Faceted Search for Mathematics

Radu Hambasan   Michael Kohlhase

Computer Science, Jacobs University Bremen

Abstract. Faceted search is one of the most practical ways to browse a large corpus of information. Information is categorized automatically for a given query and the user is given the opportunity to further refine his/her query. Many search engines offer a powerful faceted search engine, but only on the textual level. Faceted Search in the context of Math Search is still unexplored territory.

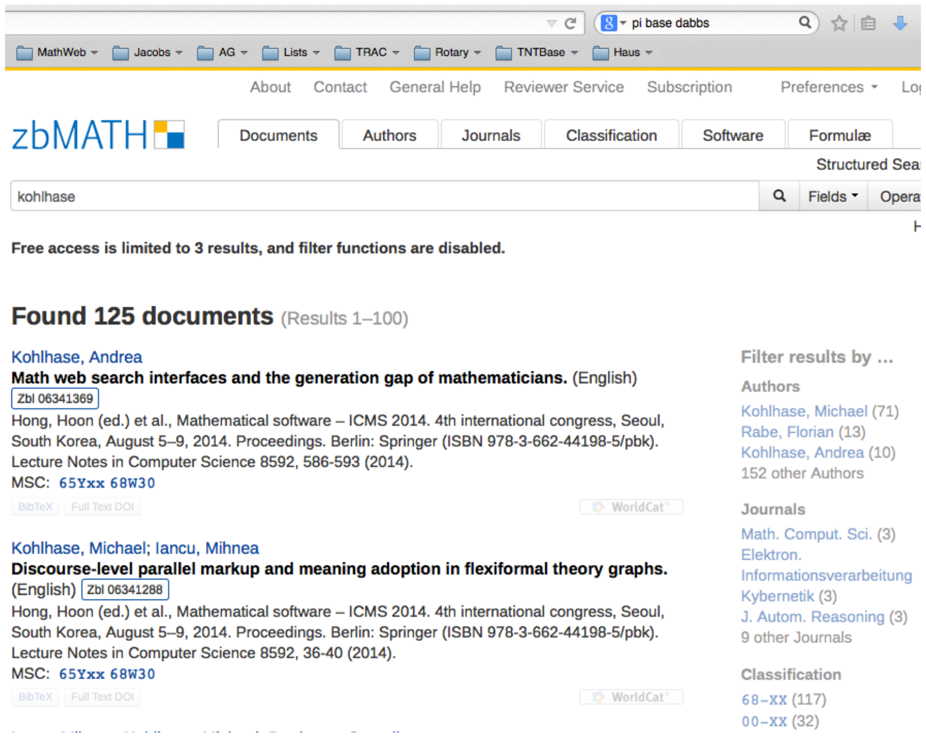
In this paper, we describe one way of solving the faceted search problem in mathematics: by extracting recognizable formula schemata from a given set of formulae and using these schemata to divide the initial set into formula classes. Also, we provide a direct application by integrating this solution with existing services.

1 Introduction

Search engines have become the prevalent tool for exploring the ever growing trove of digital data on the Internet. Although text search engines (e.g. Google or DuckDuckGo) are sufficient for most uses, they are limited when it comes to finding scientific content. STEM documents (Science, Technology, Engineering and Mathematics) contain mathematical formulae which cannot be properly indexed by a text search engine as they are structured expressions of operators (fractions, square-roots, subscripts and superscripts) and tokens.

A good math search engine is needed by several user groups. One user group would be an airline manufacturer, searching for formulae in their engineering whitepapers. In the case of research centers, like CERN, valuable time would be saved if scientists would have a fast, reliable and powerful math search engine to analyse previous related work. Still another user group is represented by university students who would be empowered by search engines, when their textbooks are not limited to text, but also include formulae. For all these applications, we first need a strong math search engine and second, a large corpus of math to index. Correspondingly, Math Information Retrieval is a small but vibrant research topic, we refer the reader to the recent Math-2 Task [1] at the NTCIR-11 IR Evaluation Campaign for an overview.

The Cornell e-Print Archive arXiv [3], is an example of such a corpus, containing over a million STEM documents from various scientific fields (Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics). Zentralblatt Math (ZBMath) [17] has abstracts/reviews of all published papers (currently 3 .5 million) since 1859. A search engine for these must provide an expressive query language and query-refining options to be able to retrieve useful information.



Free access is limited to 3 results, and filter functions are disabled.

