REPORT ON D6.1,
LEAD BY Jacobs University Bremen (JacobsUni):
Full-text Search (Formulae + Keywords) over LaTeX-based Documents (e.g. the arXiv subset)

MICHAEL KOHLHASE & ALEXANDRU GLONTARU

Due on: 31/05/2016 (Month 9)
Delivered on: 27/06/2016

Progress on and finalization of this deliverable has been tracked publicly on:
`https://github.com/OpenDreamKit/OpenDreamKit/issues/133`

ABSTRACT. A virtual research environment (VRE) contains extensive data (D), representations of knowledge (K), and software (S). This only becomes accessible to users, if the VRE provides a unified search facility for D/K/S structures. In the mathematical VRE under development in the OpenDreamKit project, we have mathematical D/K/S, so that conventional search engines only partially apply.

In this report, we present the MathWebSearch engine, an open-source, open-format, content-oriented full-text search engine for mathematical formulae encoded in Content MathML. We eventually want to use it for searching notebooks, active documents and the Math-in-the-Middle Ontology in the OpenDreamKit VRE. This report only presents the current system that can search mathematical documents written in LaTeX or HTML5.

Contents

1. Introduction: Search in OpenDreamKit

A virtual research environment (VRE) contains extensive data (D), representations of knowledge (K), and software (S). This only becomes accessible to users, if the VRE provides a unified search facility for D/K/S structures. In the mathematical VRE under development in the OpenDreamKit project, we have mathematical D/K/S, so that conventional search engines only partially apply.

In this report, we present the MathWebSearch engine, an open-source, open-format, content-oriented full-text search engine for mathematical formulae encoded in Content MathML. We eventually want to use it for searching notebooks, active documents and the Math-in-the-Middle Ontology in the OpenDreamKit VRE. This report only presents the current system that can search mathematical documents written in LaTeX or HTML5. For applications to other D/K/S structures, see Section 11.

The report is organized as follows: in Section 2 we give a general overview over the MathWebSearch system as a base for a discussion of the system's components in the ensuing sections: Sections 3 and 4 discuss the input format (MWS harvests) and how to convert TeX/LaTeX documents so that they can be harvested. Sections 5 to 8 give an overview over internals of the MathWebSearch backend: indexing of formulae and full text, and distributed querying. Section 9 presents a web front-end for MathWebSearch and Section 10 discusses the larger corpora the system is used for. Section 11 concludes the report and details future work.

2. MathWebSearch Overview

Mathematics-aware information retrieval (MIR) has often been touted as one of the "killer applications" of computer support in STEM (Science, Technology, Engineering, and Mathematics). Indeed, the yearly production of published research in mathematics exceeds 120 thousand journal articles a year alone [MK14]; the amount of internal technical documents in company archives certainly exceeds this quantity by far; being able to access the knowledge locked up in them will become a determining factor for commercial success.

MWS is a web application that provides low-latency answers to full-text queries which consist of keywords and formulae. MathWebSearch front-ends convert formula schemata (with query variables) into content MathML expressions, which the MathWebSearch formula indexer answers by unification and combines the results with keyword results from a text search engine. The modular architecture and standardized formats makes MathWebSearch applicable to a wide range of querying tasks: web-based formula search engines or editing support services. Unification queries over content MathML expressions form the basis of an expressive query language with well-defined semantics. As substitution instances of the original query, MathWebSearch results are highly significant, if the encoding of data set and search query are adequate – i.e. do not drop or spuriously introduce salient semantic features [MK14].

At its core, the MathWebSearch system is a content-based search engine for mathematical formulae. It indexes MathML formulae, using a technique derived from automated theorem proving: Substitution Tree Indexing. Recently, it was augmented with full-text search capabilities, combining keyword queries with unification-based formula search. The engine serving text queries is Elasticsearch 2.3. From now on, in order to avoid confusion, we will refer to the core system (providing just formula query capability) as MathWebSearch and to the complete service (MWS + Elasticsearch) as TeMaSearch (Text + Math Search).

The overall workflow of TeMaSearch is the following [RH15]:

(1) HTML5 documents representing mathematical articles are crawled to generate MathWebSearch harvests. The harvest format is an extension of MathML which MathWebSearch can index. Its role is to separate the math from the text in a given document.

(2) MathWebSearch indexes the harvests.

(3) a second pass is made over the harvests to generate annotated documents.

(4) Elasticsearch indexes the annotated documents.

(5) Everything is now ready for answering queries. When a query is issued, MathWebSearch will answer the mathematical part and Elasticsearch will answer the text part. The results are combined through a NodeJS proxy to send a final result set.

Each mathematical expression is encoded as a set of substitutions based on a depth-first traversal of its Content MathML tree. Furthermore, each tag from the Content MathML tree is encoded as a TokenID, to lower the size of the resulting index. The (bijective) mapping is also stored together with the index and is needed to reconstruct the original formula. The index itself is an in-memory tree of substitution paths.

To facilitate fast retrieval, MathWebSearch stores FormulaIDs in the leaves of the substitution tree. These are integers uniquely associated with formulae, and they are used to store the context in which the respective expressions occurred. These identifiers are stored in a separate LevelDB database.

