

Abstract

The design and architecture of MlaS (Math Indexer and Searcher), a system for mathematics retrieval is presented, and design decisions are discussed. We argue for an approach based on Presentation MathML using a similarity of math subformulae. The system was implemented as a math-aware search engine based on the state-of-the-art system Apache Lucene. Scalability issues were checked against more than 400,000 arXiv documents with 158 million mathematical formulae. Almost three billion MathML subformulae were indexed using a Solr-compatible Lucene.

I do not seek. I find. (Pablo Picasso)

1. Introduction and Motivation

The solution to the problem of mathematical formulae retrieval lies at the heart of building digital mathematical libraries (DML).

How to write query



Search in: MREC 2011.4.439 Search

Total hits: 15973, showing 1-30. Searching time: 584 ms

Andrew bound states in normal and ferromagnetic high-T_c superconducting tunnel junctions
 close from the [110] surface when the symmetry is $d_{x^2-y^2}$...
 score = 1.1615998
 arxiv.org/abs/cond-mat/0305446 - cached XHTML

Particle trajectories and acceleration during 3D fan reconnection
 ... # $(\frac{dx}{dt}, \frac{dy}{dt}) = 1$ and ...
 score = 1.0577431
 arxiv.org/abs/0811.1144 - cached XHTML

Pairing symmetry and long range pair potential in a weak coupling theory of ...
 does not mix with usual s_{\pm} symmetry gap in an anisotropic band structure.
 score = 1.0254444
 arxiv.org/abs/cond-mat/9906142 - cached XHTML

Computers are useless. They can only give you answers. (Pablo Picasso)

2. Approaches to Searching Mathematics

A great deal of research on has been already undertaken on searching mathematical formulae in digital libraries and on the web. The comparison of math search systems, including our new MlaS is summarized in the table below.

System	Input documents	Internal representation	Approach	Query language	Queries	Indexing core
MathDex	HTML, TeX/LaTeX, Word, PDF	Presentation MathML (as strings)	syntactic	X	text, math, mixed	Apache Lucene
LaActiveMath	OpenMath, OpenMath	OpenMath (as string)	syntactic	X	text, math, mixed	Apache Lucene
IFEX/Search	IFEX	IFEX (as string)	syntactic	X	IFEX	Apache Lucene
MathWeb Search	Presentation MathML, Content MathML, OpenMath	Content MathML, OpenMath (substitution trees)	semantic	✓	QMath, IFEX, Mathematics, Maxima, Maple, Yacas styles (palette editor)	Apache Lucene
EgoMath	Presentation MathML, Content MathML, PDF	Presentation MathML trees (as strings)	mixed	✓	IFEX	EgoThor
MlaS	any (well-formed) MathML	Canonical Presentation MathML trees (as compact strings)	math tree similarity/normalization	✓	\mathcal{A}_N or IFEX or MathML	Apache Lucene/Solr

Everything you can imagine is real. (Pablo Picasso)

3. Design of MlaS

We have developed a math-aware, full-text based search engine called MlaS (Math Indexer and Searcher).

The top-level indexing scheme, including a detailed view of the mathematical part is shown in Figure 1.

