional tags. Thus, numbers are either split into single digits or replaced by variable tags. Splitting numbers into single digits causes significantly longer token streams, which degrades performance. Substituting all multi-digit numbers with tags like `<number_01>` improved the exact match accuracy of the validation data set from 92.7% to 95.0%. We use a total of 32 of such placeholder tags as more than 99% of the formulae have less or equal to 32 multi-digit numbers. We randomly select the tags that we substitute the numbers with. Since multi-digit numbers basically always perfectly correspond in the different mathematical languages, we directly replace the tag with their corresponding numbers after the translation. Lastly, we split the MFS dataset into 97% training data, 0.5% validation data, 2.5% test data and split the semantic LaTeX data set into 90% training data, 5% validation data, and 5% test data since this set is smaller.

4.1.3 Evaluation of the Convolutional Network

In the following, we use three evaluation metrics: Exact Matches (EM), Levenshtein distance [227], and Bilingual Evaluation Understudy (BLEU) [282]. The EM and the Levenshtein distance are calculated on the comparison of the sequence of Mathematica and TeX tokens. Hence even LaTeX equivalent expressions, such as `E=mc^2` and `E=mc^{2}`, are not considered as an EM. Due to the two additional curly brackets, the Levenshtein distance between both expressions is 2. We further denote the share of translations that have a Levenshtein distance of up to 5 by $\mathrm{LD}_{\leq 5}$, and denote the average Levenshtein Distance by LD.

The BLEU score is a quality measure that compares the machine's output to a translation by a professional human translator. It compares the n-grams (specifically $n = 1, \ldots, 4$) between the prediction and the ground truth. Since the translations in the data sets are ground truth values instead of human translations, for the back-translation of formulae, this metric reflects the closeness to the ground truth. BLEU scores range from 0 to 100, with the higher value

indicating the better result. For a comparison to natural languages, state-of-the-art translators from English to German reach a 35.0 and from English to French a 45.6 BLEU score [102]. That the BLEU scores for formula translations are significantly higher than the scores for natural language can be attributed to the larger vocabularies in natural language and a considerably higher variability between correct translations.

In addition to this, we also perform round trip experiments from LaTeX into Mathematica and back again on the `im2latex-100k`[6] dataset [95]. This dataset consists about $100k$ formulae from papers of arXiv, including their renderings. The `im2latex-100k` task's concept was the conversion of mathematical formulae from images into LaTeX via OCR. We use it as an additional source for more general mathematical expressions instead. For our round trip experiment, we translate all LaTeX expressions into Mathematica with the internal LaTeX import function and our convolutional sequence to sequence model. Afterward, we use Mathematica's export function to generate LaTeX again. Finally, we compare this round trip translated LaTeX with the original input formula. Note that 66.8% of the equations in the `im2latex-100k` data set contain tokens that are not in our model's vocabulary.

4.1.3.1 Results

Table 4.1 show the results of our convolutional sequence to sequence model for translations to Mathematica and semantic LaTeX evaluated with the EM rate and the BLEU score. We achieved an EM accuracy of 95.1% and a BLEU score of 99.68 for translations to Mathematica. For the translation from LaTeX to semantic LaTeX, we achieved an EM accuracy of 90.7% and a BLEU score of 96.79. Table 4.2 we compare our model with Mathematica's internal TeX import function on the two datasets MFS and `im2latex-100k`. While the accuracy drops on a new dataset, our model still outperforms Mathematica's import function on all metrics. Lastly, for a more qualitative analysis, we evaluated our model on 100 random samples of DLMF formulae manually, i.e., we did not check the EM or BLEU score but a human annotator manually checked if a translation was correct or at least syntactically valid (which is the same as the previously used `Import` metric). All 100 samples and the results are available in Table E.1 in Appendix E.1 available in the electronic supplementary material. Table 4.3 show the comparison of our model with Mathematica's import function and our previously developed translator LaCaST [13]. As we can see, on these random samples, Mathematica outperforms our model but LaCaST performs best. Nonetheless, LaCaST was specifically designed for translations on the DLMF, which allows LaCaST to correctly anticipate the usage of constants, such as i for the imaginary unit or e for Euler's number.

Table 4.1: Results for the backward translation.

Metric	LaTeX → Mathematica	LaTeX → semantic LaTeX
EM	95.1%	90.7%
BLEU	99.68	96.79

[6] https://paperswithcode.com/dataset/im2latex-100k [accessed 2021-09-21]

Table 4.2: Comparison between Mathematica and our model on backward translation of the formulae of the MFS and `im2latex-100k` dataset. Import denotes the fraction of formulae that can be imported by Mathematica, i.e., the translation was syntactically valid.

Dataset	Method	EM	Import	$\text{LD}_{\leq 5}$	LD
MFS	Mathematica	2.7%	88.5%	16.4%	88.7
	Conv. Seq2Seq	**95.1%**	**98.3%**	**96.7%**	**0.615**
`im2latex-100k`	Mathematica	15.3%	0.153%	2.30%	18.3
	Conv. Seq2Seq	**16.3%**	**0.698%**	**2.56%**	**12.9**

Table 4.3: Qualitative comparison between Mathematica, LaCASt, and our model on 100 random DLMF samples. ✘ indicate wrong translations. ✔ indicate correct translations. As in Table 4.2, Import denotes syntactically valid translations. The full dataset is available in Appendix E.1 available in the electronic supplementary material.

Method	Import	✔	✘
Mathematica	**71%**	11%	89%
LaCASt	57%	**22%**	**78%**
Conv. Seq2Seq	45%	5%	95%

4.1.3.2 Qualitative Analysis and Discussion

We constitute that our model successfully outperforms Mathematica on various scenarios. A good example for this is the following equation[7]:

$$\wp\left(z;g_2,g_3\right)=-\frac{\sigma\left(z-z_0;g_2,g_3\right)\sigma\left(z+z_0;g_2,g_3\right)}{\sigma\left(z;g_2,g_3\right)^2\sigma\left(z_0;g_2,g_3\right)^2}/;z_0=\wp^{-1}\left(0;g_2,g_3\right). \tag{4.1}$$

The symbol $\wp$ (`\wp`) is properly interpreted by the model and Mathematica as the Weierstrass' elliptic function $\wp$ (`WeierstrassP`). That is because the symbol $\wp$ is uniquely tied to the Weierstrass $\wp$ function. The inverse of this function, $\wp^{-1}$ is also properly interpreted by both systems as the `InverseWeierstrassP`. However, σ was not properly interpreted by Mathematica as the `WeierstrassSigma` presumably due to the ambiguity of σ. Considering the expression is from the MFS and $\wp$ appears in the same expressions, we can conclude that σ is referring to the `WeierstrassSigma`. Our model was able to capture this connection and correctly translate the entire expression.

The low scores of Mathematica on their own dataset can be attributed to the fact that Mathematica does not attempt to disambiguate its own exported expressions. As we discussed earlier, an export from a computational language to a presentation language loses semantic information. Our sequence to sequence model was able to restore the semantic information under the assumption that the input was generated from the MFS via Mathematica. Hence, our model performs very well on the trained data but is unable to produce reliable translations on

[7]Extracted from https://functions.wolfram.com/EllipticFunctions/WeierstrassP/introductions/Weierstrass/04/ [accessed 2021-09-14]

Table 4.4: Examples of our machine translations from LaTeX to Mathematica for the same expressions as in Table 1.2 from the introduction. The column **MM** refers to the results of Mathematica's import function from Table 1.2 for comparison. Circled results (✘) are syntactically invalid.

LaTeX	Rendering	Conv. Seq2Seq Translation		MM
`\int_a^b x dx`	$\int_a^b x dx$	`Integrate[x^(x,x)`	✘	✘
`\int_a^b x \mathrm{d}x`	$\int_a^b x \mathrm{d}x$	`Integrate[x^x,a]`	✘	✘
`\int_a^b x\, dx`	$\int_a^b x\, dx$	`Integrate[x^bx,x]`	✘	✔
`\int_a^b x\; dx`	$\int_a^b x\; dx$	`Integrate[x^bx,x]`	✘	✘
`\int_a^b x\, \mathrm{d}x`	$\int_a^b x\, \mathrm{d}x$	`Integrate[x^a^bx,x]`	✘	✘
`\int_a^b \frac{dx}{x}`	$\int_a^b \frac{dx}{x}$	`Integrate[(dx)/x`	✘	✘
`\sum_{n=0}^N n^2`	$\sum_{n=0}^{N} n^2$	`Sum[n^2,{n,0,N}]`	✔	✔
`\sum_{n=0}^N n^2 + n`	$\sum_{n=0}^{N} n^2 + n$	`Sum[n^2+n,{n,0,N}]+n`	✘	?
`{n \choose m}`	$\binom{n}{m}$	`JacobiSymbol[n, m]`	✘	✘
`\binom{n}{m}`	$\binom{n}{m}$	`Binomial[n, m]`	✔	✔
`\int_a^b x dt`	$\int_a^b x dt$	`Integrate[x,{t,a,b}]`	✔	
`\int_a^b x^2 dx`	$\int_a^b x^2 dx$	`Integrate[x^2,{x,a,b}]`	✔	
`\int_{a}^{b} x^2 dx`	$\int_a^b x^2 dx$	`Integrate[x^2,{a,a,b}]`	✘	

unseen, more general expressions. A first hint to this problem can be found in Table 4.3 for our evaluation on the 100 DLMF formulae. While our model clearly outperforms Mathematica on the MFS dataset, the internal rule-based import function of Mathematica works more reliable on unknown expressions. One reason for the low performance of our model on the DLMF evaluation is our vocabulary. 71 of the 100 expressions contain tokens that are not in the Mathematica-export vocabulary. Hence, our model was unable to correctly interpret these expressions. This clearly underlines the limitation of the model. As an approach to mitigate this effect in the future, we could use multilingual translations [40, 174] which would allow learning translations and tokens that are not represented in the training data for the respective language pair.

Additionally, we must note that every dataset we used has a significant bias. The DLMF and MFS specifically focus on OPSF. The `im2latex-100k` dataset was created from arXiv articles in the area of high energy physics[8]. A general limitation of neural networks is that trained models inherit biases from training data. For a successful formula translation, this means that the set of symbols, as well as the style in which the formulae are written, has to be present in the training data. Rather than learning the actual semantics of an expression, a model is able to capture the notation flavor / convention another tool produces, such as Mathematica's export function or LaTeXML. The generated LaTeX from both Mathematica and LaTeXML, is limited to a specific vocabulary and does not allow variation as it is produced by rule-based translators.

[8] *Phenomenology* (`hep-ph`) and *Theory* (`hep-th`) specifically.

Because of the limited vocabularies as well as limited set of LaTeX conventions in the data sets, the translation of mathematical LaTeX expressions of different flavors is not possible.

Due to the performance on the MFS and `im2latex-100k` datasets, we conclude that our model captures more patterns compared to Mathematica's internal import methods. On the other hand, we have also shown that our model is unable to capture the semantic information of mathematical expressions but concludes semantics from patterns and token structures. Whether this semantics is correct or consistent with additional contextual information does not matter. Hence, our translation is rather unpredictable and susceptible to minor visual changes in the inputs. If we consider the simple examples from Table 1.2 from the introduction, we can see that our model is unable to correctly translate most expressions similar to Mathematica. Table 4.4 shows the translations for our model. Three of the translations even consists obvious syntax errors, such as unbalanced brackets. In comparison to the first Table 1.2, we added three more examples to show that marginal changes may have a significant impact on the final translation. For example, simply changing the variable of integration from x to t in the first examples changes the outcome from a syntactically and semantically invalid expression to a correct and valid translation. Similarly, additional curly brackets around the limits of an integral may cause a wrong translation and an error that can be difficult to trace back if not immediately noticed[9].

Considering the simplicity of the expressions, a machine translation model alone might not be the correct approach for a reliable LaTeX to CAS translator. Especially because such simple mistakes harms the trustworthiness of the entire engine. Since accuracy and precision are among the most important aspects in mathematics, our machine translator cannot be considered as compatible with existing rule-based approaches. A hybrid solution with ML-enhanced pattern recognition techniques and rule-based translations could be the more promising solution in the future.

4.2 Context-Sensitive Translation

Since the previous section has shown that machine translations are not as reliable as rule-based approaches, we continue to develop a more reliable strategy following heuristics that have been developed over time by studying mathematical notations. Specifically, we want to focus on a more broad source of mathematical expressions away from the strict notation guidelines in the DLMF and the less descriptive scientific articles in arXiv. In the following, we will focus on Wikipedia articles as our primary source for mathematical expressions.

4.2.1 Motivation

Like many other knowledge base systems, Wikipedia encodes mathematical formulae in a representational format similar to LaTeX [156, 17, 405]. 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

[9] Here the variable of integration switched from x to a in the translated expression due to the redundant curly brackets around the limits of the integral. This error can be easily overlooked.

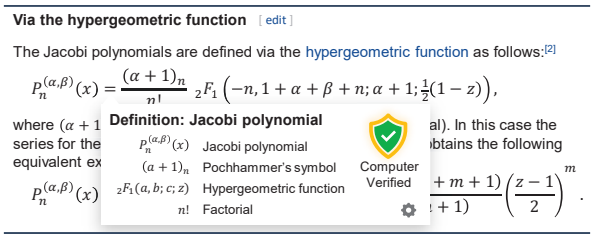


Figure 4.1: Mathematical semantic annotation in Wikipedia.

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 [144]. 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 Artificial Intelligence (AI) classification models due to the complex nature of mathematics.

In this section, we propose a semantification and translation pipeline that makes the math in Wikipedia computable via CAS. CAS, such as Maple [36] and Mathematica [393], 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 verification checks on complex mathematical equations [2, 11]. 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 [359]. Since such a translation is context-sensitive, we also propose a semantification approach for the mathematical content. This semantification uses semantic LaTeX macros [260] from the DLMF [98] and noun phrases 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 [260, 403]. Our semantification approach enables the features from the DLMF for mathematics in Wikipedia. Figure 4.1 provides an example vision of our semantic annotations and verification results in Wikipedia [17]. Head et al. [150] recently evaluated that providing readers information on the individual elements in mathematical expressions on-site [329, 394], such as shown in Figure 4.1, can significantly support users of all experience levels to read and comprehend articles more efficiently [150].

Mathematics is not a formal language. Its interpretation heavily depends on the context, e.g., $\pi(x+y)$[10] 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

[10] In the following, we use this color coding for examples to easily distinguish them from other mathematical content in this section.

reliable translation between an ambiguous, context-dependent format and an unambiguous, context-free CAS syntax. Hence, we derive the following research question:

> **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 section, 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 Figure 4.2). In addition, we perform automatic symbolic and numeric computations on the translated expressions to verify equations from Wikipedia articles [2, 11]. 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)[11].

For the evaluation of the translations, we focus on the sub-domain of OPSF. OPSF are generally well-supported by general-purpose CAS [13], 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.

In this section, we present our pipeline and discuss each of the augmentation steps. Section 4.2.2 discusses related work. In Section 4.2.3, we introduce a formal definition for translating LaTeX to CAS syntaxes. Section 4.2.4 explains necessary pre-processing steps for Wikipedia articles. Section 4.2.5 introduces our annotated dependency graph. Section 4.2.6 concludes by replacing generic LaTeX subexpressions with semantically enriched macros from the DLMF. The evaluation and discussion subsequently continue in Chapter 5.

[11] https://tpami.wmflabs.org [accessed 2021-09-01]

4.2.2 Related Work

Our proposed pipeline tangents several well-known tasks from MathIR, namely descriptive entity recognition for mathematical expressions [183, 213, 279, 320, 329], math tokenization [402], math dependency recognition [14, 214], and automatic verification [2, 11]. 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 [14, 270, 18]. Hence, existing approaches to translate LaTeX to CAS syntaxes are limited to simple arithmetic expressions [18] or require manual semantic annotations [14]. Some CAS, such as Mathematica, support LaTeX imports. Those functions fall into the first category [18] 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. [260] developed a set of the previously mentioned semantic macros that link specific mathematical expressions with definitions in the DLMF [98]. The manually generated semantic data from the DLMF [403] was successfully translated to and evaluated by CAS with our proposed framework LaCASt [2, 13]. 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 CAS via LaCASt. In this paper, we focus on the first step. The second phase is largely covered by [2, 11, 13]. A more comprehensive overview was given in Section 2.4.

4.2.3 Formal Mathematical Language Translations

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 [263], 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[12] or presentation MathML[13], and content languages $\mathcal{L}_C$, such as content MathML, OpenMath [204], or CAS syntaxes [36, 393]. Elements of these languages are often referred to as symbol layout trees for $e \in \mathcal{L}_P$ or operator trees for $e \in \mathcal{L}_C$ [92]. 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

$$X := \{f | f \in \mathcal{D} \cup \mathcal{K} \wedge (f \in \mathcal{K} \Rightarrow f \notin \mathcal{D})\}. \tag{4.2}$$

A fact f is a tuple $(\mathrm{MOI}, \mathrm{MC})$ of a Mathematical Objects of Interest (MOI) [14] 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

[12] https://www.latex-project.org/ [accessed 2021-06-29]
[13] https://www.w3.org/TR/\gls{mathml}3/ [accessed 2021-06-29]

trees, i.e., $\bar{e} \in \mathcal{L}_P \setminus \mathcal{L}_C$, to operator tree representations $\tilde{e} \in \mathcal{L}_C$. In the following, we presume that these subexpressions $\bar{e}$ are MOI so that we can derive $\tilde{e}$ from a fact $f \in X$. Then we define the semantification step as the sequence of fact-based graph transformations

$$\mathrm{t}_s(e, X) := g_{f_1} \circ \cdots \circ g_{f_n}(e), \tag{4.3}$$

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

$$\mathrm{t}_m(e) := g_{r_1} \circ \cdots \circ g_{r_n}(e). \tag{4.4}$$

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).

Definition of a Context-Sensitive Translation Function

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

$$\mathrm{t} : \mathcal{L}_P \times X \to \mathcal{L}_C \qquad \mathrm{t}(e, X) := \mathrm{t}_m(\mathrm{t}_s(e, X)). \tag{4.5}$$

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

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 [3, 13] as an implementation of $\mathrm{t}_m(e)$ between the content languages *semantic* LaTeX [403] (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 [2, 11].

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

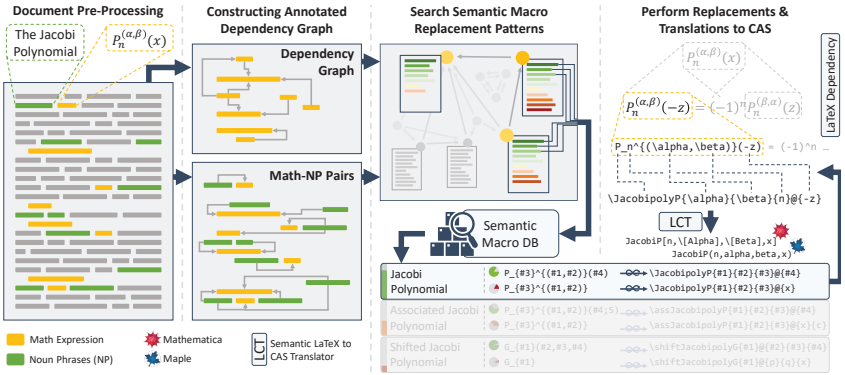


Figure 4.2: The workflow of our context-sensitive translation pipeline from LaTeX to CAS syntaxes.

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 [2, 11]. 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 Figure 4.2. 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 5.2 in Chapter 5.

4.2.3.1 Example of a Formal Translation

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

$$f = (\pi(x), \text{prime counting function}) \in \mathcal{D}. \tag{4.6}$$

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

$$r_f = \texttt{\textbackslash pi}(v_1) \rightarrow \texttt{\textbackslash nprimes@\{}v_1\texttt{\}}, \tag{4.7}$$

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. Our final pipeline will derive an entire list of ranked facts and replacement rules that are successively applied. LaCaSt defines a translation rule $r_1 \in \mathcal{R}^C_{\text{Mathematica}}$ for this function to `PrimePi[x]` and a rule $r_2 \in \mathcal{R}^C_{\text{Maple}}$ to `pi(x)` in Maple[14], respectively. Hence, the translation to Mathematica would be performed via r_1 as

$$\begin{aligned}
\mathrm{t}(\texttt{\textbackslash pi(x+y)}, X) &= \mathrm{t}_m(\mathrm{t}_s(\texttt{\textbackslash pi(x+y)}, X)) && (4.8)\\
&= g_{r_1}(g_f(\texttt{\textbackslash pi(x+y)})) && (4.9)\\
&= g_{r_1}(\texttt{\textbackslash nprimes@\{x+y\}}) && (4.10)\\
&= \texttt{PrimePi[x+y]}. && (4.11)
\end{aligned}$$

For Maple, the translation process is performed via r_2 instead

$$\begin{aligned}
\mathrm{t}(\texttt{\textbackslash pi(x+y)}, X) &= \mathrm{t}_m(\mathrm{t}_s(\texttt{\textbackslash pi(x+y)}, X)) && (4.12)\\
&= g_{r_2}(g_f(\texttt{\textbackslash pi(x+y)})) && (4.13)\\
&= g_{r_2}(\texttt{\textbackslash nprimes@\{x+y\}}) && (4.14)\\
&= \texttt{pi(x+y)}. && (4.15)
\end{aligned}$$

This underlines the modular system of our translation pipeline. Further, LaCaSt takes care of additional requirements for successful translations. In this particular example, LaCaSt informs a user about the requirement of loading the `NumberTheory` package in Maple in order to use the translated expression `pi(x+y)`. Note that the subexpression $x+y$ was not transformed by $g_f(e)$ nor by $g_{r_1}(e)$, because $x+y \in \mathcal{L}_M \cap \mathcal{L}_P$. Hence, this translation is complete and appropriate.

[14] Maple requires to pre-load the *NumberTheory* package.

4.2.4 Document Pre-Processing

For extracting the facts from a document $\mathcal{D}$, we need to identify all MOI and MC. In previous research [329], 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 [240] as our POS tagger [367, 368]. Since CoreNLP is unable to parse mathematics, we replace all math by placeholders first. In a previous project [279], 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[15]. However, the MLP approach works reasonably well in most cases.

Since Wikipedia articles are written in Wikitext, we use Sweble [99] 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 (see Appendix B available in the electronic supplementary material for more details about math formulae in Wikipedia). 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[16] 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 ϵ `< 0`, `[ϵ, {{pi}}-ϵ]`, and $0 \leq \phi \leq 4${{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 $0 \leq \phi \leq 4${{pi}} is replaced by `0 \leq \phi \leq 4\pi` 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 [123, 377].

4.2.5 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 [214], there are multiple valid options to construct a dependency graph. We decided to use the POM tagger [402] to generate parse trees from LaTeX expressions

[15] 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 4.2.2.

[16] https://en.wikipedia.org/wiki/Jacobi_polynomials [accessed 2021-06-07]

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 [3, 13]. 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 [18]. 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 Appendix F.2 available in the electronic supplementary material and on our demo page.

In addition to the new concept for addressing math stems, we also changed our approach for definition detection. Previously [214], 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 [111, 134, 183, 370] but not a single formula. We will elaborate more on this matter in Section 5.2.3. 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 [213] and later applied distance metrics [279, 329, 330] between single identifiers and textual descriptions. We were able to reach an F1 score of .36 for annotating single identifiers with textual descrip-

tions. Since we are working on more complex, less overloaded [14], MOI expressions now, we can presume an improvement if we apply the same approach again. Hence, we used our latest improvements [330] and applied some changes to annotate MOI rather than single identifiers with textual descriptions from the surrounding context. Originally, we considered only nouns, noun sequences, adjectives followed by nouns, and Wikipedia links as candidates of definiens (now MC) [329]. However, in the field of OPSF, such descriptions are generally insufficient. Hence, we include connective possessive endings and prepositions between noun phrases (see Appendix F.1 available in the electronic supplementary material for further details).

Originally [329], 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 [213, 214, 279, 330]. We denote the final score of a fact f, i.e., of an MOI and MC pair, with $\mathrm{s}_{\mathrm{MLP}}(\mathrm{MOI}, \mathrm{MC})$.

4.2.6 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 Figure 4.1, 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 [260, 403]. 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-

Table 4.5: Mappings and likelihoods for the semantic LaTeX macro of the general hypergeometric function in the DLMF.

Prob.	Semantic Macro	LaTeX	Rendered
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)`	${}_2F_1\left(\begin{smallmatrix}a,b\\c\end{smallmatrix};z\right)$
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}`	${}_2F_1$

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 4.5 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 4.5 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 $a_1, \dots, a_p$, $b_1, \dots, b_q$, and z for the variables. Hence, the last version in Table 4.5 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 $\mathrm{s}_{\mathrm{MLP}}(f)$. 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 $\mathrm{s}_{\mathrm{MLP}}(f)$ that ranks the pair of MOI and description MC. Second, we index all DLMF replacement patterns in an Elasticsearch (ES)[17] database to search for a semantic macro for a given description. ES uses the BM25 score to retrieve relevant semantic 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: $\mathrm{s}_{\mathrm{ES}}(f)$. Lastly, every semantic macro $\widetilde{m}$ has multiple rendered forms, of which some are more frequently used than others in the DLMF, see the

[17] https://github.com/elastic/elasticsearch [accessed 2021-01-01]

probability in Table 4.5. 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 $\mathrm{s}_{\mathrm{DLMF}}(r_f)$. The ranking for a fact $\mathrm{rk}(f)$ is simply the average over the three components $\mathrm{s}_{\mathrm{MLP}}(f)$, $\mathrm{s}_{\mathrm{ES}}(f)$, and $\mathrm{s}_{\mathrm{DLMF}}(r_f)$.

4.2.6.1 Common Knowledge Pattern Recognition

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[18] followed by a letter after integrals to replace it with `\diff{.}` (see [11] 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.

[18] Note the difference between normal d and the roman typestyle d.

It is possible to commit no mistakes and still lose.
That is not a weakness, that is life.

Jean-Luc Picard - *Star Trek: The Next Generation*

CHAPTER 5

Qualitative and Quantitative Evaluations

Contents

This chapter primarily contributes to the research task **V**, i.e., evaluating the effectiveness of the semantification and translation system LACAST. In Section 5.1, we also extend LACAST semantic LaTeX translations to support more mathematical operators, including sums, products, integrals, and limit notations. Hence, this chapter secondarily also contributes to research task **IV**, i.e.,

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-658-40473-4_5.

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_5

implementing an extension of the semantification approach to provide translations to CAS. We evaluate LaCaST on two different datasets: the DLMF and Wikipedia.

First, we evaluate LaCaST on the DLMF to estimate the capabilities and limitations of our rule-based translator on a semantic enhanced dataset. Translating formulae from the DLMF to CAS can be considered simpler primarily for three reasons. First, the formulae are manually enhanced and can be considered unambiguous in most cases. Second, the constraints of formulae are directly attached to equations and therefore accessible to LaCaST. Lastly, parts of equations in the DLMF are linked to their definitions which allow to resolve substitutions and fetch additional constraints. This meta information is either not available or given in the surrounding context in Wikipedia articles which greatly harms the accessibility of this crucial data. Hence, we presume that we achieve the best possible translations via LaCaST on the DLMF. For evaluating the capabilities of LaCaST, we perform numeric and symbolic evaluation techniques to evaluate a translation [3, 13]. We will further use these evaluation approaches to identify flaws in the DLMF and CAS computations.

Next, we evaluate LaCaST on Wikipedia as the direct successor of the previous Chapter 4. Here, we use the full and final version of LaCaST, including every improvement that has been discussed throughout the thesis. Specifically, it actively uses all common knowledge pattern recognition techniques discussed in Section 4.2.6.1, all heuristics for detecting math operators introduced in Section 5.1.2, and the enhanced symbolic and numeric evaluation pipeline first outlined in [3] and finally elaborated in Section 5.1.3. In combination with the automatic evaluation, we are able to perform plausibility checks on complex mathematical formulae in Wikipedia.

This chapter is split in two parts following two main motivations behind them. In Section 5.1, we elaborate the possibility to use LaCaST translations to automatically verify entire DML and CAS with one another. We specifically focus on the DLMF for our DML and Mathematica and Maple for our general-purpose CAS. In Section 5.2, we use the final context-sensitive version of LaCaST introduced in Chapter 4, including every improvement introduced in the first Section 5.1 of this chapter, with the goal to verify equations in Wikipedia articles. This chapter finalizes the improvements of LaCaST for semantic LaTeX expressions (Section 5.1) and general LaTeX expressions (Section 5.2).

The content of Section 5.1 was published at the TACAS conference [8]. Some parts in Section 5.2 have also been previously published at the CICM conference [2]. Section 5.2, as the direct successor of Chapter 4, is part of the aforementioned submission to the TPAMI journal [11].

5.1 Evaluations on the Digital Library of Mathematical Functions

Digital Mathematical Library (DML) gather the knowledge and results from thousands of years of mathematical research. Even though pure and applied mathematics are precise disciplines, gathering their knowledge bases over many years results in issues which every digital library shares: consistency, completeness, and accuracy. Likewise, CAS[1] play a crucial role in the modern era for pure and applied mathematics, and those fields which rely on them. CAS can be used to simplify, manipulate, compute, and visualize mathematical expressions. Accordingly,

[1] In the sequel, the acronyms CAS and DML are used, depending on the context, interchangeably with their plurals.

modern research regularly uses DML and CAS together. Nonetheless, DML [2, 10] and CAS [20, 100, 180] are not exempt from having bugs or errors. Durán et al. [100] even raised the rather dramatic question: "*can we trust in [CAS]*?"

Existing comprehensive DML, such as the DLMF [98], are consistently updated and frequently corrected with errata[2]. Although each chapter of the DLMF and its print analog *The NIST Handbook of Mathematical Functions* [276] has been carefully written, edited, validated, and proofread over many years, errors still remain. Maintaining a DML, such as the DLMF, is a laborious process. Likewise, CAS are eminently complex systems, and in the case of commercial products, often similar to black boxes in which the magic (i.e., the computations) happens in opaque private code [100]. CAS, especially commercial products, are often exclusively tested internally during development.

An independent examination process can improve testing and increase trust in the systems and libraries. Hence, we want to elaborate on the following research question.

Research Question

How can digital mathematical libraries and computer algebra systems be utilized to improve and verify one another?

Our initial approach for answering this question is inspired by Cohl et al. [2]. In order to verify a translation tool from a specific LaTeX dialect to Maple , they perform symbolic and numeric evaluations on equations from the DLMF. This approach presumes that a proven equation in a DML must be also valid in a CAS. In turn, a disparity in between the DML and CAS would lead to an issue in the translation process. However, assuming a correct translation, a disparity would also indicate an issue either in the DML source or the CAS implementation. In turn, we can take advantage of the same approach proposed by Cohl et al. [2] to improve and even verify DML with CAS and vice versa. Unfortunately, previous efforts to translate mathematical expressions from various formats, such as LaTeX [3, 10], MathML [18], or OpenMath [152], to CAS syntax show that the translation will be the most critical part of this verification approach.

In this section, we elaborate on the feasibility and limitations of the translation approach from DML to CAS as a possible answer to our research question. We further focus on the DLMF as our DML and the two general-purpose CAS Maple and Mathematica for this first study. This relatively sharp limitation is necessary in order to analyze the capabilities of the underlying approach to verify commercial CAS and large DML. The DLMF uses semantic macros internally in order to disambiguate mathematical expressions [260, 403]. These macros help to mitigate the open issue of retrieving sufficient semantic information from a context to perform translations to formal languages [10, 18]. Further, the DLMF and general-purpose CAS have a relatively large overlap in coverage of special functions and orthogonal polynomials. Since many of those functions play a crucial role in a large variety of different research fields, we focus in this first study mainly on these functions.

In particular, we extend the first version of LACAST [3] to increase the number of translatable functions in the DLMF significantly. Current extensions include a new handling of constraints, the support for the mathematical operators: sum, product, limit, and integral, as well as overcoming

[2] https://dlmf.nist.gov/errata/ [accessed 2021-05-01]

semantic hurdles associated with Lagrange (prime) notations commonly used for differentiation. Further, we extend its support to include Mathematica using the freely available WED[3] (hereafter, with Mathematica, we refer to the WED). These improvements allow us to cover a larger portion of the DLMF, increase the reliability of the translations via LaCaSt, and allow for comparisons between two major general-purpose CAS for the first time, namely Maple and Mathematica. Finally, we provide open access to all the results contained within this paper[4].

The section is structured as follows. Section 5.1.1 explains the data in the DLMF. Section 5.1.2 focus on the improvements of LaCaSt that had been made to make the translation as comprehensive and reliable as possible for the upcoming evaluation. Section 5.1.3 explains the symbolic and numeric evaluation pipeline. We will provide an in-depth discussion of that process in Section 5.1.3. Subsequently, we analyze the results in Section 5.1.4. Finally, we conclude the findings and provide an outlook for upcoming projects in Section 5.1.5.

Related Work Existing verification techniques for CAS often focus on specific subroutines or functions [45, 58, 107, 148, 180, 185, 225, 228], such as a specific theorems [218], differential equations [153], or the implementation of the `math.h` library [224]. Most common are verification approaches that rely on intermediate verification languages [45, 148, 153, 180, 185], such as *Boogie* [29, 225] or *Why3* [41, 185], which, in turn, rely on proof assistants and theorem provers, such as *Coq* [37, 45], *Isabelle* [153, 167], or *HOL Light* [146, 148, 180]. Kaliszyk and Wiedijk [180] proposed on entire new CAS which is built on top of the proof assistant HOL Light so that each simplification step can be proven by the underlying architecture. Lewis and Wester [228] manually compared the symbolic computations on polynomials and matrices with seven CAS. Aguirregabiria et al. [20] suggested to teach students the known traps and difficulties with evaluations in CAS instead to reduce the overreliance on computational solutions.

We [2] developed the aforementioned translation tool LaCaSt, which translates expressions from a semantically enhanced LaCaSt dialect to Maple. By evaluating the performance and accuracy of the translations, we were able to discover a sign-error in one the DLMF's equations [2]. While the evaluation was not intended to verify the DLMF, the translations by the rule-based translator LaCaSt provided sufficient robustness to identify issues in the underlying library. To the best of our knowledge, besides this related evaluation via LaCaSt, there are no existing libraries or tools that allow for automatic verification of DML.

5.1.1 The DLMF dataset

In the modern era, most mathematical texts (handbooks, journal publications, magazines, monographs, treatises, proceedings, etc.) are written using the document preparation system LaTeX. However, the focus of LaTeX is for precise control of the rendering mechanics rather than for a semantic description of its content. In contrast, CAS syntax is coercively unambiguous in order to interpret the input correctly. Hence, a transformation tool from DML to CAS must disambiguate mathematical expressions. While there is an ongoing effort towards such a process [14, 214, 329, 402, 408], there is no reliable tool available to disambiguate mathematics sufficiently to date.

[3] https://www.wolfram.com/engine/ [accessed 2021-05-01]
[4] https://lacast.wmflabs.org/ [accessed 2021-10-01]

The DLMF contains numerous relations between functions and many other properties. It is written in LaTeX but uses specific semantic macros when applicable [403]. These semantic macros represent a unique function or polynomial defined in the DLMF. Hence, the semantic LaTeX used in the DLMF is often unambiguous. For a successful evaluation via CAS, we also need to utilize all requirements of an equation, such as constraints, domains, or substitutions. The DLMF provides this additional data too and generally in a machine-readable form [403]. This data is accessible via the i-boxes (information boxes next to an equation marked with the icon ⓘ). If the information is not given in the attached i-box or the information is incorrect, the translation via LaCASt would fail. The i-boxes, however, do not contain information about branch cuts (see Section 5.1.4.1) or constraints. Constraints are accessible if they are directly attached to an equation. If they appear in the text (or even a title), LaCASt cannot utilize them. The test dataset, we are using, was generated from DLMF Version 1.0.28 (2020-09-15) and contained $9{,}977$ formulae with $1{,}505$ defined symbols, $50{,}590$ used symbols, $2{,}691$ constraints, and $2{,}443$ warnings for non-semantic expressions, i.e., expressions without semantic macros [403]. Note that the DLMF does not provide access to the underlying LaTeX source. Therefore, we added the source of every equation to our result dataset.