3. MwsHarvests

MwsHarvests is the format that the MathWebSearch accepts as crawled data. It is an extension of MathML which introduces a few tags and attributes to specify meta-information about the crawled Content MathML data [Mws].

The introduced tags are **harvest**, **expr** and **data**, all defined in the mws namespace. The root element of a MwsHarvest is a **mws:harvest**. Its children are **mws:expr** and **mws:data** nodes. **mws:expr** nodes contain the actual ContentMathML and have the following attributes:

- **url** specifies the URL+UUID of the **m:math** from which the content was extracted,
- **data_id** specifies the **id** of a **mws:data** node previously defined in this document. The respective data will be associated with this expression.

mws:data nodes contain arbitrary XML data and have the following attributes:

- **data_id** specifies an unique identifier within this XML document.

Note that **mws:data** has been introduced in version 2 of the **mws:harvest**, but it is backward compatible.

An example harvest document is provided here:

```
<?xml version="1.0"?>
<mws:harvest xmlns:mws="http://search.mathweb.org/ns"
                         xmlns:m="http://www.w3.org/1998/Math/MathML">
   <mws:data mws:data_id="foo">
      <!-- arbitrary XML data -->
   </mws:data>
   <mws:expr url="http://math.example.org/article123456#e123456"
                         mws:data_id="foo">
```

```
    <m:apply>
      <m:eq/>
      <m:apply>
        <m:apply>
          <m:csymbol cd="ambiguous">subscript</m:csymbol>
          <m:limit/>
          <m:apply>
            <m:ci>&#x2192;</m:ci>
            <m:ci>x</m:ci>
            <m:cn>0</m:cn>
          </m:apply>
        </m:apply>
        <m:apply>
          <m:divide/>
          <m:cn>1</m:cn>
          <m:apply>
            <m:csymbol cd="ambiguous">superscript</m:csymbol>
            <m:ci>x</m:ci>
            <m:cn>2</m:cn>
          </m:apply>
        </m:apply>
      </m:apply>
      <m:infinity/>
    </m:apply>
  </mws:expr>
  <!--
    More mws:data and mws:expr nodes
  -->
</mws:harvest>
```

This specifies that the data contained in the **`mws:expr`** was extracted from an **`m:math`** node found in the document `http://math.example.org/article123456` with id e123456.

4. LATEXML

An overwhelming majority of the digital scientific content is written using LATEX or TEX, due to its usability and popularity among STEM researchers. However, formulae in these formats are not good candidates for searching because they do not display the mathematical structure of the underlying idea. For this purpose, conversion engines have been developed to convert LATEX expressions to more organized formats such as MathML [RH15].

An open source example of such a conversion engine is LATEXML. The MathWebSearch project relies heavily on it, to convert arXiv documents from LATEX to XHTML which is later indexed by MathWebSearch. It exposes a powerful API, accepting custom definition files which relate TEX elements to corresponding XML fragments that should be generated. For the scope of this project, we are more interested in another feature of LATEXML: cross-referencing between Presentation MathML and Content MathML. While converting TEX entities to Presentation MathML trees, LATEXML assigns each PMML element a unique identifier which is later referenced from the corresponding Con-tent MathML element. In this manner, we can modify the Content MathML tree and reflect the changes in the Presentation MathML tree which can be displayed to the user.

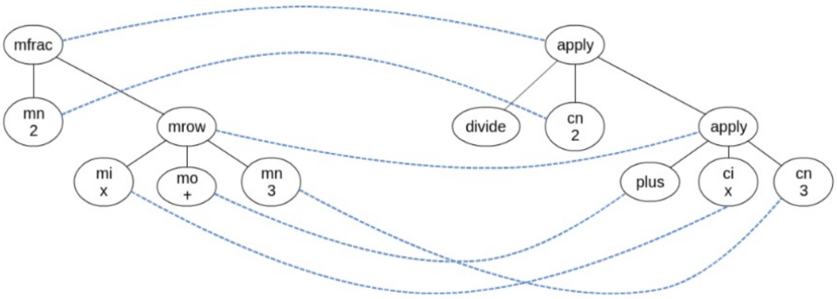


FIGURE 1. The CMML/PMML Parallel Markup

Figure 1 illustrates the parallel markup for 2 Presentation MathML and on the right side, Content MathML. As we can see, every Content element has a Presentation correspondent, except the divide operator, whose meaning is reflected in the structure of the displayed formula. On the left side we have Presentation MathML and on the right side, Content MathML. As we can see, every Content element has a Presentation correspondent, except the divide operator, whose meaning is reflected in the structure of the displayed formula.

5. INDEXING MATHEMATICAL FORMULAE

For indexing mathematical formulae on the web, we will interpret them as first-order terms. This allows us to use a technique from automated reasoning called term indexing. This is the process by which a set of terms is stored in a special purpose data structure (the index, normally stored in memory) where common parts of the terms are potentially shared, so as to minimize access time and storage. The indexing technique we work with is a form of tree-based indexing called substitution-tree indexing [KŞ06]. A substitution tree, as the name suggests, is simply a tree where substitutions are the nodes. A term is constructed by successively applying substitutions along a path in the tree, the leaves represent the terms stored in the index. Internal nodes of the tree are generic terms and represent similarities between terms.