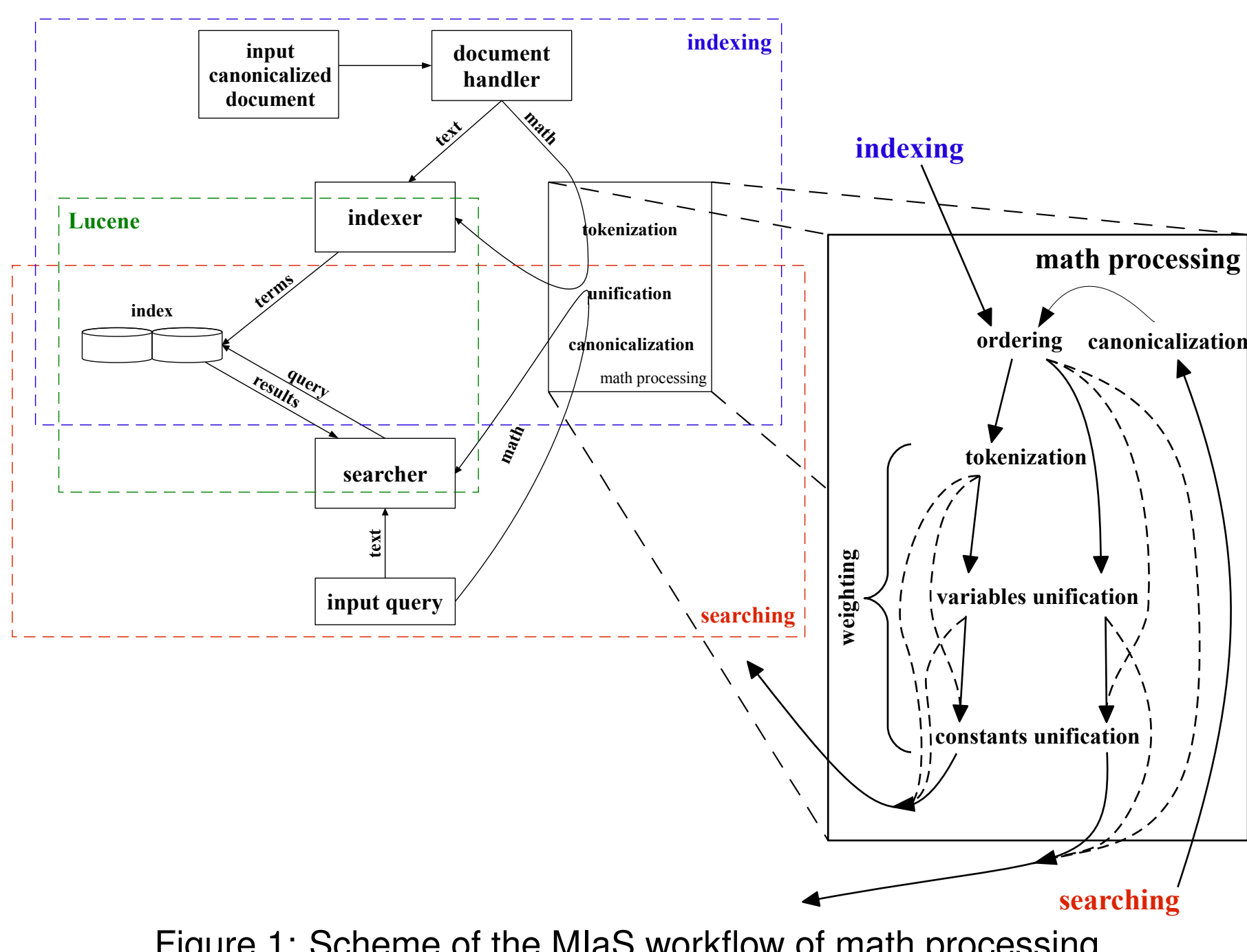


Figure 1: Scheme of the MlaS workflow of math processing

Indexing MlaS is currently able to index documents in XHTML, HTML and TXT formats. As Figure 1 shows, the input document is first split into textual and mathematical parts. The textual content is indexed in a conventional way. Mathematical expressions, on the other hand, are pre-analyzed in several steps to facilitate searches not only for exact whole formulae, but also for subparts (tokenization) and for similar expressions (formulae modifications). This addresses the issue of the static character of full-text search engines and creates several representations of each input formula all of which are indexed. Each indexed mathematical expression has a weight (relevancy score) assigned to it. It is computed throughout the whole indexing phase by individual processing steps following this basic rule of thumb—the more modified a formula and the lower the level of a subformula, the less weight is assigned to it.

