MioFFAn: an Annotation Software for Formula Formalization with LLM Automation Capabilities

Nicolas Sibuet[*1], **Horacio Saggion**[2], **Riccardo Rossi**[1,3]
[1]Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
[2]Universitat Pompeu Fabra (UPF), Barcelona, Spain
[3]International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain.
{nicolas.sibuet, riccardo.rossi}@upc.edu, horacio.saggion@upf.edu

Abstract

The automatic translation of mathematical expressions in scientific literature into executable symbolic code—a process we refer to as Formula Formalization—is hindered by a severe scarcity of high-quality, ground-truth datasets specialized for technical scientific domains. In this paper, we present MioFFAn, an open-source, document-centric, and customizable framework designed to facilitate rapid annotation for this task. Building upon the MioGatto architecture, we extend existing features to overcome structural limitations and pivot its scope by introducing specific functionalities for Formula Formalization, such as selection of equations of interest and aided symbolic code specification. By allowing users to configure custom taxonomies and properties for identified symbols, and compatible symbolic operators, we ensure the framework is adaptable to diverse specialized scientific fields. Furthermore, MioFFAn is designed to incorporate partial automation via Large Language Models. By defining a modular set of automated sub-tasks with strict output formats, we enable researchers to iteratively refine automation capabilities and evaluate competing strategies using standard NLP metrics. We specify the current automation methodology and perform a preliminary evaluation that demonstrates to efficacy of this human-in-the-loop approach.

Keywords: Symbolic Code Generation, Scientific Document Analysis, Dataset Curation

1. Introduction

Scientific literature serves as the primary vehicle for communicating complex discoveries across all disciplines. For many fields, these discoveries are expressed through mathematical formulations that are to be implemented computationally. This implementation process requires both precise semantic interpretation of mathematical notation and the technical ability to manipulate these elements within a programming language. While modern Optical Character Recognition (OCR) systems effectively extract presentational mathematical formats—such as LaTeX or MathML—from PDF documents, these lack the semantic and computational depth required for the extracted mathematical formulas to be implemented in code. As noted by Greiner-Petter (2023), converting these representations into unambiguous, computationally complete models would require systems capable of synthesizing relevant contextual information, such as explicit mentions of variables or operators and their case-specific properties. Although Large Language Models (LLMs) offer a promising mechanism for orchestrating this transformation, their progress is hindered by the scarcity of high-quality, case-specific datasets that can serve as benchmark or training corpora.

In this work, we present MioFFAn (Math identifier-oriented Formula Formalization Annotator)[1], an annotation tool designed for the fine-grained labeling of mathematical expressions within scientific documents to establish the ground-truth symbolic code required to model them in a Computer Algebra System (CAS). We hereafter refer to this task as Formula Formalization. We build upon the MioGatto framework (Asakura et al., 2021), pivoting from its original scope to introduce features tailored to our requirements. In particular, the original MioGatto software was designed for description alignment (Alexeeva et al., 2020) and coreference analysis of mathematical entities, focusing on the evolution of local notations throughout a document. Therefore, it addresses individual symbols rather than the way they are combined within formulas. In contrast, MioFFAn encompasses symbol characterization and grounding but extends to the generation of symbolic code that orchestrates these elements.

Our software provides domain-specific flexibility, allowing users to customize annotation parameters according to the unique constraints of different scientific fields, such as physics, economics and biological modeling, among others. The system is document-centric and interactive, enabling users to maintain full context by annotating directly onto the manuscript. Finally, the framework is designed for semi-automation in a modular way: by specifying certain tasks to be done automatically and providing corresponding interfaces and evaluation routines, developers may implement automation strategies in any way they see fit. While we incorpo-

[1]MioFFAn is open-source and available at https://github.com/NicolasSR/MioFFAn

Proceedings of the 3rd Int. Workshop on Natural Scientific Language Processing (NSLP 2026) @ LREC 2026, pages 206–217
12 May 2026.

rate baseline automation via LLM in this work, the system is designed to facilitate more sophisticated automated workflows in the future.

To demonstrate the applicability of our framework in realistic research and development scenarios, we present a use case centered on Finite Element Methods (FEM) (Zienkiewicz et al., 2024), cornerstone of numerical simulation. We illustrate how MioFFAn can be configured to characterize this domain's variable properties and operators and we use this use case within a proof-of-concept evaluation of current automation capabilities.

The remainder of this paper is organized as follows: Section 2 reviews related work in terms of similar NLP tasks, existing datasets and prior annotation tools. Section 3 thoroughly describes the MioFFAn framework, including a small review of the original MioGatto tool. Then, Section 4 describes the use case to which MioFFAn has been configured during this study. Following, Section 5 details the proof-of-concept evaluation for the current automation framework. Section 6 discusses possible integration of MioFFAn into other pipelines and Section 7 specifies the most important limitations of the software. Finally, Section 8 concludes the document and proposes future work directions.

2. Related work

2.1. Related NLP tasks

We review the tasks that relate the most to Formula Formalization:

Neural machine translation (NMT) for mathematics. The translation of mathematical notation from presentational formats (e.g., LaTeX) to computational formats can be framed as a particular NMT (Kalchbrenner and Blunsom, 2013; Tan et al., 2020) task. Petersen et al. (2023) use this terminology and demonstrate that end-to-end NMT can outperform rule-based systems in LaTeX-to-Mathematica translation; however, they highlight a significant bottleneck for further generalization caused by notation bias, lack of contextual integration and insufficient hand-crafted data. Prior rule-based pipeline LaCaST (Greiner-Petter et al., 2023) leverages external convention databases and entity linking to address ambiguity, though it lacks the flexibility of deep learning architectures.