Found 125 documents (Results 1–100)

Kohlhase, Andrea
Math web search interfaces and the generation gap of mathematicians. (English) Zbl 06341369
Hong, Hoon (ed.) et al., Mathematical software – ICMS 2014. 4th international congress, Seoul, South Korea, August 5–9, 2014. Proceedings. Berlin: Springer (ISBN 978-3-662-44198-5/pbk). Lecture Notes in Computer Science 8592, 586-593 (2014).
MSC: 65Yxx 68W30
BibTeX Full Text DOI WorldCat

Kohlhase, Michael; Iancu, Mihnea
Discourse-level parallel markup and meaning adoption in flexiformal theory graphs. (English) Zbl 06341288
Hong, Hoon (ed.) et al., Mathematical software – ICMS 2014. 4th international congress, Seoul, South Korea, August 5–9, 2014. Proceedings. Berlin: Springer (ISBN 978-3-662-44198-5/pbk). Lecture Notes in Computer Science 8592, 36-40 (2014).
MSC: 65Yxx 68W30
BibTeX Full Text DOI WorldCat

Filter results by ...
Authors
Kohlhase, Michael (71)
Rabe, Florian (13)
Kohlhase, Andrea (10)
152 other Authors
Journals
Math. Comput. Sci. (3)
Elektron. Informationsverarbeitung Kybernetik (3)
J. Autom. Reasoning (3)
9 other Journals
Classification
68-XX (117)
00-XX (32)

Fig. 1: Faceted Search in ZBMath

Zentralblatt Math also provides a powerful search engine called “structured search”. This engine is also capable of faceted search[1]. Figure 1 shows a typical situation: a user searched for a keyword (here an author name) and the faceted search generated links for search refinements (the **facets**) on the right. Currently, facets for the primary search dimensions are generated – authors, journals, MSC (Mathematics Subject Classification) [13]. This allows the user to further explore the result space, even without knowing in advance the specifics of what he/she is looking for. Unfortunately these facets are all metadata-driven and not specific to mathematics – the MSC facet is an exception, but it is rather vague because it will only provide information about the field of mathematics to which an article belongs. If the authors use formulae from another field in their paper, the results will suffer a drop in relevance.

[1] Faceted search originates from image search [16] and the results showed that users prefer a category-based approach to searching, even if the interface is initially unfamiliar.

With the work reported in this paper we try to lift this limitation by computing “formula facets” consisting of a set of formula schemata generated to further disambiguate the query by refining it in a new dimension. For instance, for the query in figure 1, we could have the facets in Figure 2, which allows the user to drill in on *i*) variation theory and minimal surfaces, *ii*) higher-order unification, and *iii*) type theory. The red identifiers (prefixed with a question mark), stand for query variables, their presence making the results **formula schemata**.

$$\int_{?M} ?\Phi(d_p ?f) dvol$$

$$\lambda ?X.h(H^1 ?X) \cdots H^n ?X$$

$$\frac{?\Gamma \vdash ?A \gg ?\alpha}{?D}$$

Fig. 2: Formula Facets

The formula schemata in figure 2 were manually created to judge the feasibility of using schemata as recognizable user interface entities, but for an application we need to generate them automatically from the query. Moreover, each schema should further expand to show the formula class it represents. Formula classes would consist of all formulae sharing the same schema. This is the algorithmic problem we explore in this paper.

After reviewing the preliminaries in Section 2, we present the schematization algorithm in Section 3 and discuss its implementation in Section 4. Section 5 addresses first results in finding cutoff heuristics, the main cognitively relevant parameter in the schematization algorithm. Section 6 discusses applications beyond the faceted math search problem addressed in this paper and sketches future work. Section 7 concludes the paper.

2 Preliminaries

We will now present the systems on which our work is based:

– **MathWebSearch** provides the necessary index structure for schema search.
– **Elasticsearch** provides hits in response to text queries, as well as run aggregations on the hits. These hits represent formulae to be schematized.
– **arXiv** provides a large corpus of mathematical documents that we can index and run our system on.
– **LaTeXML** converts LaTeX expressions to MathML.

As discussed in Section 1, the goal of this project is to develop a scalable formula schematization engine, capable of dividing a set of query hits into classes, according to the generated formula schemata.

We have set the following end-user requirements for our system:

R1. it should be able to generate formula schemata from a given set of formulae and the resulting schemata should be easily recognizable by the user.
R2. it should be able to classify the given set of formulae according to the generated schemata.
R3. the system should be massively scalable, i.e. capable of answering queries with hundreds of thousands of formulae in a matter of seconds.

At its core, the MathWebSearch [8] system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML [12] formulae, using a technique derived from automated theorem proving: Substitution Tree Indexing [7]. Recently, it was augmented with full-text search capabilities, combining keyword queries with unification-based formula search. The engine serving text queries is Elasticsearch (below). From now on, in order to avoid confusion, we will refer to the core system (providing just formula query capability) as MWS and to the complete service (MWS + Elasticsearch) as TeMaSearch (Text + Math Search).

Internal to MWS, 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 trie of substitution paths.

To facilitate fast retrieval, MWS 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 [11] database.

MathWebSearch exposes a RESTful HTTP API which accepts XML queries. A valid query must obey the Content MathML format, potentially augmented with *qvar* variables which match any subterms. A *qvar* is a wildcard in a query, with the restriction that if two *qvar*s have the same name, they must be substituted in the same way.

Elasticsearch [5] is a powerful and efficient full text search and analytics engine, built on top of Lucene [2]. 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 RESTful 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.

arXiv is a repository of over one million publicly accessible scientific papers in STEM fields. For the NTCIR-11 challenge [8], MWS indexed over 8.3 million paragraphs (totaling 176 GB) from arXiv. We will base our queries on this large index, because it provides a rich database of highly relevant formulae. Moreover, Elasticsearch will have more formulae on which it can run aggregations, also leading to more relevant results.