5.1.2 Semantic LaTeX to CAS translation

The aforementioned translator LaCASt was first developed by Greiner-Petter et al. [3, 10]. They reported a coverage of 53.6% translations [3] for a manually selected part of the DLMF to the CAS Maple. Afterward, they extended LaCASt to perform symbolic and numeric evaluations on the entire DLMF and reported a coverage of 58.8% translations [2]. This version of LaCASt serves as a baseline for our improvements. They reported a success rate of $\sim 16\%$ for symbolic and $\sim 12\%$ for numeric verifications.

Evaluating the baseline on the entire DLMF result in a coverage of only 31.6%. Hence, we first want to increase the coverage of LaCASt on the DLMF. To achieve this goal, we first increasing the number of translatable semantic macros by manually defining more translation patterns for special functions and orthogonal polynomials. For Maple, we increased the number from 201 to 261. For Mathematica, we define 279 new translation patterns which enables LaCASt to perform translations to Mathematica. Even though the DLMF uses 675 distinguished semantic macros, we cover $\sim 70\%$ of all DLMF equations with our extended list of translation patterns (see Zipf's law for mathematical notations [14]). In addition, we implemented rules for translations that are applicable in the context of the DLMF, e.g., ignore ellipsis following floating-point values or `\choose` always refers to a binomial expression. Finally, we tackle the remaining issues outlined by Cohl et al. [2] which can be categorized into three groups: (i) expressions of which the arguments of operators are not clear, namely sums, products, integrals, and limits; (ii) expressions with prime symbols indicating differentiation; and (iii) expressions that contain ellipsis. While we solve some of the cases in Group (iii) by ignoring ellipsis following floating-point values, most of these cases remain unresolved.

In the following, we first introduce the constraint handling via blueprints[5]. Next, we elaborate our solutions for (i) in Section 5.1.2.2 and (ii) in Section 5.1.2.3.

[5]This subsection 5.1.2.1 was previously published by Cohl et al. [2].

5.1.2.1 Constraint Handling

Correct assumptions about variable domains are essential for CAS systems, and not surprisingly lead to significant improvements in the CAS ability to simplify. The DLMF provides constraint (variable domain) metadata for formulae, and we have extracted this formula metadata. We have incorporated these constraints as assumptions for the simplification process (see Section 5.1.3.1). Note however, that a direct translation of the constraint metadata is usually not sufficient for a CAS to be able to understand it. Furthermore, testing invalid values for numerical tests returns incorrect results (see Section 5.1.3.2).

For instance different symbols must be interpreted differently depending on the usage. One must be able to interpret correctly certain notations of this kind. For instance, one must be able to interpret the command `a,b\in A`, which indicates that both variables `a` and `b` are elements of the set `A` (or more generally `a_1,\dots,a_n\in A`). Similar conventions are often used for variables being elements of other sets such as the sets of rational, real or complex numbers, or for subsets of those sets.

Also, one must be able to interpret the constraints as variables in sets defined using an equals notation such as `n=0,1,2,\dots`, which indicates that the variable `n` is a integer greater than or equal to zero, or together `n,m=0,1,2,\dots`, both the variables `n` and `m` are elements of this set. Since mathematicians who write LaTeX are often casual about expressions such as these, one should know that `0,1,2,\dots` is the same as `0,1,\dots`. Consistently, one must also be able to correctly interpret infinite sets (represented as strings) such as `=1,2,\dots`, `=1,2,3,\dots`, `=-1,0,1,2,\dots`, `=0,2,4,\dots`, or even `=3,7,11,\dots`, or `=5,9,13,\dots`. One must be able to interpret finite sets such as `=1,2`, `=1,2,3`, or `=1,2,\dots,N`.

An entire language of translation of mathematical notation must be understood in order for CAS to be able to understand constraints. In mathematics, the syntax of constraints is often very compact and contains textual explanations. Translating constraints from LaTeX to CAS is a compact task because CAS only allow precise and strict syntax formats. For example, the typical constraint $0 < x < 1$ is invalid if directly translated to Maple, because it would need to be translated to two separate constraints, namely $x > 0$ and $x < 1$.

We have improved the handling and translation of variable constraints/assumptions for simplification and numerical evaluation. Adding assumptions about the constrained variables improves the effectiveness of Maple's `simplify` function. Our previous approach for constraint handling for numerical tests was to extract a pre-defined set of test values and to filter invalid values according to the constraints. Because of this strategy, there often was no longer any valid values remaining after the filtering. To overcome this issue, instead, we chose a single numerical value for a variable that appears in a pre-defined constraint. For example, if a test case contains the constraint $0 < x < 1$, we chose $x = \frac{1}{2}$.

A naive approach for this strategy, is to apply regular expressions to identify a match between a constraint and a rule. However, we believed that this approach does not scale well when it comes to more and more pre-defined rules and more complex constraints. Hence, we used the POM-tagger to create blueprints of the parse trees for pre-defined rules. For the example LaTeX constraint `$0 < x < 1$`, rendered as $0 < x < 1$, our textual rule is given by

```
0 < var < 1 ==> 1/2.
```

The parse tree for this blueprint constraint contains five tokens, where `var` is an alphanumerical token that is considered to be a placeholder for a variable.

We can also distinguish multiple variables by adding an index to the placeholder. For example, the rule we generated for the mathematical LaTeX constraint `$x,y \in \Real$`, where `\Real` is the semantic macro which represents the set of real numbers, and rendered as $x, y \in \mathbb{R}$, is given by

```
var1, var2 \in \Real ==> 3/2,3/2.
```

A constraint will match one of the blueprints if the number, the ordering, and the type of the tokens are equal. Allowed matching tokens for the variable placeholders are Latin or Greek letters and alphanumerical tokens.

5.1.2.2 Parse sums, products, integrals, and limits

Here we consider common notations for the sum, product, integral, and limit operators. For these operators, one may consider Mathematically Essential Operator Metadata (MEOM). For all these operators, the MEOM includes *argument(s)* and *bound variable(s)*. The operators act on the arguments, which are themselves functions of the bound variable(s). For sums and products, the bound variables are referred to as *indices*. The bound variables for integrals[6] are called *integration variables*. For limits, the bound variables are continuous variables (for limits of continuous functions) and indices (for limits of sequences). For integrals, MEOM include precise descriptions of regions of integration (e.g., piecewise continuous paths/intervals/regions). For limits, MEOM include limit points (e.g., points in $\mathbb{R}^n$ or $\mathbb{C}^n$ for $n \in \mathbb{N}$), as well as information related to whether the limit to the limit point is independent or dependent on the direction in which the limit is taken (e.g., one-sided limits).

For a translation of mathematical expressions involving the LaTeX commands `\sum`, `\int`, `\prod`, and `\lim`, we must extract the MEOM. This is achieved by (a) determining the argument of the operator and (b) parsing corresponding subscripts, superscripts, and arguments. For integrals, the MEOM may be complicated, but certainly contains the argument (function which will be integrated), bound (integration) variable(s) and details related to the region of integration. Bound variable extraction is usually straightforward since it is usually contained within a differential expression (infinitesimal, pushforward, differential 1-form, exterior derivative, measure, etc.), e.g., $\mathrm{d}x$. Argument extraction is less straightforward since even though differential expressions are often given at the end of the argument, sometimes the differential expression appears in the numerator of a fraction (e.g., $\int \frac{f(x)\mathrm{d}x}{g(x)}$). In which case, the argument is everything to the right of the `\int` (neglecting its subscripts and superscripts) up to and including the fraction involving the differential expression (which may be replaced with 1). In cases where the differential expression is fully to the right of the argument, then it is a *termination symbol*. Note that some scientists use an alternate notation for integrals where the differential expression appears immediately to the right of the integral, e.g., $\int \mathrm{d}x f(x)$. However, this notation does not appear in the DLMF. If such notations are encountered, we follow the same approach that we used for sums, products, and limits (see Section 5.1.2.2).

[6] The notion of integrals includes: antiderivatives (indefinite integrals), definite integrals, contour integrals, multiple (surface, volume, etc.) integrals, Riemannian volume integrals, Riemann integrals, Lebesgue integrals, Cauchy principal value integrals, etc.

Extraction of variables and corresponding MEOM The subscripts and superscripts of sums, products, limits, and integrals may be different for different notations and are therefore challenging to parse. For integrals, we extract the bound (integration) variable from the differential expression. For sums and products, the upper and lower bounds may appear in the subscript or superscript. Parsing subscripts is comparable with the problem of parsing constraints [2] (which are often not consistently formulated). We overcame this complexity by manually defining patterns of common constraints and refer to them as blueprints (see Section 5.1.2.1). This blueprint pattern approach allows LaCaST to identify the MEOM in the sub- and superscripts.

For our MEOM blueprints, we define three placeholders: `varN` for single identifiers or a list of identifiers (delimited by commas), `numL1`, and `numU1`, representing lower and upper bound expressions, respectively. In addition, for sums and products, we need to distinguish between including and excluding boundaries, e.g., $1 < k$ and $1 \leq k$. An excluding relation, such as $0 < k < 10$, must be interpreted as a sum from 1 to 9. Table 5.1 shows the final set of sum/product subscript blueprints.

Standard notations may not explicitly show infinity boundaries. Hence, we set the default boundaries to infinity. For limit expressions we need different blueprints to capture the limit direction. We cover the standard notations with '`var1 \to numL*`', where `*` is either `+`, `-`, `^+`, `^-` or absent and the different arrow-notations where `\to` can be either `\downarrow`, `\uparrow`, `\searrow`, or `\nearrow`, specifying one-sided limits. Note that the arrow-notation (besides `\to`) is not used in the DLMF and thus, has no effect on the performance of LaCaST in our evaluation. Note further that, while the blueprint approach is very flexible, it cannot handle every possible scenario, such as the divisor sum $\sum_{(p-1)|2n} 1/p$ [98, (24.10.1)]. Proper translations of such complex cases may even require symbolic manipulation, which is currently beyond the capabilities of LaCaST.

Table 5.1: The table contains examples of the blueprints for subscripts of sums/products including an example expression that matches the blueprint.

Blueprints	Example
`numL1 \leq var1 < var2 \leq numU1`	$0 \leq n < k \leq 10$
`-\infty < varN < \infty`	$-\infty < n < \infty$
`numL1 < varN < numU1`	$0 < n, k < 10$
`numL1 \leq varN < numU1`	$0 \leq k < 10$
`numL1 < varN \leq numU1`	$0 < n, k \leq 10$
`varN \leq numU1`	$n, k \leq N + 5$
`varN \in numL1`	$n \in \{1, 2, 3\}$
`varN = numL1`	$n, k, l = 1$

Identification of operator arguments Once we have extracted the bound variable for sums, products, and limits, we need to determine the end of the argument. We analyzed all sums in the DLMF and developed a heuristic that covers all the formulae in the DLMF and potentially a large portion of general mathematics. Let x be the extracted bound variable. For

sums, we consider a summand as a part of the argument if (I) it is the very first summand after the operation; or (II) x is an element of the current summand; or (III) x is an element of the following summand (subsequent to the current summand) and there is no termination symbol between the current summand and the summand which contains x with an equal or lower depth according to the parse tree (i.e., closer to the root). We consider a summand as a single logical construct since addition and subtraction are granted a lower operator precedence than multiplication in mathematical expressions. Similarly, parentheses are granted higher precedence and, thus, a sequence wrapped in parentheses is part of the argument if it obeys the rules (I-III). Summands, and such sequences, are always entirely part of sums, products, and limits or entirely not.

A termination symbol always marks the end of the argument list. Termination symbols are relation symbols, e.g., $=$, $\neq$, $\leq$, closing parentheses or brackets, e.g., $)$, $]$, or $>$, and other operators with MEOMs, if and only if, they define the same bound variable. If x is part of a subsequent operation, then the following operator is considered as part of the argument (as in (II)). However, a special condition for termination symbols is that it is only a termination symbol for the current chain of arguments. Consider a sum over a fraction of sums. In that case, we may reach a termination symbol within the fraction. However, the termination symbol would be deeper inside the parse tree as compared to the current list of arguments. Hence, we used the depth to determine if a termination symbol should be recognized or not. Consider an unusual notation with the binomial coefficient as an example

$$\sum_{k=0}^{n}\binom{n}{k} = \sum_{k=0}^{n}\frac{\prod_{m=1}^{n} m}{\prod_{m=1}^{k} m \prod_{m=1}^{n-k} m}. \tag{5.1}$$

This equation contains two termination symbols, marked red and green. The red termination symbol $=$ is obviously for the first sum on the left-hand side of the equation. The green termination symbol $\prod$ terminates the product to the left because both products run over the same bound variable m. In addition, none of the other $=$ signs are termination symbols for the sum on the right-hand side of the equation because they are deeper in the parse tree and thus do not terminate the sum.

Note that `varN` in the blueprints also matches multiple bound variable, e.g., $\sum_{m,k\in A}$. In such cases, x from above is a list of bound variables and a summand is part of the argument if one of the elements of x is within this summand. Due to the translation, the operation will be split into two preceding operations, i.e., $\sum_{m,k\in A}$ becomes $\sum_{m\in A}\sum_{k\in A}$. Figure 5.1 shows the extracted arguments for some example sums. The same rules apply for extraction of arguments for products and limits.

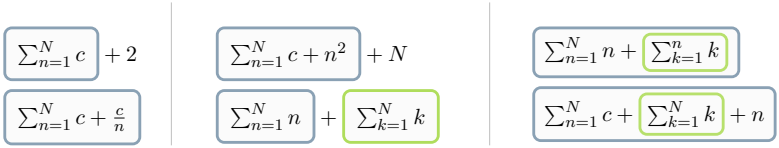


Figure 5.1: Example argument identifications for sums.

5.1.2.3 Lagrange's notation for differentiation and derivatives

Another remaining issue is the Lagrange (prime) notation for differentiation, since it does not outwardly provide sufficient semantic information. This notation presents two challenges. First, we do not know with respect to which variable the differentiation should be performed. Consider for example the Hurwitz zeta function $\zeta(s,a)$ [98, §25.11]. In the case of a differentiation $\zeta'(s,a)$, it is not clear if the function should be differentiated with respect to s or a. To remedy this issue, we analyzed all formulae in the DLMF which use prime notations and determined which variables (slots) for which functions represent the variables of the differentiation. Based on our analysis, we extended the translation patterns by meta information for semantic macros according to the slot of differentiation. For instance, in the case of the Hurwitz zeta function, the first slot is the slot for prime differentiation, i.e., $\zeta'(s,a) = \frac{\mathrm{d}}{\mathrm{d}s}\zeta(s,a)$. The identified variables of differentiations for the special functions in the DLMF can be considered to be the standard slots of differentiations, e.g., in other DML, $\zeta'(s,a)$ most likely refers to $\frac{\mathrm{d}}{\mathrm{d}s}\zeta(s,a)$.

The second challenge occurs if the slot of differentiation contains complex expressions rather than single symbols, e.g., $\zeta'(s^2,a)$. In this case, $\zeta'(s^2,a) = \frac{\mathrm{d}}{\mathrm{d}(s^2)}\zeta(s^2,a)$ instead of $\frac{\mathrm{d}}{\mathrm{d}s}\zeta(s^2,a)$. Since CAS often do not support derivatives with respect to complex expressions, we use the inbuilt substitution functions[7] in the CAS to overcome this issue. To do so, we use a temporary variable `temp` for the substitution. CAS perform substitutions from the inside to the outside. Hence, we can use the same temporary variable `temp` even for nested substitutions. Table 5.2 shows the translation performed for $\zeta'(s^2,a)$. CAS may provide optional arguments to calculate the derivatives for certain special functions, e.g., `Zeta(n,z,a)` in Maple for the n-th derivative of the Hurwitz zeta function. However, this shorthand notation is generally not supported (e.g., Mathematica does not define such an optional parameter). Our substitution approach is more lengthy but also more reliable. Unfortunately, lengthy expressions generally harm the performance of CAS, especially for symbolic manipulations. Hence, we have a genuine interest in keeping translations short, straightforward and readable. Thus, the substitution translation pattern is only triggered if the variable of differentiation is not a single identifier. Note that this substitution only triggers on semantic macros. Generic functions, including prime notations, are still skipped.

Table 5.2: Example translations for the prime derivative of the Hurwitz zeta function with respect to s^2.

System	$\zeta'(s^2,a)$
DLMF	`\Hurwitzzeta'@{s^2}{a}`
Maple	`subs(temp=(s)^(2),` `  diff(Zeta(0,temp,a),temp$(1)))`
Mathematica	`D[HurwitzZeta[temp,a],{temp,1}]` `  /.temp->(s)^(2)`

A related problem to MEOM of sums, products, integrals, limits, and differentiations are the notations of derivatives. The semantic macro for derivatives `\deriv{w}{x}` (rendered as $\frac{\mathrm{d}w}{\mathrm{d}x}$) is

[7] Note that Maple also support an evaluation substitution via the two-argument `eval` function. Since our substitution only triggers on semantic macros, we only use `subs` if the function is defined in Maple. In turn, as far as we know, there is no practical difference between `subs` and the two-argument `eval` in our case.

often used with an empty first argument to render the function behind the derivative notation, e.g., `\deriv{}{x}\sin@{x}` for $\frac{\mathrm{d}}{\mathrm{d}x}\sin x$. This leads to the same problem we faced above for identifying MEOMs. In this case, we use the same heuristic as we did for sums, products, and limits. Note that derivatives may be written following the function argument, e.g., $\sin(x)\frac{\mathrm{d}}{\mathrm{d}x}$. If we are unable to identify any following summand that contains the variable of differentiation before we reach a termination symbol, we look for arguments prior to the derivative according to the heuristic (I-III).

Wronskians With the support of prime differentiation described above, we are also able to translate the Wronskian [98, (1.13.4)] to Maple and Mathematica. A translation requires one to identify the variable of differentiation from the elements of the Wronskian, e.g., z for $\mathscr{W}\{\mathrm{Ai}(z), \mathrm{Bi}(z)\}$ from [98, (9.2.7)]. We analyzed all Wronskians in the DLMF and discovered that most Wronskians have a special function in its argument—such as the example above. Hence, we can use our previously inserted metadata information about the slots of differentiation to extract the variable of differentiation from the semantic macros. If the semantic macro argument is a complex expression, we search for the identifier in the arguments that appear in both elements of the Wronskian. For example, in $\mathscr{W}\{\mathrm{Ai}(z^a), \zeta(z^2, a)\}$, we extract z as the variable since it is the only identifier that appears in the arguments z^a and z^2 of the elements. This approach is also used when there is no semantic macro involved, i.e., from $\mathscr{W}\{z^a, z^2\}$ we extract z as well. If LaCASt extracts multiple candidates or none, it throws a translation exception.

5.1.3 Evaluation of the DLMF using CAS

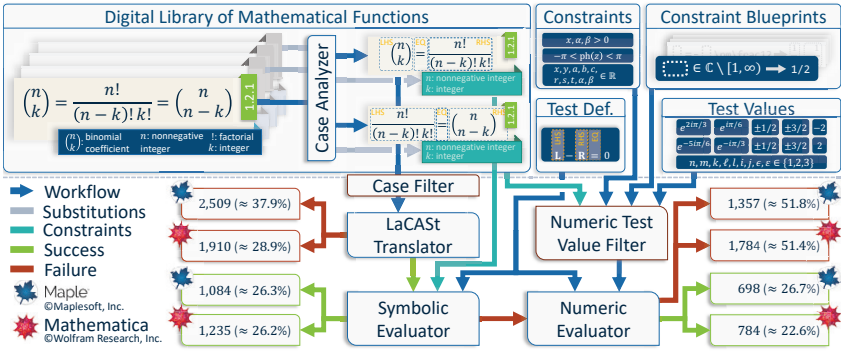


Figure 5.2: The workflow of the evaluation engine and the overall results. Errors and abortions are not included. The generated dataset contains $9,977$ equations. In total, the case analyzer splits the data into $10,930$ cases of which $4,307$ cases were filtered. This sums up to a set of $6,623$ test cases in total.

For evaluating the DLMF with Maple and Mathematica, we symbolically and numerically verify the equations in the DLMF with CAS. If a verification fails, symbolically and numerically, we identified an issue either in the DLMF, the CAS, or the verification pipeline. Note that an issue does not necessarily represent errors/bugs in the DLMF, CAS, or LaCASt (see the discussion about branch cuts in Section 5.1.4.1). Figure 5.2 illustrates the pipeline of the evaluation engine.

First, we analyze every equation in the DLMF (hereafter referred to as test cases). A case analyzer splits multiple relations in a single line into multiple test cases. Note that only the adjacent relations are considered, i.e., with $f(z) = g(z) = h(z)$, we generate two test cases $f(z) = g(z)$ and $g(z) = h(z)$ but not $f(z) = h(z)$. In addition, expressions with $\pm$ and $\mp$ are split accordingly, e.g., $i^{\pm i} = e^{\mp\pi/2}$ [98, (4.4.12)] is split into $i^{+i} = e^{-\pi/2}$ and $i^{-i} = e^{+\pi/2}$. The analyzer utilizes the attached additional information in each line, i.e., the URL in the DLMF, the used and defined symbols, and the constraints. If a used symbol is defined elsewhere in the DLMF, it performs substitutions. For example, the multi-equation [98, (9.6.2)] is split into six test cases and every ζ is replaced by $\frac{2}{3}z^{3/2}$ as defined in [98, (9.6.1)]. The substitution is performed on the parse tree of expressions [10]. A definition is only considered as such, if the defining symbol is identical to the equation's left-hand side. That means, $z = (\frac{3}{2}\zeta)^{3/2}$ [98, (9.6.10)] is not considered as a definition for ζ. Further, semantic macros are never substituted by their definitions. Translations for semantic macros are exclusively defined by the authors. For example, the equation [98, (11.5.2)] contains the Struve $\mathbf{K}_\nu(z)$ function. Since Mathematica does not contain this function, we defined an alternative translation to its definition $\mathbf{H}_\nu(z) - Y_\nu(z)$ in [98, (11.2.5)] with the Struve function $\mathbf{H}_\nu(z)$ and the Bessel function of the second kind $Y_\nu(z)$, because both of these functions are supported by Mathematica. The second entry in Table E.2 in Appendix E available in the electronic supplementary material shows the translation for this test case.

Next, the analyzer checks for additional constraints defined by the used symbols recursively. The mentioned Struve $\mathbf{K}_\nu(z)$ test case [98, (11.5.2)] contains the Gamma function. Since the definition of the Gamma function [98, (5.2.1)] has a constraint $\Re z > 0$, the numeric evaluation must respect this constraint too. For this purpose, the case analyzer first tries to link the variables in constraints to the arguments of the functions. For example, the constraint $\Re z > 0$ sets a constraint for the first argument z of the Gamma function. Next, we check all arguments in the actual test case at the same position. The test case contains $\Gamma(\nu + 1/2)$. In turn, the variable z in the constraint of the definition of the Gamma function $\Re z > 0$ is replaced by the actual argument used in the test case. This adds the constraint $\Re(\nu + 1/2) > 0$ to the test case. This process is performed recursively. If a constraint does not contain any variable that is used in the final test case, the constraint is dropped.

In total, the case analyzer would identify four additional constraints for the test case [98, (11.5.2)][8]. Note that the constraints may contain variables that do not appear in the actual test case, such as $\Re\nu + k + 1 > 0$. Such constraints do not have any effect on the evaluation because if a constraint cannot be computed to true or false, the constraint is ignored. Unfortunately, this recursive loading of additional constraints may generate impossible conditions in certain cases, such as $|\Gamma(iy)|$ [98, (5.4.3)]. There are no valid real values of y such that $\Re(iy) > 0$. In turn, every test value would be filtered out, and the numeric evaluation would not verify the equation. However, such cases are the minority and we were able to increase the number of correct evaluations with this feature.

To avoid a large portion of incorrect calculations, the analyzer filters the dataset before translating the test cases. We apply two filter rules to the case analyzer. First, we filter expressions that do not contain any semantic macros. Due to the limitations of LaCaST, these expressions most likely result in wrong translations. Further, it filters out several meaningless expressions

[8] See Table E.2 in Appendix E available in the electronic supplementary material for the applied constraints (including the directly attached constraint $\Re z > 0$ and the manually defined global constraints from Figure 5.3).

that are not verifiable, such as $z = x$ in [98, (4.2.4)]. The result dataset flag these cases with '*Skipped - no semantic math*'. Note that the result dataset still contains the translations for these cases to provide a complete picture of the DLMF. Second, we filter expressions that contain ellipsis[9] (e.g., `\cdots`), approximations, and asymptotics (e.g., $\mathcal{O}(z^2)$) since those expressions cannot be evaluated with the proposed approach. Further, a definition is skipped if it is not a definition of a semantic macro, such as [98, (2.3.13)], because definitions without an appropriate counterpart in the CAS are meaningless to evaluate. Definitions of semantic macros, on the other hand, are of special interest and remain in the test set since they allow us to test if a function in the CAS obeys the actual mathematical definition in the DLMF. If the case analyzer (see Figure 5.2) is unable to detect a relation, i.e., split an expression on $<$, $\leq$, $\geq$, $>$, $=$, or $\neq$, the line in the dataset is also skipped because the evaluation approach relies on relations to test. After splitting multi-equations (e.g., $\pm$, $\mp$, $a = b = c$), filtering out all non-semantic expressions, non-semantic macro definitions, ellipsis, approximations, and asymptotics, we end up with $6,623$ test cases in total from the entire DLMF.

After generating the test case with all constraints, we translate the expression to the CAS representation. Every successfully translated test case is then symbolically verified, i.e., the CAS tries to simplify the difference of an equation to zero. Non-equation relations simplifies to Booleans. Non-simplified expressions are verified numerically for manually defined test values, i.e., we calculate actual numeric values for both sides of an equation and check their equivalence.

5.1.3.1 Symbolic Evaluation

The symbolic evaluation was performed for Maple as described in the following (taken from [2]). Originally, we used the standalone Maple `simplify` function directly, to symbolically simplify translated formulae. See [26, 28, 148, 190] for other examples of where Maple and other CAS simplification procedures has been used elsewhere in the literature. Symbolic simplification is performed either on the difference or the division of the left-hand sides and the right-hand sides of extracted formulae. Thus the expected outcome should be respectively either a `0` or `1`. Note that other outcomes, such as other numerical outcomes, are particularly interesting, since these may be an indication of errors in the formulae.

In Maple, symbolic simplifications are made using internally stored relations to other functions. If a simplification is available, then in practice it often has to be performed over multiple defined relevant relations. Often, this process fails and Maple is unable to simplify the said expression. We have adopted some techniques which assist Maple in this process. For example, forcing an expression to be converted into another specific representation, in a pre-processing step, could potentially improve the odds that Maple is able to recognize a possible simplification. By trial-and-error, we discovered (and implemented) the following pre-processing steps which significantly improve the simplification process:

- conversion to exponential representation;
- conversion to hypergeometric representation;
- expansion of expressions (for example `(x+y)^2`); and
- combined expansion and conversion processes.

[9] Note that we filter out ellipsis (e.g., `\cdots`) but not single dots (e.g., `\cdot`).

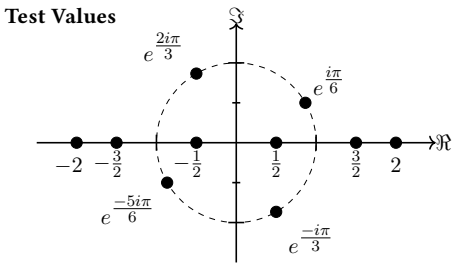


Figure 5.3: The ten numeric test values in the complex plane for general variables. The dashed line represents the unit circle $|z| = 1$. At the right, we show the set of values for special variable values and general global constraints. On the right, i is referring to a generic variable and not to the imaginary unit.

In comparison to the original approach described in [2], we use the newer version Maple 2020 now. Another feature we added to LACAST is the support of packages in Maple. Some functions are only available in modules (packages) that must be preloaded, such as `QPochhammer` in the package `QDifferenceEquations`[10]. The general `simplify` method in Maple does not cover q-hypergeometric functions. Hence, whenever LACAST loads functions from the q-hypergeometric package, the better performing `QSimplify` method is used. With the WED and the new support for Mathematica in LACAST, we perform the symbolic and numeric tests for Mathematica as well. The symbolic evaluation in Mathematica relies on the full simplification[11]. For Maple and Mathematica, we defined the global assumptions $x, y \in \mathbb{R}$ and $k, n, m \in \mathbb{N}$. Constraints of test cases are added to their assumptions to support simplification. Adding more global assumptions for symbolic computation generally harms the performance since CAS internally uses assumptions for simplifications. It turned out that by adding more custom assumptions, the number of successfully simplified expressions decreases.

5.1.3.2 Numerical Evaluation

Defining an accurate test set of values to analyze an equivalence can be an arbitrarily complex process. It would make sense that every expression is tested on specific values according to the containing functions. However, this laborious process is not suitable for evaluating the entire DML and CAS. It makes more sense to develop a general set of test values that (i) generally covers interesting domains and (ii) avoid singularities, branch cuts, and similar problematic regions. Considering these two attributes, we come up with the ten test points illustrated in Figure 5.3. It contains four complex values on the unit circle and six points on the real axis. The test values cover the general area of interest (complex values in all four quadrants, negative and positive real values) and avoid the typical singularities at $\{0, \pm 1, \pm i\}$. In addition, several variables are tied to specific values for entire sections. Hence, we applied additional global constraints to the test cases.

[10] https://jp.maplesoft.com/support/help/Maple/view.aspx?path=QDifferenceEquations/QPochhammer [accessed 2021-05-01]

[11] https://reference.wolfram.com/language/ref/FullSimplify.html [accessed 2021-05-01]

The numeric evaluation engine heavily relies on the performance of extracting free variables from an expression. Maple does not provide a function to extract free variables from an expression. Hence, we implemented a custom method first. Variables are extracted by identifying all `names` [36][12] from an expression. This will also extract constants which need to be deleted from the list first. Unfortunately, inbuilt functions in CAS, if available, and our custom implementation for Maple are not very reliable. Mathematica has the undocumented function `Reduce`FreeVariables` for this purpose. However, both systems, the custom solution in Maple and the inbuilt Mathematica function, have problems distinguishing free variables of entire expressions from the bound variables in MEOMs, e.g., integration and continuous variables. Mathematica sometimes does not extract a variable but returns the unevaluated input instead. We regularly faced this issue for integrals. However, we discovered one example without integrals. For `EulerE[n,0]` from [98, (24.4.26)], we expected to extract $\{n\}$ as the set of free variables but instead received a set of the unevaluated expression itself $\{\texttt{EulerE[n,0]}\}$[13]. Since the extended version of LACAST handles operators, including bound variables of MEOMs, we drop the use of internal methods in CAS and extend LACAST to extract identifiers from an expression. During a translation process, LACAST tags every single identifier as a variable, as long as it is not an element of a MEOM. This simple approach proves to be very efficient since it is implemented alongside the translation process itself and is already more powerful as compared to the existing inbuilt CAS solutions. We defined subscripts of identifiers as a part of the identifier, e.g., z_1 and z_2 are extracted as variables from $z_1 + z_2$ rather than z.

The general pipeline for a numeric evaluation works as follows. First, we replace all substitutions and extract the variables from the left- and right-hand sides of the test expression via LACAST. For the previously mentioned example of the Struve function [98, (11.5.2)], LACAST identifies two variables in the expression, ν and z. According to the values in Figure 5.3, ν and z are set to the general ten values. A numeric test contains every combination of test values for all variables. Hence, we generate 100 test calculations for [98, (11.5.2)]. Afterward, we filter the test values that violate the attached constraints. In the case of the Struve function, we end up with 25 test cases (see also Table E.2 in Appendix E available in the electronic supplementary material).

In addition, we apply a limit of 300 calculations for each test case and abort a computation after 30 seconds due to computational limitations. If the test case generates more than 300 test values, only the first 300 are used. Finally, we calculate the result for every remaining test value, i.e., we replace every variable by their value and calculate the result. The replacement is done by Mathematica's `ReplaceAll` method because the more appropriate method `With`, for unknown reasons, does not always replace all variables by their values. We wrap test expressions in `Normal` for numeric evaluations to avoid conditional expressions, which may cause incorrect calculations (see Section 5.1.4.1 for a more detailed discussion of conditional outputs). After replacing variables by their values, we trigger numeric computation. If the absolute value of the result is below the defined threshold of 0.001 or true (in the case of inequalities), the test calculation is considered successful. A numeric test case is only considered successful if and only if every test calculation was successful. If a numeric test case fails, we store the information on which values it failed and how many of these were successful.

[12] A *name* in Maple is a sequence of one or more characters that uniquely identifies a command, file, variable, or other entity.

[13] The bug was reported to and confirmed by Wolfram Research Version 12.0.

5.1.4 Results

The translations to Maple and Mathematica, the symbolic results, the numeric computations, and an overview PDF of the reported bugs to Mathematica are available online[14]. In the following, we mainly focus on Mathematica because of page limitations and because Maple has been investigated more closely by [2]. The results for Maple are also available online. Compared to the baseline ($\approx 31\%$), our improvements doubled the amount translations ($\approx 62\%$) for Maple and reach $\approx 71\%$ for Mathematica. The majority of expressions that cannot be translated contain macros that have no adequate translation pattern to the CAS, such as the macros for interval Weierstrass lattice roots [98, §23.3(i)] and the multivariate hypergeometric function [98, (19.16.9)]. Other errors (6% for Maple and Mathematica) occur for several reasons. For example, out of the 418 errors in translations to Mathematica, 130 caused an error because the MEOM of an operator could not be extracted, 86 contained prime notations that do not refer to differentiations, 92 failed because of the underlying LaTeX parser [402], and in 46 cases, the arguments of a DLMF macro could not be extracted.

Out of $4{,}713$ translated expressions, $1{,}235$ (26.2%) were successfully simplified by Mathematica ($1{,}084$ of $4{,}114$ or 26.3% in Maple). For Mathematica, we also count results that are equal to 0 under certain conditions as successful (called `ConditionalExpression`). We identified 65 of these conditional results: 15 of the conditions are equal to constraints that were provided in the surrounding text but not in the info box of the DLMF equation; 30 were produced due to branch cut issues (see Section 5.1.4.1); and 20 were the same as attached in the DLMF but reformulated, e.g., $z \in \mathbb{C}\backslash(1,\infty)$ from [98, (25.12.2)] was reformulated to $\Im z \neq 0 \vee \Re z < 1$. The remaining translated but not symbolically verified expressions were numerically evaluated for the test values in Figure 5.3. For the $3{,}474$ cases, 784 (22.6%) were successfully verified numerically by Mathematica (698 of $2{,}618$ or 26.7% by Maple[15]). For $1{,}784$ the numeric evaluation failed. In the evaluation process, 655 computations timed out and 180 failed due to errors in Mathematica. Of the $1{,}784$ failed cases, 691 failed partially, i.e., there was at least one successful calculation among the tested values. For $1{,}091$ all test values failed. The Appendix E, available in the electronic supplementary material, provides a Table E.2 with the results for three sample test cases. The first case is a false positive evaluation because of a wrong translation. The second case is valid, but the numeric evaluation failed due to a bug in Mathematica (see next subsection). The last example is valid and was verified numerically but was too complex for symbolic verifications.