At the end of all processing methods, formulae are converted from XML nodes to a compacted linear string form, which can be handled by the indexing core. Start and end XML tags are substituted by the tag name followed by an argument embraced by opening and closing parentheses. This creates abbreviated but still unambiguous representation of each XML node. For example, formula $a + b^2$, in MathML written as:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msup>
    <mi>a</mi>
    <+</+>
    <mi>b</mi>
    <mn>2</mn>
  </msup>
</math>
```

is converted to "math(mrow(mi(a)mo(+)msup(mi(b)mn(2))))" and this string is then indexed by Lucene.

Tokenization Tokenization is a straightforward process of obtaining subformulae from an input formula. MlaS makes use of Presentation MathML markup where all logical units are enclosed in XML tags which makes obtaining all subformulae a question of tree traversal. The inner representation of each formula is an XML node encapsulating all the member child nodes. This means the highest level formula—as it appears in the input document—is represented by a node named "math". All logical subparts of an input formula are obtained and passed on to modification algorithms.

Formulae Modifications MlaS performs three types of unification algorithms, the goal of which is to create several more or less generalized representations of all formulae obtained through the tokenization process. These steps allow the system to return similar matches to the user query while preserving the formula structure and α -equality.

Ordering Let us take a simple example: $a + 3$ and the query $3 + a$. This would not match even though it is perfectly equal. This is why a simple ordering of the operands of the commutative operations, addition and multiplication, is used. It tries to order arguments of these operations in the alphabetical order of the XML nodes denoting the operands whenever possible—it considers the priority of other relevant operators in the formula. The system applies this function to the formula being indexed as well as to the query expression. Applied to the example above, the XML node denoting variable a is named "mi", the node denoting number 3 is named "mn". "mi" < "mn" therefore $3 + a$ would be exchanged for $a + 3$ and would match.