Tasks on mathematical grounding and tagging. Relating specific mathematical identifiers with their in-document descriptions is attempted via LLM in (Dev et al., 2024) but also before, via rule-based systems (Schubotz et al., 2016; Alexeeva et al., 2020). Mathematical Entity Linking (Kristianto et al., 2016) strives to associate mathematical symbols to knowledge base entries corresponding to the concept they represent. Finally, Part-of-Math (PoM) Tagging (Youssef, 2017) classifies mathematical symbols according to the type of concept that they represent, according to a specific taxonomy. Zou et al. (2025) and Shan and Youssef (2024) approach the task by using LLMs and determine that further work is needed to recognize complex math concepts and that explicit human expert evaluation is needed for proper assessment.

Autoformalization. The Autoformalization task (Weng et al., 2025) refers to the translation of natural language mathematics into verifiable formal languages (e.g., Lean, Isabelle). Zhang et al. (2025) conclude that including well-defined contextual information and syntactic correction capabilities are determinant to achieve good results. While the task is generally focused on formal proofs, recent expansions have applied these techniques to physics lemmas and theorems (Soroco et al., 2025; Kabra et al., 2025).

Word math problem solving. Finally, research into solving word math problems has shown that offloading reasoning to symbolic solvers or CAS significantly improves accuracy (He-Yueya et al., 2023; Chen et al., 2023). These systems generate intermediate symbolic programs—comprising characterized variables and equations—that align closely with our proposed Formula Formalization output, despite differing in input modality and end goals.

2.2. Related Datasets

Ground-truth data related to Formula Formalization is notably scarce. MathMLben (Schubotz et al., 2018) provides ~300 LaTeX expressions in tree representation with Wikidata-linked variables, while MathAlign (Alexeeva et al., 2020) links identifiers to textual descriptions in arXiv snippets. However, these resources often omit complex expressions and lack the domain specificity required for real-world applications. Recent benchmark STEM-PoM (Zou et al., 2025) focuses exclusively on the PoM Tagging task. In contrast, Autoformalization benchmarks (e.g., MiniF2F (Zheng et al., 2022), LeanEuclid (Murphy et al., 2024)) are highly structured but strictly limited to proof assistants, failing to support the general-purpose CAS integration or numerical simulations required for engineering contexts. Schembera et al. (2025) present a graph knowledge base for applied mathematics and algorithms. It incorporates mathematical models and formulations for several scientific fields, indicating the nature of the quantities included in each formulation, but does not orchestrate corresponding symbolic codes or semantic trees. The knowledge base is collaborative, but limited by time-consuming manual input.

2.3. Mathematical Annotation

Existing frameworks for mathematical annotation vary significantly in input modality and granular focus. Early annotation tools compatible with mathematical notation, like KAT (Schmoll and Wiesing, 2016) and AnnoMathTeX (Scharpf et al., 2019) leverage HTML5 and LaTeX, respectively. The latter incorporates recommendation systems from Wikipedia and arXiv. For PDF-based workflows, Alexeeva et al. (2020) provide a specialized identifier annotator. MioGatto (Asakura et al., 2021) and AnnoTize (Panzer and Schaefer, 2023) both utilize MathML in HTML environments; while MioGatto emphasizes rapid annotation through mathematical identifier recognition and allows for contextual coreference resolution via text highlighting, AnnoTize prioritizes extensive user personalization for annotated object properties and marking style. VARAT (Kato and Kano, 2025) initiates a movement toward domain-specific workflows. It introduces industry-specific annotation for chemical manufacturing, although the annotation content is limited to textual definition and grounding on whole paragraphs. Unfortunately, none of these tools implement specific features for annotating the symbolic code representation of whole mathematical expressions.

The MathMLben suite (Schubotz et al., 2018) focuses on graph representations of complete equations and the STEM-PoM Labeler (Zou et al., 2025) focuses on taxonomic symbol classification. However, these frameworks process isolated equations, losing the document-level context essential for disambiguation.

Critically, while MioGatto and the STEM-PoM Labeler have been used in research involving LLM-assisted automation (Zou et al., 2025; Dev et al., 2024), the implementation of the frameworks used for LLM interaction are not publicly available.

3. The MioFFAn Framework

Designed as a document-centric suite utilizing HTML and MathML, the framework facilitates a multi-stage annotation pipeline that incrementally reduces the complexity of Formula Formalization. The main steps in this pipeline are:

1. Formula Selection: The user identifies a target mathematical expression within the manuscript.
2. Identifier Characterization: The user defines mathematical properties and attributes for the specific identifiers within that expression.
3. Contextual Grounding: These specifications are grounded by aligning identifiers with relevant context segments through document-level highlighting.
4. Symbolic Synthesis: The user writes the symbolic code for the formula, leveraging the previously grounded concepts and CAS operators.

MioFFAn utilizes a client-server model (TypeScript/Python) and operates locally via web browser. The choice of HTML+MathML over LaTeX, which is the preferred markup language for writing scientific corpora, is that they are specifically designed to be integrated into dynamic web applications, facilitating selection of elements and transformations of these via well-established web-app development suites. Nonetheless, conversion from LaTeX files to HTML and vice-versa may be added to the software at a later stage. Furthermore, there is increasing interest from scientific paper repositories such as arXiv towards using HTML as a more accessible visualization format (arXiv, n.d.).

A schematic of MioFFAn's architecture can be viewed in Figure 1, while a global view of the software's interface can be observed in Figure 2. The content of the sample shown in all the figures that contain MioFFAn's interface is sourced from (Hashemi et al., 2021).

As the framework builds upon the MioGatto architecture (Asakura et al., 2021), we first provide a preliminary analysis of the base software's characteristics before detailing our specific extensions and architectural shifts.