5.1.4.1 Error Analysis

The numeric tests' performance strongly depends on the correct attached and utilized information. The example [98, (1.4.8)] from the DLMF

$$\frac{\mathrm{d}^2 f}{\mathrm{d}x^2} = \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\mathrm{d}f}{\mathrm{d}x}\right), \tag{5.2}$$

illustrates the difficulty of the task on a relatively easy case[16]. Here, the argument of f was not explicitly given, such as in $f(x)$. Hence, LaCasT translated f as a variable. Unfortunately,

[14] https://lacast.wmflabs.org/ [accessed 2021-10-01]

[15] Due to computational issues, 120 cases must have been skipped manually. 292 cases resulted in an error during symbolic verification and, therefore, were skipped also for numeric evaluations. Considering these skipped cases as failures, decreases the numerically verified cases to 23% in Maple.

[16] This is the first example in Table E.2

this resulted in a false verification symbolically and numerically. This type of error mostly appears in the first three chapters of the DLMF because they use generic functions frequently. We hoped to skip such cases by filtering expressions without semantic macros. Unfortunately, this derivative notation uses the semantic macro `deriv`. In the future, we filter expressions that contain semantic macros that are not linked to a special function or orthogonal polynomial.

As an attempt to investigate the reliability of the numeric test pipeline, we can run numeric evaluations on symbolically verified test cases. Since Mathematica already approved a translation symbolically, the numeric test should be successful if the pipeline is reliable. Of the 1,235 symbolically successful tests, only 94 (7.6%) failed numerically. None of the failed test cases failed entirely, i.e., for every test case, at least one test value was verified. Manually investigating the failed cases reveal 74 cases that failed due to an `Indeterminate` response from Mathematica and 5 returned `infinity`, which clearly indicates that the tested numeric values were invalid, e.g., due to testing on singularities. Of the remaining 15 cases, two were identical: [98, (15.9.2)] and [98, (18.5.9)]. This reduces the remaining failed cases to 14. We evaluated invalid values for 12 of these because the constraints for the values were given in the surrounding text but not in the info boxes. The remaining 2 cases revealed a bug in Mathematica regarding conditional outputs (see below). The results indicate that the numeric test pipeline is reliable, at least for relatively simple cases that were previously symbolically verified. The main reason for the high number of failed numerical cases in the entire DLMF (1,784) are due to missing constraints in the i-boxes and branch cut issues (see Section 5.1.4.1), i.e., we evaluated expressions on invalid values.

Bug reports Mathematica has trouble with certain integrals, which, by default, generate conditional outputs if applicable. With the method `Normal`, we can suppress conditional outputs. However, it only hides the condition rather than evaluating the expression to a non-conditional output. For example, integral expressions in [98, (10.9.1)] are automatically evaluated to the Bessel function $J_0(|z|)$ for the condition[17] $z \in \mathbb{R}$ rather than $J_0(z)$ for all $z \in \mathbb{C}$. Setting the `GenerateConditions`[18] option to `None` does not change the output. `Normal` only hides $z \in \mathbb{R}$ but still returns $J_0(|z|)$. To fix this issue, for example in (10.9.1) and (10.9.4), we are forced to set `GenerateConditions` to false.

Setting `GenerateConditions` to false, on the other hand, reveals severe errors in several other cases. Consider $\int_z^\infty t^{-1}e^{-t}\,\mathrm{d}t$ [98, (8.4.4)], which gets evaluated to $\Gamma(0,z)$ but (condition) for $\Re z > 0 \wedge \Im z = 0$. With `GenerateConditions` set to false, the integral incorrectly evaluates to $\Gamma(0,z) + \ln(z)$. This happened with the 2 cases mentioned above. With the same setting, the difference of the left- and right-hand sides of [98, (10.43.8)] is evaluated to 0.398942 for $x, \nu = 1.5$. If we evaluate the same expression on $x, \nu = \frac{3}{2}$ the result is `Indeterminate` due to `infinity`. For this issue, one may use `NIntegrate` rather than `Integrate` to compute the integral. However, evaluating via `NIntegrate` decreases the number of successful numeric evaluations in general. We have revealed errors with conditional outputs in (8.4.4), (10.22.39), (10.43.8-10), and (11.5.2) (in [98]). In addition, we identified one critical error in Mathematica. For [98, (18.17.47)], WED (Mathematica's kernel) ran into a *segmentation fault (core dumped)* for $n > 1$. The kernel of the full version of Mathematica gracefully died without returning an output[19].

[17] $J_0(x)$ with $x \in \mathbb{R}$ is even. Hence, $J_0(|z|)$ is correct under the given condition.

[18] https://reference.wolfram.com/language/ref/GenerateConditions.html [accessed 2021-05-01]

[19] All errors were reported to and confirmed by Wolfram Research.

Besides Mathematica, we also identified several issues in the DLMF. None of the newly identified issues were critical, such as the reported sign error from the previous project [2], but generally refer to missing or wrong attached semantic information. With the generated results, we can effectively fix these errors and further semantically enhance the DLMF. For example, some definitions are not marked as such, e.g., $Q(z) = \int_0^\infty e^{-zt} q(t)\,\mathrm{d}t$ [98, (2.4.2)]. In [98, (10.24.4)], ν must be a real value but was linked to a *complex parameter* and x should be positive real. An entire group of cases [98, (10.19.10-11)] also discovered the incorrect use of semantic macros. In these formulae, $P_k(a)$ and $Q_k(a)$ are defined but had been incorrectly marked up as Legendre functions going all the way back to DLMF Version 1.0.0 (May 7, 2010). In some cases, equations are mistakenly marked as definitions, e.g., [98, (9.10.10)] and [98, (9.13.1)] are annotated as local definitions of n. We also identified an error in LaCaST, which incorrectly translated the exponential integrals $E_1(z)$, $\mathrm{Ei}(x)$ and $\mathrm{Ein}(z)$ (defined in [98, §6.2(i)]). A more explanatory overview of discovered, reported, and fixed issues in the DLMF, Mathematica, and Maple is provided in Appendix D available in the electronic supplementary material.

Branch cut issues Problems that we regularly faced during evaluation are issues related to multi-valued functions. Multi-valued functions map values from a domain to multiple values in a codomain and frequently appear in the complex analysis of elementary and special functions. Prominent examples are the inverse trigonometric functions, the complex logarithm, or the square root. A proper mathematical description of multi-valued functions requires the complex analysis of Riemann surfaces. Riemann surfaces are one-dimensional complex manifolds associated with a multi-valued function. One usually multiplies the complex domain into a many-layered covering space. The correct properties of multi-valued functions on the complex plane may no longer be valid by their counterpart functions on CAS, e.g., $(1/z)^w$ and $1/(z^w)$ for $z, w \in \mathbb{C}$ and $z \neq 0$. For example, consider $z, w \in \mathbb{C}$ such that $z \neq 0$. Then mathematically, $(1/z)^w$ always equals $1/(z^w)$ (when defined) for all points on the Riemann surface with fixed w. However, this should certainly not be assumed to be true in CAS, unless very specific assumptions are adopted (e.g., $w \in \mathbb{Z}$, $z > 0$). For all modern CAS[20], this equation is not true. Try, for instance, $w = 1/2$. Then $(1/z)^{1/2} - 1/z^{1/2} \neq 0$ on CAS, nor for w being any other rational non-integer number.

In order to compute multi-valued functions, CAS choose branch cuts for these functions so that they may evaluate them on their principal branches. Branch cuts may be positioned differently among CAS [84], e.g., $\operatorname{arccot}(-\frac{1}{2}) \approx 2.03$ in Maple but is ≈ -1.11 in Mathematica. This is certainly not an error and is usually well documented for specific CAS [108, 171]. However, there is no central database that summarizes branch cuts in different CAS or DML. The DLMF as well, explains and defines their branch cuts carefully but does not carry the information within the info boxes of expressions. Due to complexity, it is rather easy to lose track of branch cut positioning and evaluate expressions on incorrect values. For example, consider the equation [98, (12.7.10)]. A path of $z(\phi) = e^{i\phi}$ with $\phi \in [0, 2\pi]$ would pass three different branch cuts. An accurate evaluation of the values of $z(\phi)$ in CAS require calculations on the three branches using analytic continuation. LaCaST and our evaluation frequently fall into the same trap by evaluating values that are no longer on the principal branch used by CAS. To solve this issue, we need to utilize branch cuts not only for every function but also for every equation in the DLMF [10]. The positions of branch cuts are exclusively provided in the text

[20] The authors are not aware of any example of a CAS which treats multi-valued functions without adopting principal branches.

but not in the i-boxes. Adding the information to each equation in the DLMF would be a laborious process because a branch cut position may change according to the used values (see the example [98, (12.7.10)] from above). Our result data, however, would provide beneficial information to update, extend, and maintain the DLMF, e.g., by adding the positions of the branch cuts for every function. An extended discussion about branch cut issues is available in Appendix A available in the electronic supplementary material.

5.1.5 Conclude Quantitative Evaluations on the DLMF

We have presented a novel approach to verify the theoretical digital mathematical library DLMF with the power of two major general-purpose computer algebra systems Maple and Mathematica. With LaCaSt, we transformed the semantically enhanced LaTeX expressions from the DLMF to each CAS. Afterward, we symbolically and numerically evaluated the DLMF expressions in each CAS. Our results are auspicious and provide useful information to maintain and extend the DLMF efficiently. We further identified several errors in Mathematica, Maple [2], the DLMF, and the transformation tool LaCaSt, proving the profit of the presented verification approach. Further, we provide open access to all results, including translations and evaluations[21].

The presented results show a promising step towards an answer for our initial research question. By translating an equation from a DML to a CAS, automatic verifications of that equation in the CAS allows us to detect issues in either the DML source or the CAS implementation. Each analyzed failed verification successively improves the DML or the CAS. Further, analyzing a large number of equations from the DML may be used to finally verify a CAS. In addition, the approach can be extended to cover other DML and CAS by exploiting different translation approaches, e.g., via MathML [18] or OpenMath [152].

Nonetheless, the analysis of the results, especially for an entire DML, is cumbersome. Minor missing semantic information, e.g., a missing constraint or not respected branch cut positions, leads to a relatively large number of false positives, i.e., unverified expressions correct in the DML and the CAS. This makes a generalization of the approach challenging because all semantics of an equation must be taken into account for a trustworthy evaluation. Furthermore, evaluating equations on a small number of discrete values will never provide sufficient confidence to verify a formula, which leads to an unpredictable number of true negatives, i.e., erroneous equations that pass all tests.

After all, we conclude that the approach provides valuable information to complement, improve, and maintain the DLMF, Maple, and Mathematica. A trustworthy verification, on the other hand, might be out of reach.

5.1.5.1 Future Work

The resulting dataset provides valuable information about the differences between CAS and the DLMF. These differences had not been largely studied in the past and are worthy of analysis. Especially a comprehensive and machine-readable list of branch cut positioning in different systems is a desired goal [84]. Hence, we will continue to work closely together with the editors of the DLMF to improve further and expand the available information on the DLMF. Finally, the numeric evaluation approach would benefit from test values dependent on the actual functions involved. For example, the current layout of the test values was designed to avoid

[21] https://lacast.wmflabs.org/ [accessed 2021-10-01]

problematic regions, such as branch cuts. However, for identifying differences in the DLMF and CAS, especially for analyzing the positioning of branch cuts, an automatic evaluation of these particular values would be very beneficial and can be used to collect a comprehensive, inter-system library of branch cuts. Therefore, we will further study the possibility of linking semantic macros with numeric regions of interest.

Finally, we used LaCaSt to perform translations solely on semantic LaTeX expressions. Real-world mathematics, however, is not available in this semantically enriched format. In the previous chapter, we already developed and discussed a context-sensitive extension for LaCaSt. This enables LaCaSt to translate not only semantic LaTeX formulae from the DLMF but, considering an informative textual context, also general mathematical expressions to multiple CAS. In the following section, we will evaluate this new extension of LaCaSt on Wikipedia articles.

5.2 Evaluations on Wikipedia

In the following, resulting from our motivation outlined in Chapter 4 - improving Wikipedia articles - we use Wikipedia for our test dataset to evaluate our context-sensitive extension of LaCaSt. More specifically, we considered every English Wikipedia article that references to the DLMF via the `{{dlmf}}` template[22]. 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)[23]. For the entire dataset (the remaining 103 Wikipedia pages), we detected $6,337$ formulae in total (including potential erroneous math).

So far, one of our initial three issues from Section 4.2.3 still remains unsolved: how can we determine if a translation was appropriate and complete? We called a translation appropriate, if the intended meaning of a presentational expression $e \in \mathcal{L}_P$ is the same as the translated expression $\mathrm{t}(e, X) \in \mathcal{L}_C$. However, how can we know the intended semantic meaning of a presentational expression $e \in \mathcal{L}_P$? In natural languages, the BLEU score [282] 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 $\operatorname{arccot}(x)$ 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 [13, Section 3.2]). Previously, we developed a new approach that relies on automatic verification checks with CAS [2, 11] 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.

[22] 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.

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

5.2.1 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. As we have discussed in the previous Section 5.1, 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.

In the previous Section 5.1, we saw that the translations by LaCASt [13] 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) [2, 11]. 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, 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 [359]. 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.

5.2.2 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[24], 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

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

Table 5.3: The symbolic and numeric evaluations 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). The DLMF scores refer to the results presented in the previous Section 5.1.

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

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[25]. 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 added an overview to Appendix F.4 available in the electronic supplementary material.

5.2.3 Results

First, we evaluated the $6,337$ detected formulae with our automatic evaluation via Maple and Mathematica. Table 5.3 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 [403] reached a success rate of 26% for symbolic and 43% for numeric evaluations [11] (see the previous Section 5.1). 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's function contains

[25]Note that there are numerous specialized CAS that would cover the mentioned domains too, such as GAP [177], PARI/GP [283], or Cadabra [290].

such an edit on the equation

$$J_n(x) = \frac{1}{\pi} \int_0^{\pi} \cos(n\tau - x \sin \tau)\, d\tau. \tag{5.3}$$

Here, the edit[26] changed $J_n(x)$ to $J_Z WE(x)$. 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.

5.2.3.1 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 [330]. 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 5.4a and for relevance 2 in Table 5.4b. 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 [330], 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.

5.2.3.2 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 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)

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

Table 5.4: Performance of description extractions via MLP for low (5.4a) and high (5.4b) relevance. 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.

(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

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 (`base`) 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 5.5 compares the performance. The table shows that depth 1 outperforms depth 0, which intuitively contradicts the F1 scores in Table 5.4a. 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.

5.2.3.3 Translations from LaTeX to CAS

Mathematica's ability to import TeX expressions will serve as a baseline. While Mathematica does 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 5.6 compares the performance. The methods `base`, `ck`, `full` are the same as in Table 5.5, but now refer to translations to Mathematica, rather than semantic LaTeX. Method `full` uses the optimal setting as shown in Table 5.5. We consider a

Table 5.5: Performance of semantification from LaTeX to semantic LaTeX. **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. The methods `base` refers to no transformations $\mathrm{t}(e, X) = e$, `ck` where $X = \mathcal{K}$, and `full` use the full proposed pipeline. ✔ matches the benchmark entry and ✘ does not match the 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

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 `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 [13]. Note that we were not able to appropriately translate 18 expressions (indicated by the human performance in Table 5.6) as discussed before.

5.2.4 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 Appendix F.3 (available in the electronic supplementary material) 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 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.

Table 5.6: Performance comparison for translating LaTeX to Mathematica. A translation was successful (**ST**) if it was syntactically verified by Mathematica (otherwise: **FT**). ✔ refers to matches with the benchmark and ✘ to mismatches. The methods are explained in Section 5.2.3.3.

LaTeX Translations to Mathematica

Method	ST	FT	✔	✘
`MM_import`	57 (.60)	38 (.40)	9 (.09)	48 (.51)
`LaCaST_base`	55 (.58)	40 (.42)	11 (.12)	44 (.46)
`LaCaST_ck`	**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)

5.2.4.1 Defining Equations

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[27]. 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 5.6.

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 [214], 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 [111, 134, 183, 370] 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 [111, 134] into our dependency graph concept.

5.2.4.2 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 5.6). This naive

[27] The DLMF did not use this notation, hence LaCaST was not capable of translating $:=$ in the first place.

approach, emphasizes the importance of knowing the domain knowledge for specific articles. In combination with article classifications [320], we could activate different common knowledge sets depending on the specific domain.

5.2.4.3 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 4.5) 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 [14], 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. This is also one of the reasons why our semantic LaTeX test performed 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 [1, 15]. 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.

5.2.5 Conclude Qualitative Evaluations on Wikipedia

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 [17] and user experience [150], enabling entity linking for mathematics [320, 17], or allowing for automatic quality checks via CAS [2, 11]. In turn, we plan to integrate [401] 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 [260]. As shown in the DLMF [260], 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 [17].

In future work, we aim to mitigate the issues outlined in Section 5.2.4, 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 [111, 134, 183, 370], 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 [401] (as an extension to the existing ORES system) that makes use of this novel capability.

That can only mean one thing.
And I don't know what it is.

Sam Diamond - *Murder by Death*

CHAPTER 6

Conclusion and Future Work

Contents

This chapter summarizes and concludes the contribution of this thesis in Section 6.1 and Section 6.2, respectively. Section 6.3 provides an overview of future work projects.

6.1 Summary

In this thesis, we presented novel approaches to translate presentational mathematical encodings into computable formats and to evaluate these translations. We focused on LaTeX for the presentational encodings and Computer Algebra Systems (CAS) syntaxes for computable formats. Primarily, we focused on translations to the two major general-purpose CAS Maple and Mathematica.

Every mathematical format serves a specific purpose and encodes different amounts of semantic information into an expression. A presentational format encodes visual information, while computable formats need to uniquely link elements with specific definitions (i.e., implementations). There are numerous mathematical formats and conversion tools available. Many roads leads to Rome, thus there are several translation paths from LaTeX to CAS syntaxes available, including direct translations via CAS import functions (see Table 1.3). The most well-covered conversion path between mathematical formats is between the standard encodings LaTeX and MathML. Since content MathML explicitely encodes semantic information and many CAS are able to import content MathML, the easiest approach for translating LaTeX to CAS was to use MathML as an intermediate format. Hence, we developed MathMLben, a MathML benchmark, to evaluate the quality of the translations of several state-of-the-art LaTeX to MathML conversion tools.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-658-40473-4_6.

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_6

Our benchmark test revealed that existing LaTeX conversion tools only consider the semantic information that is explicitly encoded in the given expression, e.g., via visual pattern recognition approaches. For example, Mathematica concludes $P_n^{(\alpha,\beta)}(x)$ to be the Jacobi polynomial because there is no other expression with the same pattern available in Mathematica. Only three of the nine state-of-the-art converters supported content MathML but with insufficient accuracy. The conversion tool LaTeXML performed best and is able translate semantically enriched formulae in semantic LaTeX. Without a manual annotation with semantic macros, however, LaTeXML also create wrong and incomplete results. In addition, even though CAS often support MathML (including content MathML), there is no public mapping between functions in a Content Dictionary (DC) and functions in the CAS available. Hence, a reliable import of MathML is generally limited to K-14[1] mathematics.

Prior to this thesis, we developed LaCaST, a translator from semantic LaTeX to the CAS Maple. LaCaST was the first translator to a CAS syntax that provided additional information about the translation process and provided alternative translations if a direct mapping was unavailable. The first version of LaCaST laid the foundation to solve translation issues related to differences in the definitions of functions, e.g., branch cut positioning. However, LaCaST required manually crafted semantic LaTeX as it is used in the DLMF. Subsequently, we focused on extending LaCaST to perform a semantification step from LaTeX to semantic LaTeX based on the information gathered in the surrounding context of a formula.

The semantification of mathematical expressions, even though related to other MathIR tasks, was new due to the information needs for a translation to computable formats. Other tasks in MathIR, such as the search for relevant or similar formulae, rarely need to understand the structure of mathematical objects in an expression. For a translation to computable formats, a conversion tool needs to identify the subexpressions representing a specific formula, determine which formula it represents, what parts of the subexpression are variable or fixed (stem), and how the formula is declared in the context. Existing approaches to semantically enhance mathematical expressions with information from a textual context can be categorized into two groups. The first group takes single identifiers (or other single tokens) and attaches information from the context to these identifiers. The second group annotates entire mathematical expressions. Both approaches, however, ignore informative and crucial subexpressions.

As a first approach for a semantification process, we explored the capabilities of word embedding techniques. These models generally perform well on several natural language processing tasks and are able to capture co-occurrences of tokens in large corpora. These co-occurrences seem to model semantic relationships, as it is often shown in the infamous king-queen relationship[2]. Unfortunately, we were unable to achieve similar results for math embeddings due to fundamental issues in existing embedding approaches. While natural language sentences are a sequential order of words, math formulae are deeply nested structures in which only a few tokens are fixed. However, distinguishing fixed from variable tokens, i.e., identifying the *stem* of a mathematical function, is context-dependent. In order to overcome these representational issues of mathematical expressions, we introduced a new nested concept for mathematical expressions, MOI.

[1] Kindergarten to early college.

[2] The relationship between *king* and *man* is very similar (in terms of cosine difference between the vector representations) to the relationship between *queen* and *woman*.

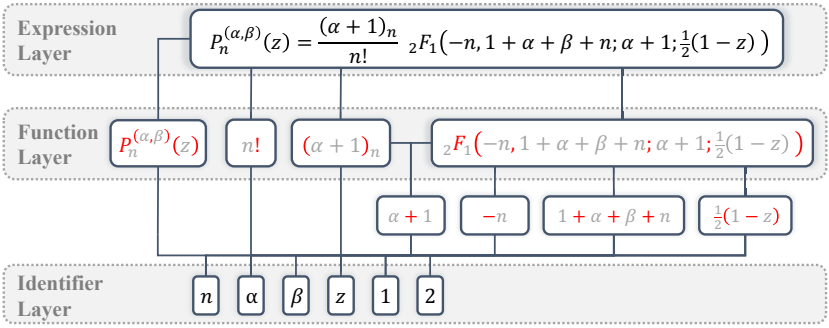


Figure 6.1: Layers of a mathematical expression with mathematical objects (MOI). MOI in the function layer can be semantically enhanced by semantic LaTeX macros. The red tokens are fixed tokens of the MOI and the gray tokens are variable (variables and parameters).

A Mathematical Objects of Interest (MOI) represents a meaningful mathematical subexpression (math object) which might be composed of other MOI. Figure 6.1 shows different layers of mathematical objects within the defining formula of Jacobi polynomials. As previously mentioned, most MathIR approaches focus on the context-independent elements in the expression or identifier layer. For translating equations from LaTeX to CAS syntaxes, however, the elements in the layers in between both extremes are generally most crucial. If we want to translate an equation to the syntax of CAS, we need to primarily translate MOI in the function layer because those elements are mapped to unique keywords in the CAS. As an approach to explore the usability of the new MOI concept, we performed the first large-scaled notation study of over 2.5 billion mathematical subexpressions in 2 million documents from arXiv and zbMATH. We have shown that the distribution of mathematical subexpressions is similar to words in natural language corpora. Following the idea that mathematical expressions are more comparable to sentences in natural languages, we analyzed the effectiveness of distribution scores, such as BM25, to retrieve MOI for given textual descriptions and achieved good results.

Consequently, we developed a novel semantification pipeline based on the MOI concept in which we presume that every isolated mathematical expression in a text is considered to be meaningful. The connections between MOI are modeled by a mathematical dependency graph that links two MOI if one is a subexpression of the other (following a specific heuristic to allow matches between $\Gamma(x)$ and $\Gamma(z)$). Each MOI (now a node in the dependency graph) is tagged with descriptions extracted from the textual context. With these descriptions, we can retrieve semantic LaTeX macros that represent the MOI. In addition, the dependency graph allows retrieving semantic LaTeX macros for each meaningful subexpression too. Finally, we semantically enhance the original LaTeX expression by replacing each MOI with the correpsonding semantic LaTeX macro. The resulted enhanced expression can be further translated to CAS syntaxes with LaCaST. Figure 6.2 shows the relevant annotations and dependencies of the defining formula of Jacobi polynomials in the English Wikipedia article. In order to replace LaTeX with semantic LaTeX macros, we retrieve all textual descriptions (green boxes) surrounding the formula and all dependent MOI (blue boxes).

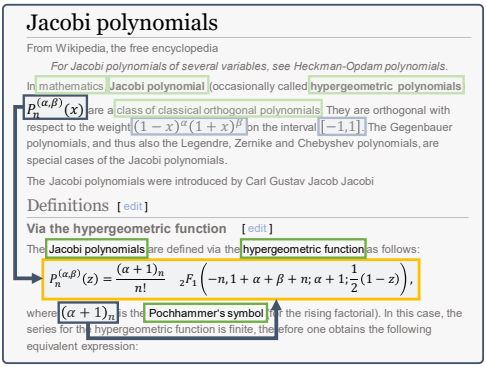


Figure 6.2: The annotated defining formula of Jacobi polynomials (yellow) in the English Wikipedia article. The defining formula depends on two other MOI (blue) in the same article: $P_n^{(\alpha,\beta)}(x)$ and $(\alpha+1)_n$. Hence, in order to properly translate the defining formula, we need to translate the dependent MOI. This can be achieved by retrieving textual annotations (green) from the surrounding context.

The proposed semantification approach requires a semantic LaTeX macro to semantically enhance an MOI. The semantic macros were developed for the DLMF and mostly covered OPSF. General-purpose CAS, like Maple and Mathematica, natively support functions from this area in general. Hence, there is a significant overlap between the functions that have a semantic macro in the DLMF and are natively supported by CAS. Translating general expressions to CAS is often not possible and may require entire new subroutines in the CAS. Consider the prime counting function $\pi(x)$ does not exist in Maple. In this case, translating $\pi(x)$ to Maple is impossible unless we are able to automatically generate subroutines that are able to compute this function. Often, however, general functions are much simpler and may be represented by known functions, e.g., $f(x) := \sin^2(x)$. In this case, we need to identify the definition of $f(x)$ in order to properly translate it. Translating $f(x)-g(x)$, for instance, is meaningless without knowing the definition of $f(x)$ and $g(x)$. However, determining whether an equation declares a definition remains an open research task for future work.

As an alternative to the new context-sensitive translation pipeline for LaCaST, we also experimented with machine translation approaches for LaTeX to CAS conversions. We discovered that our machine translation approach is very powerful in adapting conversion rules of other converters, e.g., the LaTeX export function of Mathematica or the conversion process by LaTeXML. Here, we achieved up to 95.1% exact match accuracy for undoing an export conversion by Mathematica and 90.7% accuracy for undoing a conversion by LaTeXML. However, we also identified that such machine translations are very unreliable when it comes to general mathematical expressions. On 100 random selected samples from the DLMF, our machine translation approach correctly translated only 5% of the expressions, compared to 11% by Mathematica or 7$ by SymPy. Our rule-based translator LaCaST achieved 22%. If LaCaST performs translations on the original semantic LaTeX source of the 100 samples from the DLMF, LaCaST achieves 51% accuracy. On non-semantic enhanced cases from Wikipedia articles, our new context-sensitive version of

LaCASt correctly translated 27% compared to the state-of-the-art 9% by Mathematica. We have also shown that a proper definition detection system and an improved common knowledge datatset would boost the number of correct translated expressions to 47%. In comparison, a human annotator was able to translate 81% of the expressions manually.

For determining if a translation was correct or not, one cannot directly adapt established measures for natural language translations. The known BLEU score, for instance, is inappropriate since two entire different mathematical expressions can still be equivalent. Hence, we developed a novel evaluation system based on the fact that a translated expression can be further computed by CAS. Consider an equation, which mathematicians manually proved, such as

$$\sin^2(z) + \cos^2(z) = 1. \tag{6.1}$$

If the translation of this expression was correct, the equation must be valid in the syntax of the CAS too. Most CAS are powerful enough to verify such simple equivalence, e.g., via symbolic simplifications. In combination with a comprehensive library of proven equations, such as the DLMF, we could semantically evaluate translations by LaCASt.

There is a catch to this evaluation technique. Verifying an equation to be correct can become arbitrarily complex (consider the infamous Riemann hypothesis or Fermat's last conjecture, for example). Hence, automatically verifying an equation with CAS is limited. Nonetheless, CAS are powerful and flexible tools, especially when it comes to numeric evaluations. We developed a two-step evaluation approach to verify an equation in CAS. First, we symbolically simplify the difference of the left- and right-hand sides of an equation to zero. If the result is zero, the equation is considered symbolically verified. Second, we numerically calculate the difference between the left- and right-hand sides for actual numeric test values if the symbolic verification failed. An equation is numerically verified if the difference is close to zero for all test values (due to machine accuracy). While the numeric evaluation approach never proves equivalence, it can detect disparity. A symbolically or numerically verified equation can be considered as correctly translated by LaCASt.

It turns out that the translations of LaCASt are so reliable on DLMF equations that this evaluation technique not only detects issues in the translation process but in the source and target systems as well. Consider there is an error in a test equation, such as in

$$\mathsf{Q}_{\nu}^{-1/2}(\cos\theta) = -\left(\frac{\pi}{2\sin\theta}\right)^{1/2} \frac{\cos\left(\left(\nu+\frac{1}{2}\right)\theta\right)}{\nu+\frac{1}{2}}. \tag{6.2}$$

The numeric evaluation would fail for most test values indicating that there was an error either in the source equation, i.e., the DLMF, the translator LaCASt, or in the target CAS. Hence, we evaluated the entire DLMF with this evaluation technique and identified numerous of issues in the DLMF, Wikipedia, Maple, and Mathematica. Via LaCASt translations and evaluations, for example, we identified the sign error (the red marked minus) in equation (6.2) in the DLMF [98, (14.5.14)]. This error was fixed with version 1.0.16 in the DLMF. Most notable error reports include this sign error and incorrect semantic annotations in the DLMF, wrong calculations for specific integrals and bugs in a variable extraction algorithm in Mathematica, incorrect symbolic computations in Maple, and malicious edits in Wikipedia articles[3].

[3] An overview of discovered, reported, and fixed issues in CAS, DLMF, and in the Wikipedia articles is available in Appendix D available in the electronic supplementary material.

Note that, even with our novel semantification approach, LaCASt cannot be considered as a finished project (see Section 6.3). Several improvements could be achieved in the future. A crucial issue occur, for instance, if a function is not following the DLMF standard notation, e.g., $p(n, \alpha, \beta, x)$ for the Jacobi polynomial rather than $P_n^{(\alpha,\beta)}(x)$. In that case, LaCASt is incapable of translating the expression. There is, however, no easy solution to this problem. Such a custom notation raise the question about the order of the arguments. For example, in $p(a, b, c, d)$, we cannot determine if c is referring to the degree of the Jacobi polynomial and should be mapped to the first argument in Mathematica syntax or to any other position. One possible workaround is to fetch and analyze the definition of $p(a, b, c, d)$, supposed the definition is available in the context. By comparing the definition in the context with the actual Jacobi polynomial definition in the DLMF or the CAS, we could map each argument with their respective semantics, e.g., c to the *degree* of the polynomial. Such a comparison would introduce its own challenges. For example, what if the definition is not exactly the same as in the DLMF? Moreover, as we pointed out earlier, determining an equation as a defining formula is also an open research question. Recently, a similar issue gained interest among the NLP community with the goal to determine the semantic classification of paragraphs and text spans, such as definitions, theorems, or examples [111, 134, 183, 209, 370]. Most of the remaining issues of LaCASt come along with open research questions. Some examples are:

- How can we distinguish an equation from a defining formula?
- How can we determine the *stem* of a function by a given definition?
- How can we identify constraints and their scopes in natural language contexts?
- Are there specific numeric values an equation should be tested on to increase the trustworthiness in positive numeric evaluation results?
- How can a translation process overcome different branch cut positions between domain and co-domain representations?

Nonetheless, LaCASt, in its current state, already outperforms existing presentational-to-computational translation solutions, improves the scientific work cycle of experimenting and publishing, and even helps to correct issues in DML and CAS. LaCASt increases the trustworthiness in translation with a transparent communication about the translation decisions [13]. In combination with direct access to CAS' kernels, LaCASt also performs automatic verification checks on its translations, the source formula, and the system computations. This capability was successfully demonstrated on the DLMF in which we were able to identify numerous issues, from missing or incorrect semantic annotations to wrong constraints and sign errors [2]. With the same evaluation approach, LaCASt helps discover bugs in the commercial CAS, Maple and Mathematica [8]. In Wikipedia, LaCASt computations allow for detecting malicious edits and the performed semantic enhancements potentially improve the readability and accessibility of mathematical content [11].

In addition, several of the projects on the way to the final version of LaCASt contributed towards multiple MathIR tasks. The developed MathML benchmark: MathMLben, for instance, is used for research in mathematical entity linking [321]. Our math embedding experiments enabled new approaches, such as centroid search queries and similarity measures for mathematical expressions [15, 323, 332, 404]. Our study about the frequency distributions of mathematical subexpressions in large corpora [14] enabled a new search engine for zbMATH [16], an auto-

completion for mathematical inputs, new approaches for plagiarism detection systems[4], and literature recommendation systems that will, for the first time, take mathematical content into account [50]. The mathematical dependency graph generated by LaCASt can be embedded in Wikipedia to provide additional semantic information about a formula in a pop-up information window [17]. Lastly, LaCASt is currently planned to be integrated into future versions of the DLMF to provide static translations for all DLMF equations and a live interface for general expressions. The source of LaCASt is publicly available on https://github.com/gipplab/LaCASt since February 2022.

LaCASt Translation Examples To conclude with the examples from the introduction of the thesis, LaCASt correctly translates every expression in Table 1.2 to Maple, Mathematica, and SymPy. On 100 random selected formulae from the DLMF, LaCASt correctly translated 22% and significantly outperforms existing converters, such as Mathematica (11%), SymPy (7%), and machine translations (5%). For the semantic LaTeX source, LaCASt correctly translated 51% of the 100 samples. LaCASt addresses the issues of branch cuts and differences in definitions between the system by providing additional information and a transparent decision process. For instance, $\operatorname{arccot}(z)$ is translated to Maple with `arccot(z)` but LaCASt warns about the differences in the positioning of branch cuts and informs the user about alternative translation patterns, such as `I/2*ln((($0-I)/($0+I))` or `arctan(1/($0))`. Additionally, LaCASt provides links to the definitions of the function, the domains, and the constraints, if available. By providing a textual context that declares $P_n^{(\alpha,\beta)}(x)$ as the Jacobi polynomial and $\Gamma(z)$ as the Gamma function, LaCASt also correctly translates equation (1.1) from the introduction. No CAS import functions nor alternative translations via MathML (followed by an import to the CAS) are capable of correctly translating equation (1.1), all expressions in Table 1.2, or $\pi(x+y)$ in various contexts. Further, no system, besides LaCASt, informs the user about potential issues, such as the different branch cuts of $\operatorname{arccot}(z)$.