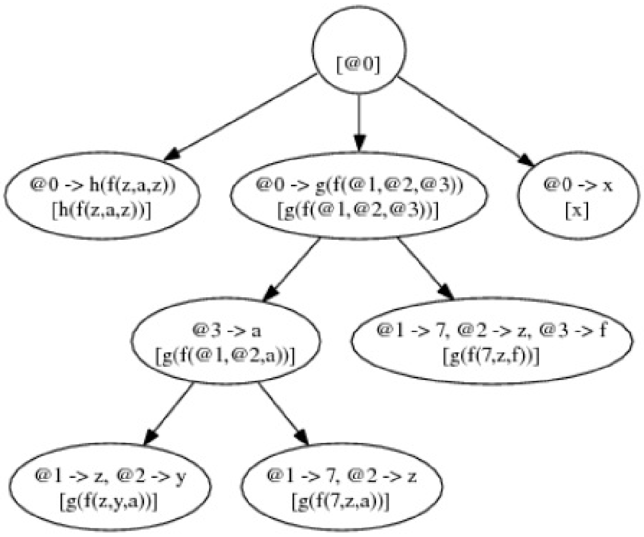


FIGURE 2. Division Tree Example

The main advantage of substitution tree indexing is that we only store substitutions, not the actual terms, and this leads to a small memory footprint. Figure 2 shows a typical index for the terms $h(f(z,a,z))$, x, $g(f(z,y,a))$, $g(f(7,z,a))$, and $g(f(7,z,f))$. For clarity we present not only the substitutions in the node, but the term produced up to that node as well (between square brackets). The variables @integer are used to denote placeholder variables for parts that differ between terms. All placeholder variables are substituted before a leaf is reached.

Adding data to an existing index is simple and fast, querying the data structure is reduced to performing a walk down the tree. In contrast to automated reasoning our application does not need tree merging. Therefore we use substitutions only when building the index. Index building is done based on Algorithm 1. Once the index is built, we keep the actual term instead of the substitution at each node, so we do not have to recompute it with every search. Structure sharing methods conserve memory and make this tractable. To each of the indexed terms, some data is attached — an identifier that relates the term to its exact location. The identifier, location and other relevant data are stored in a database external to the search engine. We use XPointer references to specify term locations.

Unfortunately, substitution tree indexing does not support subterm search in an elegant fashion, so when adding a term to the index, we add all its subterms as well. This simple trick works well: the increase in index size remains manageable and it greatly simplifies the implementation. The rather small increase is caused by the fact that many of the subterms are shared among larger terms and they are only added once.

6. Elastic Search

Elasticsearch is a powerful and efficient full text search and analytics engine, built on top of Lucene [Ela]. It can scale massively, because it partitions data in shards and is also fault tolerant, because it replicates data. It indexes schema-free JSON documents and the search engine exposes a REST-ful web interface. The query is also structured as JSON and supports a multitude of features via its domain specific language: nested queries, filters, ranking, scoring, searching using wildcards/ranges and faceted search.

The faceted search feature [1] this feature is the terms aggregation: a multi-bucket aggregation, with dynamically built buckets. We can specify an array field from a document and ask ES to count how many unique items from the array are there in the whole index. This list can also be sorted, e.g. most frequently occurring items first. Additionally, we can also impose a limit on the number of the buckets (items) for which we want to receive the count. is of particular interest to us. One way to use this feature is the terms aggregation: a multi-bucket aggregation, with dynamically built buckets. We can specify an array field from a document and ask ES to count how many unique items from the array are there in the whole index. This list can also be sorted, e.g. most frequently occurring items first. Additionally, we can also impose a limit on the number of the buckets (items) for which we want to receive the count.

LISTING 1. Elastic Search Term Aggregation Query

```
{
        "query" : {
                "match" : {
                        "body" : {
                                "query" : "Pierre Fermat",
                                "operator" : "and"
```

[1] Faceted search as such is now deprecated in ES and was replaced by the more powerful "aggregations" feature, which also allows facets of facets, i.e. nested facets.

```
                }
            }
        },

        "aggs" : {
            "formulae" : {
                "terms" : { "field" : "ids" }
            }
        }
}
```

An ES query which would return the most frequently used formulae (and subformulae) for “Pierre Fermat”, is presented in Listing 1. The key part is the aggs fields. We are specifying that we want an aggregation called Formulae on “terms” (i.e. we want bucket counting) and the target of the aggregation is the fields ids.

A possible response to the above query can be found in Listing 2. In the response we can see the returned aggregations. In our example there is only one and it is called formulae. We can find the actual result in the buckets field. The key field in the bucket corresponds to a FormulaID. Here, the most frequent formulae were the one with ID 230 and the one with ID 93. The former appeared in 10 documents and the latter appeared in 9 documents.

LISTING 2. Elastic Search Term Aggregation Response

```
{
    ...
    "aggregations" : {
        "formulae" : {
            "buckets" : [
            {
                "key" : "230",
                "doc_count" : 10
            },
            {
                "key" : "93",
                "doc_count" : 9
            },
            ...
            ]
        }
    }
}
```

7. INTEGRATION VIA THE QUERY PROXY

The query proxy is a HTTP RESTful interface which provides text and formula search. It is the operational heart of the full-text search functionality that is new for the NTCIR-11 version of MathWebSearch. As we show in figure 3, its backend is a MathWebSearch formula indexer serving the formula index and an ElasticSearch daemon serving the annotated documents.