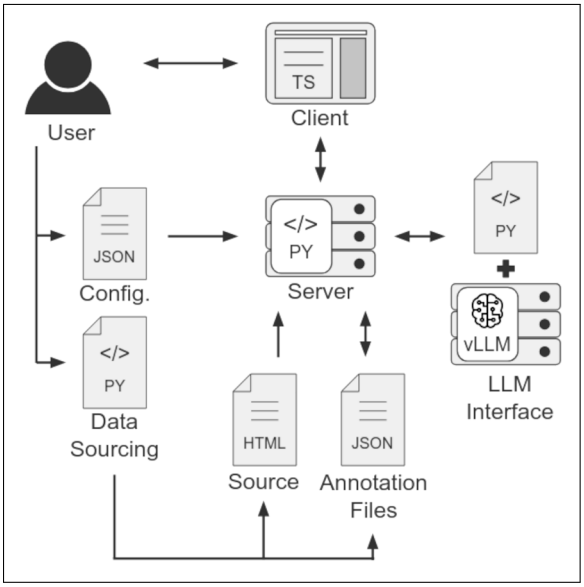

Figure 1: Architecture schematic of MioFFAn

3.1. Description of the Original MioGatto Software

We selected MioGatto as our foundational framework due to its MathML-native architecture, its document-centric approach, its interactivity and its

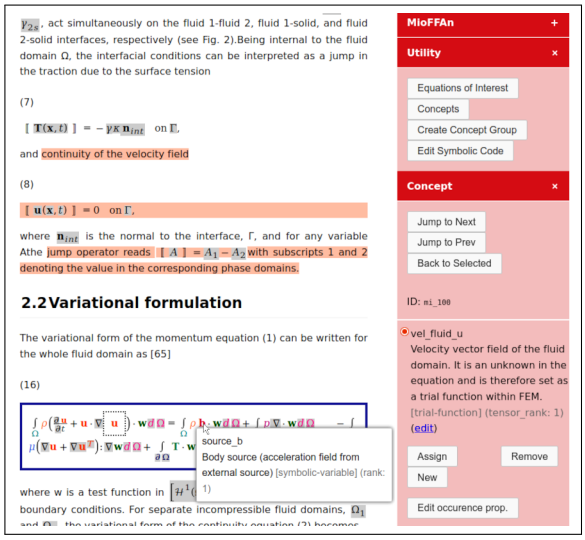

Figure 2: MioFFAn's interface with annotations. The EoI is surrounded by a blue frame. Colored symbols indicate Occurrences. SoGs are highlighted (orange) for the selected Occurrence ($\mathbf{u}$). Information is shown for the Occurrence under the mouse pointer ($\mathbf{b}$). Edition toolbar on the right, in red. The sample content is sourced from (Hashemi et al., 2021), as is in all subsequent figures.

approach to fast identifier selection. Most importantly, its features align with the identifier characterization and contextual grounding steps from our proposed pipeline at the beginning of Section 3.

The system implements three core functionalities:

- Math Identifier (MI) Management: Automated identification of <mi> MathML tags, enabling rapid selection via mouse or keyboard.
- Concept Assignment: Annotation of a given MI is done by assigning a Concept (description, arity, and notation affixes) to it. An MI associated with a Concept is termed an Occurrence[2]; all Occurrences of the same Concept share a distinct color for visual consistency and distinguishability.
- Contextual Grounding: Highlighting of text segments to serve as Sources of Grounding (SoG), linking descriptions or relevant information directly to specific Occurrences.

Despite its strengths, MioGatto presents several bottlenecks for its effective use in realistic annotation dataset creation. Architecturally, the system is constrained by a one-sample-per-session limit, requiring a server restart to switch documents. Datawise, its annotation schema pre-allocates entries for every MI regardless of activity. Functionally, we identify four critical limitations: (1) Annotations are strictly limited to single <mi> tags (generally single characters), preventing the grouping of complex symbols; (2) Concept properties are fixed and mix intrinsic characteristics (arity) with notation-specific ones (affixes); (3) SoG highlighting is restricted to individual nodes, preventing cross-node or in-line math selection; and (4) Concepts are identified by the Unicode characters they contain, making the underlying annotation structure cumbersome to manipulate.

[2]Throughout this paper, capitalized *Concept* and *Occurrence* refer specifically to these architectural entities.

3.2. Enhancement of Original Features

We implemented architectural enhancements that resolve the limitations identified in Section 3.1.

To allow for **hierarchical and compound MI management**: MioFFAn extends the definition of a Math Identifier beyond single <mi> tags. By default, the system now natively supports compound MathML structures, including sub-scripts (<msub>), super-scripts (<msup>), and their combinations (<msubsup>). The user may add more tags as they see fit via JSON configuration. All MIs are uniquely identified by tag IDs.

To capture complex notations like δu_i or $f(x)$ as single entities, we introduce Groups: user-defined MIs encompassing multiple contiguous sibling elements within the MathML tree. These are implemented via a temporary <mstyle class="custom-group"> wrapper in the server-side DOM, preserving the integrity of the original source file. They are permanently and uniquely identified in the annotation files by the first and last <mi> tags within their combined sub-trees in Depth-First Search (DFS) traversal order. The declaration process for Groups is intuitive, needing only the selection by mouse of start and end characters to include. An example of the process can be visualized in Figure 3. This way, the user does not need to deal with the explicit HTML code.

This architecture implicitly enables hierarchical annotation, where a Concept can be assigned to a holistic expression (e.g., a function $f(x)$) while its constituent symbols (e.g., x) retain their own specific Concept assignments. To navigate overlapping tags, users employ assigned keyboard keys to perform a DFS traversal of the MI tree, ensuring precise selection regardless of visual density.

To **decouple the property schema**: we allow both Concepts and Occurrences to have their own independent properties. This way, we differentiate:

- Invariant Semantic Properties (Concept): Defined once for a mathematical entity (e.g., a "velocity vector" (**u**) is always a vector quantity).

- Variant Properties (Occurrence): Specific to a single instance (e.g., whether the vector is currently transposed ($\mathbf{u}^T$)).