An overwhelming majority of the digital scientific content is written using LaTeX or TeX [10], 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.

An open source example of such a conversion engine is LaTeXML [14]. The MathWebSearch project relies heavily on it, to convert arXiv documents from LaTeX to XHTML which is later indexed by MWS. 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 Content 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 3 illustrates the parallel markup for $\frac{2}{x+3}$. 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.

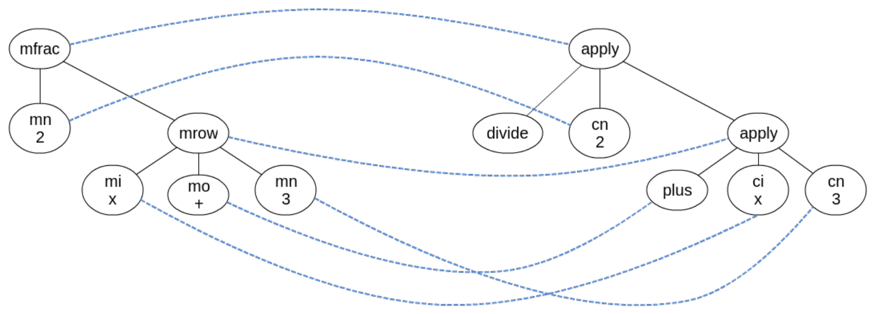


Fig. 3: The CMML/PMML Parallel Markup

3 Schematization of Formula Sets

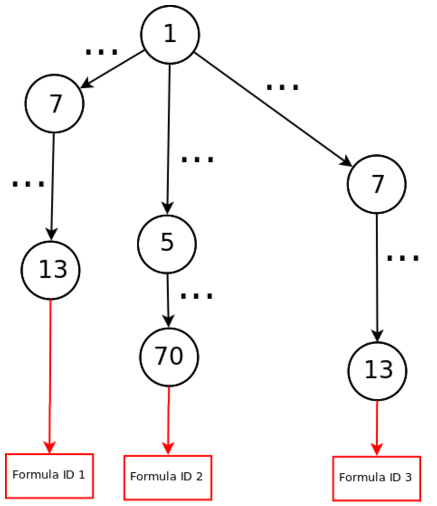
In this section, we provide a theoretical approach to the problem of generating formula schemata, by formalizing the problem and describing an efficient algorithm to solve it. First we formulate the problem at hand more carefully.

Definition 1. *Given a set $\mathcal{D}$ of documents (fragments) – e.g. generated by a search query, a* ***coverage*** *$0 < r \leq 1$, and a* ***width*** *n, the* ***Formula Schemata Generation*** *(FSG) problem requires generating a set $\mathcal{F}$ of at most n formula schemata (content MathML expressions with* `qvar` *elements for query variables), such that $\mathcal{F}$ covers $\mathcal{D}$ with coverage r.*

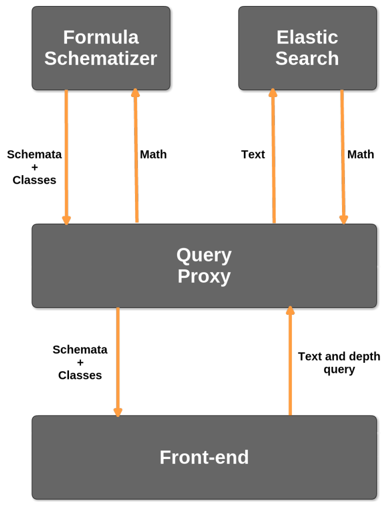
Definition 2. *We say that a set $\mathcal{F}$ of formula schemata* ***covers*** *a set $\mathcal{D}$ of document fragments, with* ***coverage*** *r, iff at least $r \cdot |\mathcal{D}|$ formulae from $\mathcal{D}$ are an instance $\sigma(f)$ of some $f \in \mathcal{F}$ for a substitution σ.*

The algorithm that we present requires a MWS index of a corpus. Given such an index, and a set $\mathcal{D}$ of formulae (as CMML expressions), we can find the set $\mathcal{F}$ in the following way:

1. Parse the given CMML expressions similarly to MWS queries, to obtain their encoded DFS representations.
2. Choose a reasonable cutoff heuristic, see below.
3. Unify each expression with the index, up to a given threshold (given by the above heuristic).
4. Keep a counter for every index path associated with the unifications. Since we only match up to a threshold, some formulae will be associated with the same path (excluding the leaves). We increase the counter each time we find a path already associated with a counter.
5. Sort these path-counter pairs by counter in descending order and take the first n (n being the width required by the FSG).
6. If the threshold depth was smaller than a formula's expression depth, the path associated with it will have missing components. We replace the missing components with `qvars` to generate the schema and return the result set.



(a) Index simplified at depth 1



(b) FS Engine Architecture

Fig. 4: Aspects of the Formula Search System

Figure 4a shows a MWS index with encoded expressions which were simplified at depth 1. This means that the Content MathML representation of the formulae was truncated at depth 1 and then encoded in the index, resulting in a "simplified index".