Upon receiving a query, it sends the formula to the formula indexer service and receives the matching formula identifiers. Using these, along with the text query, it queries ElasticSearch for documents which contain the respective text and the formula identifiers. As these documents may be large, only their unique identifiers are retrieved and a secondary query is used to provide snippets with the relevant data.

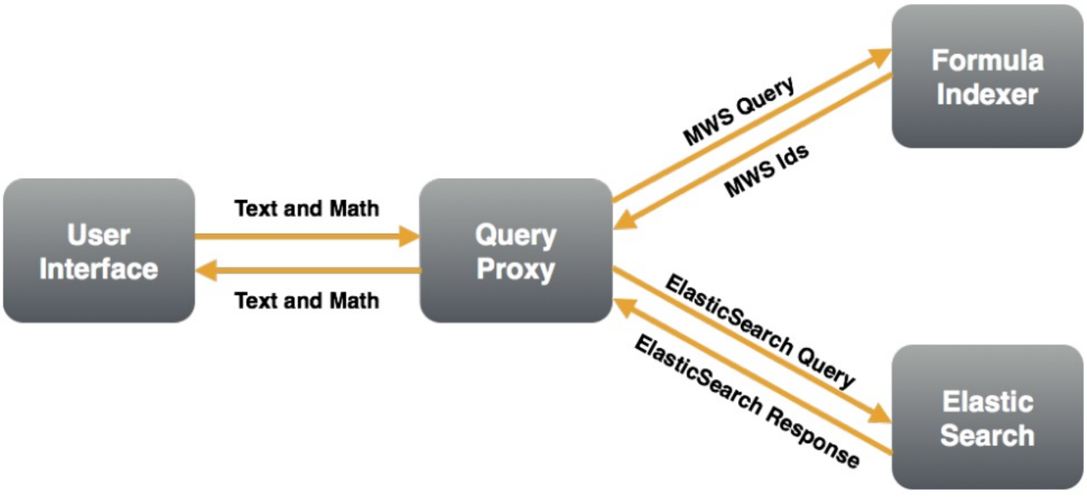


FIGURE 3. The CMML/PMML Parallel Markup

Internally, we can configure what is considered a hit by manipulating the query we present to ElasticSearch. It understands an expressive language, allowing us to define a match as any document containing the formula, and (or) at least n or a percentage of words. In the current configuration, a document is considered a hit if it contains a formula matching the formula query and at least one of the words in the text query.

8. DISTRIBUTION OF INDEXING AND QUERYING

As presented above, the main indexing data structure is a DFS substitution tree. To represent the tree in a manner which supports cross-machine links, we use three types of index nodes: [PK11]

(1) **Internal Index Nodes** They are used to navigate through most parts of the tree. Their data stores mappings from token ID to the corresponding index node.

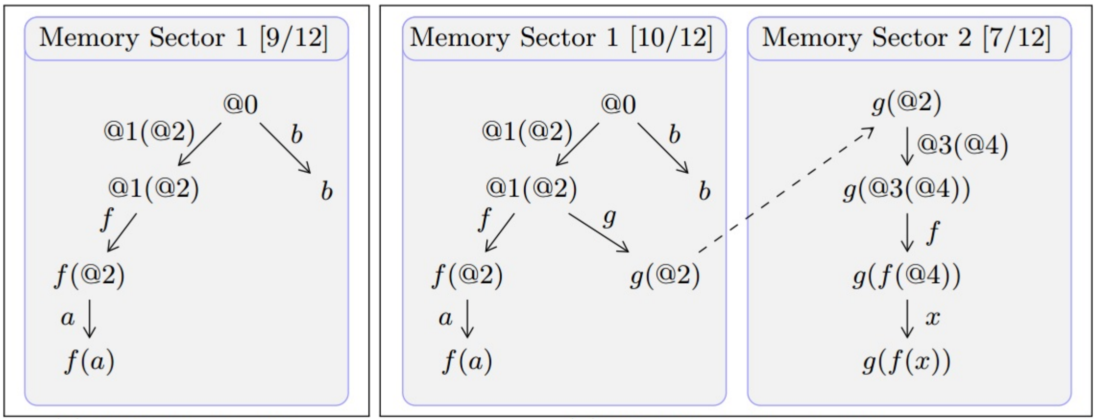
(2) **Leaf Index Nodes** They represent the end of a particular formula and its corresponding ID in the URL+URI database.

(3) **Remote Index Nodes** They represent cross-sector links. Their data consists of a pair of memory sector ID and node ID, which uniquely determine the corresponding index node.

As harvests are fed into the system, the index is built. Notice that, instead of building it on the heap (with no control over individual node's memory locations), the system places it inside specific memory sectors.

Memory sectors are continuous areas of memory of fixed sizes, usually represented as memory mapped files. They are the smallest units of replication, migration, and load balancing. When the system starts, an initial memory sector is created and the tree is built in this area. As terms get inserted, the sector fills. When it reaches a given threshold, a new sector is created and the contents of the original sector get split between the two.