Unification of Variables and Constants Let us take another example: $a + b^2$ and $x + y^2$. Again, these would not match even though the difference is only in the variables used. MlaS employs a process that unifies variables in expressions while taking bound variables into account. All variables are substituted for unified symbols (ids) in both the indexing and searching phases. Applied to the example, both expressions would unify to $id_1 + id_2^{[2]}$ and would match. This process is not applied to single symbols—this would lead to the indexing of millions of ids and searching for any symbol would end up matching all of the documents containing it.

Unification of constants is a straightforward process of substituting all the numerical constants for one unified symbol (const). This obviates the need for the exact values of constants in user queries.

Formulae Weighting During the searching phase, a query can match several terms in the index. However one match can be more important to the query than another, and the system must consider this information when scoring matched documents. For mathematical formulae the system makes use of the processing operations described above since they all produce expressions more generalized than the input ones.

Each formula has a weight attribute indexed alongside itself, which belongs to the interval $(0, 1)$. Weight w of the subformula contained on a certain level in a parent formula with the number of nodes (n) can be calculated in particular situations as follows:

- no changes made: $w = \frac{level(1+1+c+vc)}{n}$
- unified variables: $w = \frac{level(v+c+vc)}{n}$
- unified constants: $w = \frac{level(c+vc)}{n}$
- unified both variables and constants: $w = \frac{level(vc)}{n}$

To fine tune the weighting parameters, we developed a tool with verbose output in which the behavior of the model can be observed and tested. A sample from the tool mentioned above is shown in Table 1.

We have come to the conclusion that the unification of variables interferes less with original formula meaning than the unification of number constants. For this reason, its coefficient should be higher—i.e., less discriminating. The main question then became, how discriminating the level coefficient should be. Our empirical deduction is that going deeper in a structural tree should be discriminating, the precise match on a lower level should still score more than any unified formula on the level above, as could be seen in Table 1: $\frac{1}{2^4} = \frac{1}{16}$ (row 5) is an exact match on the second level and its score is higher than unified expressions matched on the first level (rows 2, 3 and 4). This led us to the valuation of level weighting coefficient $l = 0.7$, unification weighting coefficient $v = 0.8$ and constant weighting coefficient $c = 0.5$.