The formulae's paths represent their depth-first traversal. Every formula can be reconstructed given its path in the index. The circles represent index nodes and the number inside represents the token's ID. When we reach a leaf node, we completely described a formula. This is encoded in the leaf node by an ID, which can be used to retrieve the formula from the database. The length of the arrows symbolizes the depth of the omitted subterms (for higher depths, we have longer arrows). Notice how both formula with ID 1 and formula with ID 3 show the same "path" when ignoring subterms below a cutoff depth (the simplification depth), which in this case is 1.

4 Implementation

In this section, we explain the key details of the formula classifier's implementation, the overall system architecture, as well as the challenges and trade-offs associated with the taken design decisions.

Design Overview The full faceted search system comprises of the following components: the Formula Schematizer 4, Elasticsearch, a proxy to mediate communication between the Schematizer and Elasticsearch and a Web front-end. The architecture of the system is shown in Figure 4b.

Once the user enters a query (which consists of keywords and a depth), the front-end forwards the request to a back-end proxy. The proxy sends the text component of the query to Elasticsearch and receives back math contained in matching documents. Afterwards, it sends the retrieved math and the depth parameter (from the original query) to the Schematizer. The Schematizer will respond with a classification of the math in formula classes, as well as the corresponding schema for each class. Finally, the proxy forwards the result to the front-end which displays it to the user.

In the following sections, we will explain the core components of the system in detail and describe the challenges faced during implementation.

The Formula Schematizer The Schematizer is the central part of our system. It receives a set of formulae in their Content MathML representation, generates corresponding formula schemata and classifies the formulae according to the generated schemata. It provides an HTTP endpoint and is therefore self-contained, i.e. it can be queried independently, not only as part of the faceted search system. As a consequence, the Schematizer displays a high degree of versatility, and can be integrated seamlessly with other applications.

Although our algorithm works well in theory, we needed to adapt it considering various MathWebSearch implementation details, e.g. the index is read-only (therefore we cannot store extra data into the index nodes). Therefore, the overall idea/theory is the same, but now we take the following shortcut: instead of unifying every formula with the index, we just pretend we do and instead generate a “signature” for each formula. This signature is the path shown in Figure 4a. We use the MathWebSearch encoding for MathML nodes, where each node is assigned an integer ID based on its tag and text content. If the node is not a leaf, then only the tag is considered. The signature will be a vector of integer IDs, corresponding to the pre-order traversal of the Content MathML tree.

Naturally, the signature depends on the depth chosen for the cutoff heuristic. At depth 0, the signature consists only of the root token of the Content MathML expression. At full depth (the maximum depth of the expression), the signature is the same as the depth-first traversal of the Content MathML tree.

Based on these computed signatures, we divide the input set of formulae into formula classes, i.e. all formulae with the same signature belong to the same class. For this operation we keep an in-memory hash table, where the keys are given by the signatures and the values are sets of formulae which have the signature key. After filling the hash table, we sort it according to the number of formulae in a given class, since the signatures which cover the most formulae should come at the beginning of the reported result.

The Schematizer caller can place an optional limit on the maximum number of schemata to be returned. If such a limit was specified, we apply it to our sorted list of signatures and take only the top ones.

As a last step, we need to construct Content MathML trees from the signatures, to be able to show the schemata as formulae to the user. We are able to do this because we know the arity of each token and the depth used for cutoff. The tree obtained after the reconstruction might be incomplete, so we insert query variables in place of missing subtrees. We finally return these Content MathML trees with query variables (the formula schemata), together with the formulae which they cover.

Presentation by Replacement After obtaining the schemata and formula classes, we need to be able to display the result to the user. One possibility would be to have the Schematizer return Content MathML expressions for the schemata and use an XSL stylesheet [15] to convert them to Presentation MathML. This approach would unfortunately generate unrecognizable schemata due to the inherent ambiguity of CMML. For instance, a csymbol element can be represented in several different ways depending on the notation being used. Additionally, we cannot reliably foresee all possible rules that should be implemented in the stylesheet and as a consequence some formulae will be wrongly converted.

Since the XSL conversion is unreliable, we will make use of the cross-reference system provided by LaTeXML, as discussed before. Instead of returning Content MathML expressions, the Schematizer will use the first formula in each class as a template and “punch holes into it”, effectively returning the ID of the nodes that are to be substituted with query variables. We will use this IDs to replace the referenced PMML nodes with `<mi>` nodes representing the qvars.

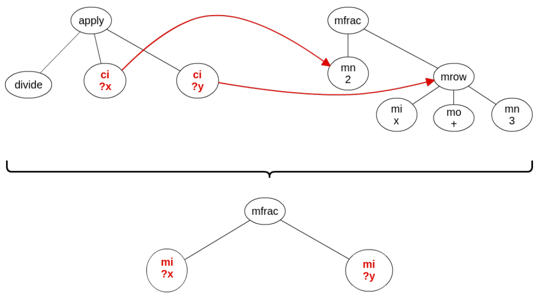


Fig. 5: Presentation by Replacement

Figure 5 shows the presentation by replacement technique for a given schema. The Schematizer returned a schema which was checked against the first formula in its class ($\frac{2}{x+3}$) to generate two substitutions, marked with red on the left side. Due to the cross-reference system provided by LaTeXML, we are able to find the corresponding PMML elements and substitute them with `<mi>` tokens. The result will be displayed to the user as $\frac{?x}{?y}$.