In Figure 4, we present an example of term insertion which causes a tree split. Consider the initial tree containing the expressions f(a) and b and Memory Sectors of size 12 units, as depicted in Figure 4(a). We consider a memory model where each link and each node costs 1 unit. Let's insert g(f(x)). As the resulting tree would overflow the current memory sector, a new one is created and parts of the tree are migrated to the new memory sector (See Figure 4(b)). To split the tree, the system performs a DFS traversal, as long as the size needed to store the explored and queued nodes does not exceed a given threshold (in our case, 10). Once a node expansion would go over it, all the internal nodes in the queue are transformed into remote nodes and the rest is moved to the other memory sector.



(a) Initial Index Tree (b) Distributed Index Tree

FIGURE 4. Tree Distribution across Memory Sectors

The advantage of the new data structures is that indexes can be distributed over multiple computers: when the memory usage on one machine goes above a specific threshold, a percentage of the memory sectors are migrated to other machines [PK11]. As the tree uses only relative pointers and the sectors are represented as memory mapped files, it is enough to flush the memory sector to the file, send it across the network, and re-map it into the new system's memory (of course, we assume endian-compatible machines). All operations on memory sectors (splits, migrations) are coordinated by a master node, which keeps track of their locations, as well as cross-sector links.

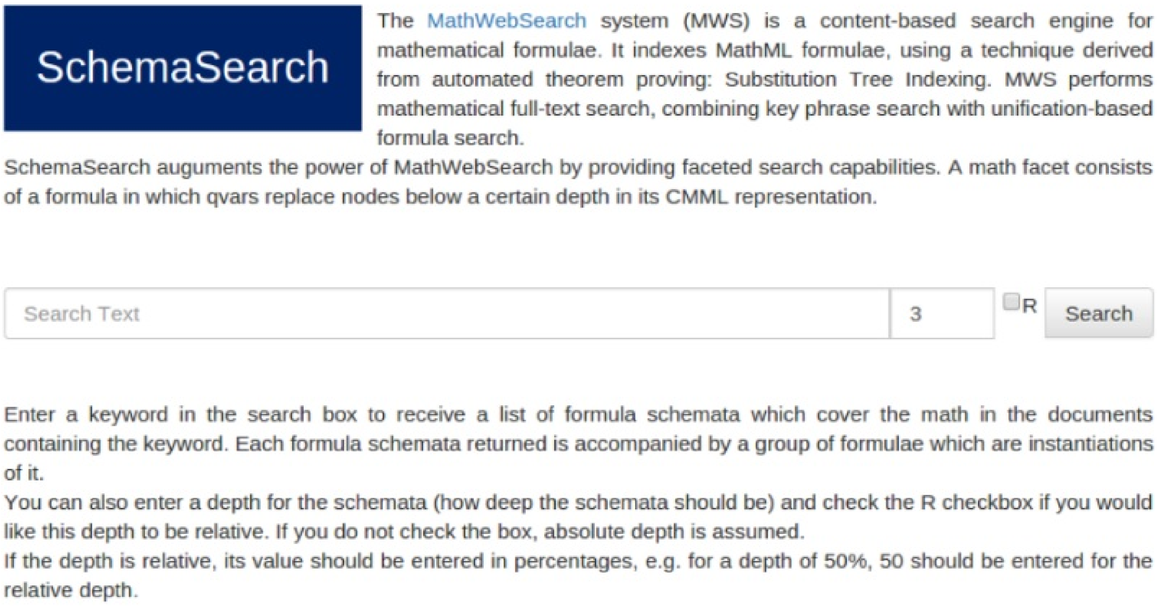
For a distributed instance of MathWebSearch we have the following setup: All nodes in the network of MathWebSearch machines run `mwsd` (the MathWebSearch daemon with the index and the URI database). MathWebSearch has only one `restd` daemon, which resides on the master node. Upon receiving Q `restd` passes it to `mwsd` on the master node, which start DFS search on its substitution tree index fragment. Whenever `mwsd` hits a remote index node whose target sector is on a different machine, it forwards the respective subquery Q0 to the `mwsd` on the remote machine, while continuing processing on the current machine.

In result-unlimited queries, the master `mwsd` will just wait for the slave `mwsd` to return the results and aggregate that with its own result set. In result-limited cases, some of the slave's results may be irrelevant, since the result limit has already been reached. The tradeoffs and efficiency issues involved in such effects will still need to be investigated. Similarly, the top of the index tree (which resides on the master node) will receive much higher processing loads, possibly becoming a bottleneck to the overall system.

9. Frontend

For exampling the functionality of MathWebSearch we will continue by presenting two frontends based on its mechanism. The first one is a text-only search engine which returns the math from the matching documents, after running it through the schematizer (the core part of MWS). This is the demo for showcasing the schematization process. The second one is a direct application of the Schematizer into a Math Search Engine which is capable of mathematical faceted search. [HK15]

9.1. **SchemaSearch.** The SchemaSearch front-end provides just a textual search input field. It is intended for users who want an overview of the formulae contained in a corpus. As shown in figure 5, the user can enter a set of keywords for the query, as well as a schema depth, which defaults to 3. The maximum result size is not accessible to the user, to prevent abuses and reduce server load. There is also an "R" checkbox which specifies if the cutoff depth should be absolute or relative. If relative, the depth should be given in percentages [RH15].



