

# Math Retrieval Using Leaf-Root Expression Trees Format

Andrew Norton<sup>\*</sup>  
University of Virginia  
Charlottesville, Virginia  
apn4za@virginia.edu

Ben Haines<sup>†</sup>  
University of Virginia  
Charlottesville, Virginia  
bmh5wx@virginia.edu

## ABSTRACT

This paper describes our implementation of a system for searching for mathematical expressions. The paper summarize existing techniques for math retrieval and describes our particular implementation. We propose a few extensions to implement and provide experimental results that measure the effectiveness of these proposals.

## 1. INTRODUCTION

### 1.1 Background

Within the field of Information Retrieval, the task of mathematical expression retrieval is a topic that has been attracting an increasing amount of attention in recent years. Math search is useful in a variety of situations, particularly for students and practitioners of technical fields. Particular scenarios include researchers who want to discover relevant work relating to a particular function or a student who needs help solving a particular problem. Existing solutions are often unsatisfying. For example, consider arxiv.org and math.stackexchange.com. These sites are two of the largest resources of collected mathematical information for both professional researchers and student. However, the usefulness of much of this information is decreased by challenges in locating it. The search feature of the arXiv does not permit searching for commonplace symbols such as "+" or "-". Searches on stackexchange often return no relevant results despite additional efforts revealing that multiple relevant results do exist. A better math retrieval system could in both cases increase productivity of many users.

Multiple approaches to parsing, indexing, and searching mathematical expression have been proposed. A specific state of the art algorithm does not exist as research in the field has not had time to converge to optimal solutions for the problems of storing and parsing expressions. Approaches

tend to fall into one of two categories, those that use established text based search methods with modifications in order to apply them to the particular problem of Mathematical Information Retrieval (MIR) and those that use tailored approaches that attempt to take advantage of the inherent structure of expressions to improve performance. We provide a brief comparison of techniques in these categories and justify our particular choice of a system to implement. Our research focuses on the effectiveness of the Leaf-Root path system for representing structured expressions. The primary contribution of the work is the suggestion of query expansion in order to allow searches for generalized expressions and an examination of how changes in the parsing grammar can effect performance.

## 2. RELATED WORK

Among the earliest discussions of a math retrieval system, and one of the few describing an actual large scale implementation is the paper by Youssef and Miller regarding the Digital Library of Mathematical Functions[3]. The idea proposed involves a sequence of steps to process mathematical notation into a format recognizable by existing search engines. This involves first using macros to map math symbols to standard alphanumeric text representations. For example "+" and "<" are mapped respectively to "plus" and "lt". Next, nested expressions, such as exponents, are flattened. Finally, expressions are normalized by sorting the leaves of the corresponding parse tree in a standardized manner.

The authors select an evolutionary approach that augments existing text search engines due to practicality constraints but they also outline a few relevant challenges that they suggest could be better addressed with a structural approach. These challenges include

- Recognition of mathematical symbols
- Capturing and indexing structure
- Accounting for mathematical "synonyms"

Since the 2003 publishing of the paper discussed above, incremental improvements have been suggested by a variety of sources. In 2007 Miner and Munavalli introduced a more involved for processing inputs while still relying on a standard text search infrastructure for the fundamental search operation.[4] Largely within the last five years a number of papers have emerged that attempt to address the problems in more fundamental ways. In 2012 an approach that bridges the gap between text and structure based methods is introduced in the paper "A structure based approach for mathe-

<sup>\*</sup>  
<sup>†</sup>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHARLOTTESVILLE '15 Charlottesville, Virginia USA  
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