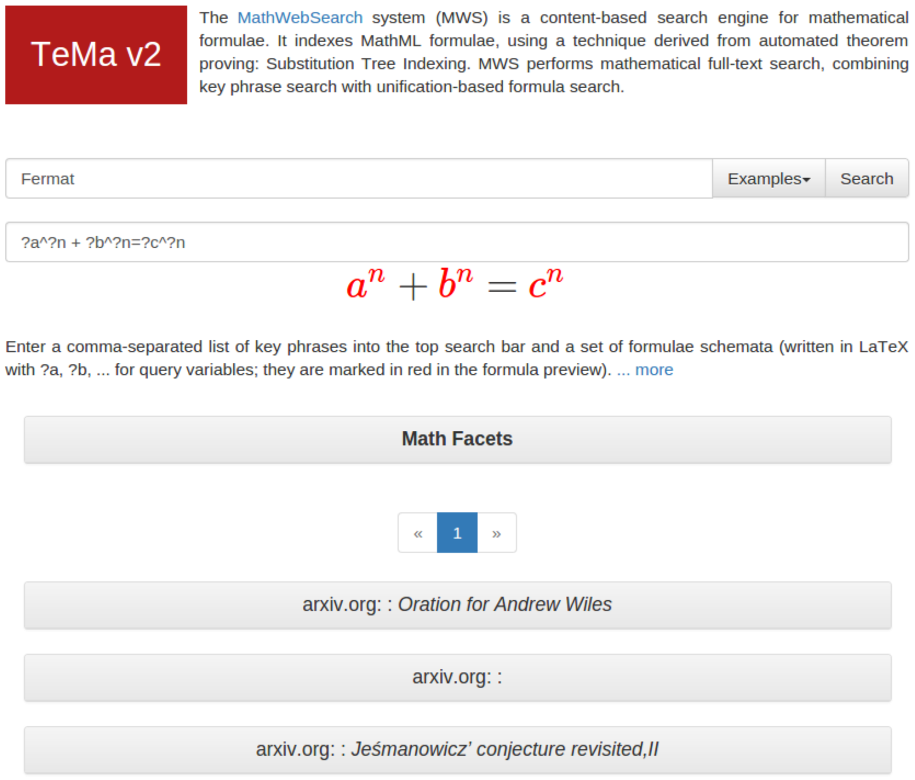
Performance We designed the Schematizer to be a very lightweight daemon, both as memory requirements and as CPU usage. To test if we achieved this goal, we benchmarked it on a server running Linux 3.2.0, with 10 cores (Intel Xeon CPU E5-2650 2.00GHz) and 80 GB of RAM.

We obtained the 1123 expressions to be schematized by querying Elasticsearch with the keyword "Fermat". While the overall time taken by the faceted search engine was around 5 seconds, less than a second was spent in the Schematizer. Also, the CPU utilized by the Schematizer never rose higher than 15% (as indicated by the top utility). Asymptotically, the algorithm would run in $O(N)$ time, where N is the number of input formulae. We are able to reach linear time performance, because each formula is processed exactly once and the signature is stored in a hash table, as discussed in Section 4.

Due to its design principles, the Schematizer is almost indefinitely scalable, because it does not require shared state between formulae and can therefore be implemented as a MapReduce [4] job, where mappers compute the signature of assigned formulae and reducers assemble the signature hash table. However, the unification algorithm currently used by MathWebSearch is linear is in the number of nodes of the Content MathML expression that is being unified. Therefore, the current search engine implementation would pose challenges for scalability, although the Schematizer itself will be able to extend easily.

The Front-End We have integrated the Schematizer into a Math Search Engine which is capable of mathematical faceted search.

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. Figure 6 shows the results of a query for “Fermat” and $?a^{?n} + ?b^{?n} = ?c^{?n}$. Besides the regular TeMaSearch results, the user is also presented with a “Math Facets” section.


TeMa v2

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.

Fermat

?a^?n + ?b^?n=?c^?n

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

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

Math Facets

arxiv.org: : *Oration for Andrew Wiles*

arxiv.org: :

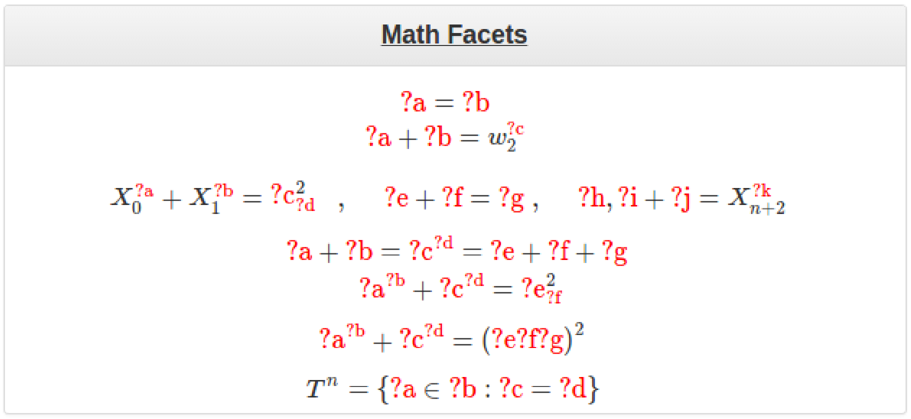
arxiv.org: : *Jeśmanowicz’ conjecture revisited,II*