SchemaSearch

The MathWebSearch system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML formulae, using a technique derived from automated theorem proving: Substitution Tree Indexing. MWS performs mathematical full-text search, combining key phrase search with unification-based formula search.

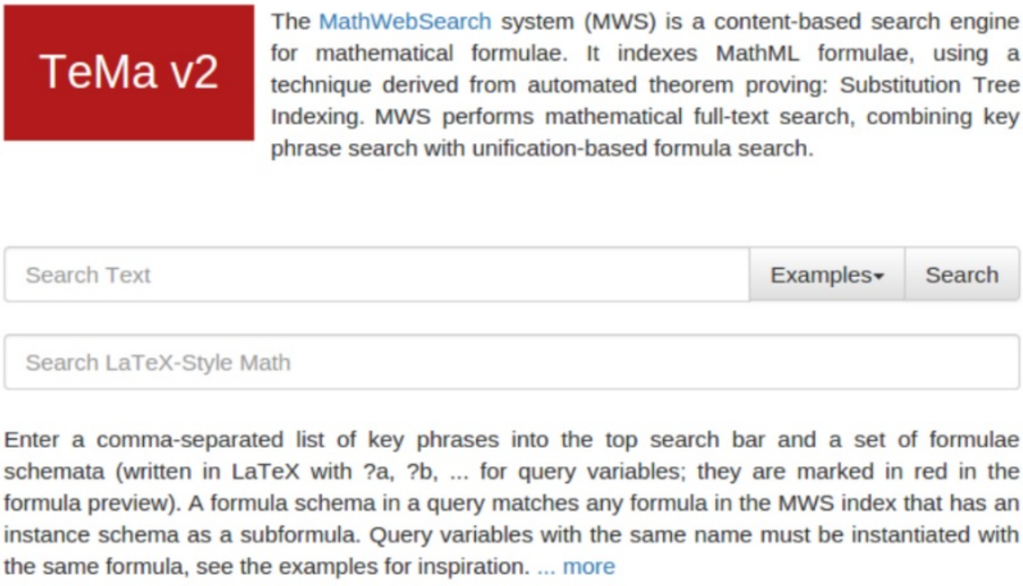
SchemaSearch augments the power of MathWebSearch by providing faceted search capabilities. A math facet consists of a formula in which qvars replace nodes below a certain depth in its CMML representation.

Enter a keyword in the search box to receive a list of formula schemata which cover the math in the documents containing the keyword. Each formula schemata returned is accompanied by a group of formulae which are instantiations of it.

You can also enter a depth for the schemata (how deep the schemata should be) and check the R checkbox if you would like this depth to be relative. If you do not check the box, absolute depth is assumed.

If the depth is relative, its value should be entered in percentages, e.g. for a depth of 50%, 50 should be entered for the relative depth.

Figure 5. SchemaSearch frontend



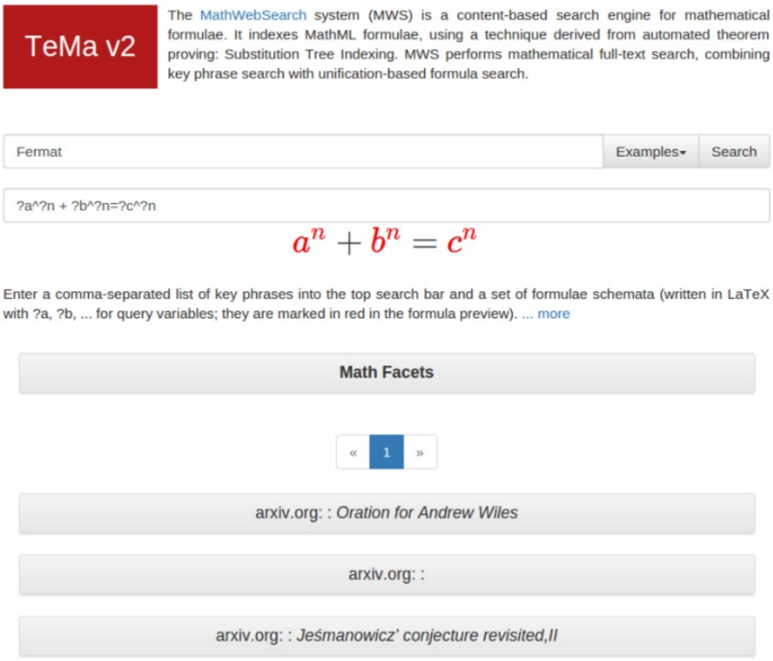
The MathWebSearch system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML formulae, using a technique derived from automated theorem proving: Substitution Tree Indexing. MWS performs mathematical full-text search, combining key phrase search with unification-based formula search.