This disentanglement allows the same Concept to be reused across different implementation modalities and notations, instead of needing to create slightly different Concepts for each scenario. Users can customize these property sets via JSON configuration, enabling the tool to adapt to the specific requirements of different scientific disciplines. A view of the interface for defining new Concepts in MioFFAn can be seen in Figure 4.

To **improve SoG flexibility**: SoGs now support inter-span highlighting, allowing users to select text across multiple HTML nodes, including in-line mathematical expressions. It also allows for the highlighting of entire displayed formulas. SoGs are now directly associated to Concepts, instead of individual Occurrences.

Additionally, we allow the annotation schema to be dynamically populated as needed and the dependency to Unicode is replaced by unique IDs for Concepts. Finally, users can now navigate between document samples within a single session without requiring a server restart, facilitating large-scale dataset curation.

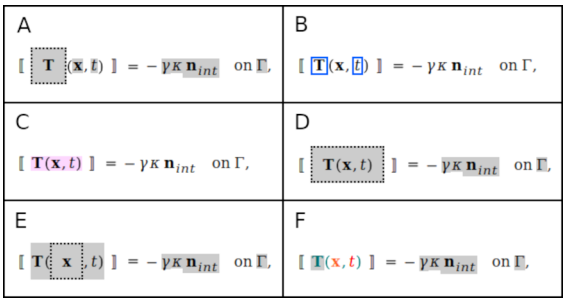

Figure 3: Group designation process and example. Sequentially: (A) Annotation options (shadowed symbols) prior to grouping. (B) Within Group creation tool: first and last Group elements selected by mouse click. (C) Within Group creation tool: complete group visualization. (D) Annotation options after grouping. (E) Demonstration of inner MI selection within Group. (F) Example view of hierarchically assigned concepts.

3.3. Formula Formalization-Specific Features

We introduce three core features that are vital to the Formula Formalization workflow.

First, we introduce **Equations of Interest (EoI)**. Rather than treating all document content equally, users designate specific EoIs, susceptible of formalization. This visually marks the expression and initializes a dedicated entry for that formula in the annotation schema. Each EoI is indexed via the HTML ID of its parent tag. The view of a marked EoI within the document can be seen within Figures 2 and 4.

Second, we introduce **canonical Variable Names**. To ensure that defined Concepts can be uniquely represented within the annotated symbolic code, the framework requires each Concept to be mapped to a Variable Name.

Third, we introduce **integrated symbolic code construction**. The final stage of the pipeline is the synthesis of the symbolic code representation. To minimize manual syntax errors and accelerate the translation process, MioFFAn provides an interactive construction environment with a palette containing the Variable Names for all Concepts defined in previous steps and a list of available operators specific to the target CAS. When a user selects an element from these lists, it is automatically injected into the text box for the code. However, these aids do not stop the user from defining their own intermediate variables and logic manually in order to maintain readability or handle complex local logic. A view of this functionality can be seen in Figure 5.

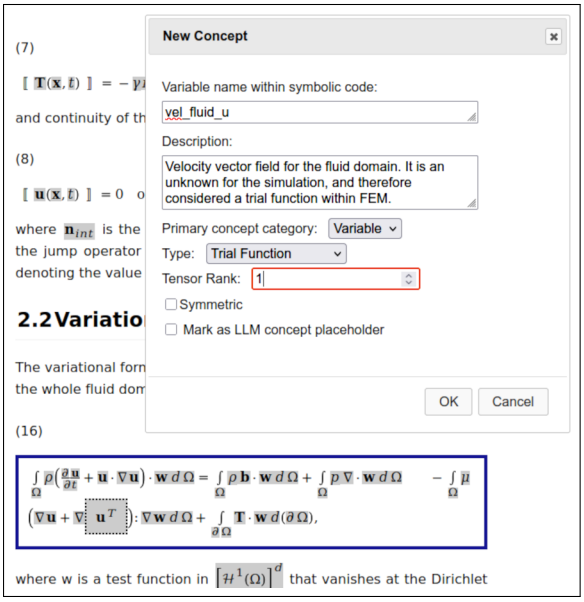

Figure 4: Menu for the registration of a new concept for the selected MI ($\mathbf{u}^T$). Fields according to taxonomy in Section 4. EoI visibly marked by a blue frame.

3.4. Data Sourcing, Management and Preprocessing

To ensure high-fidelity structural representation, we implement a preprocessing routine tailored for the ScienceDirect API[3], as it supports the retrieval of

[3] https://dev.elsevier.com/

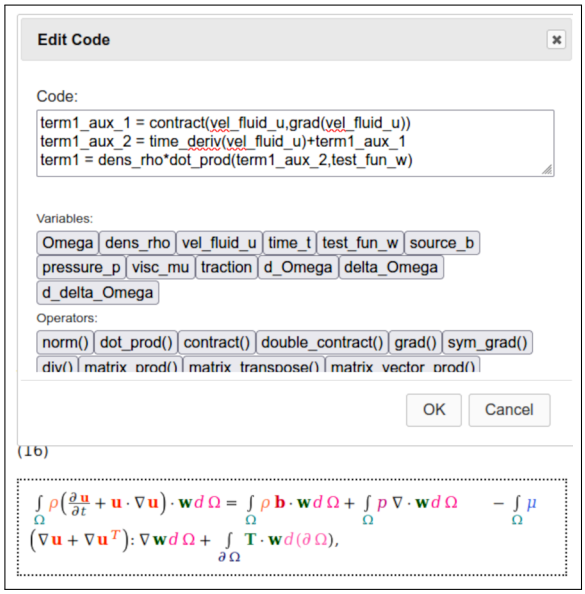

Figure 5: Menu for the specification of the symbolic code for an EoI. Shows palettes for both variables and operators.

research papers in XML format, bypassing the need for non-deterministic OCR and instead performing a direct transformation into MathML-compliant HTML.