Fig. 6: TeMa v2 Query Results

When the “Math Facets” section is expanded the user can see the top 10 schemata (ranked with respect to their coverage), as shown in Figure 7. We have also implemented a “search-on-click” functionality that allows the user to do a fresh search using the clicked schema and the initial keyword, which effectively filters the current results.

5 Finding a Cutoff Heuristic

To generate formula schemata, we must define a “cutoff heuristic”, which tells the program when two formulae belong to the same schema class. If there is


Math Facets

$$?a = ?b$$
$$?a + ?b = w_2^{?c}$$
$$X_0^{?a} + X_1^{?b} = ?c^2_{?d} \quad , \quad ?e + ?f = ?g \, , \quad ?h, ?i + ?j = X^{?k}_{n+2}$$
$$?a + ?b = ?c^{?d} = ?e + ?f + ?g$$
$$?a^{?b} + ?c^{?d} = ?e^2_{?f}$$
$$?a^{?b} + ?c^{?d} = (?e?f?g)^2$$
$$T^n = \{?a \in ?b : ?c = ?d\}$$

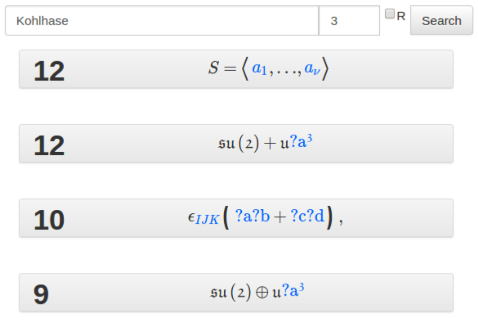
Fig. 7: Math Facets in TeMa v2

no heuristic, two formulae would belong to the same class, only if they were identical. However, we want formulae that have something in common to be grouped together, even if they are not perfectly identical. The cutoff heuristic is the parameter in the schematization algorithm that determines the suitability of schemata for the various information access tasks at hand. As this is essentially a user-driven, cognitive task it is not a priori clear what cutoff heuristics will perform best.

To explore the space of heuristics, we have implemented a special front-end and used that to evaluate heuristics for the math search task discussed above. As a proper user-level evaluation was beyond the scope of this paper, we have implemented various heuristics and discussed them with the ZBMath group in the context of the ZBMath corpus, this led to the development of the dynamic cutoff heuristic presented at the end of this section.

A Schema Evaluation Front-End 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 8a, 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.

Figure 8a shows the formula schemata at depth 3, over the arXiv corpus, for a query containing the keyword “Kohlhase”. By default, the top 40 schemata are shown, but the results are truncated for brevity. The bold number on the left side of each result item indicates how many formulae are present in each formula class. For instance, the third schema represents a formula class containing 10 formulae. The entities marked in blue are query variables (qvars).



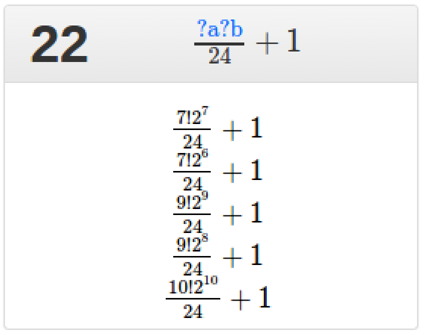
12 $S = \langle a_1, \ldots, a_\nu \rangle$

12 $\mathfrak{su}(2) + \mathfrak{u}?a^3$

10 $\epsilon_{IJK}\left(?a?b + ?c?d \right),$

9 $\mathfrak{su}(2) \oplus \mathfrak{u}?a^3$

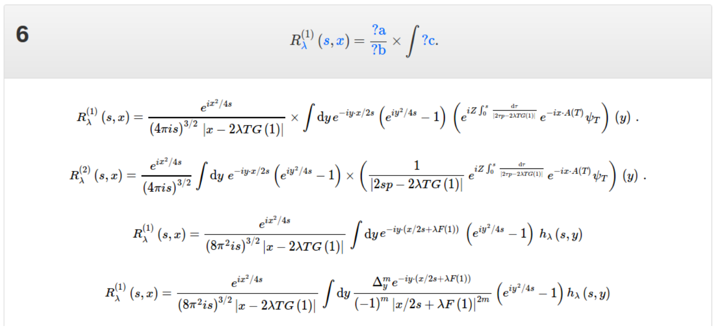
(a) Faceted Results at depth 3



22 $\frac{?a?b}{24} + 1$

$$\frac{7!2^7}{24} + 1$$
$$\frac{7!2^6}{24} + 1$$
$$\frac{9!2^9}{24} + 1$$
$$\frac{9!2^8}{24} + 1$$
$$\frac{10!2^{10}}{24} + 1$$