In Figure 2 the whole formula preprocessing process is illustrated together with its subformulae weightings.

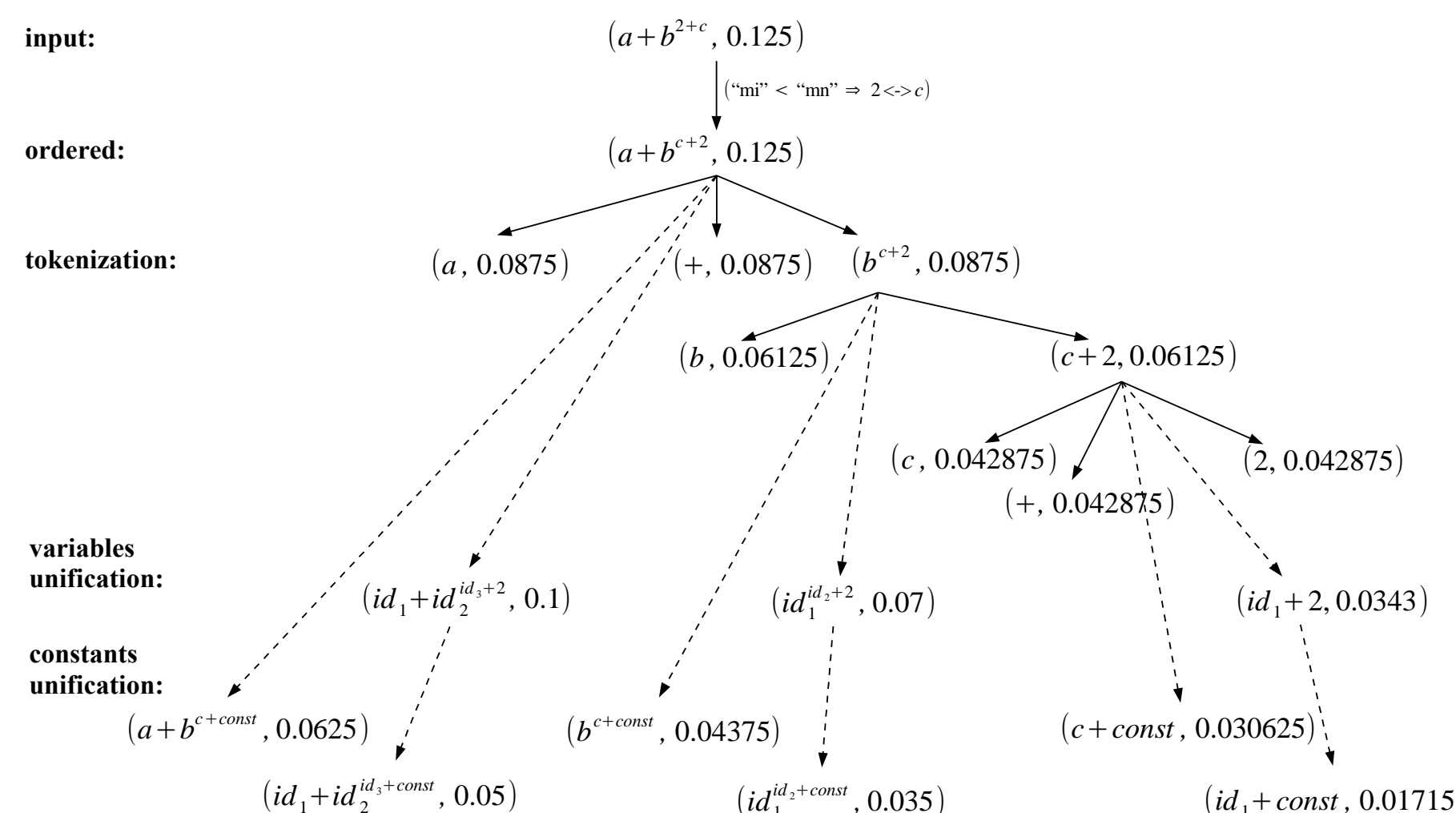


Figure 2: Example of formula preprocessing. Ordered pairs are (\langle expression written naturally \rangle , \langle it's weight \rangle). All expressions as shown are indexed, except for the original one.

Table 1: Example of weighting function on several formulae. Original query is $a + 3$ —all queried expressions are $a + 3$, $id_1 + 3$, $a + const$, $id_1 + const$.

Formula	Indexed Expressions	Score	Matched
$a + 3$	$0.25 \cdot (a + 3)$, $0.2 \cdot (id_1 + 3)$, $0.175 \cdot (a, 3)$, $0.125 \cdot (a + const)$, $0.1 \cdot (id_1 + const)$, $0.2 \cdot (id_1 + const)$, $0.125 \cdot (a + const)$, $0.1 \cdot (id_1 + const)$, $0.125 \cdot (a + const)$, $0.1 \cdot (id_1 + const)$	2.7	0.1 $(id_1 + const)$, $0.25 \cdot (a + 3)$, $0.2 \cdot (id_1 + 3)$
$b + 3$	$0.25 \cdot (b + 3)$, $0.2 \cdot (id_1 + 3)$, $0.175 \cdot (b, 3)$, $0.125 \cdot (b + const)$, $0.1 \cdot (id_1 + const)$	1.2	0.1 $(id_1 + const)$, $0.2 \cdot (id_1 + 3)$
$a + 5$	$0.25 \cdot (a + 5)$, $0.2 \cdot (id_1 + 5)$, $0.175 \cdot (a, 5)$, $0.125 \cdot (a + const)$, $0.1 \cdot (id_1 + const)$	0.9	0.1 $(id_1 + const)$, $0.2 \cdot (id_1 + 5)$
$c + 10$	$0.25 \cdot (c + 10)$, $0.2 \cdot (id_1 + 10)$, $0.175 \cdot (c, 10)$, $0.125 \cdot (c + const)$, $0.1 \cdot (id_1 + const)$	0.4	0.1 $(id_1 + const)$
$\frac{1}{2}a$	$0.16667 \cdot (\frac{1}{2}a)$, $0.133334 \cdot (\frac{1}{2}a)$, $0.116667 \cdot (\frac{1}{2}a + 3)$, $0.093334 \cdot (id_1 + 3)$, $0.083334 \cdot (id_1 + const)$, $0.081667 \cdot (\frac{1}{2}a, 3)$, $0.066667 \cdot (\frac{1}{2}a + const)$, $0.058333 \cdot (b + const)$	1.26	$0.046667 \cdot (id_1 + const)$, $0.116667 \cdot (\frac{1}{2}a + 3)$, $0.093334 \cdot (id_1 + 3)$
$\frac{1}{2}b$	$0.16667 \cdot (\frac{1}{2}b)$, $0.133334 \cdot (\frac{1}{2}b)$, $0.116667 \cdot (\frac{1}{2}b + 3)$, $0.093334 \cdot (id_1 + 3)$, $0.083334 \cdot (id_1 + const)$, $0.081667 \cdot (\frac{1}{2}b, 3)$, $0.066667 \cdot (\frac{1}{2}b + const)$, $0.058333 \cdot (b + const)$	0.56	$0.046667 \cdot (id_1 + const)$, $0.116667 \cdot (\frac{1}{2}b + 3)$