During this transformation, we perform a deterministic ID injection phase:

- Mathematical Entities: All <mi> tags, designated compound MI tags and containers are assigned unique persistent identifiers if they are not already present in the source XML.
- Textual Context: To facilitate precise grounding, the text is segmented into <p> nodes and nodes within them, each receiving a unique ID.

This system allows the annotation schema to reference specific tags in the document. Said annotation system is based on two different JSON files: *anno.json*, used for Groups and *mcdict.json*, used for Concepts, EoIs and Occurrences. Both of them are generated during the preprocessing step.

3.5. Partial Automation via LLM

MioFFAn integrates infrastructure for LLM-assisted annotation. Rather than an end-to-end approach, we implement a modular, stage-wise pipeline that allows for human-in-the-loop verification and refinement. The automation logic is implemented in Python and is decoupled from the core application, enabling researchers to swap or evaluate different strategies without modifying the software's primary interface and functionalities.

In terms of interface, we utilize the OpenAI-compatible vLLM API to interact with local Large Language Models. This choice ensures data privacy—a critical requirement in scientific research.

The framework currently supports three automation tasks designed to populate the annotation schema:

- **Symbol Segmentation:** The model identifies which MathML elements or compound structures within an EoI should be grouped as unique Occurrences. For example, in $\delta u F(x)$, the system suggests segments for δu and $F(x)$. These are assigned placeholder Concepts to allow for easy visual verification and correction. The software enables the user to assign placeholder Concepts manually as well, in order to annotate ground-truth samples for this specific task, and in order to correct the automated results before continuing to the next steps. These placeholders do not require any description or Concept property specification.
- **Concept Assignment:** The system suggests Concepts, populated with at least their canonical Variable Names and a textual description. Within this same stage, the model maps these Concepts to existing placeholder Concepts.
- **SoG Identification:** The model identifies text spans or formulas in the HTML tree that serve as the SoG for each Concept. By providing the start and end node IDs, the system automatically generates document-level highlights.

By providing the HTML tree and the content within the annotation files as input for the routines and enforcing a fixed output format at each stage (lists of Groups, Concepts, Occurrences or SoGs to be added), MioFFAn enables methodology-agnostic evaluation. Predicted annotation outputs can be directly compared against gold-standard annotation files, and the annotation files can be processed together with the source HTML file to apply standard NLP metrics. The fact that MioFFAn annotations are based on pointers towards content included in the original HTML file, and not the textual content itself, makes it less susceptible to format-related bias during evaluation. We acknowledge that this limits the scores for systems that generate correct natural language content but are not able to locate it correctly within the original document. However, we believe this relation capability to be an integral necessity for the transparency of these systems towards the human annotator and therefore we consider this effect to be fair. In any case, the user may implement intermediate LLM logs within their automation framework, together with custom evaluation metrics, to directly evaluate natural language responses against the gold standard, bypassing this constraint.

3.5.1. Provisional Automation Methodology

The automation code and LLM prompts used for this paper are available within the examples directory of MioFFAn's GitHub repository. Here we describe the process followed for each task. For the **symbol segmentation**:

1. The LLM is tasked with recreating the MathML sub-structures for candidate symbols.
2. A Python routine using the lxml library performs a search within the source MathML tree to validate these candidates. Proposed structures that lack an exact match in the original source are discarded.
3. If overlapping structures are detected (e.g., x nested within $F(x)$), a second LLM call provides the conflict context and solves it, either removing the inner structure instance or maintaining it as an independent symbol.

For the **concepts assignment**:

1. The LLM proposes Concepts relevant to the EoI. For each one, it defines a unique Variable Name, a textual description, and a Justification for its choice (appended to the description).
2. The LLM performs a mapping from these new Concepts to the available segmented symbols. Concepts that fail to find a specific symbol occurrence are pruned.
3. For every Concept individually, the LLM determines appropriate domain-specific properties.

For the **SoG identification**:

1. For each Concept, the LLM identifies the IDs of specific , <p> or tags that serve as textual evidence for the Concept's meaning and properties.
2. The framework verifies that the returned IDs exist within the document before finalizing the SoG highlights in the annotation file.

4. Use Case

We configure the framework for its use in Finite Element Methods (FEM), a domain characterized by high-dimensional tensors and variational formulations that demand precise characterization. Specifically, we target a custom Sympy-based symbolic compiler[4] designed to translate certain variational forms of partial differential equations into simulation-ready C++ code for the KratosMultiphysics (Dadvand et al., 2010) framework.

[4] https://github.com/NicolasSR/KratosFECompiler

The configuration encompasses, on one side, the **domain taxonomy and properties schema**: We define a tiered schema (Table 1) that establishes primary categories (variables, functions, operators, domains and integration variables) and determines, for each one, invariant Concept properties and instance-specific Occurrence attributes. These can be categorical (multiple choice) or single-valued (boolean, numerical, textual). On the other side, it encompasses the **compiler grammar**: We define a symbolic grammar in terms of a list of operators such as `grad()` (gradient), `div()` (divergence) or `contract()` (tensor contraction), to be utilized from the Sympy-based compiler. These will be available within the MioFFAn symbolic code synthesis palette.

This characterization is the one applied within all figures of this document, and marks the setting for the evaluations in the next section. The full files containing the complete taxonomy and operators list can be found within the examples directory of the MioFFAn repository.

5. Automation Evaluation

We conduct a preliminary evaluation to assess the efficacy of the MioFFAn framework and its provisional LLM routines on a curated set of variational formulations from FEM literature.

5.1. Setup and Dataset

The framework was setup to interface a local vLLM server hosting a Qwen3-4B-Instruct-2507 (Yang et al., 2025) model with 65000 tokens of context length. This particular LLM model was chosen for its reported good performance in scientific content and mathematics, while being able to run on a consumer-grade GPU (RTX 4090 in our setup). A comparison against other models in the same parameters count range is shown in Appendix A.