(b) Expansion of a Formula Class 1



6 $R^{(1)}_\lambda (s,x) = \frac{?a}{?b} \times \int ?c.$

$$R^{(1)}_\lambda (s,x) = \frac{e^{ix^2/4s}}{(4\pi is)^{3/2}\,|x - 2\lambda TG(1)|} \times \int \mathrm{d}y\, e^{-iy\cdot x/2s}\left(e^{iy^2/4s} - 1\right)\left(e^{iZ\int_0^s \frac{\mathrm{d}\tau}{|2\tau p - 2\lambda TG(1)|}}\, e^{-ix\cdot A(T)}\psi_T\right)(y)\,.$$

$$R^{(2)}_\lambda (s,x) = \frac{e^{ix^2/4s}}{(4\pi is)^{3/2}} \int \mathrm{d}y\; e^{-iy\cdot x/2s}\left(e^{iy^2/4s} - 1\right) \times \left(\frac{1}{|2sp - 2\lambda TG(1)|}\, e^{iZ\int_0^s \frac{\mathrm{d}\tau}{|2\tau p - 2\lambda TG(1)|}}\, e^{-ix\cdot A(T)}\psi_T\right)(y)\,.$$

$$R^{(1)}_\lambda (s,x) = \frac{e^{ix^2/4s}}{(8\pi^2 is)^{3/2}\,|x - 2\lambda TG(1)|} \int \mathrm{d}y\, e^{-iy\cdot(x/2s + \lambda F(1))}\left(e^{iy^2/4s} - 1\right) h_\lambda (s,y)$$

$$R^{(1)}_\lambda (s,x) = \frac{e^{ix^2/4s}}{(8\pi^2 is)^{3/2}\,|x - 2\lambda TG(1)|} \int \mathrm{d}y\, \frac{\Delta^m_y e^{-iy\cdot(x/2s + \lambda F(1))}}{(-1)^m\,|x/2s + \lambda F(1)|^{2m}}\left(e^{iy^2/4s} - 1\right) h_\lambda (s,y)$$

(c) Expansion of a Formula Class 2

Fig. 8: Generated Schemata

Figure 8b shows the expansion of a formula class. There are 22 formulae in the class given by this particular math schema, as indicated by the count on the left upper side, out of which only ten are shown. We can see 2 unnamed query variables marked with blue as $?a$ and $?b$. By seeing the schema, the user can form an impression about the general structure of the formulae from that class. After expanding the class, the listing of concrete formulae appears. If the user clicks on one of them, he is redirected to the source document from which that expression was extracted.

Another class expansion which showcases the schematization can be seen in Figure 8c. By seeing this schema, the user can abstract away the complexity of the formulae and obtain a "summary" of the meaning behind it. Also, by expanding the class he can explore several related formulae easily, because they are grouped together.

Dynamic Cutoff We have experimented with several possibilities for the heuristic and found out that a *dynamic cutoff* which preserves the operators leads to more intuitive results. We can identify the operators by looking at the first child of the `apply` token in the CMML tree. The user is given the option to have an absolute (fixed) or relative (depending on the depth of the CMML tree) cutoff for the operands.

Figure 9 illustrates this heuristic at depth 1. The divide element was kept, because it was the first child of apply, while the other children were removed. If we were to use a depth of 2, the plus element would also be included in the schema.

This heuristic is not simply keeping another tree node uncut. If the current node is an operator, it can also have multiple levels of children and therefore we need to keep that entire subtree from being cut. What this means, is that the cutoff depth can vary significantly, depending on how deep the operator's subtree is.

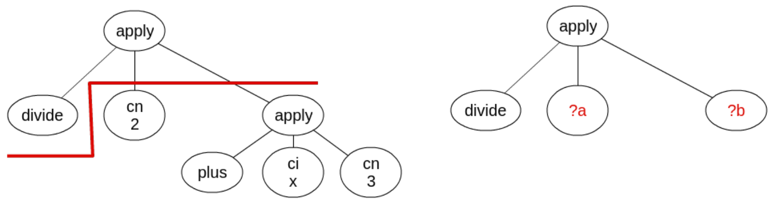


Fig. 9: Dynamic Cutoff

6 Applications and Future Work

One improvement angle that can be worked on is the ranking of the schemata. We have used a simple method, ranking them in decreasing order of coverage, thus having the schema with most formulae in its class on the first place. However, this is not always a good approach. When users look at the facets, it is usually because they were not able to find what they were looking for (because the result set is too large). The first schemata cover most of the formulae users have already looked at, so they are not of interest. However, the last schemata are not of interest either, because they typically only cover very rare formulae (1-2 occurrences). An alternative ranking approach might place the *medium-coverage* schemata first, then the top-coverage and then the low-coverage. In order to define precisely what is the range for medium-coverage, further research is required.