$\frac{1}{2}c$	$0.16667 \cdot (\frac{1}{2}c)$, $0.133334 \cdot (\frac{1}{2}c)$, $0.116667 \cdot (\frac{1}{2}c + 5)$, $0.093334 \cdot (id_1 + 5)$, $0.083334 \cdot (id_1 + const)$, $0.081667 \cdot (\frac{1}{2}c, 5)$, $0.066667 \cdot (\frac{1}{2}c + const)$, $0.058333 \cdot (c + const)$	0.42	$0.046667 \cdot (id_1 + const)$, $0.116667 \cdot (\frac{1}{2}c + 5)$
$\frac{1}{10}a$	$0.16667 \cdot (\frac{1}{10}a)$, $0.133334 \cdot (\frac{1}{10}a)$, $0.116667 \cdot (\frac{1}{10}a + 10)$, $0.093334 \cdot (id_1 + 10)$, $0.083334 \cdot (id_1 + const)$, $0.081667 \cdot (\frac{1}{10}a, 10)$, $0.066667 \cdot (\frac{1}{10}a + const)$, $0.058333 \cdot (c + const)$	0.19	$0.046667 \cdot (id_1 + const)$

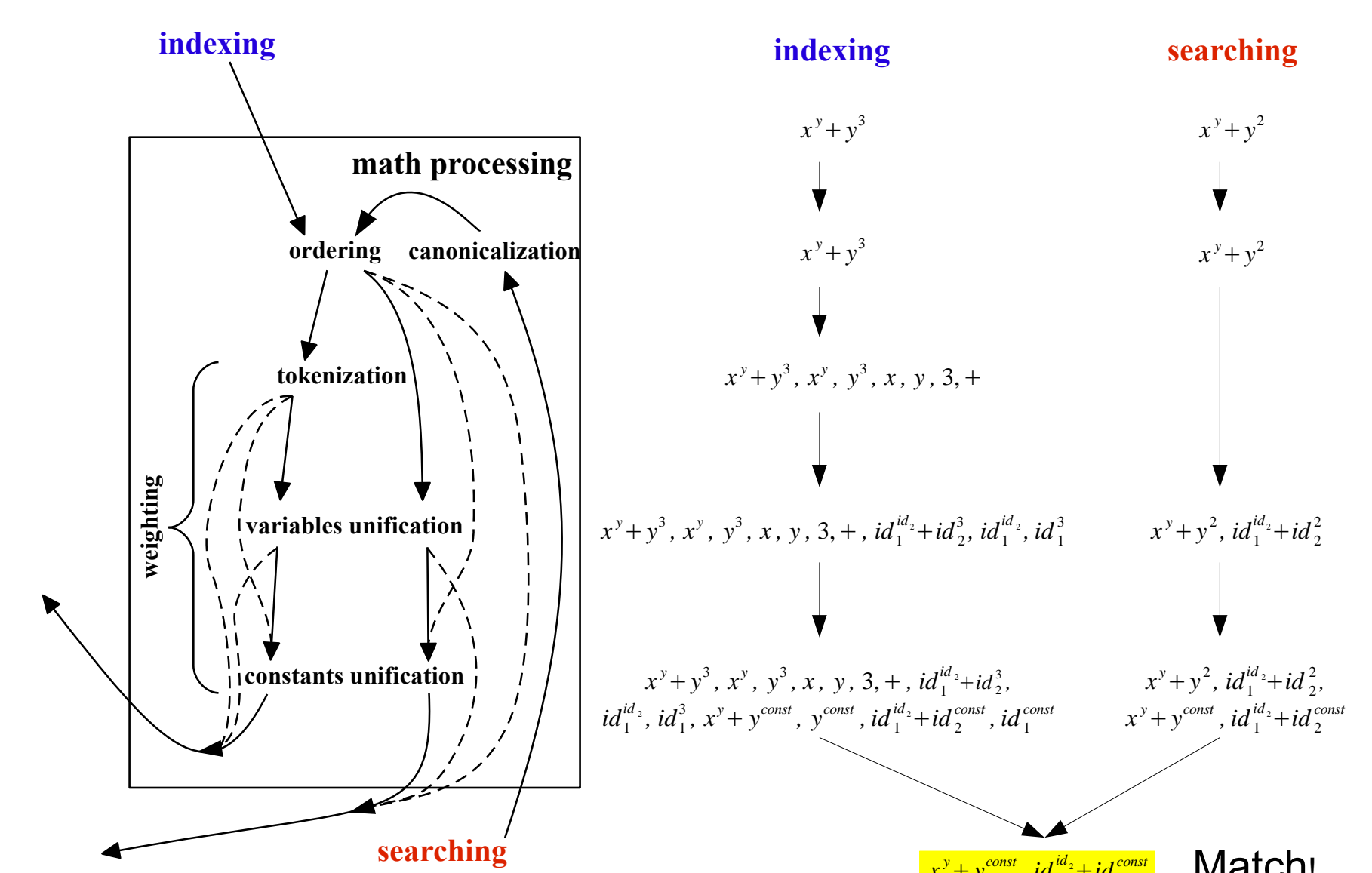


Figure 3: Math-aware search in MlaS

Searching In the search phase, user input is again split into mathematical and textual parts. Formulae are then reprocessed in the same way as in the indexing phase, except for tokenization—which we doubt that users are likely to query, for example $\frac{a+b}{c}$ wanting to find documents only with occurrences of variable c . That means the queried expressions are first ordered, then unified. This produces several representations which are connected to the final query by the logical OR operator.

A very positive value has its price in negative terms...
 the genius of Einstein leads to Hiroshima. (Pablo Picasso)

4. Evaluation and Implementation

For large scale evaluation, we needed an experimental implementation and a corpus of mathematical texts. The Math Indexer and Searcher is written in Java. The role of full-text indexing and searching core is performed by Apache Lucene 3.1.0. The mathematical part of document processing can be seen as a standalone pluggable extension to any full-text library, however it would need custom integration for each one. In the case of Lucene, a custom Tokenizer (MathTokenizer) has been implemented.

For the textual content of documents, Lucene's StandardAnalyzer is employed. In MathTokenizer, TermAttributes are used for carrying strings of math expressions and PayloadAttribute for storing weights of formulae. Lucene's practical scoring function for every hit document d by query q with each query term t is as follows:

$$score(q, d) = coord(q, d) \cdot queryNorm(q) \cdot \sum_{t \in q} (tf(i \text{ in } d) \cdot idf(i)^2 \cdot l.getBoost(i) \cdot norm(t, d))$$