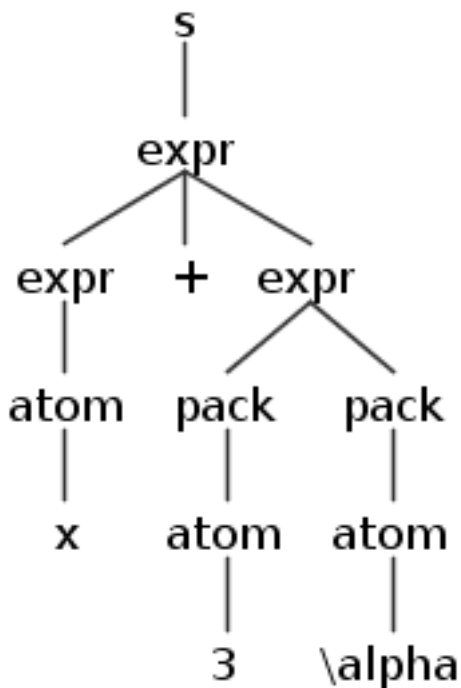


Figure 1: A sample substitution tree

mathematical expression retrieval”. [5] The fundamental idea is to use a modified longest common substring (LCS) algorithm to measure similarity between expressions. The expressions are tokenized and each token is given a label to indicate its nested depth. The LCS algorithm then weights the similarities between two expressions by how closely depth markers are aligned. An alternate structure capturing approach that has been explored in a number of papers is that of substitution trees. [2][1] In such a tree each internal node represents a generalized expression form and leaves represent specific expressions for which all variables have a substituted value. A simple example tree is shown in the next figure. This structure stores relationships between abstract expression forms. Within each node individual expressions are stored as symbol layout trees that map spatial relationships between individual components of an expression. The authors of this approach reported performance improvements when compared to a standard Lucene search that they attribute to substructure queries enabled by the tree structures.

The approach that most directly influenced our implementation is an alternative method for using trees to store structure. Furthering our intuition that a parse trees could be a useful representation of structure the particular implementation by Zhong provided inspiration and solutions to some technical challenges. [6]

### 3. PROPOSED SOLUTION

There are three general components to our proposed solution. Each will be discussed separately and the particular decisions we made will be justified.

#### 3.1 Representation

The system was inspired by and particularly targets the

Stack Exchange and arXiv use cases which both store math information as LaTeX. Additionally, the corpus of expressions made available to us contains LaTeX expressions. For these reasons we chose to focus on searching for LaTeX expressions rather than alternatives such as MathML. As a result of this we decided to accept LaTeX formatted inputs. The advantage of this choice is that LaTeX is a widely used system for expressing mathematical structures, in particular, users who want to search a LaTeX corpus are probably also capable of composing queries with it. As well as this, many simple expressions can be given without any overt formatting and will still be valid which increases ease of use.

When given a raw string, whether it be as part of the corpus or in a query the system first performs a number of preprocessing operations. First, operations such as “\left and \right” purely change the display of an expression and not the structure or content are removed. Second, semantically similar operations are grouped roughly into equivalence classes and occurrences are replaced with a single symbol. For example, “\pmod, \bmod, and \mod” which have the same meaning but different presentations are replaced with the common command “MOD”. Other operations that might have clear differences in meaning but are also similar in some way such as “+, \sum, and \bigoplus” are replaced with the common command “SUM”.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

### 3.2 Citations

Citations to articles [?, ?, ?, ?], conference proceedings [?] or books [?, ?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the .tex file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author’s surname and a word from the title. This identifying key is included with each item in the .bib file for your article.

The details of the construction of the .bib file are beyond the scope of this sample document, but more information can be found in the *Author’s Guide*, and exhaustive details in the *LaTeX User’s Guide* [?].

This article shows only the plainest form of the citation command, using \cite. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

### 4. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

### 5. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the `.cls` and `.tex` files that it describes.

## 6. REFERENCES

- [1] To be found... *Can't find it*, —.
- [2] M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In J. Calmet, T. Ida, and D. Wang, editors, *Artificial Intelligence and Symbolic Computation*, volume 4120 of *Lecture Notes in Computer Science*, pages 241–253. Springer Berlin Heidelberg, 2006.
- [3] B. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1-3):121–136, 2003.
- [4] R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 342–355. Springer Berlin Heidelberg, 2007.
- [5] P. Pavan Kumar, A. Agarwal, and C. Bhagvati. A structure based approach for mathematical expression retrieval. In C. Sombattheera, N. Loi, R. Wankar, and T. Quan, editors, *Multi-disciplinary Trends in Artificial Intelligence*, volume 7694 of *Lecture Notes in Computer Science*, pages 23–34. Springer Berlin Heidelberg, 2012.
- [6] W. Zhong. A novel similarity-search method for mathematical content in latex markup and its implementation. *Annals of Mathematics and Artificial Intelligence*, 2015.