To provide a more sophisticated example that underlines the capabilities of LaCASt, consider Bailey's transformation of very-well-poised ${}_8\phi_7$ from the DLMF [98, (17.9.16)]

$$
\begin{aligned}
&{}_8\phi_7\left(\begin{matrix} a, qa^{\frac{1}{2}}, -qa^{\frac{1}{2}}, b, c, d, e, f \\ a^{\frac{1}{2}}, -a^{\frac{1}{2}}, aq/b, aq/c, aq/d, aq/e, aq/f \end{matrix}; q, \frac{a^2q^2}{bcdef}\right) \\
&\quad = \frac{(aq, aq/(de), aq/(df), aq/(ef); q)_\infty}{(aq/d, aq/e, aq/f, aq/(def); q)_\infty}\, {}_4\phi_3\left(\begin{matrix} aq/(bc), d, e, f \\ aq/b, aq/c, def/a \end{matrix}; q, q\right) \\
&\qquad + \frac{\left(aq, aq/(bc), d, e, f, a^2q^2/(bdef), a^2q^2/(cdef); q\right)_\infty}{(aq/b, aq/c, aq/d, aq/e, aq/f, a^2q^2/(bcdef), def/(aq); q)_\infty} \\
&\qquad \times {}_4\phi_3\left(\begin{matrix} aq/(de), aq/(df), aq/(ef), a^2q^2/(bcdef) \\ a^2q^2/(bdef), a^2q^2/(cdef), aq^2/(def) \end{matrix}; q, q\right).
\end{aligned}
\tag{6.3}
$$

No CAS nor other translation approaches are capable of interpreting and translating this expression correctly with (or without) semantic annotations or textual descriptions. Mathematica, for example, cannot interpret leading indexes correctly, such as in ${}_8\phi_7$, and is unable to understand $(a, b; q)_n$ because the multiple q-pochhammer symbol does not exist in Mathematica.

[4] See the DFG (German Research Foundation) fund: *Analyzing Mathematics to Detect Disguised Academic Plagiarism* (https://gepris.dfg.de/gepris/projekt/437179652 [accessed 2021-09-08])

Since the DLMF source uses semantic macros to unambiguously describe the expression, LaCaST translates this complicated equation from the DLMF to Mathematica effortlessly by exploiting the definition of the multiple q-pochhammer symbol. Additionally, LaCaST provides useful information about the internal decision process (see Figure 6.3). Outside of the DLMF, e.g., in Wikipedia, LaCaST would require a context that explains the functions in equation (6.3) to properly disambiguate the components.

A short example context that enables LaCaST to properly understand equation (6.3)

The basic hypergeometric function ${}_2\phi_2\left(\begin{smallmatrix}a,b\\c,d\end{smallmatrix};q,z\right)$ and the multiple q-pochhamer symbol $(a,b;q)_n$ describes Bailey's transformation of very-well-poised ${}_8\phi_7$.

In combination with this context, LaCaST identifies the function patterns and semantically enhances the input expression with DLMF macros. Consequently, LaCaST correctly translates the expression to Mathematica, as it did for the original DLMF source equation, and provides the same useful information about the translation decisions, see Figure 6.3. Unfortunately, the equation is too complex for our automatic evaluation approach.

Performing a manual translation for such significant expressions is very exhaustive and requires a deep understanding of the CAS. Simple mistakes, such as a sign error or a switched order of arguments, can lead to errors that are very difficult to detect. Additionally, even performing translations to appropriate counterparts in the CAS can quickly yield to undesired behaviour (as we haven seen for translations of $\operatorname{arccot}(-1)$). By providing information about the internal translation decisions, LaCaST translations are more trustworthy and comprehensible. LaCaST notifies a user about potential issues in regard of branch cut positions or questionable translation decisions, mitigating the chance of wrong, untracable errors. For instance, LaCaST is aware of the issue that the q-multi-pochhammer symbol is not natively supported by Mathematica but performs an alternative translation instead. Further, LaCaST sensitizes users for potential ambiguity issues, such as the use of abbreviations[5] or the ambiguity[6] of e.

Translation of Bailey's Transformation of Very-Well-Poised ${}_8\phi_7$ (see equation (6.3) and [98, (17.9.16)])

```
QHypergeometricPFQ[{a, q*(a)^(Divide[1,2]),-q*(a)^(Divide[1,2]),b,c,d,e,f},{(
    a)^(Divide[1,2]), -(a)^(Divide[1,2]),a*q/b,a*q/c,a*q/d,a*q/e,a*q/f},q,
    Divide[(a)^(2)*(q)^(2),b*c*d*e*f]]
 == Divide[Product[QPochhammer[Part[{a*q,a*q/(d*e),a*q/(d*f),a*q/(e*f)},i],q,
     Infinity],{i,1, Length[{a*q,a*q/(d*e),a*q/(d*f),a*q/(e*f)}]}], Product[
    QPochhammer[Part[{a*q/d,a*q/e,a*q/f,a*q/(d*e*f)},i],q, Infinity],{i,1,
    Length[{a*q/d,a*q/e,a*q/f,a*q/(d*e*f)}]}]* QHypergeometricPFQ[{a*q/(b*c
    ),d,e,f},{a*q/b,a*q/c,d*e*f/a},q,q]
 + Divide[Product[QPochhammer[Part[{a*q,a*q/(b*c),d,e,f,(a)^(2)*(q)^(2)/(b*d*
    e*f),(a)^(2)*(q)^(2)/(c*d*e*f)},i],q, Infinity],{i,1, Length[{a*q,a*q/(b
    *c),d,e,f,(a)^(2)*(q)^(2)/(b*d*e*f),(a)^(2)*(q)^(2)/(c*d*e*f)}]}],
    Product[QPochhammer[Part[{a*q/b,a*q/c,a*q/d,a*q/e,a*q/f,(a)^(2)*(q)^(2)/
    (b*c*d*e*f),d*e*f/(a*q)},i],q, Infinity],{i,1, Length[{a*q/b,a*q/c,a*q/
    d,a*q/e, a*q/f,(a)^(2)*(q)^(2)/(b*c*d*e*f),d*e*f/(a*q)}]}]]
 * QHypergeometricPFQ[{a*q/(d*e),a*q/(d*f), a*q/(e*f),(a)^(2)*(q)^(2)/(b*c*d*
    e*f)},{(a)^(2)*(q)^(2)/(b*d*e*f),(a)^(2)*(q)^(2)/(c*d*e*f), a*(q)^(2)/(d
    *e*f)},q,q]
```

Linebreaks are manually added to improve readability.

[5] An abbreviation may refer to a single variable. For instance, def may refers to a variable *definition* earlier in the article. However, an interpretation of three individual variables (i.e., d, e, and f) is often more reasonable.

[6] The letter e is commonly used for the Euler's number but can also simply refer to a Latin letter variable.

Free Variables

`a, b, c, d, e, f, q`

Abbreviation Warning

Found a potential abbreviation: `def`. This program cannot translate abbreviations. Hence the expression was interpreted as a sequence of multiplications, e.g., `etc -> e*t*c`.

Math Constant e

You used a typical letter for a constant (the mathematical constant e, known as *Napier's constant* with a value of $2.71828182845\ldots$). We keep it like it is! But you should know that Mathematica uses `E` for this constant. If you want to translate it as the constant, use the corresponding DLMF macro `\expe`.

Translation Information for ${}_r\phi_s$

Name: Basic hypergeometric (or q-hypergeometric) function
Example: \qgenhyperphi{r}{s}@@@{a_1,...,a_r}{b_1 ,..., b_s}{q}{z}

Translation Pattern: `QHypergeometricPFQ[{$2},{$3},$4,$5]`

Relevant Links
DLMF: http://dlmf.nist.gov/17.4#E1
Mathematica: https://reference.wolfram.com/language/ref/QHypergeometricPFQ.html

Translation Information for $(x;q)_n$

Name: q-Multi-Pochhammer symbol
Example: \qmultiPochhammersym{a_1,\ldots,a_n}{q}{n}

Translation pattern unavailable. Use alternative translation pattern instead.
Alternative Translation Pattern:
`Product[QPochhammer[Part[{$0},i],$1,$2],{i,1,Length[{$0}]}]`

Relevant Links
DLMF: http://dlmf.nist.gov/17.2.E5
Mathematica: unavailable

Figure 6.3: Translation information about the translation of Bailey's transformation of very-well-poised ${}_8\phi_7$ to Mathematica of equation (6.3) with LaCaST (see also the DLMF [98, (17.9.16)]). Since the q-Multi-Pochhammer symbol is not natively supported in Mathematica, LaCaST uses the alternative translation pattern based on the definition of the function [98, (17.2.5)]. The information about abbreviations and name of constants are fetched from the POM tagger's lexicon files [402] that LaCaST relies on.

6.2 Contributions and Impact of the Thesis

This thesis made three main contributions:

1. It presented a novel semantification process that replaces MOI with semantic enhanced LaTeX macros based on information extracted from a near-by textual context and a common knowledge database;
2. It demonstrated the first context-sensitive LaTeX to CAS translator LaCaST, which performs manually crafted rule-based translations to multiple CAS syntaxes from semantic LaTeX expressions generated by the previously developed semantification process; and
3. It showcased the efficiency and usability of LaCaST with a novel evaluation approach that symbolically and numerically verifies equations from a source database, e.g., the DLMF or Wikipedia, with the power of CAS.

These contributions resulted in 14 peer-reviewed publications [1, 2, 3, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18] with 2 doctoral program participations [4, 5], and 2 invited talks [6, 7]. The publications were 63 times cited[7] overall. In addition, 3,782 commits[8] to a variety of different open source projects were performed during the time of the thesis. In the following, we briefly summarize the contributions of this thesis for each of the five research tasks that were defined in the introduction, Section 1.3.

Research Tasks I

Analyze the strengths and weaknesses of existing semantification approaches for translating mathematical expressions to computable formats.

Contributing Publications: [1, 9, 12, 18]

To analyze the strengths and weaknesses of existing translation tools, we performed a new evaluation on nine state-of-the-art LaTeX to MathML converters, including Mathematica as CAS. We developed a new benchmark for MathML, called MathMLben, to evaluate translations against a manually crafted golden dataset. All converters solely rely on the semantic information that can be retrieved from the structure of an expression, e.g., by pattern matching approaches. In addition, only three converters supported content MathML with an unsatisfactory accuracy.

The main identified weakness of all analyzed tools was the lack of taking local contextual information into account for the translation process. Through our evaluation, we were able to significantly improve LaTeXML translations by manually annotating LaTeX expressions with semantic information via semantic LaTeX macros. This performance improvement underlines the need for a semantification process that automatically performs semantic annotations based on information from a given context. The poor accuracy of all evaluated conversion tools showed, that translations from LaTeX over MathML to CAS have no advantages compared to other translation paths, e.g., over semantic LaTeX. Since, semantic LaTeX translations to Maple were successfully implemented with the first version of LaCaST, and the accuracy of LaTeXML significantly improved by semantic annotations with semantic macros, we choosed semantic LaTeX as an intermediate format to translate expressions from LaTeX to CAS syntaxes.

[7] According to Google Scholar evaluated on 2021-09-16.
[8] According to github.com evaluated on 2021-19-08.

Research Tasks II

Develop a semantification process that will improve on the weaknesses of current approaches.

Contributing Publications: [10, 14, 15]

We accomplished this research task by developing a novel semantification process that relies on the textual information in the nearby context of a formula combined with a set of standard knowledge information. As a first attempt at creating a new common knowledge dataset, we studied math embeddings (i.e., word embeddings for mathematical expressions) to retrieve common co-occurrences between math objects and textural descriptions. This attempt was unsuccessful due to the flexible and nested nature of mathematical notations. Instead, we relied on the DLMF and the lexicon files of the POM tagger for our common knowledge database.

To analyze the nearby textual context, we retrieve noun phrases as descriptions for mathematical objects. Since the concept of mathematical objects was barely studied in the past, we introduced a new concept of so-called Mathematical Objects of Interest (MOI). The idea behind MOI is that every mathematical subexpression is potentially meaningful. Previous research efforts in the MathIR area only focused either on single identifiers or entire mathematical expressions, ignoring the interconnectivity between subexpressions in math formulae. The new MOI concept has proven successful on a variety of different tasks in MathIR. Consequentially, we developed a novel semantification process based on MOI. The semantification process generates a mathematical dependency graph of MOI and annotates each MOI with textual descriptions from their textual context. The dependencies provide access to relevant descriptions of an MOI and its subexpression (which are also MOI). With these descriptions, we retrieve semantic LaTeX macros from the DLMF that replace the original LaTeX subexpression. This semantification gradually transforms the original LaTeX expression into the semantically enhance semantic LaTeX encoding.

Research Tasks III

Implement a system for the automated semantification of mathematical expressions in scientific documents.

Contributing Publications: [11, 16, 17]

We achieved this research task by relying on the results of several previous research projects. The nearby textual analysis was performed with a modified version of the mathosphere system [279, 329, 330] which was initially designed to retrieve identifier-definiens pairs from a mathematical text. We updated the system to retrieve facts, i.e., pairs of MOI and textual descriptions, from a given text. We further generated the dependency graph of MOI in a document with the approaches outlined by Kristianto et al. [214]. Finally, we extended the POM tagger [402] to create tree patterns of semantic LaTeX macros from the DLMF.

This new semantification pipeline is performed in four steps. First, we analyze a given text, e.g., a Wikipedia page, to identify all MOI and noun phrases. Second, we build a mathematical dependency graph by defining directed edges between MOI if an MOI is a subexpression of another MOI. Further, each MOI is annotated with noun phrases taken from the same sentence the MOI appears in (including subexpression appearances). Third, we use the noun phrases of an MOI and the noun phrases of dependant MOI to determine replacement patterns to semantic

DLMF LaTeX macros. This replaces generic LaTeX subexpressions by semantic LaTeX macros. Fourth, the resulted semantic LaTeX expression will be translated towards the target CAS syntax by LaCaST (see the next research task).

For this research task, we also elaborated the capabilities of machine translation techniques. We discovered that our sequence-to-sequence model outperforms other machine translation models and achieves very good scores on undoing conversions of rule-based translators, such as Mathematica's LaTeX export function and LaTeXML translations of semantic LaTeX. However, we also show that our machine translation are unreliable on other general mathematical expressions that have not been generated by Mathematica or LaTeXML. We constitute that our machine translation model in its current form is, therefore, unsuitable for performing LaTeX to CAS translations.

Research Tasks IV

Implement an extension of the system to provide translations to computer algebra systems.

Contributing Publications: [3, 11, 13]

We accomplished the research task **IV** with the previously developed translator LaCaST. LaCaST was originally implemented as a rule-based translator for semantic LaTeX expressions in the DLMF and solely supported Maple as a target CAS. In this thesis, we extended LaCaST to support more CAS, especially focusing our efforts on Mathematica and (more recently) on SymPy. Further, we implemented additional semantification heuristics in order to correctly translate the mathematical operators for integrals, sums, products, and limits. With a study of the prime notations (for derivatives) in the DLMF, we further expand the coverage of LaCaST translations specifically for functions in the DLMF.

Lastly, we added the previously developed semantification pipeline to LaCaST which finally turns LaCaST into the first context-sensitive LaTeX to CAS translator. LaCaST is currently able to parse the context of a given English Wikipedia article. However, the pipeline currently allows analyzing any English text document that encodes mathematical formulae in LaTeX.

Research Tasks V

Evaluate the effectiveness of the developed semantification and translation system.

Contributing Publications: [2, 8, 11]

We accomplished the research task **V** with a combination of a qualitative and quantitative evaluation pipeline. For the qualitative evaluation of LaCaST, we manually crafted a benchmark dataset of 95 equations from English Wikipedia articles about OPSF. LaCaST was able to correctly transform LaTeX into semantic LaTeX for 48% of the equations and achieved 27% correct translations to Mathematica overall. In comparison, Mathematica's LaTeX import function correctly imported 9% of the expressions and a human annotator was able to translate 81% of the equations to Mathematica. We were able to show that a theoretical concept of definition detection and a domain-dependent common knowledge database (rather than a fixed common knowledge database) would increase the number of correct translations via LaCaST to Mathematica from 27% to 49%. Performing translations from the semantic LaTeX dataset DLMF underlines that the most

pressing issue still remains in a reliable semantification pipeline. LaCaST was able to translate 62.9% and 72% of all DLMF equations to Maple and Mathematica, respectively. To evaluate the semantification, we further analyzed LaCaST's ability to retrieve relevant descriptions from the context of a given formula and achieved an F1 score of .495 (.508 precision and .483 recall respecitvely).

Further, we developed a new concept to verify a translated expression based on the assumption that a correct equation in the source database must remain valid after translating to the target system. The computational ability of CAS allows us to perform verification checks on translated equations enable us to evaluate large datasets. In particular, we performed two novel approaches, symbolic and numeric evaluations. The symbolic evaluation tries to simplify the difference between the left- and right-hand sides of an equation to zero. The numeric evaluation performs actual numeric calculations on test values and numerically checks the equivalence of an equation's left- and right-hand sides. On the DLMF, LaCaST was able to symbolically verify 26.3% and 26.2% translations to Maple and Mathematica, respectively. Symbolically unverified expressions were further evaluated numerically. LaCaST achieved a numeric verification rate of 26.7% for Maple and 22.6% for Mathematica. In combination, both evaluation techniques verified 43.3% translations for Maple and 42.9% translations for Mathematica. Performing the same techniques on the Wikipedia articles resulted in an overall evaluation of 18.1% and 23.6% for Maple and Mathematica respectively.

The novel verification approach has proven to be very successful and even identified issues in the source database, i.e., Wikipedia articles and the DLMF, and bugs in the commercial target CAS, Maple and Mathematica. With the automatic evaluations from LaCaST, we identified bugs regarding integrals and the variable extraction function in Mathematica, discovered numerous minor issues in the DLMF including a sign error and incorrect semantic annotations, and detected a malicious edit in the Wikipedia edit history in the domain of OPSF. The errors in the Mathematica and the DLMF has been reported and mostly fixed[9]. An overview of the reports are available in Appendix D available in the electronic supplementary material.

6.3 Future Work

The research advances in MathIR and the development of LaCaST in this thesis motivates several follow-up projects. Current plans include to incorporate LaCaST into the DLMF for providing translations, automatic evaluation results, and peculiarities compared to multiple CAS for each equation. Additionally, plans are made for including LaCaST as a translation-as-a-service endpoint. The developed semantification process is also planned to find its way into MediaWiki to semantically enhance mathematical content in Wikipedia pages. LaCaST has not been open source due to its dependency to the POM tagger [402] and the semantic LaTeX macros [260], when the research on this thesis took place. Since February 2022, the source code is publicly available at https://github.com/gipplab/LaCASt.

In this section, we provide a brief overview of four specific projects for our future work. Section 6.3.1 discusses ideas to improve the shortcomings of LaCaST and related open research questions that motivate follow-up projects. Section 6.3.2 discusses how we plan to improve existing LaTeX to MathML converters with our semantification pipeline. Section 6.3.3 explains the Wikipedia extension for semantic enhanced mathematical expressions. This section was

[9] As of 2021-10-01.

published as a poster together with M. Schubotz [17]. In Section 6.3.4, we discuss a potential multilingual support of LaCaST. The multilingual research project will be part of a DAAD-funded post-doctoral scholarship.

6.3.1 Improved Translation Pipeline

The performance of the presented context-sensitive translator LaCaST leaves some room for improvements and even motivates entire new research projects. The most pressing shortcoming of LaCaST is the lack of generalizability beyond OPSF. The main reason for this shortcoming is the open research task of identifying equations as definitions. Recent advances of definition detections in natural languages [111, 134, 183, 370] may pave the way to a reliable classification of mathematical equations in the near future. An equation tagged as definition enables correct translations of dependant formulae in the same document. This enables LaCaST to translate general functions, such as $f(x)$, which are not directly defined in the CAS. Further, a definition detection of equations may help to build a comprehensive definition library across entire scientific corpora with numerous use cases for the mathematical community.

Another issue that remains woefully neglected by our translation tool is the positioning of branch cuts for multi-valued functions. The main reason for that shortcoming is that there is no database or standard available to store and describe branch cuts uniformly across multiple systems and libraries. While branch cuts are openly discussed and presented, their description is often embedded in natural language text descriptions, which harms the machine readability and consequentially the accessibility of the information. In order to consider branch cut positions for a more reliable translation, we need to develop a standard to describe positions uniformly in a machine-readable format. Subsequently, a manual analysis across multiple CAS and libraries, including the DLMF, is required to build a comprehensive database that stores this information. Translation tools may finally use the database to either provide additional information during a translation process or automatically perform alternative translations based on the stored positioning of branch cuts. The latter, while considerately more difficult, is beneficial to improve the verification of equations in the DLMF further.

Lastly, the powerful numeric evaluation approach used to verify a translated expression heavily relies on the chosen numeric test values. LaCaST currently uses the same ten numeric test values for all tested equations and filters invalid combinations regarding the constraints. While easy to maintain for many test cases, this approach ignores function-specific attributes such as domains, branch cuts, singularities, and other essential characteristics. Testing functions on specific *values of interest* enable several valuable applications. For example, numeric calculations specifically along the defined branch cuts of the involved functions could help to automatically detect definition disparity on branch cuts between the systems, e.g., evaluating $\operatorname{arccot}(-1)$. In addition, testing values of interest potentially increases the trustworthiness of a numerically verified equation significantly. However, no study about values of interest for functions has been undertaken to the best of our knowledge. It might even be questioned if such values exist for all functions in the DLMF. Further, the value of interest may change depending on the actual argument of the functions. In this case, LaCaST would need to automatically adjust the tested values accordingly, which increases the complexity of the task even further.

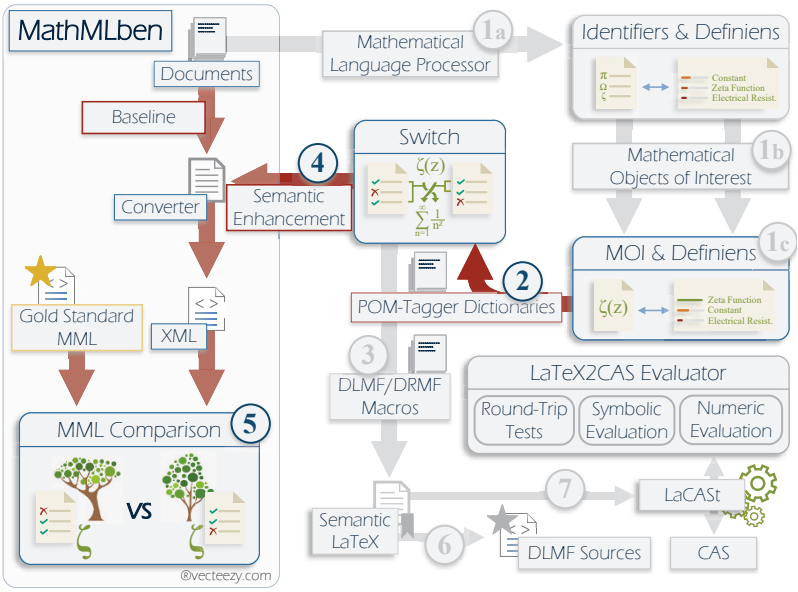


Figure 6.4: Proposed pipeline to improve existing LaTeX to MathML converters.

6.3.2 Improve LaTeX to MathML Converters

As we have described in Section 3.3 in Chapter 3, our outlined translation pipeline can also be used to improve existing LaTeX to MathML translators. Figure 6.4 highlights this additional remaining pipeline. In this thesis, we primarily focused on the main pipeline along (1), (2), (3), and (7). However, the information we gathered in the steps (1) and (2) can also be forwarded to a MathML converter. In Chapter 2, we developed MathMLben, the MathML benchmark, with the help of LaTeXML, a LaTeX to XML converter. We manually added semantic annotations to the source expression in order to improve the conversion by LaTeXML. For example, the first entry contains the expression about Van der Waerden numbers $W(2, k)$. Here, we manually added the link to the corresponding Wikidata ID Q7913892 for W, which (together with additional scripts) enabled LaTeXML to generate a proper, annotated content MathML representation of the expression.

We can now use our semantification steps to automate the annotation process. In combination with existing Wikidata entity linking approaches [320, 321, 327], we can also annotate the original expressions with Wikidata IDs as we did manually for MathMLben. While this semantic enrichment process through Wikidata IDs was developed specifically for LaTeXML, other LaTeX to MathML converters can also profit from such annotations. SnuggleTeX, for example, is a LaTeX to XML converter that allows users to pre-define the semantics of symbols in order to improve the so-called *upconversion*[10] process. One option in particular is the `assumeSymbol` command.

[10]SnuggleTeX uses this term for referring to a conversion process that requires semantic enrichment steps, e.g., from LaTeX to content MathML or Maxima syntax.

Besides annotating single symbols, e.g., via

```
\assumeSymbol{e}{exponentialNumber} $e$,
```
(6.4)

we can also define generic functions, such as

```
\assumeSymmbol{f_{n_k}}{function} $f_{n_k}(x)$.
```
(6.5)

These pre-defined assumptions enable SnuggleTeX to perform a correct conversion to content MathML or the CAS Maxima.

6.3.3 Enhanced Formulae in Wikipedia

Recently[11], we deployed a feature that enables enhancing mathematical formulae in Wikipedia with semantics from Wikidata [308]. For instance, the wikitext code

Annotated Wikitext Formula

```
<math qid="Q35875">E=mc^2</math>
```

now connects the formula $E = mc^2$ to the corresponding Wikidata item by creating a hyperlink from the formula to the special page shown in Figure 6.5[12]. The special page displays the formulae together with its name, description, and type, which the page fetches from Wikidata. This information is available for most formulae in all languages. Moreover, the page displays elements of the formula modeled as `has part` annotations of the Wikidata item.

The `has part` annotation is not limited to individual identifiers but also applicable to complex terms, such as $\frac{1}{2}m_0v^2$, i.e., the kinetic energy approximation for slow velocities[13]. For example, we demonstrated using the annotation for the Grothendieck–Riemann–Roch theorem[14]

$$\mathrm{ch}(f_!\mathcal{F}^\bullet)\mathrm{td}(Y) = f_*(\mathrm{ch}(\mathcal{F}^\bullet)\mathrm{td}(X)). \tag{6.6}$$

The smooth quasi-projective schemes X and Y in the theorem lack Wikipedia articles. However, dedicated articles on *quasi-projective variety* and *smooth scheme* exist. We proposed modeling this situation by creating the new Wikidata item *smooth quasi-projective scheme*[15], which links to the existing articles as subclasses. To create a clickable link from the Wikidata item to Wikipedia, we could create a new Wikipedia article on *smooth quasi-projective scheme*. Alternatively, we could add a new section on *smooth quasi-projective scheme* to the article on *quasi-projective variety* and create a redirect from the Wikidata item to the new section.

Aside from implementing the new feature, defining a decision-making process for the integration of math rendering features into Wikipedia was equally important. For this purpose, we

[11] A. Greiner-Petter: *Link Wikipedia Articles from Specialpage Math Formula Information*, GitHub Commit to `mediawiki-extensions-math` on 27th November 2020: https://github.com/wikimedia/mediawiki-extensions-Math/commit/912866b976fbdcd94fda3062244d23a34c5e7a76

[12] https://en.wikipedia.org/wiki/Special:MathWikibase?qid=Q35875 [accessed 2021-08-18]

[13] https://en.wikipedia.org/w/index.php?oldid=939835125#Mass\T1\textendashvelocity_relationship [accessed 2021-08-18]

[14] https://en.wikipedia.org/w/index.php?title=Special:MathWikibase&qid=Q1899432 [accessed 2021-08-18]

[15] https://www.wikidata.org/wiki/Q85397895 [accessed 2021-08-18]

founded the Wikimedia Community Group Math[16] as an international steering committee with authority to decide on future features of the math rendering component of Wikipedia.

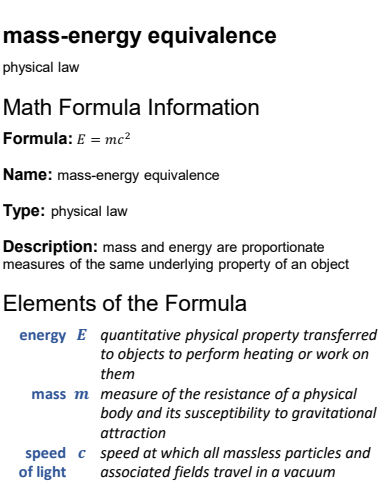


Figure 6.5: Semantic enhancement of the formula $E = mc^2$.

The new feature helps Wikipedia users to better understand the meaning of mathematical formulae by providing details on the elements of formulae. Because the new feature is available in all language editions of Wikipedia, all users benefit from the improvement. Rolling out the feature for all languages was important to us because using Wikipedia for more in-depth investigations is significantly more prevalent in languages other than English [226]. Nevertheless, also in the English Wikipedia, fewer than one percent of the articles have a quality rating of good or higher [299]. Providing better tool support to editors can help in raising the quality of articles. In that regard, our semantic enhancements of mathematical formulae will flank other semi-automated methods, such as recommending sections [299] and related articles [337].

To stimulate the wide-spread adoption of semantic annotations for mathematical formulae, we are currently working on tools that support editors in creating the annotations and, therefore, successively determing the ground truth of mathematics in Wikipedia. With `AnnoMathTex` [319], we are developing a tool that facilitates annotating mathematical formulae by providing a graphical user interface that includes machine learning assisted suggestions [14] for annotations. Moreover, we will integrate a field into the visual wikitext editor that will suggest Wikipedia authors to link the Wikidata id of a formula if the formula is in the Wikidata database. Improved tool support will particularly enable smaller language editions of Wikipedia to benefit from the new feature because the annotations performed in any language will be available in all languages automatically.

Additionally, our recent advances with LaCaST on the Wikipedia dataset allows us to automatically verify equations in Wikipedia to some degree. We currently working on a system that automatically triggers the verification engine on edits in mathematical content. This would allow us to generate a live feed of verified and not verified mathematical edits in the entire Wikipedia. While this presumably generates a lot of interesting data for numerous of projects, it will also serve as a proof-of-concept to integrate the system into existing quality control mechanisms. On the long run, we hope to integrate the verification technique into the existing *Objective Revision Evaluation Service* (ORES) [144], such as other recently ermeged ORES extensions [359, 401].

[16] https://meta.wikimedia.org/wiki/Wikimedia_Community_User_Group_Math [accessed 2021-08-18]

6.3.4 Language Independence

The multilingual aspect of our translator becomes more and more important with the focus on Wikipedia. Since Wikipedia is a multilingual encyclopedia, providing a language-independent semantification process is a desired task. In general, the concept of our developed semantification approach is language independent. The pipeline relies on a POS tagger to tag tokens and generate parse trees of the sentences. The score of an MOI-description pair is calculated based on the distance between both tokens in the parse tree. Consequentially, we can presume that our semantification pipeline works for other languages too, as long as there is a reliable POS tagger for that language available. However, we already noticed minor issues with the well-developed CoreNLP's POS tagger for the English language when using the MLP approach. As a reminder, the MLP approach suggested masking mathematical elements by placeholders before using a POS tagger on the sentence. For example, in the following sentence

Example sentence including math

```
1 The Jacobi polynomial P_n^(α,β)(x) is an orthogonal polynomial.
```

the mathematical expressions is replaced by a placeholder `MATH_1`.

Example sentence with masked math

```
1 The Jacobi polynomial MATH_1 is an orthogonal polynomial.
```

While this approach works well in many cases, in this particular example, CoreNLP's POS tagger[17] tags both `polynomial` tokens as adjectives (`JJ`) while both should be tagged as nouns (`NN`).

The underlying issue is that the MLP approach presumes math expressions to represent noun tokens. However, the *mathematical language* is generally more complex compared to that simple scheme [138]. This language can become quite different from general natural language communication. The mathematical language introduces a technical terminology with entirely new terms, such as '*functor*', changes the meaning of existing vocabulary, such as '*group*' or '*ring*', and even define entire phrases to represent math concepts, such as '*without loss of generality*' or '*almost surely*'. All these specifics need to be adopted by a POS tagger. Math notation is often part of a natural language sentence but does not necessarily represent a logical token. In addition, we presume that mathematical expressions are generally language-independent. However, its notation style may change from language to language, even for simple cases. For example, while the US or Germany uses $\geq$ to express a greater or equal relation, the notation $\geqq$ is more common in Japan. Considering the sheer amount of different math notations, it might not be obvious to a student from Japan that $\geq$ and $\geqq$ refer to the same relation. Yet, these symbols are so basic that most authors, even in educational literature, would probably not explicitly declare their meaning in the context. This issue grows with a more and more educated audience. For example, math educational books written for math students in universities rarely mention the specific meanings of logic symbols (e.g. $\wedge$, $\vee$), quantifiers (e.g. $\forall$, $\exists$), or set notations (e.g. $\cap$ and $\cup$).

[17] Tested with CoreNLP's version 4.2.2.

Unfortunately, the multilingual aspects of mathematics have barely been studied in the past. D. Halbach [143] recently tried to take advantage of the multilingual versions of Wikipedia articles to identify defining formulae of that article. A defining formula of an article is the mathematical expression that is the main subject of that article. For example, $P_n^{(\alpha,\beta)}(x)$ can be considered as the defining formula of the article about Jacobi polynomials. D. Halbach assumed that a mathematical expression that appears in multiple language versions of the same article is a good candidate for such a defining formula. Unfortunately, it turned out that different languages tend to use different visualizations of the same formula. For example, he showed that Schwarz's theorem in the Polish, English, German and French Wikipedia articles use different mathematical formulae for the same concept. This result indicates that the semantification approach we developed in this thesis may not be easily generalized for other languages. In addition, there is no POS tagger available that is specialized in mathematical content.

In combination with researchers from the National Institute of Standards and Technology (NIST) in the US, the National Institute of Informatics (NII) in Japan, and the University of Wuppertal in Germany, we plan to study the multilingual aspects of mathematical languages to analyze language-specific notation and declaration differences. This project is part of a post-doctoral DAAD scholarship and includes training a math-specific NLP model for better POS tagging of mathematical content articles.