Corpus of Mathematical Documents MREC A document corpus MREC (version 2011.3.324) was used to evaluate the behaviour of the system we modelled. The documents come from the arXMLiv project that is converting document sets from arXiv into XHTML + MathML (both Content and Presentation) [2].

We were able to gather great amount of documents in MREC corpus version 2011.4.439 to test our indexing system. This corpus consists of 439,423 arXiv documents containing 158,106,118 mathematical formulae. 2,910,314,146 expressions were indexed and the resulting size of the index is 47 GB. Sizes of uncompressed and compressed corpora size are 124 GB and 15 GB, respectively. MREC corpora are available to the community for download from MREC web page <http://nlp.fi.muni.cz/projekty/eudml/MREC/> so that other math indexing engines could be compared with MlaS on the same data.

Results MlaS demonstrated the ability to index and search a relatively vast corpus of real scientific documents. Its usability is highly elevated thanks to its preprocessing functions together with formulae weighting model. The ability to search for exact and similar formulae and subformulae, more so with customizable relevancy computation, demonstrates an unquestionable contribution to the whole search experience.

We have created a demo web interface WebMlaS which is publicly available on the MlaS web page <http://nlp.fi.muni.cz/projekty/eudml/mias/>. Our WebMlaS interface supports queries in two different notations—in \mathcal{A}_N -IFEX and MathML. Mathematical queries are additionally canonized using XSLT transformations from UMCL library [1] to improve the query and to avoid notation flaws restraining proper results retrieval. Portability of the interface is increased by using MathJax for rendering of mathematical formulae in snippets.

Scalability Testing and Efficiency We have devised a scalability test to see how the system behaves with an increasing number of documents and formulae indexed. Subsets containing 10,000, 50,000, 200,000 and the complete 324,060 documents were gradually indexed and several values were measured: the number of input formulae, the number of indexed formulae, the indexing time and the average query time. Indexing time of this corpus was 1378.82 min, e.g. almost 23 hours.

Table 2: Scalability test results (run on 32 GB RAM, quad core AMD Opteron™ Processor 850 driven machine).

Documents	Input formulae	Indexed formulae	Indexing time [min]	Average query time [ms]