Enter a comma-separated list of key phrases into the top search bar and a set of formulae schemata (written in LaTeX with ?a, ?b, ... for query variables; they are marked in red in the formula preview). A formula schema in a query matches any formula in the MWS index that has an instance schema as a subformula. Query variables with the same name must be instantiated with the same formula, see the examples for inspiration. ... more

FIGURE 6. TemaV2 frontend

9.2. **TemaV2 frontend.** The TemaV2 front-end extends TeMaSearch to be able to perform mathematical faceted search. It is intended for users who want to filter query results based on a given facet (formula schema in this case). The look and feel is similar to the previous version of TeMaSearch, as shown in Figure 6, where the first input field is used to specify keywords and the second one is used to specify LATEX-style formulae for the query. When returning results, a "Math Facets" menu will be presented to the user [RH15].

Figure 7 shows the results of a query for "Fermat" and $?a^{?n}+?b^{?n} =?c^{?n}$.



The MathWebSearch system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML formulae, using a technique derived from automated theorem proving: Substitution Tree Indexing. MWS performs mathematical full-text search, combining key phrase search with unification-based formula search.

$$a^n + b^n = c^n$$

Enter a comma-separated list of key phrases into the top search bar and a set of formulae schemata (written in LaTeX with ?a, ?b, ... for query variables; they are marked in red in the formula preview). ... more

arxiv.org: : *Oration for Andrew Wiles*

arxiv.org: :

arxiv.org: : *Jeśmanowicz' conjecture revisited,II*

FIGURE 7. TeMa v2 Query Results

10. MathWebSearch Instances and Corpora: arXiv, zbMath, Mizar

The MathWebSearch system has been used to supply search instances on various corpora of mathematical documents, we describe three here, others can be seen on `http://search.mathweb.org`.

10.1. arXiv search.

Begun on August 14, 1991, created by Paul Ginsparg, the "Cornell e-Print arXiv" (`http://arXiv.org`) is a repository of scientific papers and electronic preprints in fields of mathematics, computer science, physics, astronomy, biology and statistics or finance written in TEX/LATEX for an optimized transfer over the internet and an easily rendered client-side. In present the project is hosted by Cornell University and includes over a million articles and increases with around 8000 per month.

The KWARC group have converted by arXiv corpus into HTML5 [Sta+10] and harvested it for MathWebSearch. The instance at `http://arxivsearch.mathweb.org` indexes over 105 000 math-heavy papers. This subcorpus has also been used for the NTCIR Math Information Retrieval Challenges [AKO13; Aiz+14; Aiz+16].

10.2. zbMATH search.

Zentralblatt Math (zbMath) is a mathematical information service that curates a database of reviews and classifications (MSC, see [Msc]) for all articles in mathematics since the middle of the 19th century. The database currently contains 3.8 million reviews and grows at a rate of ca 120 000 reviews per year. The zbMATH portal at `https://zbmath.org/` offers a faceted search engine for reviews based on bibliographic metadata, MSC classification, and *formulae*. The latter is driven by MathWebSearch.

10.3. Search on a Formal Library of Mathematical Knowledge.

We have converted the Mizar Mathematical Library [MizLib] to the OMDoc/MMT format and indexed the results in MathWebSearch [Ian+13], enabling formula-based search on a formal knowledge of contains over 1000 articles with over 50000 theorems and over 10000 definitions encoded in first-order logic based on Tarski-Grothendiek set theory. As the data is fully formal, we can identify universal variables in the corpus (explicitly marked up universal quantifications) and search for applicable theorems, e.g. we can issue queries like formula query: $\int_a^b |V(t)I(t)|dt \leq ?R$, and the OpenDreamKit system will return Hölder's equation as one of the hits:

Theorem 17. (Hölder's Inequality)
If f and g are measurable real functions, $l, h \in \mathbb{R}$, and $p, q \in [0, \infty)$, such that $1/p + 1/q = 1$, then

$$\int_l^h |f(x)g(x)|\,dx \leq \left(\int_l^h |f(x)|^p\,dx\right)^{\frac{1}{p}} \left(\int_l^h |g(x)|^q\,dx\right)^{\frac{1}{q}} \tag{1}$$

11. Conclusion

We have presented a search engine capable of dealing with pure mathematical queries but also with text and math queries. The MathWebSearch web service is open source software released under the GNU public license. The code is available from the developer portal [MWSa]. Search front-ends for various corpora and applications are referenced on the MathWebSearch project page [MWSb]. In contrast to other approaches, MathWebSearch uses the full content structure of formulae and is easily extensible to other content-oriented formats. Our evaluations show that query times are low and essentially constant in index/harvest size, so that a search engine can scale up to web proportions. Contrary to our expectations, index size is linear in harvest size for the arXiv corpus, which transcends the main memory limits of standard servers. Therefore, implementing parallelization/distribution strategies is a priority.

Adaptation to OpenDreamKit. For utilization in the OpenDreamKit VRE we need to solve two development problems:

ODK1 for all types of data (D), representations of knowledge (K), and software (S) in OpenDreamKit we need MathWebSearch harvesters – i.e. systems that transform the D/K/S strutures into MathWebSearch harvests (see Section 3).

ODK2 for all interfaces in the OpenDreamKit VRE – see e.g. [Koh16] – we need to embed search front-ends that make use of the user context to allow precise and convenient entry of queries and selection/ranking of search results. This also means that we will integrate novel features like faceted search for mathematics, e.g. along the lines of [HK16]

Work on these has already begun. For instance we have parsed the formulae in the "Open Encyclopedia of Integer Sequences" (OEIS, see [Inc] and Task **T6.6** in the OpenDreamKit proposal) and harvested them for MathWebSearch (**ODK1**) and built an OEIS-specific front-end (**ODK2**), see [LK16]. Further more we are currently working on exporting the D/K/S structures in the LMFDB, GAP, and SageMath into MathWebSearch-harvestable OMDoc/MMT format (**ODK1**).