BACK MATTER

Glossary

Symbols

$X = \{f | f \in \mathcal{D} \cup \mathcal{K} \wedge (f \in \mathcal{K} \Rightarrow f \notin \mathcal{D})\}$
: Our definition of a mathematical context X defined in (4.2) on page 104. A context is a set of facts f in a document $\mathcal{D}$ and a set of common knowledge facts $\mathcal{K}$ so that document facts overwrite common knowledge facts. 106, 107, 138

$\mathcal{L}_C$ **— Mathematical Content Languages**
: Denotes mathematical content languages (CL), such as semantic LaTeX, content MathML, or CAS syntaxes. 106, 107, 112, 134, 137, 138

$\mathcal{L}_M$ **— Computer Algebra System Languages**
: Refers to CAS languages in general, such as the syntax of Mathematica, Maple, or SymPy inputs.. 107, 109, 110

$\mathcal{L}_P$ **— Mathematical Presentation Languages**
: Denotes mathematical presentational languages (PL), such as presentation MathML or LaTeX. 106, 107, 109–112, 134, 138

$$\mathrm{mBM25}(t,d) = \max_{d \in D} \frac{(k+1)\,\mathrm{IDF}(t)\,\mathrm{ITF}(t,d)\,\mathrm{TF}(t,d)}{\max\limits_{t' \in d|_{c(t)}} \mathrm{TF}(t',d) + k\left(1 - b + \frac{b\,\mathrm{AVG}_{\mathrm{DL}}}{|d|\,\mathrm{AVG}_C}\right)}$$
: Our mathematical BM25 ranking to measure the importance of a given MOI t in a document $d \in D$ which is part of a corpora D. $\mathrm{IDF}(t)$ is the inverse-document frequency, $\mathrm{ITF}(t,d)$ the inverse-term frequency of t in d, $\mathrm{TF}(t,d)$ the term frequency of t in d, $\mathrm{AVG}_{\mathrm{DL}}$ the average document length (number of terms) in D, AVG_C the average complexity of terms in D, $c(t)$ the complexity of t, and b, k are parameters. 85

$\mathrm{s}_{\mathrm{DLMF}}(r_f)$**:**
: The probability score for a replacement rule $r_f = m \to \widetilde{m}$. This score is the probability that $\widetilde{m}$ is rendered as m in the DLMF. For example, the general hypergeometric function never omits arguments, such as in ${}_2F_1(z)$ in the DLMF. Hence, the probability of ${}_2F_1(z)$ is 0. In contrast, in 19.7%, the function uses the linear rendered form ${}_2F_1(a,b;c;z)$. 114, 137

$\mathrm{s}_{\mathrm{ES}}(f) = \mathrm{s}_{\mathrm{ES}}(\mathrm{MLP}, \mathrm{MC})$
: The normalized Elasticsearch score for a retrieved semantic macro $\widetilde{m}$ for the given $\mathrm{MC} \in f$. This score is higher if MC better matches the description of the semantic macro $\widetilde{m}$. Since ES provide absolute scores, this score is normalized to the best fitting hit, i.e., the first retrieved result is always scored 1. 113, 114

$\mathrm{s}_{\mathrm{MLP}}(f) = \mathrm{s}_{\mathrm{MLP}}(\mathrm{MLP}, \mathrm{MC})$
: The score of the MLP engine [330] for a given fact f which depends on (1) the distance between the MOI and its first occurrence in the document $\mathcal{D}$, (2) the distance in the

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4

natural language syntax tree between the MOI and the MC, and (3) if the MOI and MC matches pre-defined patterns. 112–114, 137

$\mathrm{t}(e, X) = \mathrm{t}_m(\mathrm{t}_s(e, X))$
Our translator function follows a two step strategy of which the first step is a semantification $\mathrm{t}_s(e, X)$ followed by a rule-based transformation $\mathrm{t}_m(e)$. 106, 107, 134, 138

$\mathrm{t}_m(e) = g_{r_1} \circ \cdots \circ g_{r_n}(e)$
A rule-based translation function that performs translations on a set of rules $r_k \in \mathcal{R}_{C_2}^{C_1}, k = 1, \ldots, n$ from a content language C_1 to another content language C_2. Similar to the semantification function, it performs graph transformations g_r based on the rules. Example implementations are LaCasT or SymPy's `latex2sympy` function. 106, 107

$\mathrm{t}_s(e, X) = g_{f_1} \circ \cdots \circ g_{f_n}(e)$
A fact-based semantification translation function takes an expression e and a context X to perform a series of graph transformations g_f defined by the facts f to semantically enhance subtrees of e. 106–108, 137

A

AI — Artificial Intelligence
A broad research field with the focus on machine (artificial) intelligence. 103, 142

AJIM — Aslib Journal of Information Management
An international journal with an 5-year IF of 2.653 in library and information science with focus on information and data management. According to https://academic-accelerator.com/5-Year-Impact-Factor/Aslib-Journal-of-Information-Management [accessed 2021-10-01] it is placed 33 of 227 journals in the field of library and information sciences. 9, 15, 163

arXiv:
Is a pre-print archive for scientific papers in a variaty of different fields, such as mathematics, physics, or computer science. See arxiv.org [accessed 2021-10-01] for more information. 40, 62–66, 68, 70, 71, 73–75, 78–84, 86, 91, 92, 99, 101, 103, 144, 192

arXMLiv:
An HTML5 (including MathML) dataset based on the arXiv articles. The HTML5 was generated via LaTeXML and is available at https://sigmathling.kwarc.info/resources/arxmliv-dataset-2020/ [accessed 2021-10-01] [132]. 65, 74

Axiom:
Is a free, general-purpose CAS first developed by IBM around 1965 (named *Scratchpad* at that time). Since 2001, Axiom is open source under a moified BSD license and available on GitHub at https://github.com/daly/axiom [accessed 2021-10-01] [173]. 5, 34, 35

B

BLEU — Bilingual Evaluation Understudy
Is an algorithm to measure the quality of translated texts first described by Papineni et al. [282] in 2001. The algorithm presumes the closer (more sharing n-grams) a translation is to human translations the better it is. 14, 99, 100, 134, 146

BM25 — Okapi BM25
Is a ranking function to calculate the relevance of results in a search engine [310]. The underlying idea of BM25 is that words that appear regularly only in a few documents are more *important* for that document than words that appear everywhere across the entire corpora. 12, 73, 83, 85, 113, 145

C

CAS — Computer Algebra System(s)
A mathematical software that allows one to work with mathematical expressions, e.g., by manipulating, computing, or ploting them. The acronym CAS, in this thesis, is referring to a single or multiple systems depending on the context. ix, xi, xii, 1–8, 10, 13–15, 19–22, 24–36, 38, 40–43, 47, 52, 55, 58–60, 93, 95, 97, 103–108, 111, 115–120, 123–129, 131–136, 138, 139, 141, 143–150, 152, 154–156, 158, 163–165, 168, 171, 174, 175, 180, 193

CD — Content Dictionary
Content dictionaries are structured documents that contain the definition of mathematical concepts. See the OpenMath specification for more details [53]. 23–26, 31, 57, 58, 143

CICM — Conference on Intelligent Computer Mathematics
An annual international conference on mathematical computation and information systems (has a CORE rank of C since 2021). 9, 10, 15, 116

CL — Content Language
Content languages are languages that encode mainly semantic (content) information, such as content MathML, OpenMath, or CAS syntaxes. 43

CLEF — Conference and Labs of the Evaluation Forum
An annual international conference for systematic evaluation of information access systems. 9

cMML — Content MathML
Content MathML encodes the meaning of mathematical notations. For more information see the explanations about MathML. 22, 23

CORE — Computing Research and Education Association of Australasia
Is an association of university departments that provide assessments of major conferences in the computing disciplines. The main categories are A* (flagship), A (excellent), B good to very good, and C for other ranked conferences that meet minimum standards, see http://portal.core.edu.au/conf-ranks/ [accessed 2021-10-01]. 8, 9

CoreNLP:
CoreNLP is a Java library for natural language processing tasks developed by Stanford NLP Group and includes tokenizer, POS taggers, lemmatizers and more [240]. 109, 110, 160, 185, 186

D

DBOW-PV — Distributed Bag-of-Words of Paragraph Vectors
An approach to embed entire paragraphs into single vectors introduced by Le and Mikolov [222]. 67–69

DL — Deep Learning
Is a broad family of machine learning methods that uses neural networks for learning features. 61

DLMF — Digital Library of Mathematical Functions
A digital version [98] of *NIST's Handbook of Mathematical Functions* [276]. The DLMF (or the book respectively) is a standard reference for OPSF and provides access to numerous of definitions, identities, plots, and more. ix, x, xii, 1, 4, 5, 8, 12, 14, 15, 17, 25, 28, 30–33, 35, 40, 46, 47, 49–51, 56, 58, 62, 63, 65, 66, 93–95, 97, 98, 100, 101, 103–109, 112–119, 121–126, 129–137, 139–142, 144–156, 163–165, 168, 174–183, 190–192

DML — Digital Mathematical Library
A general digital library that specifically focuses on mathematics. 63, 115–118, 123, 128, 132, 133, 148, 164

DRMF — Digital Library of Mathematical Formulae
An outgrowth of the DLMF project [77, 78]. 30, 32

E

EMNLP — Empirical Methods in Natural Language Processing
An annual international conference on natural language processing (has a CORE rank of A). 9

ES — Elasticsearch
A search engine written in Java that uses the open-source search engine library Apache Lucene, see https://www.elastic.co/ and https://lucene.apache.org/ [accessed 2021-07-02]. 86, 88, 113, 193

G

GUI — Graphical User Interface
A visual interface that allows for interacting with data or software. 48, 49

H

HTML — HyperText Markup Language
The standard markup language for web documents. 23, 74

I

ICMS — International Congress on Mathematical Software
A bi-annual congress that gathers the mathematicians, scientists and programmers who are interested in the development of mathematical sofware. 9, 13, 60

J

JCDL — Joint Conference on Digital Libraries
An annual major conference in the field of digital libraries (had a CORE rank of A* until it was unranked in 2021 because the CORE committee removed the entire digital library domain from their ranking scheme). 9, 10, 14, 19, 163, 166

L

LaCaSt — LaTeX to CAS translator

Is the name of the framework we developed in this thesis to translate mathematical LaTeX to CAS. The first version of LaCaSt was part of the author's Master's thesis and supported translations only from semantic LaTeX to Maple [3, 13]. Within this thesis, we extended LaCaSt by supporting general LaTeX [11] expressions and additional CAS [8], such as Mathematica. The source of LaCaSt is publicly available on https://github.com/gipplab/LaCASt since February 2022.. ix–xii, 7, 8, 10, 14–17, 28–30, 32, 58, 95, 100, 101, 105–107, 109–111, 114–119, 121, 122, 124–134, 139, 141, 144–152, 154–156, 159, 163, 168, 171, 174, 180, 191

LaTeX:

Is an extension of the typesetting system TeX used for document preparation. LaTeX provides additional macros on top of TeX allowing the writer to focus more on the content of a document rather than on the exact layout. Since this thesis focus on mathematical expressions in LaTeX, there is not much difference between TeX and LaTeX. ix, xi, 1–3, 5–8, 10, 13, 19–22, 24, 25, 27–35, 37–42, 45–54, 56–60, 74, 83, 88, 93, 94, 97–100, 102–108, 110, 112, 113, 116, 118, 121, 129, 132, 135, 138–141, 143–146, 152–154, 156, 157, 166, 174–180, 188–193

LaTeXML:

Is a tool developed by B. Miller to convert LaTeX documents to a variaty of other formats, such as XML or HTML. The tool can also be used to transform single mathematical LaTeX expressions to math specific formats, such as MathML, or image formats, such as SVG. More infomation can be found at *LaTeXML: A LaTeX to XML/HTML/MathML Converter*, https://dlmf.nist.gov/LaTeXML/ [accessed 2021-10-01]. 11, 32, 33, 38, 46–51, 53, 58, 74, 75, 77, 78, 83, 94, 98, 102, 143, 146, 152, 154, 157

M

Maple:

One of the major general-purpose CAS [36] developed by *Maplesoft*. If not stated otherwise, we refer to the version 2020.2. ix, xii, 1, 2, 4–8, 10, 15, 20, 21, 26, 28, 31, 32, 34, 35, 38, 43, 52, 58, 103, 104, 107–109, 115–120, 123–125, 127–136, 141, 143–145, 147–149, 152, 154, 155, 164, 165, 168, 169, 180, 189, 193

Mathematica:

One of the major general-purpose CAS [393] developed by *Wolfram Research*. If not stated otherwise, we refer to version 12.1.1. ix, xii, 1–6, 8, 10, 15, 20, 21, 26, 28–31, 35, 41, 42, 52, 97–105, 107–109, 114, 115, 117, 119, 124, 125, 127–136, 138–141, 143, 145–152, 154, 155, 164, 169–174, 180, 181, 189, 193

MathIR — Mathematical Information Retrieval

Is a sub-field of the Information Retrieval (IR) research area and as such focusing on obtaining information (mostly semantics) or retrieving relevant mathematical expressions. Note that MIR is another common acronym for mathematical information retrieval. In this thesis, we stick with the less overloaded and more precise abbreviation MathIR. ix, xi, 1, 6, 8, 11, 19, 39, 40, 54, 55, 59–63, 65, 71–73, 83, 105, 144, 148, 153, 155

MathML — Mathematical Markup Language
An XML structured standard for representing mathematical notations in web pages and other digital documents [169]. MathML allows to encode the meaning of mathematical notations to some degree, which is often referred to *content MathML*. In contrast, *presentational MathML* refers only on the visual encoding of math formulae. In case a math formula is encoded in presentational and content MathML at the same time, it is often called *parallel markup MathML*. 2, 4, 6–8, 10–12, 19–28, 32–35, 37, 39, 41, 43–47, 49–53, 57, 58, 62, 63, 65, 74–78, 92, 94, 105, 106, 117, 133, 143, 144, 148, 149, 152, 156–158, 166

MathMLben — MathML Benchmark
We developed MathMLben as a benchmark dataset for measuring the quality of MathML markup of mathematical formulae appearing in a textual context. See Section 2.3.2 on page 43 for further details. 10, 11, 45, 46, 51, 67, 94, 143, 148, 152, 157

MATLAB:
Is one of the major proprietary CAS with a specific focus on numeric computations developed by MathWorks. MATLAB is also the name of the underlying programming language the CAS MATLAB uses [164, 246]. 1, 5, 10, 35

Maxima:
Is an open source general-purpose CAS first released in 1982 (originally developed as a branch of the predecessor CAS Macsyma [264]) and is still actively maintained [324]. 2–4, 28, 29, 35, 157, 158

mBM25 — Mathematical Okapi BM25
Our extension of the BM25 score for mathematical expressions. 85, 88–90

MC — Mathematical Concept
Is a term referring to the concept that defines a mathematical expression including its visual appearance, underlying definition, constraints, domains, and other semantic information [9]. In the context of this thesis, we simplify this concept and presume that a name (or noun phrase) sufficiently specifies a concept so that the name (or noun phrase) is considered a representative MC. 106, 108–113, 137, 185

MEOM — Mathematically Essential Operator Metadata
Describes the metadata, i.e., argument(s) and bound variable(s), in sums, products, integrals, and limit operators. 120–122, 124, 128, 129

MFS — Mathematical Functions Site
A dataset of mathematical functions and relations maintained by Wolfram Research. The dataset is available at https://functions.wolfram.com/ [accessed 2021-10-01]. 98–102

MKM — Mathematical Knowledge Management
Is the general study of harvesting, maintaining, or managing mathematical information in literature and databases. 61, 62, 65

ML — Machine Learning
Is a computer science research field (often described as a subfield of artifical intelligence) with the relatively broad goal of making predections for unseen data based on trained data. 40, 61, 63, 69–71, 97, 103

MLP — Mathematical Language Processing
Mathematical language processing describes to the technical process of analyzing mathematical texts. A specific MLP task is the mapping of textual descriptions to components of mathematical formulae (see Schubotz et al. [279]), such as mathematical identifier. 61, 62, 65, 72, 110, 137, 160, 185, 186, 188

MOI — Mathematical Objects of Interest
Is a term referring to subexpressions in mathematical formulae with a specific meaning [9]. One can consider these parts as elements of general interest. 12, 13, 60, 73, 76, 86, 91–94, 106, 108–113, 136–138, 140, 144–146, 152–154, 160, 185–188, 191, 192

N

NIST — National Institute of Standards and Technology
An US government research institution. 30, 86, 161

NLP — Natural Language Processing
Is a research field with the focus on analyzing and processing natural languages in texts, images, videos, or audio formats. In this thesis, we mainly refer to natural language processing on texts rather than other multimedia formats. 39, 61, 64, 65, 72, 148, 161

NN — Neural Network
A graph network that aims to mathematically mimic biological neural networks. 61

O

OCR — Optical Character Recognition
Is a research field that focuses on identifying text and other symbols in images or videos. 28, 39, 99

OMDoc — Open Mathematical Document
Is a markup format developed by Michael Kohlhase [198] to describe mathematical documents. 22, 23, 26, 27, 32, 33, 36

OpenMath:
Is a markup language similar to MathML which uses an XML format to encode semantic information of mathematical expressions. The standard is maintained by the OpenMath Society. See http://openmath.org/ [accessed 2021-10-01] for more information. 6, 7, 19, 21–27, 34–37, 41, 58, 62, 106, 117, 133

OPSF — Orthogonal Polynomials and Special Functions
The set of orthogonal polynomials and special functions. Special functions are functions that, due to their general importance in certain fields, have specific names and standard notations. Note that there is no formal definition of the term *special function*. The *NIST Handbook of Mathematical Functions* [276] is a standard resource that covers a comprehensive set of functions (and orthogonal polynomials) that are widely accepted as *special*. 1, 3, 31–33, 35, 93, 101, 105, 111, 112, 114, 133, 140, 141, 145, 154–156, 185

ORES — Objective Revision Evaluation Service
A system used by Wikipedia to classify edits in potential damaging changes or changes made in good faith [144]. 103–105, 135, 136, 141, 142, 159

P

PL — Presentation Language
Presentation languages are languages that encode mainly visual information, such as LaTeX or presentation MathML. 43, 51

pMML — Presentation MathML
Presentational MathML refers only to the visual encoding of math formulae. For more information see the explanations about MathML. 22, 23, 75–77

POM — Part-of-Math
Is a LaTeX parser developed by Abdou Youssef [402] that tags each token in the parse tree with additional information similar to Part-of-Speech (POS) taggers in natural languages. 28, 32, 38, 52, 56, 93, 94, 110, 111, 151, 153, 155

POS — Part-of-Speech
Part-of-Speech tagging describes the process of tagging words in text with grammatical properties of the word. 45, 109, 160, 161, 185

R

Reduce:
Probably the first CAS from 1963 by Anthony C. Hearn [151] with a large impact on any other CAS that followed after. Since 2008, Reduce is open-source under the BSD license. 5, 34, 35, 164

S

Scientometrics:
An international journal with an 5-year IF of 3.702 for quantitative aspects of the science of science, communication in science and science policy. According to https://academic-accelerator.com/5-Year-Impact-Factor/Scientometrics [accessed 2021-10-01] it is placed 18 of 227 journals in the field of library and information sciences. 9, 19, 60

SCSCP — Symbolic Computation Software Composability Protocol
Is a protocol to communicate mathematical formulae between mathematical software, specifically CAS. It was developed as part of the *SCIEnce project* funded with 3 Million Euro by the Euorpean Union. More information can be found in the two publications about the project [119, 361]. 24, 26, 35, 36, 58

semantic LaTeX:
Refers to mathematical expressions that uses semantic macros developed by B. Miller for the DLMF. Each of these LaTeX macros is tied to a specific definition in the DLMF. Hence, a semantic LaTeX macro represents a unique unambiguous mathematical function as defined in the DLMF. An alternative name for semantic LaTeX is content LaTeX. ix, xi, 2, 7, 8, 10, 12, 15, 19, 22, 28, 30–33, 35, 38, 58, 93–95, 97–100, 115, 116, 133, 138, 143–146, 149, 152–155, 174

Semantification:
Refers to a process that semantically enhances mathematical expressions. Other authors

may also refer to this via *semantic enrichment* [71, 270, 402]. ix, xi, 7–11, 13, 24, 54, 57–59, 94, 95, 97, 103, 104, 106, 107, 115, 138, 144, 145, 147, 152–157, 160, 161, 193

SIGIR — Special Interest Group on Information Retrieval
A premier annual international conference on research and development in information retrieval (has a CORE rank of A*). 9, 11, 19

SnuggleTeX:
Is an open source Java program for converting LaTeX to XML, mainly MathML. SnuggleTeX is one of the rare converters that offer a semantic enrichment process to content MathML and the only LaTeX to CAS converter (supports Maxima) that is not part of a CAS itself [251]. SnuggleTeX is no longer developed with the most recent version 1.2.2 from 2010. See also https://www2.ph.ed.ac.uk/snuggletex [accessed 2021-10-01]. 2–4, 28, 29, 157, 158

STEM — Science, Technology, Engineering, and Mathematics
A group of academic disciplines. ix, xi, 2, 20, 27

sTeX — Semantic TeX
Semantic extension of TeX developed by Michael Kohlhase [200]. 19, 22, 30, 32, 33

SVG — Scalable Vector Graphics
An XML vector image format. 38, 49, 51, 52

SymPy:
An open-source CAS [252] written in Python. 2, 4, 5, 10, 15, 28–30, 34, 35, 146, 149, 154, 164, 174

T

t-SNE — t-distributed Stochastic Neighbor Embedding
Is a statistical method to visualize high-dimensional data in more convenient and easy to analyze one-, two-, or three-dimensional plots. t-SNE uses a nonlinear dimensional reduction method that tries to preserve structural groups of data. The method was first introduced by Hinton and Roweis [154]. 69, 70

TACAS — Tools and Alg. for the Construction and Analysis of Systems
TACAS is a forum for researchers, developers and users interested in rigorously based tools and algorithms for the construction and analysis of systems (has a CORE rank of A). 9, 15, 116, 163, 168, 180

TF-IDF — Term Frequency-Inverse Document Frequency
Is a statistical measure intend to reflect the importance of tokens (e.g., words) to a document in a larger corpus. The underlying assumption behind the measure is that frequent tokens across an entire corpus are less important compared to tokens that appear frequently in single documents but rarely somewhere else. The BM25 ranking function bases on the principle of TF-IDF scores. 79, 83, 85, 89, 90

TPAMI — Transactions on Pattern Analysis and Machine Intelligence
An IEEE published top monthly journal with an 5-year IF of 25.816 and a focus on pattern analysis and recognition and related fields. According to https://academic-accelerator.com/5-Year-Impact-Factor/jp/IEEE-Transactions-on-

Pattern-Analysis-and-Machine-Intelligence [accessed 2021-10-01] it is the top journal in three categories and 2nd in 2 additional categories. 9, 13, 16, 97, 116, 163

V

VMEXT — Visual Tool for Mathematical Expression Trees
A visualization tool for mathematical expression trees developed by Schubotz et al. [331]. 37, 46, 49, 50

W

W3C — World Wide Web Consortium
Is an international organization for standards for the world wide web. See www.w3.org [accessed 2021-06-09]. 23, 24

WED — Wolfram Engine for Developers
Is a free interface for the Wolfram engine (the engine behind Mathematica). Since 2019, this interface allows developers to interact and use most of Mathematica's core features without purchasing a full license. More information are available at https://www.wolfram.com/engine/ [accessed 2021-09-07] first. 117, 127, 131

WSDM — Web Search and Data Mining
A premier conference on web-inspired research involving search and data mining (has a CORE rank of A*). 9, 97

WWW — The Web Conference
An annual major conference with the focus on the world wide web (has a CORE rank of A*). 9, 12, 60

X

XML — Extensible Markup Language
A markup language mainly used for the representation of many different data structures. 20, 23–25, 27, 32, 33, 37, 43, 47, 51, 52, 74, 76, 77, 157

XSLT — Extensible Stylesheet Language (SLT) Transformation
A language to transform XML documents. 23, 24, 26

Z

zbMATH — Zentralblatt MATH
Is an international reviewing service for abstracts and articles in mathematics. zbMATH provide access to the abstracts and reviews of research articles mostly in the field of pure and applied mathematics, see also https://zbmath.org/ [accessed 2021-10-01]. 13, 73–75, 78–80, 83, 84, 86, 88–90, 92, 144, 148

BACK MATTER

Bibliography of Publications, Submissions & Talks

[1] T. Asakura, **A. Greiner-Petter**, A. Aizawa, and Y. Miyao. "Towards Grounding of Formulae". In: *Proceedings of the First Workshop on Scholarly Document Processing (SDP@EMNLP)*. Online: ACL, 2020, pp. 138–147. DOI: 10/gjzg2r. URL: https://www.aclweb.org/anthology/2020.sdp-1.16 (visited on 2021-08-02) (cit. on pp. 9, 54, 139, 150).

[2] H. S. Cohl, **A. Greiner-Petter**, and M. Schubotz. "Automated Symbolic and Numerical Testing of DLMF Formulae Using Computer Algebra Systems". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11006. Hagenberg, Austria: Springer International Publishing, 2018, pp. 39–52. DOI: 10/ggv8dn. URL: https://arxiv.org/abs/2109.08899 (visited on 2021-09-08) (cit. on pp. 6, 9, 14, 93, 102–106, 114–117, 120, 125, 126, 128, 130–133, 140, 146, 150, 152).

[3] H. S. Cohl, M. Schubotz, A. Youssef, **A. Greiner-Petter**, J. Gerhard, B. V. Saunders, M. A. McClain, J. Bang, and K. Chen. "Semantic Preserving Bijective Mappings of Mathematical Formulae Between Document Preparation Systems and Computer Algebra Systems". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 10383. Edinburgh, UK: Springer, 2017, pp. 115–131. DOI: 10.1007/978-3-319-62075-6_9. URL: https://arxiv.org/abs/2109.08655 (visited on 2021-09-08) (cit. on pp. 5, 9, 10, 14, 26, 27, 29, 36, 41, 49, 55, 105, 109, 114, 115, 117, 150, 152, 165).

[4] **A. Greiner-Petter**. "Automatic Mathematical Information Retrieval to Perform Translations up to Computer Algebra Systems". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 2307. Hagenberg, Austria: CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2307/DP1.pdf (cit. on p. 150).

[5] **A. Greiner-Petter**. "Automatic Mathematical Information Retrieval to Perform Translations up to Computer Algebra Systems". In: *Bulletin of IEEE Technical Committee on Digital Libraries (TCDL)* 15.1 (2019). URL: https://arxiv.org/pdf/2011.14616.pdf (cit. on p. 150).

[6] **A. Greiner-Petter**. "Comparative Verification of Digital Mathematical Libraries and Computer Algebra Systems". Invited talk. SIGMathLing - Seminar. 2021-03-29. URL: https://sigmathling.kwarc.info/seminar/ (cit. on p. 150).

[7] **A. Greiner-Petter**. "Semantic Preserving Translations between NIST's Digital Library of Mathematical Functions and Computer Algebra Systems". Invited talk. NIST Applied and Computational Mathematics Division (ACMD) Seminar Series. 2021-07-20. URL: https://www.nist.gov/itl/math/acmd-seminar-semantic-preserving-translations-between-nists-digital-library-mathematical (cit. on p. 150).

[8] **A. Greiner-Petter**, H. S. Cohl, A. Youssef, M. Schubotz, A. Trost, R. Dey, A. Aizawa, and B. Gipp. "Comparative Verification of the Digital Library of Mathematical Functions and Computer Algebra Systems". In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, (TACAS)*. Munich, Germany: Springer, 2022-04, pp. 87–105. DOI: 10.1007/978-3-030-99524-9_5. URL: https://arxiv.org/abs/2201.09488 (cit. on pp. 2, 6, 9, 15, 29, 34, 114, 146, 150, 152, 165).

[9] **A. Greiner-Petter**, T. Ruas, M. Schubotz, A. Aizawa, W. I. Grosky, and B. Gipp. "Why Machines Cannot Learn Mathematics, Yet". In: *Proc. Workshop on Bibliometric-Enhanced Information Retrieval and Natural Language Processing (BIRNDL@SIGIR)*. Vol. 2414. Paris, France: CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2414/paper14.pdf (cit. on pp. 9, 11, 18, 26, 35, 38, 59, 71, 150, 166, 167).

[10] **A. Greiner-Petter**, M. Schubotz, A. Aizawa, and B. Gipp. "Making Presentation Math Computable: Proposing a Context Sensitive Approach for Translating LaTeX to Computer Algebra Systems". In: *International Congress of Mathematical Software (ICMS)*. Vol. 12097. Braunschweig, Germany: Springer, 2020, pp. 335–341. DOI: 10/gn3sv2. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-030-52200-1_33.pdf (visited on 2021-07-30) (cit. on pp. 9, 13, 59, 115, 117, 124, 130, 150, 151).

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4

[11] **A. Greiner-Petter**, M. Schubotz, C. Breitinger, P. Scharpf, A. Aizawa, and B. Gipp. "Do the Math: Making Mathematics in Wikipedia Computable". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2022). In Press, pp. 1–12. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2022.3195261 (cit. on pp. 4, 9, 13, 21, 96, 102–106, 112, 114, 132–134, 140, 146, 150–152, 165).

[12] **A. Greiner-Petter**, M. Schubotz, H. S. Cohl, and B. Gipp. "MathTools: An Open API for Convenient MathML Handling". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11006. Hagenberg, Austria: Springer International Publishing, 2018, pp. 104–110. DOI: 10.1007/978-3-319-96812-4_9. URL: https://arxiv.org/abs/2109.08539 (visited on 2021-09-14) (cit. on pp. 9, 150).

[13] **A. Greiner-Petter**, M. Schubotz, H. S. Cohl, and B. Gipp. "Semantic Preserving Bijective Mappings for Expressions Involving Special Functions between Computer Algebra Systems and Document Preparation Systems". In: *Aslib Journal of Information Management* 71.3 (2019-05-20), pp. 415–439. ISSN: 2050-3806. DOI: 10/ggv8gx. URL: https://arxiv.org/abs/1906.11485 (visited on 2021-09-06) (cit. on pp. 5, 6, 9, 14, 15, 20, 21, 26, 27, 29, 36, 55, 98, 103–105, 109, 114, 132, 133, 137, 146, 150, 152, 165).

[14] **A. Greiner-Petter**, M. Schubotz, F. Müller, C. Breitinger, H. Cohl, A. Aizawa, and B. Gipp. "Discovering Mathematical Objects of Interest — A Study of Mathematical Notations". In: *Proceedings of The Web Conference (WWW)*. Taipei, Taiwan: ACM, 2020-04-20, pp. 1445–1456. DOI: 10/ggv8gw. URL: https://arxiv.org/abs/2002.02712 (visited on 2021-07-30) (cit. on pp. 6, 9, 12, 59, 104, 110, 116, 117, 139, 146, 150, 151, 157).

[15] **A. Greiner-Petter**, A. Youssef, T. Ruas, B. R. Miller, M. Schubotz, A. Aizawa, and B. Gipp. "Math-Word Embedding in Math Search and Semantic Extraction". In: *Scientometrics* 125.3 (2020-12), pp. 3017–3046. ISSN: 0138-9130, 1588-2861. DOI: 10/gg2cx9. URL: https://link.springer.com/10.1007/s11192-020-03502-9 (visited on 2021-06-30) (cit. on pp. 9, 12, 18, 21, 26, 37, 38, 59, 60, 63, 139, 146, 150, 151).

[16] P. Scharpf, M. Schubotz, **A. Greiner-Petter**, M. Ostendorff, O. Teschke, and B. Gipp. "ARQMath Lab: An Incubator for Semantic Formula Search in zbMATH Open?" In: *Working Notes of (CLEF) 2020 - Conference and Labs of the Evaluation Forum*. Vol. 2696. Thessaloniki, Greece: CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_200.pdf (cit. on pp. 9, 146, 150, 151).

[17] M. Schubotz, **A. Greiner-Petter**, N. Meuschke, O. Teschke, and B. Gipp. "Mathematical Formulae in Wikimedia Projects 2020". In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. Virtual Event, China: ACM, 2020-08, pp. 447–448. DOI: 10/ghn2t2. URL: https://arxiv.org/abs/2003.09417 (visited on 2021-08-02) (cit. on pp. 9, 14, 36, 101, 102, 140, 147, 150, 151, 154).

[18] 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: *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries (JCDL)*. Fort Worth, Texas, USA: ACM, 2018-05-23, pp. 233–242. DOI: 10.1145/3197026.3197058. URL: https://arxiv.org/abs/1804.04956 (visited on 2021-09-06) (cit. on pp. 2, 9, 10, 18–21, 37, 39, 61, 65, 68, 72, 81, 104, 109, 115, 131, 150).

BACK MATTER

Bibliography

[19] J. Abbott, A. Díaz, and R. S. Sutor. "A Report on OpenMath: A Protocol for the Exchange of Mathematical Information". In: *ACM SIGSAM Bulletin* 30.1 (1996-03), pp. 21–24. ISSN: 0163-5824. DOI: 10/ctnngk (cit. on pp. 18, 20, 23, 24).

[20] J. M. Aguirregabiria, A. Hernández, M. Rivas, and D. Donnelly. "Are We Careful Enough When Using Computer Algebra?" In: *Computers in Physics* 8.1 (1994), p. 56. ISSN: 0894-1866. DOI: 10/gn3svw (cit. on pp. 5, 115, 116).

[21] A. Aizawa, M. Kohlhase, and I. Ounis. "NTCIR-10 Math Pilot Task Overview". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-10)*. Tokyo, Japan: National Institute of Informatics (NII), 2013. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/OVERVIEW/01-NTCIR10-OV-MATH-AizawaA.pdf (visited on 2021-08-19) (cit. on pp. 52, 70).

[22] A. Aizawa, M. Kohlhase, I. Ounis, and M. Schubotz. "NTCIR-11 Math-2 Task Overview". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014, pp. 88–98. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/OVERVIEW/01-NTCIR11-OV-MATH-AizawaA.pdf (visited on 2021-08-19) (cit. on pp. 43, 52, 70).

[23] A. N. Aizawa. "An information-theoretic perspective of tf-idf measures". In: *Inf. Process. Manage.* 39.1 (2003), pp. 45–65. DOI: 10.1016/S0306-4573(02)00021-3 (cit. on p. 77).

[24] G. Araujo and D. Pellegrino. "On the constants of the Bohnenblust-Hille inequality and Hardy–Littlewood inequalities". In: *CoRR* abs/1407.7120 (2014). URL: https://arxiv.org/abs/1407.7120 (cit. on p. 89).

[25] A. Asperti, H. Geuvers, and R. Natarajan. "Social Processes, Program Verification and All That". In: *Mathematical Structures in Computer Science* 19.5 (2009-10), pp. 877–896. ISSN: 1469-8072. DOI: 10/c3st7r (cit. on p. 23).

[26] D. H. Bailey, J. M. Borwein, and A. D. Kaiser. "Automated Simplification of Large Symbolic Expressions". In: *Journal of Symbolic Computation* 60 (2014-01), pp. 120–136. ISSN: 0747-7171. DOI: 10/f5kzhg (cit. on p. 125).

[27] J. B. Baker, A. P. Sexton, and V. Sorge. "MaxTract: Converting PDF to LaTeX, MathML and Text". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 7362. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 422–426. DOI: 10.1007/978-3-642-31374-5_29 (cit. on pp. 19, 22).

[28] C. Ballarin, K. Homann, and J. Calmet. "Theorems and Algorithms: An Interface between Isabelle and Maple". In: *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation - ISSAC '95*. Montreal, Quebec, Canada: ACM Press, 1995, pp. 150–157. DOI: 10/fd8b67 (cit. on pp. 20, 34, 125).

[29] M. Barnett, B.-Y. E. Chang, R. DeLine, B. Jacobs, and K. R. M. Leino. "Boogie: A Modular Reusable Verifier for Object-Oriented Programs". In: *8th International Symposium of Formal Methods for Components and Objects*. Vol. 4111. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 364–387. DOI: 10/bjst7m (cit. on p. 116).

[30] J. Beel, B. Gipp, S. Langer, and C. Breitinger. "Research-Paper Recommender Systems: A Literature Survey". In: *International Journal on Digital Libraries* 17.4 (2016), pp. 305–338. ISSN: 1432-5012. DOI: 10/gddp66 (cit. on pp. 6, 78, 81).

[31] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nuernberger. "Research Paper Recommender System Evaluation: A Quantitative Literature Survey". In: *Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys)*. 2013. DOI: 10/ggv8d7 (cit. on p. 6).

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4

[32] R. Behrends, K. Hammond, V. Janjic, et al. "HPC-GAP: Engineering a 21st-Century High-Performance Computer Algebra System". In: *Concurrency and Computation: Practice and Experience* 28.13 (2016), pp. 3606–3636. DOI: 10/f82fwp (cit. on p. 32).

[33] Y. Bengio, R. Ducharme, and P. Vincent. "A Neural Probabilistic Language Model". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. MIT Press, 2000, pp. 932–938. URL: https://proceedings.neurips.cc/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html (visited on 2021-09-05) (cit. on p. 59).

[34] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155. URL: http://jmlr.org/papers/v3/bengio03a.html (visited on 2021-09-05) (cit. on p. 59).

[35] K. Bercic, J. Carette, W. Farmer, et al. *The Space of Mathematical Software Systems - A Survey of Paradigmatic Systems*. 2020. URL: https://arxiv.org/abs/2002.04955 (cit. on p. 6).

[36] L. Bernardin, P. Chin, P. DeMarco, et al. *Maple 2016 Programming Guide*. Maplesoft, a division of Waterloo Maple Inc., 2016. ISBN: 978-1-926902-46-3 (cit. on pp. 2, 4, 5, 18, 19, 32, 33, 102, 104, 127, 165).