10,000	3,450,114	65,194,737	39.15	32
50,000	17,734,342	334,078,835	201.68	178
200,000	70,102,960	1,316,603,055	889.28	576
324,060	112,055,559	2,129,261,646	1,292.16	789

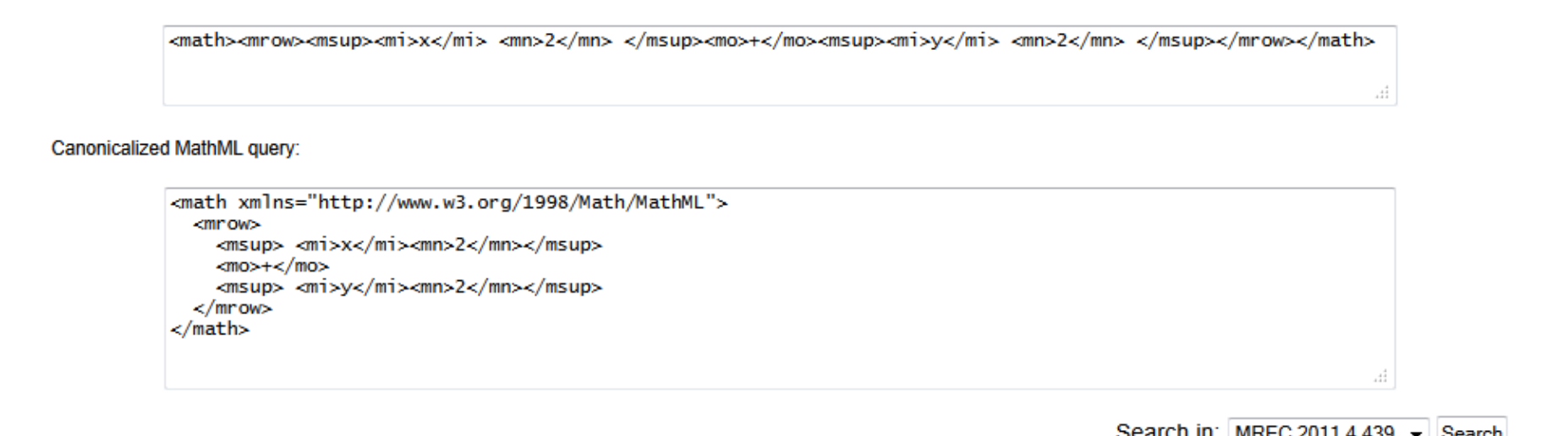
I am always doing that which I can not do, in order that I may learn how to do it. (Pablo Picasso)

5. Conclusions, Credits

We have presented an approach to mathematics searching and indexing—the architecture and design of the MlaS system. The feasibility of our approach has been verified on large corpora of real mathematical papers from arXMLiv. Scalability tests have confirmed that the computing power needed for fine math similarity computations is readily available; this would allow the use of this technology for projects on a European or world-wide scale.



How to write query



Search in: MREC 2011.4.439 Search

Finite Precision Measurement Nullifies Euclid's Postulates
 ... and the unit circle $x^2 + y^2 = 1$ are both dense but they do not intersect, in contradiction to Euclid's postulates ...
 score = 3.2980976
 arxiv.org/abs/quant-ph/0310035 - cached XHTML

COMMENT ON RECENT TUNNELING MEASUREMENTS ON Bi22Sr2CaCu2O8x
 ... gap, (b) s-wave gap and (c) S_{\pm} gap.
 ...

Canonical MathML query:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <math display="block">x^2 + y^2 = 1</math>
  </math>
```

Total hits: 36817, showing 1-30. Searching time: 116 ms

Figure 4: Web interface of MlaS for The European Digital Mathematics Library **Acknowledgements.** Our work has been partially supported by the Ministry of Education of Czech Republic within the Center of Basic Research LC536 and by the European Union through its Competitiveness and Innovation Programme (Policy Support Programme, "Open access to scientific information", Grant Agreement No. 250503).

References

- D. Archambault, F. Berger, and V. Moço. Overview of the "Universal Maths Conversion Library". In A. Pruski and H. Knops, editors, *Assistive Technology: From Virtuality to Reality: Proceedings of 8th European Conference for the Advancement of Assistive Technology in Europe AAATE 2005*, Lille, France, pages 256–260, Amsterdam, The Netherlands, Sept. 2005. IOS Press.
- H. Stamerjohanns, M. Kohlhas, D. Genev, C. David, and B. Miller. Transforming Large Collections of Scientific Publications to XML. *Mathematics in Computer Science*, 3:299–307, 2010. <http://dx.doi.org/10.1007/s11786-010-0024-7>.