The evaluation set consists of 6 samples sourced via the ScienceDirect API. The candidate samples collection was done in a greedy way by searching for keywords such as "finite element" or "variational" and keeping only those works published under a Creative Commons license. These were further filtered manually to ensure that they incorporate mathematical formulation aligned with the use case. Furthermore, we did a selective pruning of the original XML files, in order to remove irrelevant content and keep the HTML (in string format) within 20000 tokens long. We define the gold standard data through expert manual annotation via MioFFAn. We provide the data and processing scripts for this study within the examples directory of the MioFFAn repository.

Category	Level	Categorical Properties	Single-Value Properties
Variable	Concept	**Type**: [Trial, Test, Nodal, Symbolic, Numerical]	Tensor Rank, Symmetry
	Occurrence	—	Transpose, Voigt
Function	Concept	**Type**: [Defined, Undefined]	Tensor Rank, Symmetry, Dependencies
	Occurrence	—	Transpose, Voigt
Operator	Concept	**Type**: [Differential, Tensorial, Algebraic]	Linearity
Domain	Concept	**Type**: [Whole, Boundary]	Dimension
Integration var.	Occurrence	—	Domain Name Variable

Table 1: Taxonomy of mathematical concepts and occurrence-specific properties for the FEM use case.

	Symbol Segm.		Concepts Assign.	SoGs Ident.
	Coverage (%)	CoNLL (%)	CoNLL (%)	Avg. SDI (%)
Explicit Names	36.2	26.9	97.0	15.3
Original Characters	57.7	36.8	96.9	14.9

Table 2: Average results for each evaluation metric, comparing the unmodified automation workflow (Original Characters) with a modified workflow that substitutes non-standard Unicode characters by their explicit names (Explicit Names).

5.2. Evaluation Methodology

We evaluate the performance of the automation routines by comparing LLM-generated annotation files against their counterparts for the ground truth. To ensure valid comparisons, we setup the automated framework to perform the Concept Assignment and SoG Identification stages parting from the ground-truth outputs of the Symbol Segmentation stage; this ensures a one-to-one mapping of symbols and prevents error propagation from initial segmentation.

The following metrics are utilized within each automation phase:

- Symbol Segmentation. We assess Coverage, defined as the ratio of <mi> tags correctly identified relative to the gold standard. We define a tag as correctly identified if it is assigned to an Occurrence within both the predicted and ground-truth sets, regardless of the specific placeholder Concept. This does not account for the tags that are being identified in the automated results but not in the ground truth, however we consider these cases to be a minority (most symbols should be identified in the ground truth). To evaluate the accuracy of symbol clustering (distribution of <mi> tags among placeholder Concepts), we utilize the standard coreference metric: CoNLL score (average of MUC, B3, and CEAFe).
- Concept Assignment. The CoNLL score is reapplied here to determine if the segmented symbols are mapped to the correct final Concepts.
- SoG Identification. To measure the quality of the Sources of Grounding (SoG), we treat the identified text and formula IDs as sets and employ the Sørensen–Dice (SDI) index. This index measures the overlap between the LLM-predicted highlights and the expert ground truth for each Occurrence. The final score for a given sample is the average of Sørensen–Dice indexes from all Occurrences.

5.3. Results and Discussion

In this sub-section, we present a proof-of-concept evaluation of different automation methodologies. Specifically, we compare two slightly different variants of our task automation implementations:

- Original Characters (original): No modification is done to the HTML content given to the LLM. So non-standard Unicode characters will stay as they are, e.g "λ".
- Explicit Names (modification): The HTML given to the LLM is processed in order to replace all non-standard Unicode characters by the explicit names given by Python's unicodedata.name() function, e.g. "GREEK SMALL LETTER LAMBDA".

For each approach and metric, the scores for all samples are averaged. These results are shown in Table 2, where we see that the Original Characters variant yields superior performance in the Symbol Segmentation task, with an advantage of over 20% in terms of coverage and 10% in terms of clustering accuracy. Meanwhile, the Explicit Names approach

provides a marginal advantage in both Concept Assignment and SoGs Identification tasks. Such results may guide the developer towards choosing one strategy for the first task and another one for the two latter tasks. Therefore, proving the potential of the MioFFAn tool as an automation development playground.

Apart from this proof-of-concept comparison, we can take the Original Characters results as the current performance of the automation system. In this sense, the system is far from being ideal in the Symbol Segmentation task and, mostly, in the SoGs Identification task. Qualitative analysis for the latter one suggests that the model often identifies correct semantic context but struggles with the precise specification of tag IDs, sometimes highlighting entire paragraphs instead of a small sentence. These results reinforce our decision to prioritize a human-in-the-loop interface, as the automated annotations should serve only as a baseline that can be rapidly refined by a human expert, rather than a final result.

6. MioFFAn in External Pipelines

We briefly discuss how MioFFAn could fit within other tools and pipelines in the field of mathematical formula processing and annotation. In the context of OCR and document mining for scientific content, standard pipelines such as Mathpix[5] and Nougat (Blecher et al., 2023) focus on high-fidelity visual transcription (LaTeX or MathML) but typically lack semantic depth. MioFFAn could serve as a semantic post-processor for these outputs by appending explicit Concepts and symbolic codes to the formulas as textual metadata. Furthermore, within the Mathematical Information Retrieval field, reviewed by Malik et al. (2026), MioFFAn could enhance both text-based and tree-based formula retrieval systems. One could use annotated symbolic code as a precursor for generating semantic trees, or incorporate Concepts and SoGs as additional textual information to improve formula searchability and context-aware similarity comparisons. Finally, MioFFAn may be used as an input tool for knowledge bases such as the ones presented in (Schembera et al., 2025) with little modification efforts, as MioFFAn's data structure aligns closely with their framework: mathematical formulations correspond to EoIs, while quantities and their metadata map to Concepts and Concept properties.