One other application of the faceted search engine can be providing mathematical definitions with the help of NNexus [6]. NNexus is an auto-linker for mathematical concepts from several encyclopedias, e.g. PlanetMath, Wikipedia. Assuming we are able to generate relevant schemata in response to keyword queries, we can target the faceted search engine with all the concepts stored by NNexus and store a schema for each such concept. Afterwards, for a given query, we can obtain the schema and check it against our stored set of schemata. If we find it, we can link the given expression to its mathematical definition. Given a large number of stored concepts and a high schemata relevance, the user should be able to see the definition of any encountered formulae on the Web. For example, hovering over $a^2 + b^2 = c^2$ will show the definition of the Pythagorean theorem.

Another, more direct, application of the Schematizer would be *Similarity Search*. One could create a MathWebSearch based search engine, which accepts an input formula and a similarity degree (between 0% and 100%). The engine would then create a formula schema at a relative depth corresponding to the similarity degree and use this schema to search the corpus. This approach defines the similarity between two formulae as the percentage of the CMML tree depth that they share.

Last, but not least, we will need to invest in a full user-level evaluation of the utility of formula facets, and the influence of various cutoff heuristics on that.

7 Conclusion

We have presented the design and implementation of a system capable of mathematical faceted search. Moreover, we have described a general-purpose scalable Schematizer which can generate intuitive and recognizable formula schemata and divide expressions into formula classes according to said schemata. Consequently, we have successfully addressed all challenges outlined in Section 2.

Although the Schematizer provides recognizable formulae, some queries to SchemaSearch (e.g. using an author as keyword) provide hits with a very low relevance. This is because we cannot distinguish between the work of the author and work where the author is cited at the textual level. As a consequence, searching for “Fermat” would also show formulae from papers where Fermat was cited and if these papers are numerous, as it happens with known authors, would provide the user with misleading results. This suggests that a better source of mathematical expressions might be required for the SchemaSearch demo.

The implementation of the Schematizer presented here is licensed under GPL v3.0 and code is available at `http://github.com/KWARC/mws/`.

Acknowledgements This work has been supported by the Leibniz Association under grant SAW-2012-FIZ_KA-2 (Project MathSearch). The authors gratefully acknowledge fruitful discussions with Fabian Müller, Wolfram Sperber, and Olaf Teschke in the MathSearch Project, which led to this research (the ZBMath information service uses faceted search on the non-formula dimensions very successfully) and clarified the requirements from an application point of view.

References

[1] Akiko Aizawa et al. "NTCIR-11 Math-2 Task Overview". In: *NTCIR Workshop 11 Meeting*. 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.

[2] *Apache Lucene*. Oct. 4, 2015. URL: https://lucene.apache.org/ (visited on 10/04/2015).

[3] *arxiv.org e-Print archive*. URL: http://www.arxiv.org (visited on 06/12/2012).

[4] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. 2004.

[5] *Elastic Search*. Dec. 7, 2014. URL: http://www.elasticsearch.org/ (visited on 12/07/2014).

[6] Deyan Ginev and Joseph Corneli. "NNexus Reloaded". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt et al. LNCS 8543. Springer, 2014, pp. 423–426. ISBN: 978-3-319-08433-6. URL: http://arxiv.org/abs/1404.6548.

[7] Peter Graf. *Substitution Tree Indexing*. 1994.

[8] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. "MathWebSearch at NTCIR-11". In: *NTCIR Workshop 11 Meeting*. Ed. by Noriko Kando, Hideo Joho, and Kazuaki Kishida. Tokyo, Japan: NII, Tokyo, 2014, pp. 114–119. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/05-NTCIR11-MATH-HambasanR.pdf.

[9] Noriko Kando, Hideo Joho, and Kazuaki Kishida, eds. *NTCIR Workshop 11 Meeting*. Tokyo, Japan: NII, Tokyo, 2014.

[10] *LaTeX - A document preparation system*. Oct. 4, 2015. URL: https://www.latex-project.org/ (visited on 10/04/2015).

[11] *LevelDB*. Dec. 21, 2014. URL: http://leveldb.org/ (visited on 12/21/2014).

[12] *Mathematical Markup Language*. URL: http://www.w3.org/TR/MathML3/.

[13] *Mathematics Subject Classification (MSC) SKOS*. 2012. URL: http://msc2010.org/resources/MSC/2010/info/ (visited on 08/31/2012).

[14] Bruce Miller. *LaTeXML: A LaTeX to XML Converter*. URL: http://dlmf.nist.gov/LaTeXML/ (visited on 03/12/2013).

[15] *XSLT for Presentation MathML in a Browser*. Dec. 20, 2000. URL: http://dpcarlisle.blogspot.de/2009/12/xslt-for-presentation-mathml-in-browser.html#uds-search-results (visited on 04/04/2015).

[16] Ka-Ping Yee et al. "Faceted Metadata for Image Search and Browsing". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 2003, pp. 401–408. URL: http://dl.acm.org/citation.cfm?id=642681.

[17] *Zentralblatt Math Website*. Dec. 7, 2014. URL: http://zbmath.org/ (visited on 12/07/2014).