[37] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development*. Texts in Theoretical Computer Science An EATCS Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. ISBN: 978-3-642-05880-6. DOI: 10.1007/978-3-662-07964-5 (cit. on pp. 18, 116).

[38] J. Betzendahl and M. Kohlhase. "Translating the IMPS Theory Library to MMT/OMDoc". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11006. Hagenberg, Austria: Springer International Publishing, 2018, pp. 7–22. DOI: 10.1007/978-3-319-96812-4_2 (cit. on p. 25).

[39] G. Bilbeisi, S. Ahmed, and R. Majumdar. "DeepEquaL: Deep Learning Based Mathematical Equation to Latex Generation". In: *International Conference on Neural Information Processing*. Vol. 1333. Cham: Springer International Publishing, 2020, pp. 324–332. DOI: 10/gn3sx7 (cit. on pp. 18, 19).

[40] G. W. Blackwood, M. Ballesteros, and T. Ward. "Multilingual Neural Machine Translation with Task-Specific Attention". In: *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. Association for Computational Linguistics, 2018, pp. 3112–3122. URL: https://aclanthology.org/C18-1263/ (visited on 2021-09-10) (cit. on p. 100).

[41] F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich. "Why3: Shepherd Your Herd of Provers". In: *Boogie 2011: First International Workshop on Intermediate Verification Languages* (2011-05), pp. 53–64. URL: https://hal.inria.fr/hal-00790310/document (cit. on p. 116).

[42] J. Böhm, W. Decker, S. Keicher, and Y. Ren. "Current Challenges in Developing Open Source Computer Algebra Systems". In: *Mathematical Aspects of Computer and Information Sciences (MACIS)*. Vol. 9582. Berlin, Germany: Springer, 2015, pp. 3–24. DOI: 10.1007/978-3-319-32859-1_1 (cit. on p. 33).

[43] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. "Enriching Word Vectors with Subword Information". In: *Trans. Assoc. Comput. Linguistics* 5 (2017), pp. 135–146. DOI: 10/gfw9cs (cit. on pp. 59, 61, 62).

[44] K. Bosa. *SCSCP4Mathematica*. Version 0.1. Research Institute for Symbolic Computation, 2011. URL: https://www3.risc.jku.at/projects/science/jra/ (cit. on pp. 25, 33).

[45] S. Boulmé, T. Hardin, D. Hirschkoff, V. Ménissier-Morain, and R. Rioboo. "On the Way to Certify Computer Algebra Systems". In: *Electronic Notes in Theoretical Computer Science* 23.3 (1999), pp. 370–385. ISSN: 1571-0661. DOI: 10/bqv3xz (cit. on p. 116).

[46] Y. Bouzidi, A. Quadrat, and F. Rouillier. "Computer Algebra Methods for Testing the Structural Stability of Multidimensional Systems". In: *2015 IEEE 9th International Workshop on Multidimensional (nD) Systems (nDS)*. Vila Real, Portugal: IEEE, 2015-09, pp. 1–6. DOI: 10/gn3swg (cit. on p. 32).

[47] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. "A Large Annotated Corpus for Learning Natural Language Inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal: The Association for Computational Linguistics, 2015, pp. 632–642. DOI: 10/gf74f9 (cit. on p. 70).

[48] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts. "A Fast Unified Model for Parsing and Sentence Understanding". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2016. DOI: 10/gfw97m (cit. on p. 96).

[49] R. Breh and V. Kumar. "Making Mathematical Problem Solving Exploratory and Social-Synergizing Computer Algebra Systems with Semantic and Web-2.0 Technology". In: *Proceedings of the Third Annual ACM Bangalore Conference on - COMPUTE '10*. Bangalore, India: ACM Press, 2010, pp. 1–7. DOI: 10/chxtdw (cit. on p. 32).

[50] C. Breitinger and H. Reiterer. "Visualizing Feature-based Similarity for Research Paper Recommendation". In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2021*. Online: ACM, 2021. DOI: 10/gn3sxc (cit. on pp. 6, 72, 147).

[51] C. Bright, I. Kotsireas, and V. Ganesh. "Applying Computer Algebra Systems with SAT Solvers to the Williamson Conjecture". In: *Journal of Symbolic Computation* 100 (2020-09), pp. 187–209. ISSN: 0747-7171. DOI: 10/gn3sx2 (cit. on pp. 6, 32).

[52] C. Bright, I. Kotsireas, and V. Ganesh. "SAT Solvers and Computer Algebra Systems: A Powerful Combination for Mathematics". In: *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. USA: IBM Corp., 2019-11-04, pp. 323–328. DOI: 10.5555/3370272.3370309 (cit. on p. 6).

[53] S. Buswell, O. Caprotti, D. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase. *The OpenMath Standard Version 2.0*. 2004-06. URL: http://openmath.org/standard/om20-2004-06-30/ (visited on 2021-10-06) (cit. on pp. 6, 23, 163).

[54] F. Cajori. *A History of Mathematical Notations*. Vol. 1 and 2. 2 vols. New York: Dover Publications, 1993. 451 pp. ISBN: 978-0-486-67766-8 (cit. on pp. 4, 45, 58, 87).

[55] J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. "A Unified Multilingual Semantic Representation of Concepts". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2015, pp. 741–751. DOI: 10/gn3sxf (cit. on p. 63).

[56] O. Caprotti and D. Carlisle. "OpenMath and MathML: Semantic Markup for Mathematics". In: *XRDS: Crossroads, The ACM Magazine for Students* 6.2 (1999-11), pp. 11–14. ISSN: 1528-4972, 1528-4980. DOI: 10/bbpk38 (cit. on p. 24).

[57] O. Caprotti and A. M. Cohen. "Connecting Proof Checkers and Computer Algebra Using OpenMath". In: *Theorem Proving in Higher Order Logics*. Vol. 1690. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 109–112. DOI: 10.1007/3-540-48256-3_8 (cit. on pp. 6, 20, 21, 25, 34, 39).

[58] J. Carette and M. Kucera. "Partial Evaluation of Maple". In: *Science of Computer Programming* 76.6 (2011-06), pp. 469–491. ISSN: 0167-6423. DOI: 10/fnkpct (cit. on p. 116).

[59] D. Carlisle. "OpenMath, MathML, and XSL". In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 6–11. ISSN: 0163-5824. DOI: 10/bwsggv (cit. on pp. 6, 21–24).

[60] D. Carlisle, P. Ion, and R. Miner. *Mathematical Markup Language (MathML) Version 3.0, 2nd Edition*. W3C Recommendation. World Wide Web Consortium (W3C), 2014-04-10. URL: https://www.w3.org/TR/MathML3/ (visited on 2021-09-06) (cit. on pp. 6, 18, 21).

[61] D. Carlisle and M. Wang. *Web-Xslt: A Collection of XSLT Stylesheets Designed for Processing MathML*. 2021-05-03. URL: https://github.com/davidcarlisle/web-xslt (visited on 2021-09-22) (cit. on pp. 21, 22).

[62] M. M. Carneiro. "Conversion of HOL Light Proofs into Metamath". In: *Journal of Formalized Reasoning* 9.1 (2016), pp. 187–200. DOI: 10/gn3swz (cit. on p. 21).

[63] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier. "Word2vec Applied to Recommendation: Hyperparameters Matter". In: *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. ACM, 2018, pp. 352–356. DOI: 10/ggbpnn (cit. on p. 64).

[64] J. B. Cassel. "Wolfram|Alpha: A Computational Knowledge "Search" Engine". In: *Google It*. New York, NY: Springer New York, 2016, pp. 267–299. ISBN: 978-1-4939-6415-4. DOI: 10.1007/978-1-4939-6415-4_11 (cit. on p. 27).

[65] D. Cer, Y. Yang, S.-y. Kong, et al. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 169–174. DOI: 10/ghjqhk (cit. on pp. 59, 61, 63).

[66] B. B. Chaudhuri and U. Garain. "Automatic Detection of Italic, Bold and All-Capital Words in Document Images". In: *Fourteenth International Conference on Pattern Recognition, ICPR 1998, Brisbane, Australia, 16-20 August, 1998*. IEEE Computer Society, 1998, pp. 610–612. DOI: 10/bmz7zg (cit. on p. 37).

[67] R. Chauhan, I. Murray, and R. Koul. "Audio Rendering of Mathematical Expressions for Blind Students: A Comparative Study between MathML and Latex". In: *IEEE International Conference on Engineering, Technology and Education, TALE 2019, Yogyakarta, Indonesia, December 10-13, 2019*. IEEE, 2019, pp. 1–5. DOI: 10/gn3sx5 (cit. on p. 22).

[68] C. Chelba, T. Mikolov, M. Schuster, et al. "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling". In: *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*. ISCA, 2014, pp. 2635–2639. URL: http://www.isca-speech.org/archive/interspeech%5C%5C_2014/i14%5C%5C_2635.html (visited on 2021-09-05) (cit. on p. 70).

[69] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen. "Enhanced LSTM for Natural Language Inference". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2017, pp. 1657–1668. DOI: 10/gf5hs2 (cit. on p. 59).

[70] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Y. Wang. "How Large a Vocabulary Does Text Classification Need? A Variational Approach to Vocabulary Selection". In: *Proceedings of the 2019 Conference of the North*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 3487–3497. DOI: 10/gn3sv8 (cit. on p. 96).

[71] P.-Y. Chien and P.-J. Cheng. "Semantic Tagging of Mathematical Expressions". In: *Proceedings of the 24th International Conference on World Wide Web (WWW)*. Florence, Italy: ACM, 2015-05-18, pp. 195–204. DOI: 10/gmjqgn (cit. on pp. 6, 26, 36, 53, 54, 58, 71, 90, 169).

[72] J. P. C. Chiu and E. Nichols. "Named Entity Recognition with Bidirectional LSTM-CNNs". In: *Trans. Assoc. Comput. Linguistics* 4 (2016), pp. 357–370. DOI: 10/ggj7fx (cit. on p. 59).

[73] K. Cho, B. van Merrienboer, Ç. Gülçehre, et al. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: ACL, 2014, pp. 1724–1734. DOI: 10/gddmvq (cit. on pp. 59, 61).

[74] S. P. Chowdhury, S. Mandal, A. K. Das, and B. Chanda. "Automated Segmentation of Math-Zones from Document Images". In: *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*. IEEE Computer Society, 2003, pp. 755–759. DOI: 10/bqjkk9 (cit. on p. 36).

[75] C. Clark and M. Gardner. "Simple and Effective Multi-Paragraph Reading Comprehension". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2018, pp. 845–855. DOI: 10/ggvxfb (cit. on p. 59).

[76] J. S. Cohen. *Computer Alegebra and Symbolic Computation: Mathematical Methods*. Natick, Mass: AK Peters, 2003. 448 pp. ISBN: 978-1-56881-159-8 (cit. on p. 32).

[77] H. S. Cohl, M. A. McClain, B. V. Saunders, M. Schubotz, and J. C. Williams. "Digital Repository of Mathematical Formulae". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 8543. Springer, 2014, pp. 419–422. DOI: 10.1007/978-3-319-08434-3_30 (cit. on pp. 29, 43, 164).

[78] H. S. Cohl, M. Schubotz, M. A. McClain, et al. "Growing the Digital Repository of Mathematical Formulae with Generic Sources". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 9150. Springer, 2015, pp. 280–287. DOI: 10.1007/978-3-319-20615-8_18 (cit. on pp. 29, 43, 164).

[79] *Colt - Open Source Libraries for High Performance Scientific and Technical Computing in Java*. Version 1.2.0. CERN - European Organization for Nuclear Research, 2004. URL: https://dst.lbl.gov/ACSSoftware/colt/ (visited on 2021-09-23) (cit. on pp. 18, 32).

[80] *Computable Document Format (CDF)*. Wolfram Research. URL: https://www.wolfram.com/cdf/ (visited on 2021-10-06) (cit. on p. 25).

[81] *Computer Algebra Systems: A Practical Guide*. New York: Wiley, 1999. 436 pp. ISBN: 978-0-471-98353-8 (cit. on pp. 19, 32).

[82] T. Cool. *The Disappointment and Embarrassment of MathML*. General Economics and Teaching JEL A00. University Library of Munich, Germany, 2000-04-16. URL: https://econpapers.repec.org/paper/wpawuwpgt/0004002.htm (visited on 2021-08-13) (cit. on pp. 19, 23).

[83] R. M. Corless and D. J. Jeffrey. "Graphing Elementary Riemann Surfaces". In: *ACM SIGSAM Bulletin* 32.1 (1998-03), pp. 11–17. ISSN: 0163-5824. DOI: 10/dcpb9j (cit. on pp. 5, 34).

[84] R. M. Corless, D. J. Jeffrey, S. M. Watt, and J. H. Davenport. ""According to Abramowitz and Stegun" or Arccoth Needn't Be Uncouth". In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 58–65. ISSN: 0163-5824. DOI: 10/cc53nd (cit. on pp. 5, 34, 130, 131).

[85] H. Cuypers, A. M. Cohen, J. W. Knopper, R. Verrijzer, and M. Spanbroek. "MathDox, a System for Interactive Mathematics". In: Association for the Advancement of Computing in Education (AACE), 2008-06-30, pp. 5177–5182. ISBN: 978-1-880094-65-5. URL: https://www.learntechlib.org/primary/p/29092/ (visited on 2021-06-10) (cit. on pp. 6, 25).

[86] P. Dadure, P. Pakray, and S. Bandyopadhyay. "An Analysis of Variable-Size Vector Based Approach for Formula Searching". In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*. Vol. 2696. CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_150.pdf (visited on 2021-09-12) (cit. on p. 21).

[87] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2019, pp. 2978–2988. DOI: 10/gf8zxx (cit. on p. 63).

[88] A. Dakkak, T. Wickham-Jones, and W.-m. Hwu. "The Design and Implementation of the Wolfram Language Compiler". In: *Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization*. San Diego CA USA: ACM, 2020-02-22, pp. 212–228. DOI: 10/gn3sv3 (cit. on pp. 20, 33).

[89] S. Dalmas, M. Gaëtano, and S. Watt. "An OpenMath 1.0 Implementation". In: *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation - ISSAC '97*. Kihei, Maui, Hawaii, United States: ACM Press, 1997, pp. 241–248. DOI: 10/fhs7ng (cit. on p. 23).

[90] J. H. Davenport. "On Writing OpenMath Content Dictionaries". In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 12–15. ISSN: 0163-5824. DOI: 10/fr42qr (cit. on p. 24).

[91] J. H. Davenport. "The Challenges of Multivalued "Functions"". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 6167. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–12. DOI: 10.1007/978-3-642-14128-7_1 (cit. on pp. 5, 34).

[92] K. Davila and R. Zanibbi. "Layout and Semantics: Combining Representations for Mathematical Formula Search". In: *Proc. ACM SIGIR*. Shinjuku, Tokyo: ACM, 2017, pp. 1165–1168. DOI: 10.1145/3077136.3080748 (cit. on pp. 6, 72, 75, 89, 104).

[93] K. Davila, R. Zanibbi, A. Kane, and F. W. Tompa. "Tangent-3 at the NTCIR-12 MathIR Task". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-12)*. Tokyo, Japan: National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/06-NTCIR12-MathIR-DavilaK.pdf (visited on 2021-08-19) (cit. on pp. 36, 52, 53).

[94] P. Dehaye, M. Iancu, M. Kohlhase, et al. "Interoperability in the OpenDreamKit Project: The Math-in-the-Middle Approach". In: 2016, pp. 117–131. DOI: 10.1007/978-3-319-42547-4_9 (cit. on p. 42).

[95] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush. "Image-to-Markup Generation with Coarse-to-Fine Attention". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Sydney, NSW, Australia: PMLR, 2017, pp. 980–989. URL: http://proceedings.mlr.press/v70/deng17a.html (visited on 2021-09-10) (cit. on pp. 96, 98).

[96] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. Conf. North American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10/ggbwf6 (cit. on p. 63).

[97] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. M. Schwartz, and J. Makhoul. "Fast and Robust Neural Network Joint Models for Statistical Machine Translation". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2014, pp. 1370–1380. DOI: 10/ggnw4r (cit. on p. 59).

[98] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.0.25 of 2019-12-15. F. W. J. Olver, et al. eds. (cit. on pp. 1, 24, 28, 41, 43, 54, 60, 62, 71, 86, 102, 104, 115, 120, 122–125, 127–131, 145, 147–149, 164).

[99] H. Dohrn and D. Riehle. "Design and Implementation of the Sweble Wikitext Parser: Unlocking the Structured Data of Wikipedia". In: *Proceedings of the 7th International Symposium on Wikis and Open Collaboration - WikiSym '11*. Mountain View, California: ACM Press, 2011, p. 72. DOI: 10/c6g63f (cit. on p. 108).

[100] A. Durán, M. Pérez, and J. Varona. "The Misfortunes of a Trio of Mathematicians Using Computer Algebra Systems. Can We Trust in Them?" In: *Notices of the AMS* 61.10 (2014), pp. 1249–1252. URL: https://www.ams.org/notices/201410/rnoti-p1249.pdf (cit. on pp. 5, 115).

[101] J. Earley. "An Efficient Context-Free Parsing Algorithm". In: *Communications of the ACM* 13.2 (1970-02), pp. 94–102. ISSN: 0001-0782, 1557-7317. DOI: 10/csjctz (cit. on p. 35).

[102] S. Edunov, M. Ott, M. Auli, and D. Grangier. "Understanding Back-Translation at Scale". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 489–500. DOI: 10/gf6rn5 (cit. on p. 98).

[103] A. M. Elizarov, S. Khaydarov, and E. K. Lipachev. "The Formation Method of Recommendations in the Process of Scientific Peer Review of Mathematical Papers". In: *Proceedings of the 21st Conference on Scientific Services & Internet (SSI-2019)*. Novorossiysk-Abrau, Russia: CEUR-WS.org, 2019, p. 10. URL: http://ceur-ws.org/Vol-2543/rpaper12.pdf (cit. on p. 6).

[104] A. M. Elizarov, A. Kirillovich, E. K. Lipachev, and O. Nevzorova. "Digital Ecosystem OntoMath: Mathematical Knowledge Analytics and Management". In: *Data Analytics and Management in Data Intensive Domains - XVIII International Conference, DAMDID/RCDL 2016, Ershovo, Moscow, Russia, October 11-14, 2016, Revised Selected Papers*. Vol. 706. 2016, pp. 33–46. DOI: 10/gjnf52 (cit. on pp. 67, 69).

[105] A. M. Elizarov, A. Kirillovich, E. K. Lipachev, and O. Nevzorova. "OntoMath Digital Ecosystem: Ontologies, Mathematical Knowledge Analytics and Management". In: *CoRR* abs/1702.05112 (2017). URL: http://arxiv.org/abs/1702.05112 (visited on 2021-09-05) (cit. on pp. 67, 69).

[106] A. M. Elizarov, E. K. Lipachev, and S. Khaydarov. "Recommender System in the Process of Scientific Peer Review in Mathematical Journal". In: *Russian Digital Libraries Journal* 23.4 (2020), pp. 708–732. DOI: 10/gn3szc (cit. on p. 6).

[107] D. Elphick, M. Leuschel, and S. Cox. "Partial Evaluation of MATLAB". In: *Generative Programming and Component Engineering*. Vol. 2830. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 344–363. DOI: 10/b6wtqf (cit. on p. 116).

[108] M. England, E. Cheb-Terrab, R. Bradford, J. H. Davenport, and D. Wilson. "Branch Cuts in Maple 17". In: *ACM Communications in Computer Algebra* 48.1/2 (2014-07-10), pp. 24–27. ISSN: 1932-2240. DOI: 10/gknppb (cit. on pp. 34, 130).

[109] M. England, E. S. Cheb-Terrab, R. J. Bradford, J. H. Davenport, and D. J. Wilson. "Branch cuts in Maple 17". In: *ACM Comm. Comp. Algebra* 48.1/2 (2014), pp. 24–27. DOI: 10.1145/2644288.2644293 (cit. on p. 41).

[110] L. Eskor, M. Lepp, and E. Tonisson. "Automatic Translation of Computer Algebra Systems' Worksheets". In: *2011 International Conference on Computational Science and Its Applications*. Santander, Spain: IEEE, 2011-06, pp. 118–123. DOI: 10/cv8n44 (cit. on pp. 6, 33, 39).

[111] L. Espinosa Anke and S. Schockaert. "Syntactically Aware Neural Architectures for Definition Extraction". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 378–385. DOI: 10/ggwczd (cit. on pp. 109, 138, 140, 146, 154).

[112] K.-C. Fan and C. H. Huang. "Italic Detection and Rectification". In: *Journal of Information Science and Engineering* 23.2 (2007), pp. 403–419. URL: https://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=47_798 (cit. on p. 37).

[113] K.-C. Fan, C. H. Huang, and T. C. Chuang. "Italic Detection and Rectification". In: *Proceedings of the 2005 International Conference on Image Processing, ICIP 2005, Genoa, Italy, September 11-14, 2005*. IEEE, 2005, pp. 530–533. DOI: 10/cdqqp5 (cit. on p. 37).

[114] R. Fateman. *A Critique of OpenMath and Thoughts on Encoding Mathematics, January, 200*. 2001-01-17. URL: https://people.eecs.berkeley.edu/~fateman/papers/openmathcrit.pdf (visited on 2021-10-06) (cit. on p. 23).

[115] R. Fateman. "Partitioning of Algebraic Subexpressions in Computer Algebra Systems: An Alternative to Matching with an Application to Symbolic Integration". In: *ACM Communications in Computer Algebra* 49.2 (2015-08-14), pp. 38–47. ISSN: 1932-2240. DOI: 10/gn3swj (cit. on p. 32).

[116] C. Fellbaum. "A Semantic Network of English: The Mother of All WordNets". In: *Comput. Humanit.* 32.2-3 (1998), pp. 209–220. DOI: 10/dcz9jc (cit. on p. 69).

[117] D. Formánek, M. Líska, M. Ruzicka, and P. Sojka. "Normalization of Digital Mathematics Library Content". In: *Proc. of OpenMath/MathUI/CICM-WiP*. Vol. 921. Bremen, Germany: CEUR-WS.org, 2012, pp. 91–103. URL: http://ceur-ws.org/Vol-921/wip-05.pdf (visited on 2021-07-01) (cit. on pp. 73, 75, 91).

[118] D. S. Foundation. *Django 1.7 - The Django Template Language*. The Django template language. 2017-02-24. URL: https://django.readthedocs.io/en/1.7.x/topics/templates.html (visited on 2021-09-08) (cit. on p. 19).

[119] S. Freundt, P. Horn, A. Konovalov, S. Linton, and D. Roozemond. "Symbolic Computation Software Composability". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 5144. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 285–295. DOI: 10/fj2cr2 (cit. on pp. 6, 7, 33, 168).

[120] S. Freundt and S. Lesseni. *KANT 4 SCSCP Package*. Version 1.2.1. 2010. URL: http://page.math.tu-berlin.de/~kant/kantscscp.html (visited on 2021-09-23) (cit. on p. 33).

[121] L. Gao, Z. Jiang, Y. Yin, K. Yuan, Z. Yan, and Z. Tang. *Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: Can Mathematical Formulae Be Embedded like a Natural Language?* 2017-08-29. URL: http://arxiv.org/abs/1707.05154 (visited on 2021-06-30) (cit. on pp. 11, 26, 37, 38, 52, 53, 59, 62, 67).

[122] L. Gao, Y. Wang, L. Hao, and Z. Tang. "ICST Math Retrieval System for NTCIR-11 Math-2 Task". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/02-NTCIR11-MATH-GaoL.pdf (visited on 2021-08-19) (cit. on p. 53).

[123] L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang. "A Deep Learning-Based Formula Detection Method for PDF Documents". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto: IEEE, 2017-11, pp. 553–558. DOI: 10/ghpj5q (cit. on p. 108).

[124] L. Gao, K. Yuan, Y. Wang, Z. Jiang, and Z. Tang. "The Math Retrieval System of ICST for NTCIR-12 MathIR Task". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-12)*. Tokyo, Japan: National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/03-NTCIR12-MathIR-GaoL.pdf (visited on 2021-08-19) (cit. on p. 53).

[125] U. Garain. "Identification of Mathematical Expressions in Document Images". In: *10th International Conference on Document Analysis and Recognition, ICDAR 2009, Barcelona, Spain, 26-29 July 2009*. IEEE Computer Society, 2009, pp. 1340–1344. DOI: 10/fnf89r (cit. on p. 36).

[126] U. Garain, B. B. Chaudhuri, and A. R. Chaudhuri. "Identification of Embedded Mathematical Expressions in Scanned Documents". In: *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*. IEEE Computer Society, 2004, pp. 384–387. DOI: 10/cqv5wb (cit. on p. 36).

[127] A. Gárate-García, L. Márquez-Martínez, J. Cuesta-García, and E. García-Ramírez. "A Computer Algebra System for Analysis and Control of Nonlinear Time-Delay Systems". In: *Advances in Engineering Software* 65 (2013-11), pp. 138–148. ISSN: 0965-9978. DOI: 10/gn3sv7 (cit. on p. 32).

[128] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge: Cambridge University Press, 2013. 795 pp. ISBN: 978-1-107-03903-2 (cit. on pp. 18–20, 32).

[129] A. Gaudeul. "Do Open Source Developers Respond to Competition?: The LaTeX Case Study". In: *Review of Network Economics* 6 (2 2007-06), pp. 239–263. DOI: 10.2202/1446-9022.1119 (cit. on pp. 2, 19, 25, 72).

[130] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. "Convolutional Sequence to Sequence Learning". In: *Proceedings of the 34th International Conference on Machine Learning ICML*. Vol. 70. Sydney, NSW, Australia: PMLR, 2017, pp. 1243–1252. URL: http://proceedings.mlr.press/v70/gehring17a.html (visited on 2021-09-10) (cit. on pp. 96, 97).

[131] J. Giceva, C. Lange, and F. Rabe. "Integrating Web Services into Active Mathematical Documents". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 5625. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 279–293. DOI: 10/cmwkfx (cit. on pp. 6, 25).

[132] D. Ginev. *arXMLiv:08.2018 dataset, an HTML5 conversion of arXiv.org*. SIGMathLing – Special Interest Group on Math Linguistics. 2018. URL: https://sigmathling.kwarc.info/resources/arxmliv/ (cit. on pp. 64, 73, 162).

[133] D. Ginev and B. R. Miller. *LaTeXML 2012 - A Year of LaTeXML.* 2014-04-25. URL: http://arxiv.org/abs/1404.6549 (visited on 2021-08-19) (cit. on pp. 26, 30, 36).

[134] D. Ginev and B. R. Miller. "Scientific Statement Classification over arXiv.Org". In: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020.* European Language Resources Association, 2020, pp. 1219–1226. URL: https://aclanthology.org/2020.lrec-1.153/ (visited on 2021-08-15) (cit. on pp. 109, 138, 140, 146, 154).

[135] D. Ginev, H. Stamerjohanns, B. R. Miller, and M. Kohlhase. "The LaTeXML Daemon: Editable Math on the Collaborative Web". In: *Proc. Conf. Intelligent Computer Mathematics (CICM).* Vol. 6824. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 292–294. DOI: 10.1007/978-3-642-22673-1_25 (cit. on pp. 26, 36, 44).

[136] C. Goller and A. Küchler. "Learning Task-Dependent Distributed Representations by Backpropagation through Structure". In: *Proceedings of International Conference on Neural Networks (ICNN'96).* Washington, DC, USA: IEEE, 1996, pp. 347–352. DOI: 10/d5q65t (cit. on p. 96).

[137] Y. Gong, H. Luo, and J. Zhang. "Natural Language Inference over Interaction Space". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.* OpenReview.net, 2018. URL: https://openreview.net/forum?id=r1dHXnH6- (visited on 2021-09-05) (cit. on p. 59).

[138] T. Gowers, J. Barrow-Green, and I. Leader. "The Language and Grammar of Mathematics". In: *The Princeton Companion to Mathematics.* Princeton: Princeton University Press, 2008, pp. 8–16. ISBN: 978-0-691-11880-2 (cit. on pp. 36, 158).

[139] M. Grigore, M. Wolska, and M. Kohlhase. "Towards Context-Based Disambiguation of Mathematical Expressions". In: *Joint Conference ASCM and MACIS: Asian Symposium on Computer Mathematics and Mathematical Aspects of Computer and Information Sciences.* 2009, pp. 262–271. URL: file:///C:/Users/andre/Zotero/storage/UBJCMNIU/43bfae6781b11f906db53c3bfd24a7da62278e48.html#paper-header (visited on 2021-09-13) (cit. on pp. 12, 22, 23, 53–55, 58).

[140] C. Grün, S. Gath, A. Holupirek, and M. Scholl. "XQuery Full Text Implementation in BaseX". In: *Database and XML Technologies.* Springer Berlin, 2009, pp. 114–128 (cit. on p. 72).

[141] F. Guidi and C. Sacerdoti Coen. "A Survey on Retrieval of Mathematical Knowledge". In: *Mathematics in Computer Science* 10.4 (2016-12), pp. 409–427. ISSN: 1661-8270, 1661-8289. DOI: 10/gfj4mz (cit. on pp. 6, 37, 51, 70).

[142] S. A. Gutnik and V. A. Sarychev. "Application of Computer Algebra Methods to Investigate the Dynamics of the System of Two Connected Bodies Moving along a Circular Orbit". In: *Programming and Computer Software* 45.2 (2019-03), pp. 51–57. ISSN: 0361-7688. DOI: 10/gn3swd (cit. on p. 32).

[143] D. T. Halbach. "Mathematical World Knowledge Contained in the Multilingual Wikipedia Project". In: *Mathematical Software – ICMS 2020.* Vol. 12097. Cham: Springer International Publishing, 2020, pp. 353–361. DOI: 10.1007/978-3-030-52200-1_35 (cit. on p. 159).

[144] A. Halfaker and R. S. Geiger. "ORES: Lowering Barriers with Participatory Machine Learning in Wikipedia". In: *Proceedings of the ACM on Human-Computer Interaction* 4 (CSCW2 2020-10-14), 148:1–148:37. DOI: 10/ghf4vb (cit. on pp. 102, 157, 167).

[145] R. Hambasan, M. Kohlhase, and C.-C. Prodescu. "MathWebSearch at NTCIR-11". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11).* Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/05-NTCIR11-MATH-HambasanR.pdf (visited on 2021-08-19) (cit. on p. 53).

[146] J. Harrison. "HOL Light: A Tutorial Introduction". In: *Formal Methods in Computer-Aided Design (FMCAD).* Vol. 1166. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 265–269. DOI: 10/dtgcxg (cit. on pp. 25, 34, 116).

[147] J. Harrison, J. Urban, and F. Wiedijk. "History of Interactive Theorem Proving". In: *Handbook of the History of Logic.* Vol. 9. Elsevier, 2014, pp. 135–214. ISBN: 978-0-444-51624-4. DOI: 10.1016/B978-0-444-51624-4.50004-6 (cit. on pp. 19, 34).

[148] J. R. Harrison and L. Théry. "A Skeptic's Approach to Combining HOL and Maple". In: *Journal of Automated Reasoning* 21.3 (1998), pp. 279–294. ISSN: 0168-7433. DOI: 10/bn4tnw (cit. on pp. 34, 116, 125).

[149] L. He, K. Lee, M. Lewis, and L. Zettlemoyer. "Deep Semantic Role Labeling: What Works and What's Next". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2017, pp. 473–483. DOI: 10/ggv89s (cit. on p. 59).

[150] A. Head, K. Lo, D. Kang, et al. "Augmenting Scientific Papers with Just-in-Time, Position-Sensitive Definitions of Terms and Symbols". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Yokohama Japan: ACM, 2021-05-06, pp. 1–18. DOI: 10/gksmrc (cit. on pp. 6, 13, 25, 102, 140).

[151] A. C. Hearn. "REDUCE: The First Forty Years". In: *Algorithmic Algebra and Logic. Proceedings of the {A3L}; Conference in Honor of the 60th Birthday of Volker Weispfenning*. Books on Demand, 2005, pp. 19–24. URL: http://www.reduce-algebra.com/reduce40.pdf (cit. on pp. 5, 33, 168).

[152] J. Heras, V. Pascual, and J. Rubio. "Using Open Mathematical Documents to Interface Computer Algebra and Proof Assistant Systems". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 5625. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 467–473. DOI: 10/frfjbc (cit. on pp. 6, 20, 21, 23, 25, 34, 115, 131).

[153] T. Hickman, C. P. Laursen, and S. Foster. *Certifying Differential Equation Solutions from Computer Algebra Systems in Isabelle/HOL*. 2021-02-04. URL: http://arxiv.org/abs/2102.02679 (visited on 2021-08-19) (cit. on pp. 6, 116).

[154] G. E. Hinton and S. T. Roweis. "Stochastic Neighbor Embedding". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. MIT Press, 2002, pp. 833–840. URL: https://proceedings.neurips.cc/paper/2002/hash/6150ccc6069bea6b5716254057a194ef-Abstract.html (visited on 2021-09-05) (cit. on pp. 66, 169).

[155] P. Horn. *MuPAD OpenMath Package*. 2009. URL: http://mupad.symcomp.org/ (visited on 2021-09-23) (cit. on p. 33).

[156] X. Hu, L. Gao, X. Lin, Z. Tang, X. Lin, and J. B. Baker. "WikiMirs: A Mathematical Information Retrieval System for Wikipedia". In: *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries - JCDL '13*. Indianapolis, Indiana, USA: ACM Press, 2013, p. 11. DOI: 10/ggwj8v (cit. on p. 101).

[157] F. Hueske and T. Walther. "Apache Flink". In: *Encyclopedia of Big Data Technologies*. Cham: Springer International Publishing, 2018, pp. 1–8. ISBN: 978-3-319-63962-8. DOI: 10.1007/978-3-319-63962-8_303-1 (cit. on p. 84).

[158] I. Huseyinov and F. S. Tabak. "The Evaluation of Computer Algebra Systems Using Fuzzy Multi-Criteria Decision-Making Models: Fuzzy AHP and Fuzzy TOPSIS". In: *International Journal of Software Innovation* 8.1 (2020-01), pp. 1–16. ISSN: 2166-7160. DOI: 10/gn3swq (cit. on p. 32).

[159] S. Hussain, S. Bai, and S. A. Kohja. "Rule Based Conversion of LaTeX Math Equations into Content MathML (CMML)". In: *Journal of Information Science and Engineering* 36.5 (2020), pp. 1021–1034 (cit. on pp. 6, 19, 21–23, 26, 52, 54).

[160] I. Iacobacci, M. T. Pilehvar, and R. Navigli. "Embeddings for Word Sense Disambiguation: An Evaluation Study". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2016. DOI: 10/ggv7nk (cit. on pp. 59, 63).

[161] I. Iacobacci, M. T. Pilehvar, and R. Navigli. "SensEmbed: Learning Sense Embeddings for Word and Relational Similarity". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2015, pp. 95–105. DOI: 10/gfzt8q (cit. on p. 63).

[162] M. Iancu, C. Jucovschi, M. Kohlhase, and T. Wiesing. "System Description: MathHub.Info". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 8543. Cham: Springer International Publishing, 2014, pp. 431–434. DOI: 10/gn3szf (cit. on pp. 6, 25).

[163] M. Iancu, M. Kohlhase, F. Rabe, and J. Urban. "The Mizar Mathematical Library in OMDoc: Translation and Applications". In: *Journal of Automated Reasoning* 50.2 (2013-02), pp. 191–202. ISSN: 0168-7433, 1573-0670. DOI: 10/gn3sxq (cit. on pp. 6, 25).