7. Limitations

Arguably, the biggest limitation of the MioFFAn software at current time is the sourcing system's dependency on the ScienceDirect API. Apart from integrating other API's that directly work with XML/HTML formats, conversion from PDF or LaTeX files to MioFFAn-compatible HTML format should be enabled. Annotating web content would also be challenging because of its dynamic nature. This could be solved by developing an output modality that provides the annotations in an explicit way to be appended to the web content or by embedding annotation information directly onto corresponding MathML nodes. Other limitations include not being able to integrate links to external knowledge bases natively, not checking the applicability of CAS operators on given Concepts automatically and not allowing concurrent, collaborative annotation. All of these functionalities can be added progressively in the future.

8. Conclusions and Future Work

We have presented MioFFAn, a document-centric, interactive, and customizable software for the annotation of symbolic code corresponding to mathematical expressions. This software is proposed as a direct solution to the current scarcity of ground-truth data required for symbolic code generation in real-world research and development fields.

By extending the original MioGatto architecture, MioFFAn addresses critical challenges in structural flexibility through compound and hierarchical MI management, a decoupled properties system, and flexible source grounding. Furthermore, it incorporates the necessary features for the Formula Formalization task, specifically Equation of Interest selection and aided symbolic code specification. To demonstrate MioFFAn's adaptability to specialized scientific domains, we have detailed a use case in Finite Element Method by configuring an appropriate concept taxonomy and operator library.

A major feature of this framework is its modular partial automation pipeline via LLMs. By decomposing parts of the annotation process into well-defined, implementation-agnostic sub-tasks, we enable researchers to iteratively test and evaluate distinct automation strategies. Our preliminary evaluation confirms that even provisional LLM routines provide a significant baseline for informed refinement, validating the effectiveness of our human-in-the-loop design.

Moving forward, we aim to enhance MioFFAn through refining current automation strategies and specifying further automation tasks, including a wider range of APIs for source gathering and support for manual document uploads, and incorporating external knowledge bases to enrich the context of identified Concepts.

[5] https://mathpix.com/

9. Acknowledgements

Nicolas Sibuet acknowledges the Secretariat of Universities and Research of the Department of Research and Universities of the Generalitat of Catalonia, as well as the European Social Plus Fund for their financial support through the pre-doctoral scholarship AGAUR-FI (2024 FI-1 00089) Joan Oró.

10. Bibliographical References

Maria Alexeeva, Rebecca Sharp, Marco A. Valenzuela-Escárcega, Jennifer Kadowaki, Adarsh Pyarelal, and Clayton Morrison. 2020. MathAlign: Linking formula identifiers to their contextual natural language descriptions. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2204–2212. European Language Resources Association.

arXiv. n.d. Accessible HTML - arXiv info. https://info.arxiv.org/about/accessible_HTML.html. Accessed: 2026-03-23.

Takuto Asakura, Yusuke Miyao, Akiko Aizawa, and Michael Kohlhase. 2021. MioGatto: A math identifier-oriented grounding annotation tool. In *Workshop Papers of the 14th Conference on Intelligent Computer Mathematics (CICM 2021)*, volume 3377 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural Optical Understanding for Academic Documents. ArXiv:2308.13418 [cs].

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. ArXiv:2211.12588 [cs].

Pooyan Dadvand, Riccardo Rossi, and Eugenio Oñate. 2010. An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications. *Archives of Computational Methods in Engineering*, 17(3):253–297.

Aamin Dev, Takuto Asakura, and Rune Sætre. 2024. An approach to co-reference resolution and formula grounding for mathematical identifiers using large language models. In *Proceedings of the 2nd Workshop on Mathematical Natural Language Processing @ LREC-COLING 2024*, pages 1–10. ELRA and ICCL.

André Greiner-Petter. 2023. *Making Presentation Math Computable: A Context-Sensitive Approach for Translating LaTeX to Computer Algebra Systems*. Springer Fachmedien, Wiesbaden.

André Greiner-Petter, Moritz Schubotz, Corinna Breitinger, Philipp Scharpf, Akiko Aizawa, and Bela Gipp. 2023. Do the Math: Making Mathematics in Wikipedia Computable. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4384–4395.

Mohammad R. Hashemi, Pavel B. Ryzhakov, and Riccardo Rossi. 2021. Three dimensional modeling of liquid droplet spreading on solid surface: An enriched finite element/level-set approach. *Journal of Computational Physics*, 442:110480.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. Solving Math Word Problems by Combining Language Models With Symbolic Solvers. ArXiv:2304.09102 [cs].

Aditi Kabra, Jonathan Laurent, Sagar Bharadwaj, Ruben Martins, Stefan Mitsch, and André Platzer. 2025. Can Large Language Models Autoformalize Kinematics? ArXiv:2509.21840 [cs].

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Shota Kato and Manabu Kano. 2025. VARAT: Variable Annotation Tool for Documents on Manufacturing Processes. *Journal of Chemical Engineering of Japan*, 58(1):2454461.

Giovanni Yoko Kristianto, Goran Topić, and Akiko Aizawa. 2016. Entity Linking for Mathematical Expressions in Scientific Documents. In *Digital Libraries: Knowledge, Information, and Data in an Open Access Society*, pages 144–149, Cham. Springer International Publishing.

Ayushi Malik, Pankaj Dadure, and Sahinur Rahman Laskar. 2026. A Review of Mathematical Information Retrieval: Bridging Symbolic Representation and Intelligent Retrieval. *Archives of Computational Methods in Engineering*, 33(1):577–611.

Logan Murphy, Kaiyu Yang, Jialiang Sun, Zhaoyu Li, Anima Anandkumar, and Xujie Si. 2024. Autoformalizing Euclidean Geometry. ArXiv:2405.17216 [cs].