References

[Aiz+14] Akiko Aizawa et al. "NTCIR-11 Math-2 Task Overview". In: *NTCIR 11 Conference*. Ed. by Noriko Kando, Hideo Joho, and Kazuaki Kishida. Tokyo, Japan: NII, Tokyo, 2014, pp. 88–98. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/OVERVIEW/01-NTCIR11-OV-MATH-AizawaA.pdf.

[Aiz+16] Akiko Aizawa et al. "NTCIR-12 MathIR Task Overview". In: *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*. Ed. by Noriko Kando, Tetsuya Sakai, and Mark Sanderson. Tokyo, Japan: NII, Tokyo, 2016, pp. 299–308. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/OVERVIEW/01-NTCIR12-OV-MathIR-ZanibbiR.pdf.

[AKO13] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. "NTCIR-10 Math Pilot Task Overview". In: *NTCIR Workshop 10 Meeting*. Ed. by Noriko Kando and Kazuaki Kishida. Tokyo, Japan: NII, Tokyo, 2013, pp. 1–8. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/OVERVIEW/01-NTCIR10-OV-MATH-AizawaA.pdf.

[Ela] *Elastic Search*. Accessed: June 05, 2016. URL: http://www.elasticsearch.org/.

[HK15] Radu Hambasan and Michael Kohlhase. "Faceted Search for Mathematics". In: *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*. Ed. by Ralph Bergmann, Sebastian Görg, and Gilbert Müller. Oct. 2015, pp. 33–44. URL: http://ceur-ws.org/Vol-1458/D05_CRC1_Hambasan.pdf.

[HK16] Radu Hambasan and Michael Kohlhase. "Faceted Search for Mathematics". In: *MACIS 2015: Sixth International Conference on Mathematical Aspects of Computer and Information Sciences*. LNAI. in press. Springer Verlag, 2016. URL: http://kwarc.info/kohlhase/papers/macis15.pdf.

[Ian+13] Mihnea Iancu et al. "The Mizar Mathematical Library in OMDoc: Translation and Applications". In: *Journal of Automated Reasoning* 50.2 (2013), pp. 191–202. DOI: 10.1007/s10817-012-9271-4. URL: https://svn.omdoc.org/repos/latin/public/MizarOMDocAppl.pdf.

[Inc] OEIS Foundation Inc., ed. *The On-Line Encyclopedia of Integer Sequences*. URL: http://oeis.org (visited on 05/28/2013).

[Koh16] Michael Kohlhase. *Active/Structured Documents Requirements and existing Solutions*. Deliverable D4.2. OpenDreamKit, 2016.

[KŞ06] Michael Kohlhase and Ioan Şucan. "A Search Engine for Mathematical Formulae". In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*. Ed. by Tetsuo Ida, Jacques Calmet, and Dongming Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: http://kwarc.info/kohlhase/papers/aisc06.pdf.

[LK16] Enxhell Luzhnica and Michael Kohlhase. "Formula Semantification and Automated Relation Finding in the OEIS". In: *Mathematical Software - ICMS 2016 - 5th International Congress*. LNCS. accepted. Springer, 2016.

[MizLib] *Mizar Mathematical Library*. URL: http://www.mizar.org/library (visited on 09/27/2012).

[MK14] Radu Hambasan Michael Kohlhase Corneliu Prodescu. "MathWebSearch at NTCIR-11". In: 2014. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/05-NTCIR11-MATH-HambasanR.pdf.

[Msc] *Mathematics Subject Classification MSC2010*. 2010. URL: `http://msc2010.org` (visited on 11/16/2011).

[Mws] *MWS Harvest*. Accessed: June 02, 2016. URL: `https://trac.mathweb.org/MWS/wiki/MwsHarvest`.

[MWSa] *MathWebSearch*. URL: `https://github.com/KWARC/MWS` (visited on 06/11/2016).

[MWSb] *MathWebSearch – Searching Mathematics on the Web*. URL: `http://search.mathweb.org` (visited on 06/11/2016).

[PK11] Corneliu C. Prodescu and Michael Kohlhase. "MathWebSearch 0.5 - Open Formula Search Engine". In: *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) Conference Proceedings*. Sept. 2011. URL: `https://svn.mathweb.org/repos/mws/doc/2011/newmws/main.pdf`.

[RH15] Michael Kohlhase Radu Hambasan. "Faceted Search for Mathematics". B. Sc. Thesis. Jacobs University Bremen, 2015.

[Sta+10] Heinrich Stamerjohanns et al. "Transforming large collections of scientific publications to XML". In: *Mathematics in Computer Science* 3.3 (2010): *Special Issue on Authoring, Digitalization and Management of Mathematical Knowledge*. Ed. by Serge Autexier, Petr Sojka, and Masakazu Suzuki, pp. 299–307. URL: `http://kwarc.info/kohlhase/papers/mcs10.pdf`.