[164] "Introduction to Matlab". In: B. Radi and A. El Hami. *Advanced Numerical Methods with Matlab 2*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2018-05-25, pp. 179–188. ISBN: 978-1-119-49223-8. DOI: 10.1002/9781119492238.app1 (cit. on pp. 32, 33, 166).

[165] 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. 2017, pp. 56–69 (cit. on p. 43).

[166] V. Irtegov and T. Titorenko. "On the Study of the Motion of a System of Two Connected Rigid Bodies by Computer Algebra Methods". In: *International Workshop on Computer Algebra in Scientific Computing (CASC)*. Vol. 12291. Linz, Austria: Springer, 2020, pp. 266–281. DOI: 10.1007/978-3-030-60026-6_15 (cit. on p. 32).

[167] *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. Vol. 2283. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. ISBN: 978-3-540-43376-7. DOI: 10.1007/3-540-45949-9 (cit. on pp. 21, 116).

[168] ISO. *ISO/IEC 29124:2010: Information technology — Programming languages, their environments and system software interfaces — Extensions to the C++ Library to support mathematical special functions*. Technical report. 2010. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50511 (visited on 2021-09-23) (cit. on pp. 18, 32).

[169] ISO. *ISO/IEC 40314:2016: Mathematical Markup Language (MathML) Version 3.0 2nd Edition*. Technical report. 2016, p. 403. URL: https://www.iso.org/standard/58439.html (visited on 2021-09-23) (cit. on pp. 21, 166).

[170] ISO. *ISO/IEC 60559:2020: Information technology — Microprocessor Systems — Floating-Point arithmetic*. Technical report. 2020, p. 74. URL: https://www.iso.org/standard/80985.html (visited on 2021-09-23) (cit. on p. 5).

[171] D. J. Jeffrey and A. C. Norman. "Not Seeing the Roots for the Branches: Multivalued Functions in Computer Algebra". In: *ACM SIGSAM Bulletin* 38.3 (2004-09), pp. 57–66. ISSN: 0163-5824. DOI: 10/bzw56h (cit. on pp. 34, 130).

[172] D. J. Jeffrey. "Multivalued Elementary Functions in Computer-Algebra Systems". In: *Artificial Intelligence and Symbolic Computation*. Vol. 8884. Cham: Springer International Publishing, 2014, pp. 157–167. DOI: 10.1007/978-3-319-13770-4_14 (cit. on pp. 5, 34).

[173] R. D. Jenks and R. S. Sutor. *Axiom: The Scientific Computation System*. Berlin, Heidelberg: Springer-Verlag, 1992. 742 pp. ISBN: 978-0-387-97855-0 (cit. on pp. 5, 18, 162).

[174] M. Johnson, M. Schuster, Q. V. Le, et al. "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation". In: *Transactions of the Association for Computational Linguistics* 5.0 (2017), pp. 339–351. ISSN: 2307-387X. DOI: 10/gfzt8s (cit. on p. 100).

[175] D. Joyner. "AMS Special Session on SAGE and Mathematical Research Using Open Source Software". In: *ACM Communications in Computer Algebra* 43.1/2 (2009-09-09), pp. 49–54. ISSN: 1932-2240. DOI: 10/fs4nvd (cit. on p. 18).

[176] D. Joyner. "Open Source Computer Algebra Systems: Axiom". In: *ACM Communications in Computer Algebra* 42.1-2 (2008-07-25), pp. 39–47. ISSN: 1932-2240. DOI: 10/c67rv6 (cit. on pp. 18, 33).

[177] D. Joyner. "Open Source Computer Algebra Systems: GAP". In: *ACM Communications in Computer Algebra* 43.3/4 (2010-06-24), pp. 110–118. ISSN: 1932-2240. DOI: 10/d58v72 (cit. on pp. 18, 33, 134).

[178] D. Joyner, O. Čertík, A. Meurer, and B. E. Granger. "Open Source Computer Algebra Systems: SymPy". In: *ACM Communications in Computer Algebra* 45.3/4 (2012-01-23), pp. 225–234. ISSN: 1932-2240. DOI: 10/fzb6dm (cit. on pp. 18, 33).

[179] N. Kajler and N. Soiffer. "A Survey of User Interfaces for Computer Algebra Systems". In: *Journal of Symbolic Computation* 25.2 (1998-02), pp. 127–159. ISSN: 0747-7171. DOI: 10/ctkv49 (cit. on p. 20).

[180] C. Kaliszyk and F. Wiedijk. "Certified Computer Algebra on Top of an Interactive Theorem Prover". In: *Towards Mechanized Mathematical Assistants*. Vol. 4573. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 94–105. DOI: 10.1007/978-3-540-73086-6_8 (cit. on pp. 115, 116).

[181] S. Kamali and F. W. Tompa. "A New Mathematics Retrieval System". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management - CIKM '10*. Toronto, ON, Canada: ACM Press, 2010, p. 1413. DOI: 10/ckfzxp (cit. on pp. 6, 72).

[182] S. Kamali and F. W. Tompa. "Retrieving Documents with Mathematical Content". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Dublin Ireland: ACM, 2013-07-28, pp. 353–362. DOI: 10/gn3szd (cit. on pp. 6, 72).

[183] D. Kang, A. Head, R. Sidhu, K. Lo, D. Weld, and M. A. Hearst. "Document-Level Definition Detection in Scholarly Documents: Existing Models, Error Analyses, and Future Directions". In: *Proceedings of the First Workshop on Scholarly Document Processing*. Online: Association for Computational Linguistics, 2020, pp. 196–206. DOI: 10/gjzg7z (cit. on pp. 4, 53–55, 104, 109, 138, 140, 146, 154).

[184] N. P. Karampetakis and A. I. G. Vardulakis. "Special Issue on the Use of Computer Algebra Systems for Computer Aided Control System Design". In: *International Journal of Control* 79.11 (2006-11), pp. 1313–1320. ISSN: 0020-7179, 1366-5820. DOI: 10/fbft5v (cit. on pp. 2, 6).

[185] M. T. Khan. "Formal Specification and Verification of Computer Algebra Software". PhD thesis. Linz Austria: Johannes Kepler University Linz, 2014-04. 188 pp. URL: https://www3.risc.jku.at/publications/download/risc_4981/main.pdf (cit. on p. 116).

[186] Y. Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: ACL, 2014, pp. 1746–1751. DOI: 10/gf6d4z (cit. on p. 59).

[187] D. E. Knuth. "Literate Programming". In: *The Computer Journal* 27.2 (1984-02-01), pp. 97–111. ISSN: 0010-4620, 1460-2067. DOI: 10/c73jjw (cit. on pp. 26, 36).

[188] D. E. Knuth. "Semantics of Context-Free Languages". In: *Mathematical Systems Theory* 2.2 (1968-06), pp. 127–145. ISSN: 0025-5661, 1433-0490. DOI: 10/fmtcmt (cit. on p. 35).

[189] D. E. Knuth. *Digital Typography*. Vol. 78. CSLI Lecture Notes. Stanford, Calif: Cambridge University Press, 1999. 685 pp. ISBN: 978-1-57586-010-7 (cit. on p. 25).

[190] W. Koepf. "On Two Conjectures of M. S. Robertson". In: *Complex Variables, Theory and Application: An International Journal* 16.2-3 (1991-04), pp. 127–130. ISSN: 0278-1077. DOI: 10/c2gs3m (cit. on p. 125).

[191] A. Kohlhase. "Factors for Reading Mathematical Expressions". In: *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", LWDA 2018, Mannheim, Germany, August 22-24, 2018*. Vol. 2191. Mannheim, Germany: CEUR-WS.org, 2018, pp. 195–202. URL: http://ceur-ws.org/Vol-2191/paper24.pdf (cit. on p. 71).

[192] A. Kohlhase and M. Fürsich. "Understanding Mathematical Expressions: An Eye-Tracking Study". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 1785. Bialystok, Poland: CEUR-WS.org, 2016, pp. 42–50. DOI: 10/gn3szj (cit. on p. 35).

[193] A. Kohlhase, M. Kohlhase, and M. Fürsich. "Visual Structure in Mathematical Expressions". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Edinburgh, UK: Springer, 2017, pp. 208–223. DOI: 10.1007/978-3-319-62075-6_15 (cit. on p. 71).

[194] A. Kohlhase, M. Kohlhase, and C. Lange. "Dimensions of Formality: A Case Study for MKM in Software Engineering". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 6167. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 355–369. DOI: 10/ckpnmz (cit. on p. 30).

[195] A. Kohlhase, M. Kohlhase, and C. Lange. "sTeX+: A System for Flexible Formalization of Linked Data". In: *Proceedings of the 6th International Conference on Semantic Systems - I-SEMANTICS '10*. Graz, Austria: ACM Press, 2010, p. 1. DOI: 10/d8gqrz (cit. on pp. 21, 30, 31).

[196] A. Kohlhase, M. Kohlhase, and T. Ouypornkochagorn. "Discourse Phenomena in Mathematical Documents". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11006. Hagenberg, Austria: Springer, 2018, pp. 147–163. DOI: 10.1007/978-3-319-96812-4_14 (cit. on p. 71).

[197] M. Kohlhase. "Math Object Identifiers - Towards Research Data in Mathematics". In: *Lernen, Wissen, Daten, Analysen (LWDA) Conference Proceedings, Rostock, Germany, September 11-13, 2017*. Vol. 1917. CEUR-WS.org, 2017, p. 241. URL: http://ceur-ws.org/Vol-1917/paper33.pdf (visited on 2021-09-05) (cit. on p. 61).

[198] M. Kohlhase. *OMDoc – An Open Markup Format for Mathematical Documents [Version 1.2]*. Vol. 4180. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN: 978-3-540-37898-3. DOI: 10.1007/11826095 (cit. on pp. 20, 21, 23, 25, 167).

[199] M. Kohlhase. "OMDoc: An Infrastructure for OpenMath Content Dictionary Information". In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 43–48. ISSN: 0163-5824. DOI: 10/b6kd75 (cit. on p. 25).

[200] M. Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008-12), pp. 279–304. ISSN: 1661-8270, 1661-8289. DOI: 10/d8f56x (cit. on pp. 18, 20, 30, 31, 169).

[201] M. Kohlhase, J. Corneli, C. David, et al. "The Planetary System: Web 3.0 & Active Documents for STEM". In: *Procedia Computer Science* 4 (2011), pp. 598–607. ISSN: 1877-0509. DOI: 10/fhcbcr (cit. on pp. 6, 25, 33).

[202] M. Kohlhase and M. Iancu. "Discourse-Level Parallel Markup and Meaning Adoption in Flexiformal Theory Graphs". In: *Mathematical Software – ICMS 2014*. Vol. 8592. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 36–40. DOI: 10.1007/978-3-662-44199-2_7 (cit. on p. 22).

[203] M. Kohlhase, B. A. Matican, and C.-C. Prodescu. "MathWebSearch 0.5: Scaling an Open Formula Search Engine". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Bremen, Germany: Springer Berlin Heidelberg, 2012, pp. 342–357. DOI: 10.1007/978-3-642-31374-5_23 (cit. on p. 6).

[204] M. Kohlhase and F. Rabe. "Semantics of OpenMath and MathML3". In: *Mathematics in Computer Science* 6.3 (2012-09), pp. 235–260. ISSN: 1661-8289. DOI: 10/gn3sw5 (cit. on pp. 22, 23, 104).

[205] M. Kohlhase, F. Rabe, and M. Wenzel. "Making Isabelle Content Accessible in Knowledge Representation Formats". In: *25th International Conference on Types for Proofs and Programs, TYPES 2019, June 11-14, 2019, Oslo, Norway*. Vol. 175. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 1:1–1:24. DOI: 10/gn3sw3 (cit. on pp. 20, 21).

[206] A. Konovalov and S. Linton. *Symbolic Computation Software Composability Protocol - GAP Package*. Version 2.3.1. 2020. URL: https://www.gap-system.org/Manuals/pkg/SCSCP/doc/chap0.html (visited on 2021-09-23) (cit. on p. 33).

[207] I. S. Kotsireas and E. Martínez-Moro. *Applications of Computer Algebra*. Vol. 198. Springer Proceedings in Mathematics & Statistics. Cham: Springer International Publishing, 2017. 515 pp. ISBN: 978-3-319-56930-7. DOI: 10.1007/978-3-319-56932-1 (cit. on pp. 1, 2).

[208] G. Y. Kristianto and A. Aizawa. "Linking Mathematical Expressions to Wikipedia". In: *Proceedings of the 1st Workshop on Scholarly Web Mining - SWM '17*. Cambridge, United Kingdom: ACM Press, 2017, pp. 57–64. DOI: 10/gn3szb (cit. on pp. 6, 52, 53).

[209] G. Y. Kristianto, M.-Q. Nghiem, Y. Matsubayashi, and A. Aizawa. "Extracting Definitions of Mathematical Expressions in Scientific Papers". In: *Proceedings of The Japanese Society for Artificial Intelligence*. The Japanese Society for Artificial Intelligence, 2012. DOI: 10/gn3sxh (cit. on pp. 52, 58, 146).

[210] G. Y. Kristianto, G. Topic, and A. Aizawa. "MCAT Math Retrieval System for NTCIR-12 MathIR Task". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-12)*. Tokyo, Japan: National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/04-NTCIR12-MathIR-KristiantoGY.pdf (visited on 2021-08-19) (cit. on p. 53).

[211] G. Y. Kristianto, G. Topic, F. Ho, and A. Aizawa. "The MCAT Math Retrieval System for NTCIR-11 Math Track". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/06-NTCIR11-MATH-KristiantoGY.pdf (visited on 2021-08-19) (cit. on pp. 6, 53, 72).

[212] G. Y. Kristianto, G. Topić, and A. Aizawa. "Entity Linking for Mathematical Expressions in Scientific Documents". In: *Digital Libraries: Knowledge, Information, and Data in an Open Access Society*. Vol. 10075. Cham: Springer International Publishing, 2016, pp. 144–149. DOI: 10/gn3sx8 (cit. on pp. 6, 53).

[213] G. Y. Kristianto, G. Topić, and A. Aizawa. "Extracting Textual Descriptions of Mathematical Expressions in Scientific Papers". In: *D-Lib Magazine* 20.11/12 (2014-11). ISSN: 1082-9873. DOI: 10/ggwj88 (cit. on pp. 58, 61, 104, 109, 110).

[214] G. Y. Kristianto, G. Topić, and A. Aizawa. "Utilizing Dependency Relationships between Math Expressions in Math IR". In: *Information Retrieval Journal* 20.2 (2017-04), pp. 132–167. ISSN: 1386-4564, 1573-7659. DOI: 10/ggv8gd (cit. on pp. 4, 12, 53–55, 63, 71, 76, 90, 104, 108–110, 116, 138, 151).

[215] K. Krstovski and D. M. Blei. *Equation Embeddings*. 2018-03-24. URL: http://arxiv.org/abs/1803.09123 (visited on 2021-06-30) (cit. on pp. 11, 26, 37, 38, 52, 53, 59, 62, 70).

[216] D. S. Kulyabov. "Using Two Types of Computer Algebra Systems to Solve Maxwell Optics Problems". In: *Programming and Computer Software* 42.2 (2016-03), pp. 77–83. ISSN: 0361-7688. DOI: 10/gn3swk (cit. on p. 32).

[217] S. Lai, K. Liu, S. He, and J. Zhao. "How to Generate a Good Word Embedding". In: *IEEE Intelligent Systems* 31.6 (2016-11), pp. 5–14. ISSN: 1541-1672. DOI: 10/gg3jbp (cit. on p. 59).

[218] L. Lamban, J. Rubio, F. J. Martin-Mateos, and J. L. Ruiz-Reina. "Verifying the Bridge between Simplicial Topology and Algebra: The Eilenberg-Zilber Algorithm". In: *Logic Journal of IGPL* 22.1 (2014-02-01), pp. 39–65. ISSN: 1367-0751, 1368-9894. DOI: 10/gn3sxb (cit. on p. 116).

[219] G. Lample and F. Charton. "Deep Learning For Symbolic Mathematics". In: *8th International Conference on Learning Representations*. Addis Ababa, Ethiopia: OpenReview.net, 2020. URL: https://openreview.net/forum?id=S1eZYeHFDS (visited on 2021-09-10) (cit. on p. 96).

[220] L. Lamport. *LaTeX: A Document Preparation System*. Reading, Mass: Addison-Wesley Pub. Co, 1986. 242 pp. ISBN: 978-0-201-15790-1 (cit. on pp. 2, 18, 25).

[221] J. H. Lau and T. Baldwin. "An Empirical Evaluation of Doc2vec with Practical Insights into Document Embedding Generation". In: *Proceedings of the 1st Workshop on Representation Learning for NLP, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016*. Association for Computational Linguistics, 2016, pp. 78–86. DOI: 10/ggv7nm (cit. on p. 64).

[222] Q. V. Le and T. Mikolov. "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. Vol. 32. JMLR.org, 2014, pp. 1188–1196. URL: http://proceedings.mlr.press/v32/le14.html (visited on 2021-09-05) (cit. on pp. 59, 62, 64, 65, 163).

[223] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. "End-to-End Neural Coreference Resolution". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 188–197. DOI: 10/gfwtnp (cit. on p. 59).

[224] W. Lee, R. Sharma, and A. Aiken. "On Automatically Proving the Correctness of Math.h Implementations". In: *Proceedings of the ACM on Programming Languages* 2.47 (2018-01), pp. 1–32. ISSN: 2475-1421. DOI: 10/gn3sw9 (cit. on p. 116).

[225] K. R. M. Leino. "Program Proving Using Intermediate Verification Languages (IVLs) like Boogie and Why3". In: *ACM SIGAda Ada Letters* 32.3 (2012-11-29), pp. 25–26. ISSN: 1094-3641. DOI: 10/gn3sw8 (cit. on p. 116).

[226] F. Lemmerich, D. Sáez-Trumper, R. West, and L. Zia. "Why the World Reads Wikipedia: Beyond English Speakers". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. Melbourne VIC Australia: ACM, 2019-01-30, pp. 618–626. DOI: 10/fgnn (cit. on p. 157).

[227] V. I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals". In: *Soviet Physics Doklady* 10.8 (1966). URL: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf (cit. on p. 97).

[228] R. H. Lewis and M. Wester. "Comparison of Polynomial-Oriented Computer Algebra Systems". In: *ACM SIGSAM Bulletin* 33.4 (1999-12), pp. 5–13. ISSN: 0163-5824. DOI: 10/bbgvz8 (cit. on p. 116).

[229] J. Li and D. Jurafsky. "Do Multi-Sense Embeddings Improve Natural Language Understanding?" In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal: The Association for Computational Linguistics, 2015, pp. 1722–1732. DOI: 10/gfzmfk (cit. on p. 63).

[230] X. Lin, L. Gao, Z. Tang, X. Hu, and X. Lin. "Identification of Embedded Mathematical Formulas in PDF Documents Using SVM". In: *Document Recognition and Retrieval XIX, Part of the IS&T-SPIE Electronic Imaging Symposium, Burlingame, California, USA, January 25-26, 2012, Proceedings*. Vol. 8297. SPIE, 2012, p. 82970D. DOI: 10/bs8gkt (cit. on p. 36).

[231] A. Lipani, L. Andersson, F. Piroi, M. Lupu, and A. Hanbury. "TUW-IMP at the NTCIR-11 Math-2". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/09-NTCIR11-MATH-LipaniA.pdf (cit. on pp. 72, 75).

[232] M. Líska, P. Sojka, and M. Ruzicka. "Similarity Search for Mathematics: Masaryk University Team at the NTCIR-10 Math Task". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-10)*. Tokyo, Japan: National Institute of Informatics (NII), 2013. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/MATH/06-NTCIR10-MATH-LiskaM.pdf (visited on 2021-08-19) (cit. on p. 53).

[233] J. Liu, H. Li, S. Zhang, and W. Liang. "A Novel Italic Detection and Rectification Method for Chinese Advertising Images". In: *2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18-21, 2011*. IEEE Computer Society, 2011, pp. 698–702. DOI: 10/fmqzm6 (cit. on p. 37).

[234] X. Liu, Y. Shen, K. Duh, and J. Gao. "Stochastic Answer Networks for Machine Reading Comprehension". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2018, pp. 1694–1704. DOI: 10/gkz6nx (cit. on p. 59).

[235] Y. Liu, M. Ott, N. Goyal, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692 (visited on 2021-09-05) (cit. on p. 63).

[236] A. Lohia, K. Sinha, S. Vadapalli, and K. Karlapalem. "An Architecture for Searching and Indexing Latex Equations in Scientific Literature". In: *Proc. COMAD*. Goa, India: Computer Society of India, 2005, pp. 122–130. URL: http://comad2005.persistent.co.in/COMAD2005Proc/pages122-130.pdf (cit. on pp. 6, 72).

[237] H. Makishita. "Practice with Computer Algebra Systems in Mathematics Education and Teacher Training Courses". In: *Proceedings of the International Conference on Mathematical Software (ICMS)*. Vol. 8592. Seoul, South Korea: Springer, 2014, pp. 594–600. DOI: 10/gn3swr (cit. on pp. 2, 32).

[238] M. Mancini, J. Camacho-Collados, I. Iacobacci, and R. Navigli. "Embedding Words and Senses Together via Joint Knowledge-Enhanced Training". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3-4, 2017*. Association for Computational Linguistics, 2017, pp. 100–111. DOI: 10/gn3sxd (cit. on p. 63).

[239] J. Mannes. *Facebook's fastText library is now optimized for mobile*. https://techcrunch.com/2017/05/02/facebooks-fasttext-library-is-now-optimized-for-mobile/ [Accessed: 21st Sep. 2019]. 2017 (cit. on p. 59).

[240] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 55–60. DOI: 10/gf3xhp (cit. on pp. 108, 163).

[241] B. Mansouri, A. Agarwal, D. W. Oard, and R. Zanibbi. "Advancing Math-Aware Search: The ARQMath-2 Lab at CLEF 2021". In: *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*. Vol. 12657. Springer, 2021, pp. 631–638. DOI: 10/gn3sxk (cit. on p. 52).

[242] B. Mansouri, S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi. "Tangent-CFT: An Embedding Model for Mathematical Formulas". In: *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*. ACM, 2019, pp. 11–18. DOI: 10/ghp438 (cit. on pp. 11, 21, 36, 52).

[243] *Maple SCSCP Client*. Version Maple 16. Maplesoft. URL: https://www.maplesoft.com/products/maple/new_features/maple16/connectivity.aspx (visited on 2021-09-23) (cit. on pp. 25, 33).

[244] N. Marshall, C. Buteau, D. H. Jarvis, and Z. Lavicza. "Do Mathematicians Integrate Computer Algebra Systems in University Teaching? Comparing a Literature Review to an International Survey Study". In: *Computers & Education* 58.1 (2012-01), pp. 423–434. ISSN: 0360-1315. DOI: 10/c72gss (cit. on p. 32).

[245] R. Martinez. *Latex2mathml*. 2018. URL: https://github.com/roniemartinez/latex2mathml (visited on 2021-09-23) (cit. on pp. 49, 54).

[246] *MATLAB*. Version R2021a. MathWorks, 2021. URL: https://www.mathworks.com/products/matlab.html (visited on 2021-09-23) (cit. on pp. 2, 5, 166).

[247] T. Matsuzaki, H. Iwane, H. Anai, and N. H. Arai. "The Most Uncreative Examinee: A First Step toward Wide Coverage Natural Language Math Problem Solving". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. AAAI Press, 2014, pp. 1098–1104. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8524 (visited on 2021-09-05) (cit. on p. 69).

[248] D. Matthews. "Craft Beautiful Equations in Word with LaTeX". In: *Nature* 570.7760 (2019-06), pp. 263–264. ISSN: 0028-0836, 1476-4687. DOI: 10/gf3wdq (cit. on pp. 2, 19).

[249] A. Mazzei, M. Monticone, and C. Bernareggi. "Using NLG for Speech Synthesis of Mathematical Sentences". In: *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*. Association for Computational Linguistics, 2019, pp. 463–472. DOI: 10/gn3sww (cit. on p. 21).

[250] B. McCann, J. Bradbury, C. Xiong, and R. Socher. "Learned in Translation: Contextualized Word Vectors". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. 2017, pp. 6294–6305. URL: https://proceedings.neurips.cc/paper/2017/hash/20c86a628232a67e7bd46f76fba7ce12-Abstract.html (visited on 2021-09-05) (cit. on p. 59).

[251] D. McKain. *SnuggleTeX*. Version 1.2.2. The University of Edinburgh, 2010. URL: https://www2.ph.ed.ac.uk/snuggletex (visited on 2021-09-23) (cit. on pp. 3, 4, 26, 49, 54, 169).

[252] A. Meurer, C. P. Smith, M. Paprocki, et al. "SymPy: Symbolic Computing in Python". In: *PeerJ Computer Science* 3 (2017-01-02), e103. ISSN: 2376-5992. DOI: 10/gfb682 (cit. on pp. 2, 4, 5, 18, 32, 33, 169).

[253] N. Meuschke, M. Schubotz, F. Hamborg, T. Skopal, and B. Gipp. "Analyzing Mathematical Content to Detect Academic Plagiarism". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. Singapore Singapore: ACM, 2017-11-06, pp. 2211–2214. DOI: 10/cf9g (cit. on pp. 6, 42, 72, 89).

[254] N. Meuschke, V. Stange, M. Schubotz, M. Kramer, and B. Gipp. "Improving Academic Plagiarism Detection for STEM Documents by Analyzing Mathematical Content and Citations". In: *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. Champaign, IL, USA: IEEE, 2019-06, pp. 120–129. DOI: 10/ggv8jd (cit. on pp. 6, 72, 89).

[255] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781 (visited on 2021-09-05) (cit. on pp. 59, 64).

[256] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. Lake Tahoe, Nevada, USA: Curran Associates Inc., 2013, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html (visited on 2021-09-05) (cit. on pp. 11, 37, 59, 61–64).

[257] B. R. Miller. *LaTeXML: A LaTeX to XML Converter*. http://dlmf.nist.gov/LaTeXML/. Accessed: 2019-09-01. URL: http://dlmf.nist.gov/LaTeXML/ (cit. on pp. 20–22, 26, 30, 31, 36, 44, 49, 54, 73, 81).

[258] B. R. Miller. "RFC: DLMF Content Dictionaries". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 2307. Hagenberg, Austria, 2018. URL: http://ceur-ws.org/Vol-2307/paper52.pdf (cit. on pp. 24, 29).

[259] B. R. Miller. "Strategies for Parallel Markup". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 9150. Cham: Springer International Publishing, 2015, pp. 203–210. DOI: 10/gn3sxx (cit. on pp. 22, 23, 52, 54).

[260] B. R. Miller and A. Youssef. "Technical Aspects of the Digital Library of Mathematical Functions". In: *Annals of Mathematics and Artificial Intelligence* 38.1/3 (2003), pp. 121–136. ISSN: 1012-2443. DOI: 10.1023/A:1022967814992 (cit. on pp. 18, 20, 28, 29, 47, 91, 102, 104, 110, 115, 140, 153).

[261] G. A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10/d2hsft (cit. on pp. 54, 69).

[262] M. Minimair. "Collaborative Computer Algebra Systems". In: *ACM Communications in Computer Algebra*. Springer Proceedings in Mathematics & Statistics 49.2 (2017), pp. 56–57. DOI: 10/gn3swm (cit. on pp. 2, 32).

[263] P.-E. Moreau, C. Ringeissen, and M. Vittek. "A Pattern Matching Compiler for Multiple Target Languages". In: *Compiler Construction*. Vol. 2622. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 61–76. DOI: 10/d5325w (cit. on p. 104).

[264] J. Moses. "Macsyma: A Personal History". In: *Journal of Symbolic Computation* 47.2 (2012-02), pp. 123–130. ISSN: 0747-7171. DOI: 10/fkvt9t (cit. on pp. 33, 166).

[265] A. Navarro and V. Vassiliadis. "Computer Algebra Systems Coming of Age: Dynamic Simulation and Optimization of DAE Systems in Mathematica™". In: *Computers & Chemical Engineering* 62 (2014-03), pp. 125–138. ISSN: 0098-1354. DOI: 10/f5q9pj (cit. on p. 32).

[266] M. S. Nawaz, M. Malik, Y. Li, M. Sun, and M. I. U. Lali. *A Survey on Theorem Provers in Formal Methods*. 2019-12-06. URL: http://arxiv.org/abs/1912.03028 (visited on 2021-08-19) (cit. on pp. 18, 19, 34).

[267] A. Nazemi, I. Murray, and D. A. McMeekin. "Mathematical Information Retrieval (MIR) from Scanned PDF Documents and MathML Conversion". In: *IPSJ Transactions on Computer Vision and Applications* 6.0 (2014), pp. 132–142. ISSN: 1882-6695. DOI: 10/gn3sx3 (cit. on pp. 19, 22, 26).

[268] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: ACL, 2014, pp. 1059–1069. DOI: 10/ggwj6j (cit. on p. 63).

[269] M.-Q. Nghiem, G. Y. Kristianto, and A. Aizawa. "Mining Coreference Relations between Formulas and Text Using Wikipedia". In: *Proceedings of 2nd Workshop on NLP Challenges in the Information Explosion Era (NLPIX)*. 2010, pp. 69–74 (cit. on p. 54).

[270] M.-Q. Nghiem, G. Y. Kristianto, and A. Aizawa. "Using MathML Parallel Markup Corpora for Semantic Enrichment of Mathematical Expressions". In: *IEICE Transactions on Information and Systems* E96.D.8 (2013), pp. 1707–1715. ISSN: 0916-8532, 1745-1361. DOI: 10/gbdkh3 (cit. on pp. 6, 18, 20, 22, 52, 54, 104, 169).

[271] M.-Q. Nghiem, G. Y. Kristianto, Y. Matsubayashi, and A. Aizawa. "Automatic Approach to Understanding Mathematical Expressions Using Mathml Parallel Markup Corpora". In: *Proceedings of the 26th Annu. Conf. of the Japanese Society for Artificial Intelligence*. 2012. URL: http://inftyreader.org/inftyreader-kb/Automatic%20Approach%20to%20Understanding%20Mathematical%20Expressions%20Using%20MathML%20Parallel%20Markup%20Corpora.pdf (cit. on pp. 42, 54).

[272] M. Nickel and D. Kiela. "Poincaré Embeddings for Learning Hierarchical Representations". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. 2017, pp. 6338–6347. URL: https://proceedings.neurips.cc/paper/2017/hash/59dfa2df42d9e3d41f5b02bfc32229dd-Abstract.html (visited on 2021-09-05) (cit. on p. 59).

[273] G. Nishizawa, J. Liu, Y. Diaz, A. Dmello, W. Zhong, and R. Zanibbi. "MathSeer: A Math-Aware Search Interface with Intuitive Formula Editing, Reuse, and Lookup". In: *Advances in Information Retrieval*. Vol. 12036. Cham: Springer International Publishing, 2020, pp. 470–475. DOI: 10.1007/978-3-030-45442-5_60 (cit. on pp. 32, 53).

[274] S. Ohashi, G. Y. Kristianto, G. Topic, and A. Aizawa. "Efficient Algorithm for Math Formula Semantic Search". In: *IEICE Transactions* 99-D.4 (2016), pp. 979–988. DOI: 10.1587/transinf.2015DAP0023 (cit. on pp. 6, 72).

[275] A. Ohri and T. Schmah. "Machine Translation of Mathematical Text". In: *IEEE Access* 9 (2021), pp. 38078–38086. ISSN: 2169-3536. DOI: 10/gnbwh7 (cit. on p. 96).

[276] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark. *NIST Handbook of Mathematical Functions*. Cambridge New York: Cambridge University Press NIST, 2010. 968 pp. ISBN: 9780521192255 (cit. on pp. 5, 29, 115, 164, 167).

[277] M. Ott, S. Edunov, D. Grangier, and M. Auli. "Scaling Neural Machine Translation". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, 2018, pp. 1–9. DOI: 10/ggcmmz (cit. on pp. 96, 97).

[278] L. Padovani. "On the Roles of LaTeX and MathML in Encoding and Processing Mathematical Expressions". In: *Mathematical Knowledge Management (MKM), Bertinoro, Italy*. Vol. 2594. 2003, pp. 66–79. ISBN: 3-540-00568-4. DOI: 10.1007/3-540-36469-2_6 (cit. on p. 42).

[279] R. Pagel and M. Schubotz. "Mathematical Language Processing Project". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 1186. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1186/paper-23.pdf (visited on 2021-08-19) (cit. on pp. 12, 52–55, 63, 92, 104, 108–110, 151, 167).

[280] S. Pakin. *The Comprehensive LaTeX Symbol List*. 2021-05-05, p. 422. URL: https://ftp.kddilabs.jp/CTAN/info/symbols/comprehensive/symbols-a4.pdf (cit. on p. 31).

[281] M. Palmer, P. R. Kingsbury, and D. Gildea. "The Proposition Bank: An Annotated Corpus of Semantic Roles". In: *Comput. Linguistics* 31.1 (2005), pp. 71–106. DOI: 10.1162/0891201053630264 (cit. on p. 70).

[282] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. DOI: 10/dmgshg (cit. on pp. 14, 97, 132, 162).

[283] *PARI/GP*. Version 2.11.2. University of Bordeaux: The PARI Group, 2019. URL: http://pari.math.u-bordeaux.fr/ (visited on 2021-09-23) (cit. on pp. 33, 134).

[284] B. Parisse. *Compiling LATEX to Computer Algebra-Enabled HTML5*. 2017-07-05. URL: http://arxiv.org/abs/1707.01271 (visited on 2021-07-01) (cit. on p. 6).

[285] A. Patel, S. Bhattamishra, and N. Goyal. "Are NLP Models Really Able to Solve Simple Math Word Problems?" In: *Proc. Conf. North American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. ACL, 2021, pp. 2080–2094. URL: https://www.aclweb.org/anthology/2021.naacl-main.168/ (cit. on p. 53).

[286] N. Pattaniyil and R. Zanibbi. "Combining TF-IDF Text Retrieval with an Inverted Index over Symbol Pairs in Math Expressions: The Tangent Math Search Engine at NTCIR 2014". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/08-NTCIR11-MATH-PattaniyilN.pdf (visited on 2021-08-19) (cit. on p. 36).

[287] L. C. Paulson. "Isabelle: The next Seven Hundred Theorem Provers". In: *9th International Conference on Automated Deduction*. Vol. 310. Berlin/Heidelberg: Springer-Verlag, 1988, pp. 772–773. DOI: 10.1007/BFb0012891 (cit. on pp. 18, 34).

[288] M. Pawlik and N. Augsten. "RTED: A Robust Algorithm for the Tree Edit Distance". In: *CoRR* abs/1201.0230 (2012). URL: http://arxiv.org/abs/1201.0230 (cit. on pp. 48, 50).

[289] K. Peeters. *Cadabra*. Version 2.3.5. 2020-11-17. URL: https://cadabra.science/ (visited on 2021-09-23) (cit. on p. 33).

[290] K. Peeters. "Cadabra: A Field-Theory Motivated Symbolic Computer Algebra System". In: *Computer Physics Communications* 176.8 (2007-04), pp. 550–558. ISSN: 0010-4655. DOI: 10/c7mpdq (cit. on pp. 33, 134).

[291] K. Peeters. *Introducing Cadabra: A Symbolic Computer Algebra System for Field Theory Problems*. 2018-04-03. URL: http://arxiv.org/abs/hep-th/0701238 (visited on 2021-08-23) (cit. on p. 33).