NVIDIA, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, et al. 2025. NVIDIA Nemotron 3: Efficient and Open Intelligence. ArXiv:2512.20856 [cs].

Lukas Panzer and Jan Frederik Schaefer. 2023. AnnoTize: A Flexible Annotation Tool for Documents with Mathematical Formulae. In *Proceedings of the 14th MathUI Workshop*, Cambridge, UK.

Felix Petersen, Moritz Schubotz, Andre Greiner-Petter, and Bela Gipp. 2023. Neural machine translation for mathematical formulae. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11534–11550. Association for Computational Linguistics.

Philipp Scharpf, Ian Mackerracher, Moritz Schubotz, Joeran Beel, Corinna Breitinger, and Bela Gipp. 2019. AnnoMathTeX - a formula identifier annotation recommender system for STEM documents. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 532–533, New York, NY, USA. Association for Computing Machinery.

Björn Schembera, Frank Wübbeling, Hendrik Kleikamp, Burkhard Schmidt, Aurela Shehu, Marco Reidelbach, Christine Biedinger, Jochen Fiedler, Thomas Koprucki, Dorothea Iglezakis, and Dominik Göddeke. 2025. Towards a knowledge graph for models and algorithms in applied mathematics. In *Metadata and Semantic Research*, pages 95–109. Springer Nature Switzerland.

Felix Schmoll and Tom Wiesing. 2016. KAT: Enabling the Semantification of STEM Documents. In *Joint Proceedings of the FM4M, MathUI, and ThEdu Workshops, Doctoral Program, and Work in Progress at the Conference on Intelligent Computer Mathematics 2016 (CICM 2016)*, volume 1785 of *CEUR Workshop Proceedings*, pages 66–72, Bialystok, Poland. CEUR-WS.org.

Moritz Schubotz, André Greiner-Petter, Philipp Scharpf, Norman Meuschke, Howard S. Cohl, and Bela Gipp. 2018. 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 '18, pages 233–242. Association for Computing Machinery.

Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S. Cohl, Norman Meuschke, Bela Gipp, Abdou S. Youssef, and Volker Markl. 2016. 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*, SIGIR '16, pages 135–144. Association for Computing Machinery.

Ruocheng Shan and Abdou Youssef. 2024. Using large language models to automate annotation and part-of-math tagging of math equations. In *Intelligent Computer Mathematics*, pages 3–20. Springer Nature Switzerland.

Mauricio Soroco, Jialin Song, Mengzhou Xia, Kye Emond, Weiran Sun, and Wuyang Chen. 2025. PDE-Controller: LLMs for Autoformalization and Reasoning of PDEs. ArXiv:2502.00963 [cs].

Zhixing Tan, Shuo Wang, Zonghan Yang, Gang Chen, Xuancheng Huang, Maosong Sun, and Yang Liu. 2020. Neural machine translation: A review of methods, resources, and tools. *AI Open*, 1:5–21.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, et al. 2025. Gemma 3 Technical Report. ArXiv:2503.19786 [cs].

Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu, and Tiancheng Zhang. 2025. Autoformalization in the Era of Large Language Models: A Survey. ArXiv:2505.23486 [cs].

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. 2025. Qwen3 Technical Report. ArXiv:2505.09388 [cs].

Abdou Youssef. 2017. Part-of-Math Tagging and Applications. In *Intelligent Computer Mathematics*, volume 10383 of *Lecture Notes in Computer Science*, pages 356–374. Springer International Publishing, Cham.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025. Autoformalization in the Wild: Assessing LLMs on Real-World Mathematical Definitions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1738. Association for Computational Linguistics.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics. ArXiv:2109.00110 [cs].

O.C. Zienkiewicz, R.L. Taylor, and S. Govindjee. 2024. *The Finite Element Method: Its Basis and Fundamentals*, 8th edition. Butterworth-Heinemann.

Jiaru Zou, Qing Wang, Pratyush Thakur, and Nickvash Kani. 2025. STEM-POM: Evaluating Language Models Math-Symbol Reasoning in Document Parsing. ArXiv:2411.00387 [cs].

A. LLM models comparison

	Symbol Segm.	
	Coverage (%)	CoNLL (%)
Nemotron	33.9	26.4
Gemma	28.2	18.8
Qwen	57.7	36.8

Table 3: Evaluation scores for the Symbol Segmentation task, comparing three different LLM models. For each model and metric, the results from all samples have been averaged to produce a unique score.

	Concepts Assign.	SoGs Ident.
	CoNLL (%)	Avg. SDI (%)
Nemotron	98.0	15.1
Gemma	95.6	18.3
Qwen	96.9	14.9

Table 4: Evaluation scores for the Concept Assignment and the SoG Identification tasks, comparing three different LLM models. For each model and metric, the results from all samples have been averaged to produce a unique score.

Three LLM models are chosen for comparison in the current implementation of the automation framework: NVIDIA's NVIDIA-Nemotron-3-Nano-4B-BF16 (NVIDIA et al., 2025), Google's gemma-3-4b-it (Team et al., 2025) and Alibaba's Qwen3-4B-Instruct-2507 (Yang et al., 2025).

The evaluation data is the same as described in Section 5.1 and the evaluation methodology is the one described in Section 5.2.

We run each model on all samples and compute the average scores for all of them. These results are listed in Table 3 for the Symbol Segmentation task and in Table 4 for the Concepts Assignment and SoG Identification tasks. The performance in the two latter tasks is similar for all models, with less than 2% of standard deviation. However, the variability of results for the Symbols Segmentation task is significant: the Qwen3-4B-Instruct-2507 model achieves an advantage of over 20% compared to the other models in terms of coverage, and of over 10% in terms of clustering accuracy. Given this significant difference, we choose Qwen3-4B-Instruct-2507 as the model to use in our proof-of-concept study in Section 5.3.