[292] D. Pellegrino. "A short communication on the constants of the multilinear Hardy–Littlewood inequality". In: *CoRR* abs/1510.00367 (2015). URL: https://arxiv.org/abs/1510.00367 (cit. on p. 89).

[293] J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: ACL, 2014, pp. 1532–1543. DOI: 10/gfshwg (cit. on pp. 59, 61–63).

[294] A. S. Perminov and E. D. Kuznetsov. "The Implementation of Hori–Deprit Method to the Construction Averaged Planetary Motion Theory by Means of Computer Algebra System Piranha". In: *Mathematics in Computer Science* 14.2 (2020-06), pp. 305–316. ISSN: 1661-8270. DOI: 10/gn3swf (cit. on p. 32).

[295] M. E. Peters, M. Neumann, M. Iyyer, et al. "Deep Contextualized Word Representations". In: *Proc. Conf. North American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10/gft5gf (cit. on pp. 59, 61, 62).

[296] F. Petersen, M. Schubotz, and B. Gipp. "Towards Formula Translation Using Recursive Neural Networks". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 2307. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2307/WiP3.pdf (visited on 2021-09-23) (cit. on pp. 38, 54, 96).

[297] S. T. Piantadosi. "Zipf's word frequency law in natural language: A critical review and future directions". In: *Psychonomic Bulletin & Review* 21.5 (2014-03), pp. 1112–1130. DOI: 10.3758/s13423-014-0585-6 (cit. on pp. 71, 77, 78).

[298] M. Piastra and R. Bolognesi. "An Efficient Context-Free Parsing Algorithm with Semantic Actions". In: *Trends in Artificial Intelligence*. Vol. 549. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 271–280. DOI: 10.1007/3-540-54712-6_239 (cit. on p. 35).

[299] T. Piccardi, M. Catasta, L. Zia, and R. West. "Structuring Wikipedia Articles with Section Recommendations". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. Ann Arbor MI USA: ACM, 2018-06-27, pp. 665–674. DOI: 10/gkktmc (cit. on p. 157).

[300] E. Pietriga. *MathML Content2Presentation Transformation (MathMLc2p)*. Version Version 2007-05-22. The JEuclid Project. URL: http://jeuclid.sourceforge.net/content.html (visited on 2021-09-23) (cit. on p. 21).

[301] M. T. Pilehvar, J. Camacho-Collados, R. Navigli, and N. Collier. "Towards a Seamless Integration of Word Senses into Downstream NLP Applications". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2017, pp. 1857–1869. DOI: 10/gfzt8r (cit. on p. 63).

[302] M. T. Pilehvar and N. Collier. "De-Conflated Semantic Representations". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, USA: The Association for Computational Linguistics, 2016, pp. 1680–1690. DOI: 10/ggwj6n (cit. on p. 63).

[303] H. Prieto, S. Dalmas, and Y. Papegay. "Mathematica as an OpenMath Application". In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 22–26. ISSN: 0163-5824. DOI: 10/b8pq3h (cit. on pp. 6, 7, 21, 25, 39).

[304] J. R. Campos, W. Cavalcante, V. V. Fávaro, D. Nuñez-Alarcón, D. Pellegrino, and D. M. Serrano-Rodríguez. "Polynomial and multilinear Hardy–Littlewood inequalities: analytical and numerical approaches". In: *CoRR* abs/1503.00618 (2015). URL: https://arxiv.org/abs/1503.00618 (cit. on p. 89).

[305] A. Raganato, C. D. Bovi, and R. Navigli. "Neural Sequence Learning Models for Word Sense Disambiguation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 1156–1167. DOI: 10/gn3sv4 (cit. on p. 59).

[306] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, USA: The Association for Computational Linguistics, 2016, pp. 2383–2392. DOI: 10/ghnmj5 (cit. on p. 70).

[307] R. Řehůřek and P. Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, 2010-05, pp. 45–50 (cit. on p. 64).

[308] M. Reschop. "Wissenschaftler Der Uni Wuppertal Entwickeln Funktionserweiterung Für Wikipedia". In: *Pressestelle Bergische Universität Wuppertal* (2020-01-24), p. 1. URL: https://idw-online.de/de/news730459 (visited on 2021-09-08) (cit. on p. 156).

[309] W. Research. *Wolfram|Alpha*. WolframAlpha LLC, 2021. URL: https://www.wolframalpha.com/ (visited on 2021-09-30) (cit. on p. 27).

[310] S. E. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. DOI: 10.1561/1500000019 (cit. on pp. 71, 77, 81, 163).

[311] D. Roozemond. *Macaulay2 SCSCP Support*. Version 0.2.1. URL: https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2-1.15/share/doc/Macaulay2/SCSCP/html/index.html (visited on 2021-09-23) (cit. on p. 33).

[312] T. Ruas, W. I. Grosky, and A. Aizawa. "Multi-Sense Embeddings through a Word Sense Disambiguation Process". In: *Expert Syst. Appl.* 136 (2019), pp. 288–303. DOI: 10/gf4wc6 (cit. on pp. 63, 64).

[313] M. R. Rudolph, F. J. R. Ruiz, S. Athey, and D. M. Blei. "Structured Embedding Models for Grouped Data". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. 2017, pp. 251–261. URL: https://proceedings.neurips.cc/paper/2017/hash/bd686fd640be98efaae0091fa301e613-Abstract.html (visited on 2021-09-05) (cit. on p. 61).

[314] M. Ruzicka, P. Sojka, and M. Líska. "Math Indexer and Searcher under the Hood: Fine-tuning Query Expansion and Unification Strategies". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-12)*. Tokyo, Japan: National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/MathIR/05-NTCIR12-MathIR-RuzickaM.pdf (visited on 2021-08-19) (cit. on p. 53).

[315] M. Ruzicka, P. Sojka, and M. Líska. "Math Indexer and Searcher under the Hood: History and Development of a Winning Strategy". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/07-NTCIR11-MATH-RuzickaM.pdf (visited on 2021-08-19) (cit. on p. 53).

[316] T. Sáez and A. Hogan. "Automatically Generating Wikipedia Info-boxes from Wikidata". In: *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*. Lyon, France: ACM Press, 2018, pp. 1823–1830. DOI: 10/gnd6fm (cit. on pp. 6, 53).

[317] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*. Version 9.4. 2021. DOI: 10.5281/zenodo.593563 (cit. on p. 33).

[318] P. Sandhu. "Chapter 11: Using MathML for Computations". In: *The MathML Handbook*. 1st ed. Hingham, Mass: Charles River Media, 2003, p. 127. ISBN: 978-1-58450-249-4. URL: https://flylib.com/books/en/4.532.1.69/1/ (visited on 2021-09-06) (cit. on pp. 6, 7, 20–23).

[319] P. Scharpf, I. Mackerracher, M. Schubotz, J. Beel, C. Breitinger, and B. Gipp. "AnnoMathTeX - a Formula Identifier Annotation Recommender System for STEM Documents". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. Copenhagen Denmark: ACM, 2019-09-10, pp. 532–533. DOI: 10/ggv8jt (cit. on pp. 6, 52, 157).

[320] P. Scharpf, M. Schubotz, H. S. Cohl, and B. Gipp. "Towards Formula Concept Discovery and Recognition". In: *Proc. Workshop on Bibliometric-Enhanced Information Retrieval and Natural Language Processing (BIRNDL@SIGIR)*. Vol. 2414. CEUR-WS.org, 2019, pp. 108–115. URL: http://ceur-ws.org/Vol-2414/paper11.pdf (visited on 2021-09-09) (cit. on pp. 104, 139, 140, 155).

[321] P. Scharpf, M. Schubotz, and B. Gipp. "Fast Linking of Mathematical Wikidata Entities in Wikipedia Articles Using Annotation Recommendation". In: *Companion Proceedings of the Web Conference 2021*. Ljubljana Slovenia: ACM, 2021-04-19, pp. 602–609. DOI: 10/gk5d3d (cit. on pp. 6, 52, 53, 146, 155).

[322] P. Scharpf, M. Schubotz, and B. Gipp. "Representing Mathematical Formulae in Content MathML Using Wikidata". In: *Proc. Workshop on Bibliometric-Enhanced Information Retrieval and Natural Language Processing (BIRNDL@SIGIR)*. Vol. 2132. CEUR-WS.org, 2018, pp. 46–59. URL: http://ceur-ws.org/Vol-2132/paper5.pdf (cit. on p. 6).

[323] P. Scharpf, M. Schubotz, A. Youssef, F. Hamborg, N. Meuschke, and B. Gipp. "Classification and Clustering of arXiv Documents, Sections, and Abstracts, Comparing Encodings of Natural and Mathematical Language". In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. Virtual Event China: ACM, 2020-08, pp. 137–146. DOI: 10/gn3sx4 (cit. on p. 146).

[324] W. Schelter. *Maxima*. Version 5.45.1. The Maxima Group, 2021. URL: https://maxima.sourceforge.io (visited on 2021-09-30) (cit. on pp. 3, 26, 33, 166).

[325] C. Scholl, A. Konrad, A. Mahzoon, D. Grobe, and R. Drechsler. "Verifying Dividers Using Symbolic Computer Algebra and Don't Care Optimization". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Grenoble, France: IEEE, 2021-02-01, pp. 1110–1115. DOI: 10/gn3swb (cit. on p. 32).

[326] M. Schubotz. "Augmenting Mathematical Formulae for More Effective Querying & Efficient Presentation". PhD thesis. Technische Universität Berlin, 2017. ISBN: 978-3-7450-6208-3. DOI: 10.14279/depositonce-6034 (cit. on pp. 42, 52).

[327] M. Schubotz. "Generating OpenMath Content Dictionaries from Wikidata". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 2307. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2307/paper51.pdf (visited on 2021-09-16) (cit. on pp. 24, 155).

[328] M. Schubotz. "Implicit Content Dictionaries in the NIST Digital Repository of Mathematical Formulae". English. Talk presented at the OpenMath workshop CICM. Bialystok, Poland, 2016. URL: http://cicm-conference.org/2016/cicm.php?event=&menu=talks#03 (visited on 2016-10-03) (cit. on p. 44).

[329] M. Schubotz, A. Grigorev, M. Leich, et al. "Semantification of Identifiers in Mathematics for Better Math Information Retrieval". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Pisa Italy: ACM, 2016-07-07, pp. 135–144. DOI: 10/ggv8jm (cit. on pp. 4, 12, 42, 43, 52–55, 58, 61, 63, 71, 90, 92, 102, 104, 108–110, 116, 151).

[330] M. Schubotz, L. Krämer, N. Meuschke, F. Hamborg, and B. Gipp. "Evaluating and Improving the Extraction of Mathematical Identifier Definitions". In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Vol. 10456. Cham: Springer International Publishing, 2017, pp. 82–94. DOI: 10/gn3sxn (cit. on pp. 12, 42, 52–55, 61, 63, 65, 67, 68, 76, 81, 90–92, 109, 110, 135, 151, 161).

[331] M. Schubotz, N. Meuschke, T. Hepp, H. S. Cohl, and B. Gipp. "VMEXT: A Visualization Tool for Mathematical Expression Trees". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 10383. Cham: Springer International Publishing, 2017, pp. 340–355. DOI: 10/gkj7fk (cit. on pp. 12, 24, 35, 44, 170).

[332] M. Schubotz, P. Scharpf, O. Teschke, A. Kühnemund, C. Breitinger, and B. Gipp. "AutoMSC: Automatic Assignment of Mathematics Subject Classification Labels". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 12236. Cham: Springer International Publishing, 2020, pp. 237–250. DOI: 10.1007/978-3-030-53518-6_15 (cit. on p. 146).

[333] M. Schubotz and O. Teschke. “Four Decades of TeX at zbMATH”. In: *EMS Newsletter* 2019–6.112 (2019-06-06), pp. 50–52. ISSN: 1027-488X. DOI: 10/ggv8mq (cit. on p. 73).

[334] M. Schubotz, O. Teschke, V. Stange, N. Meuschke, and B. Gipp. “Forms of Plagiarism in Digital Mathematical Libraries”. In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11617. Cham: Springer International Publishing, 2019, pp. 258–274. DOI: 10.1007/978-3-030-23250-4_18 (cit. on pp. 6, 72, 89).

[335] M. Schubotz and G. Wicke. “Mathoid: Robust, Scalable, Fast and Accessible Math Rendering for Wikipedia”. In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 8543. Springer, 2014, pp. 224–235. DOI: 10.1007/978-3-319-08434-3_17 (cit. on pp. 22, 26, 47, 49).

[336] M. Schubotz, A. Youssef, V. Markl, H. S. Cohl, and J. J. Li. “Evaluation of Similarity-Measure Factors for Formulae Based on the NTCIR-11 Math Task”. In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-11)*. Tokyo, Japan: National Institute of Informatics (NII), 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/04-NTCIR11-MATH-SchubotzM.pdf (visited on 2021-08-19) (cit. on pp. 44, 48).

[337] M. Schwarzer, M. Schubotz, N. Meuschke, C. Breitinger, V. Markl, and B. Gipp. “Evaluating Link-based Recommendations for Wikipedia”. In: *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. Newark New Jersey USA: ACM, 2016-06-19, pp. 191–200. DOI: 10/ggv7ns (cit. on p. 157).

[338] O. Seddiki. “Linking HOL Light to Mathematica Using OpenMath”. Montréal, Canada: Concordia University, 2014. 90 pp. URL: http://hvg.ece.concordia.ca/Publications/Thesis/Thesis-Ons2.pdf (visited on 2021-06-08) (cit. on pp. 6, 20, 21, 25, 34, 39).

[339] R. Shan and A. Youssef. “Towards Math Terms Disambiguation Using Machine Learning”. In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 12833. Timisoara, Romania: Springer, 2021, pp. 90–106. DOI: 10/gn3sv9 (cit. on pp. 52, 53).

[340] K. Slind and M. Norrish. “A Brief Overview of HOL4”. In: *Theorem Proving in Higher Order Logics (TPHOL)*. Vol. 5170. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 28–32. DOI: 10/bcqt9j (cit. on pp. 18, 25, 34).

[341] G. G. Smith and D. Ferguson. “Diagrams and math notation in e-learning: growing pains of a new generation”. In: *International Journal of Mathematical Education in Science and Technology* 35 (5 2004), pp. 681–695. DOI: 10.1080/00207390420002325 83 (cit. on p. 72).

[342] C. M. So and S. M. Watt. “On the Conversion between Content MathML and OpenMath”. In: *Proc. Conf. Communicating Mathematics in the Digital Era*. Aveiro, Portugal, 2006, pp. 169–182. DOI: 10/dv5m7r (cit. on pp. 6, 20–24).

[343] L. A. Sobreviela. “A Reduce-Based OpenMath ↔ MathML Translator”. In: *ACM SIGSAM Bulletin* 34.2 (2000-06), pp. 31–32. ISSN: 0163-5824. DOI: 10/ckp4qs (cit. on pp. 7, 22–25).

[344] R. Socher, A. Perelygin, J. Wu, et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Seattle, Washington, USA: ACL, 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170/ (visited on 2021-09-05) (cit. on p. 59).

[345] P. Sojka. “Exploiting Semantic Annotations in Math Information Retrieval”. In: *Proceedings of the Fifth Workshop on Exploiting Semantic Annotations in Information Retrieval - ESAIR ’12*. Maui, Hawaii, USA: ACM Press, 2012, p. 15. DOI: 10.1145/2390148.2390157 (cit. on p. 22).

[346] P. Sojka and M. Líaka. “The Art of Mathematics Retrieval”. In: *Proceedings of the 11th ACM Symposium on Document Engineering - DocEng ’11*. Mountain View, California, USA: ACM Press, 2011, p. 57. DOI: 10/b5667d (cit. on p. 52).

[347] P. Sojka and M. Líška. “Indexing and Searching Mathematics in Digital Libraries”. In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 6824. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 228–243. DOI: 10.1007/978-3-642-22673-1_16 (cit. on p. 22).

[348] P. Sojka, M. Růžička, and V. Novotný. “MIaS: Math-Aware Retrieval in Digital Mathematical Libraries”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. Torino Italy: ACM, 2018-10-17, pp. 1923–1926. DOI: 10.1145/3269206.3269233 (cit. on p. 22).

[349] A. Souza and D. Freitas. "Towards a Prosodic Model for Synthesized Speech of Mathematical Expressions in MathML". In: *9th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*. Online Portugal: ACM, 2020-12-02, pp. 105–110. DOI: 10/gn3swx (cit. on pp. 21, 22).

[350] K. Stacey, P. Flynn, and L. Berenson. "Pushing the Pen or Pushing the Button : A Catalyst for Debate over Future Goals for Mathematical Proficiency in the CAS Age". In: *Australian Senior Mathematics Journal* 16.2 (2002), pp. 7–19. ISSN: 0819-4564. DOI: 10.3316/aeipt.129371 (cit. on p. 32).

[351] H. Stamerjohanns, D. Ginev, C. David, D. Misev, V. Zamdzhiev, and M. Kohlhase. "MathML-aware Article Conversion from LaTeX". In: *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada*. 2009, pp. 109–120. URL: http://eudml.org/doc/220017 (cit. on pp. 18, 20, 42).

[352] P. S. Stanimirović, Y. Wei, D. Kolundžija, J. R. Sendra, and J. Sendra. "An Application of Computer Algebra and Dynamical Systems". In: *Algebraic Informatics*. Vol. 11545. Cham: Springer International Publishing, 2019, pp. 225–236. DOI: 10/gn3swp (cit. on p. 32).

[353] Y. Stathopoulos and S. Teufel. "Mathematical Information Retrieval Based on Type Embeddings and Query Expansion". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, 2016, pp. 2344–2355. DOI: 10/gn3sx6 (cit. on p. 37).

[354] G. Sutcliffe and C. Suttner. "Evaluating General Purpose Automated Theorem Proving Systems". In: *Artificial Intelligence* 131.1-2 (2001-09), pp. 39–54. ISSN: 0004-3702. DOI: 10/ft88fr (cit. on pp. 18, 19, 34).

[355] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. Montreal, Quebec, Canada, 2014, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html (visited on 2021-09-10) (cit. on p. 96).

[356] S. Takato, A. McAndrew, J. A. Vallejo, and M. Kaneko. "Collaborative Use of KeTCindy and Free Computer Algebra Systems". In: *Mathematics in Computer Science* 11.3-4 (2017-12), pp. 503–514. ISSN: 1661-8270. DOI: 10/gn3swn (cit. on p. 32).

[357] S. D. Team. *LaTeX Parsing Caveats - Sympy 1.8 Documentation*. Parsing. URL: https://docs.sympy.org/latest/modules/parsing.html#mathrm-latex-parsing-caveats (visited on 2021-09-12) (cit. on p. 26).

[358] T. M. Team. *MathJax*. Version 3.2.0. American Mathematical Society (AMS) and Society for Industrial and Applied Mathematics (SIAM), 2021. URL: https://www.mathjax.org (visited on 2021-09-23) (cit. on p. 21).

[359] N. TeBlunthuis. *Measuring Wikipedia Article Quality in One Dimension by Extending ORES with Ordinal Regression*. 2021-08-31. DOI: 10/gn3sw7 (cit. on pp. 102, 133, 157).

[360] A. Thawani, J. Pujara, and P. A. Szekely. "Representing Numbers in NLP: A Survey and a Vision". In: *Proc. Conf. North American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. ACL, 2021, pp. 644–656. URL: https://www.aclweb.org/anthology/2021.naacl-main.53/ (cit. on pp. 26, 37, 53).

[361] The SCIEnce project. "Symbolic Computation Software Composability Protocol and Its Implementations". In: *ACM Communications in Computer Algebra* 44.3/4 (2011-01-28), pp. 210–212. ISSN: 1932-2240. DOI: 10/bs3m5n (cit. on pp. 6, 7, 20, 21, 23, 33, 168).

[362] D. Tidwell. *XSLT*. 1st ed. Cambridge [Mass.]: O'Reilly, 2001. 460 pp. ISBN: 978-0-596-00053-0 (cit. on pp. 21, 22).

[363] P. G. Tiffany. "Effectively Melding Computer Algebra Systems into the Calculus Curriculum". In: *ACM Communications in Computer Algebra* 49.2 (2015-08-14), pp. 49–50. ISSN: 1932-2240. DOI: 10/gn3swh (cit. on pp. 2, 32).

[364] I. Toloaca and M. Kohlhase. "Notation-Based Semantification". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 1785. CEUR-WS.org, 2016, pp. 73–81. URL: http://ceur-ws.org/Vol-1785/M6.pdf (visited on 2021-08-20) (cit. on pp. 20–22, 52, 54).

[365] E. Tonisson. "Students' Comparison of Their Trigonometric Answers with the Answers of a Computer Algebra System". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 7961. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 216–229. DOI: 10/gn3swt (cit. on pp. 2, 32).

[366] G. Topic, G. Y. Kristianto, M.-Q. Nghiem, and A. Aizawa. "The MCAT Math Retrieval System for NTCIR-10 Math Track". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-10)*. Tokyo, Japan: National Institute of Informatics (NII), 2013. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/MATH/05-NTCIR10-MATH-TopicG.pdf (visited on 2021-08-19) (cit. on p. 53).

[367] 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 American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. The Association for Computational Linguistics, 2003. DOI: 10/b8gqkd (cit. on p. 108).

[368] K. Toutanvoa and C. D. Manning. "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger". In: *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP)*. Hong Kong: Association for Computational Linguistics, 2000, pp. 63–70. DOI: 10/cnnbnt (cit. on p. 108).

[369] T. H. Trinh, A. M. Dai, T. Luong, and Q. V. Le. "Learning Longer-term Dependencies in RNNs with Auxiliary Losses". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Stockholm, Sweden: PMLR, 2018, pp. 4972–4981. URL: http://proceedings.mlr.press/v80/trinh18a.html (visited on 2021-09-10) (cit. on p. 97).

[370] N. Vanetik, M. Litvak, S. Shevchuk, and L. Reznik. "Automated Discovery of Mathematical Definitions in Text". In: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*. European Language Resources Association, 2020, pp. 2086–2094. URL: https://aclanthology.org/2020.lrec-1.256/ (visited on 2021-08-15) (cit. on pp. 109, 138, 140, 146, 154).

[371] A. Vaswani, N. Shazeer, N. Parmar, et al. "Attention Is All You Need". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. Long Beach, CA, USA, 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 2021-09-10) (cit. on pp. 96, 97).

[372] J. A. M. Vermaseren. *Symbolic Manipulation with FORM*. Amsterdam: CAN (Computer Algebra Netherland), 1991. 258 pp. ISBN: 978-90-74116-01-5 (cit. on p. 33).

[373] J. Wang, Y. Sun, and S. Wang. "Image To Latex with DenseNet Encoder and Joint Attention". In: *2018 International Conference on Identification, Information and Knowledge in the Internet of Things*. Vol. 147. Beijing, China: Elsevier, 2018, pp. 374–380. DOI: 10/ghcf4v (cit. on pp. 36, 96).

[374] K. Wang, X. Li, and X. Tian. "On Ambiguity Issues of Converting LaTeX Mathematical Formula to Content MathML". In: *Collaborative Computing: Networking, Applications, and Worksharing - 11th International Conference, CollaborateCom 2015, Wuhan, China, November 10-11, 2015. Proceedings*. Vol. 163. Springer, 2015, pp. 289–295. DOI: 10/gn3sxw (cit. on pp. 19, 22, 23, 26, 52, 54).

[375] Q. Wang, C. Brown, C. Kaliszyk, and J. Urban. "Exploration of Neural Machine Translation in Autoformalization of Mathematics in Mizar". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. New Orleans LA USA: ACM, 2020-01-20, pp. 85–98. DOI: 10/gn3sxr (cit. on pp. 6, 96).

[376] Q. Wang, C. Kaliszyk, and J. Urban. "First Experiments with Neural Translation of Informal to Formal Mathematics". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11006. Hagenberg, Austria: Springer, 2018, pp. 255–270. DOI: 10/gn3sw4 (cit. on p. 96).

[377] X. Wang, Z. Wang, and J.-C. Liu. "Bigram Label Regularization to Reduce Over-Segmentation on Inline Math Expression Detection". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sydney, Australia: IEEE, 2019-09, pp. 387–392. DOI: 10/ghp886 (cit. on p. 108).

[378] Z. Wang and J.-C. Liu. "PDF2LaTeX: A Deep Learning System to Convert Mathematical Documents from PDF to LaTeX". In: *Proceedings of the ACM Symposium on Document Engineering 2020*. Virtual Event CA USA: ACM, 2020-09-29, pp. 1–10. DOI: 10/ghp44g (cit. on p. 19).

[379] Z. Wang and J.-C. Liu. "Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-Level Training". In: *International Journal on Document Analysis and Recognition (IJDAR)* 24.1-2 (2021-06), pp. 63–75. ISSN: 1433-2833, 1433-2825. DOI: 10/ghpj5t (cit. on pp. 19, 96).

[380] S. M. Watt. "Exploiting implicit mathematical semantics in conversion between TeX and MathML". In: *Proc. Internet Accessible Math. Commun. (IAMC)* (2002) (cit. on p. 42).

[381] S. M. Watt. "How to Build a Global Digital Mathematics Library". In: *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. Timisoara, Romania: IEEE, 2016-09, pp. 37–40. DOI: 10.1109/SYNASC.2016.019 (cit. on pp. 22, 29).

[382] M. Wattenberg, F. Viégas, and I. Johnson. "How to Use T-SNE Effectively". In: *Distill* 1.10 (2016-10-13), 10.23915/distill.00002. DOI: 10/gffk7g (cit. on p. 66).

[383] E. Weisstein. "Computable Data, Mathematics, and Digital Libraries in Mathematica and Wolfram|Alpha". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 8543. Cham: Springer International Publishing, 2014, pp. 26–29. DOI: 10/gn3svt (cit. on p. 27).

[384] N. White, S. Matthews, and R. Chapman. "Formal Verification: Will the Seedling Ever Flower?" In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375.2104 (2017-10-13), p. 20150402. ISSN: 1364-503X, 1471-2962. DOI: 10/gfgq2v (cit. on pp. 18, 19, 34).

[385] A. N. Whitehead and B. A. Russell. *Principia Mathematica*. 2nd ed. Cambridge University Press, 1927. URL: https://cds.cern.ch/record/268025 (cit. on p. 34).

[386] A. M. Wigmore, G. Hunter, E. Pflügel, and J. Denholm-Price. "Talkmaths: A Speech User Interface for Dictating Mathematical Expressions into Electronic Documents". In: *ISCA International Workshop on Speech and Language Technology in Education, SLaTE 2009, Warwickshire, England, UK, September 3-5, 2009*. ISCA, 2009, pp. 125–128. URL: http://www.eee.bham.ac.uk/SLaTE2009/papers/SLaTE2009-07-v2.pdf (visited on 2021-09-13) (cit. on p. 21).

[387] A. M. Wigmore, G. Hunter, E. Pflügel, J. Denholm-Price, and V. Binelli. ""Let Them TalkMaths!" - Developing an Intelligent System to Assist Disabled People to Learn and Use Mathematics on Computers through a Speech Interface: The TalkMaths and VoiceCalc Systems". In: *Proceedings of the 5th International Conference on Intelligent Environments*. Vol. 2. Barcelona, Spain: IOS Press, 2009, pp. 25–33. DOI: 10/gn3sxz (cit. on pp. 21, 22).

[388] S. Wiseman, A. M. Rush, and S. M. Shieber. "Learning Global Features for Coreference Resolution". In: *Proc. Conf. North American Chapter Association for Computaitonal Linguistics: Human Language Technology (NAACL-HLT)*. The Association for Computational Linguistics, 2016, pp. 994–1004. DOI: 10/gn3sxg (cit. on p. 59).

[389] W. Wojas and J. Krupa. "Supporting Education in Algorithms of Computational Mathematics by Dynamic Visualizations Using Computer Algebra System". In: *Computational Science – ICCS 2020*. Vol. 12143. Amsterdam, Netherlands: Springer, 2020, pp. 634–647. DOI: 10/gn3swc (cit. on pp. 2, 32).

[390] W. Wojas and J. Krupa. "Teaching Students Nonlinear Programming with Computer Algebra System". In: *Mathematics in Computer Science* 13.1-2 (2019-06), pp. 297–309. ISSN: 1661-8270. DOI: 10/gn3sws (cit. on pp. 2, 32).

[391] Wolfram. *Wolfram Language & System: Importing and Exporting*. Importing and Exporting. URL: https://reference.wolfram.com/language/tutorial/ImportingAndExporting.html (visited on 2021-09-12) (cit. on pp. 20, 21, 26, 54).

[392] S. Wolfram. *What We've Built Is a Computational Language*. 2019-05-09. URL: https://writings.stephenwolfram.com/2019/05/what-weve-built-is-a-computational-language-and-thats-very-important/ (visited on 2021-07-29) (cit. on pp. 19, 33, 36).

[393] *Wolfram Mathematica*. [Accessed: 2021-06-10]. URL: http://www.wolfram.com/mathematica (cit. on pp. 2–5, 18, 19, 32, 33, 102, 104, 165).

[394] M. Wolska and M. Grigore. "Symbol Declarations in Mathematical Writing - A Corpus Study". In: *Towards a Digital Mathematics Library*. Paris, France: Masaryk University Press, 2010, pp. 119–127. DOI: 10338.dmlcz/702580 (cit. on pp. 4, 58, 68, 92, 102).

[395] *Word*. Version 2107. Microsoft, 2021-08-23 (cit. on p. 32).

[396] L. Wörteler, M. Grossniklaus, C. Grün, and M. Scholl. "Function inlining in XQuery 3.0 optimization". In: *Proc. 15th DBLP*. Pittsburgh, PA, USA: ACM, 2015, pp. 45–48. DOI: 10.1145/2815072.2815079 (cit. on p. 72).

[397] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli. "Pay Less Attention with Lightweight and Dynamic Convolutions". In: *7th International Conference on Learning Representations*. New Orleans LA USA: OpenReview.net, 2019. URL: https://openreview.net/forum?id=SkVhlh09tX (visited on 2021-09-10) (cit. on p. 97).

[398] S. Yamazaki, F. Furukori, Q. Zhao, K. Shirai, and M. Okamoto. "Embedding a Mathematical OCR Module into OCRopus". In: *2011 International Conference on Document Analysis and Recognition, ICDAR 2011, Beijing, China, September 18-21, 2011*. IEEE Computer Society, 2011, pp. 880–884. DOI: 10/b8khqf (cit. on p. 36).

[399] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. 2019, pp. 5754–5764. URL: https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html (visited on 2021-09-05) (cit. on p. 63).

[400] M. Yasunaga and J. D. Lafferty. "TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Honolulu, Hawaii, USA: AAAI Press, 2019-07-17, pp. 7394–7401. DOI: 10/ghkj9w (cit. on pp. 6, 11, 21, 26, 37, 52, 53, 59, 62, 70, 89).

[401] 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: *Designing Interactive Systems Conference 2021*. Virtual Event USA: ACM, 2021-06-28, pp. 1554–1565. DOI: 10/gn3sw6 (cit. on pp. 140, 157).

[402] A. Youssef. "Part-of-Math Tagging and Applications". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 10383. Cham: Springer International Publishing, 2017, pp. 356–374. DOI: 10/ggv8nf (cit. on pp. 2, 6, 19, 25–27, 29, 36, 42, 49, 53, 54, 58, 61, 69, 90, 91, 104, 108, 116, 128, 149, 151, 153, 168, 169).

[403] A. Youssef and B. R. Miller. "A Contextual and Labeled Math-Dataset Derived from NIST's DLMF". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 12236. Cham: Springer International Publishing, 2020, pp. 324–330. DOI: 10/gn3sv6 (cit. on pp. 2, 28, 29, 102, 104, 105, 110, 115, 117, 134).

[404] A. Youssef and B. R. Miller. "Explorations into the Use of Word Embedding in Math Search and Math Semantics". In: *Proc. Conf. Intelligent Computer Mathematics (CICM)*. Vol. 11617. Cham: Springer International Publishing, 2019, pp. 291–305. DOI: 10/gn3szg (cit. on pp. 11, 21, 26, 37, 53, 59, 146).

[405] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila. "NTCIR-12 MathIR Task Overview". In: *Proc. Conf. Evaluation of Information Access Technologies (NTCIR-12)*. Tokyo, Japan: National Institute of Informatics (NII), 2016. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/OVERVIEW/01-NTCIR12-OV-MathIR-ZanibbiR.pdf (visited on 2021-08-19) (cit. on pp. 52, 63, 70, 101).

[406] R. Zanibbi and D. Blostein. "Recognition and Retrieval of Mathematical Expressions". In: *International Journal on Document Analysis and Recognition (IJDAR)* 15.4 (2012-12), pp. 331–357. ISSN: 1433-2833, 1433-2825. DOI: 10/c7qf6r (cit. on pp. 18–22, 26, 36).

[407] R. Zanibbi, K. Davila, A. Kane, and F. W. Tompa. "Multi-Stage Math Formula Search: Using Appearance-Based Similarity Metrics at Scale". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Pisa Italy: ACM, 2016-07-07, pp. 145–154. DOI: 10/ggv8nj (cit. on pp. 72, 75, 89).

[408] R. Zanibbi, D. W. Oard, A. Agarwal, and B. Mansouri. "Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math". In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22-25, 2020, Proceedings*. Vol. 12260. Springer, 2020, pp. 169–193. DOI: 10/gn3sxm (cit. on pp. 52, 116).

[409] D. Zhang, L. Wang, L. Zhang, B. T. Dai, and H. T. Shen. "The Gap of Semantic Parsing: A Survey on Automatic Math Word Problem Solvers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.9 (2020-09-01), pp. 2287–2305. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10/gn3sxs (cit. on p. 53).

[410] X. Zhang, J. J. Zhao, and Y. LeCun. "Character-Level Convolutional Networks for Text Classification". In: *Proc. Ann. Conf. Neural Information Processing Systems (NeurIPS)*. Vol. 28. Montreal, Quebec, Canada, 2015, pp. 649–657. URL: http://arxiv.org/abs/1509.01626 (visited on 2021-09-20) (cit. on p. 96).

[411] D. Zhelezniakov, V. Zaytsev, and O. Radyvonenko. "Online Handwritten Mathematical Expression Recognition and Applications: A Survey". In: *IEEE Access* 9 (2021), pp. 38352–38373. ISSN: 2169-3536. DOI: 10/gn3sxv (cit. on pp. 19, 20, 26, 36).

[412] J. Zhou and W. Xu. "End-to-End Learning of Semantic Role Labeling Using Recurrent Neural Networks". In: *Proc. Ann. Meeting Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2015, pp. 1127–1137. DOI: 10/ggv89w (cit. on p. 59).

[413] K. Zotos. “Computer Algebra Systems – New Strategies and Techniques”. In: *Applied Mathematics and Computation* 198.1 (2008-04), pp. 123–127. ISSN: 0096-3003. DOI: 10/ft93rj (cit. on pp. 19, 32).

[414] E. Zulkoski, C. Bright, A. Heinle, I. Kotsireas, K. Czarnecki, and V. Ganesh. “Combining SAT Solvers with Computer Algebra Systems to Verify Combinatorial Conjectures”. In: *Journal of Automated Reasoning* 58.3 (2017-03), pp. 313–339. ISSN: 1573-0670. DOI: 10/gnd3vg (cit. on pp. 6, 32).

[415] E. Zulkoski, V. Ganesh, and K. Czarnecki. “MATHCHECK: A Math Assistant via a Combination of Computer Algebra Systems and SAT Solvers”. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI/AAAI Press, 2016, pp. 4228–4233. URL: http://www.ijcai.org/Abstract/16/636 (visited on 2021-08-19) (cit. on p. 6).