

INFORMATION SOCIETY TECHNOLOGIES  
(IST)  
PROGRAMME

Project IST-2001-37057 MKM-NET

**Deliverable D5.4**  
**Mathematical Knowledge Management and searchability**

I.Dahn (University of Koblenz-Landau)  
A. Asperti (University of Bologna)

Project Acronym: MKM-NET  
Proposal/Contract no.: IST-2001-37057 MKM-NET

## **Contents**

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Fulltext Search</b>	<b>3</b>
<b>3</b>	<b>Semi-textual search</b>	<b>3</b>
<b>4</b>	<b>Semantic Search for Terms</b>	<b>4</b>
<b>5</b>	<b>Fast filtering of the search space</b>	<b>5</b>
<b>6</b>	<b>Semantic Search for Knowledge</b>	<b>5</b>
<b>7</b>	<b>Key Phrase Search</b>	<b>6</b>
<b>8</b>	<b>Exploiting Semantic Relations</b>	<b>7</b>
<b>9</b>	<b>Summary</b>	<b>8</b>

## 1 Introduction

In this document we describe the state of the art for searching mathematical content. We do not discuss general standards for document markup such as *Dublin Core* or *Z39.50*.

Mathematical content is usually searched by people who intend to retrieve specific content for specific purposes. Ultimately these purposes determine whether a certain retrieved item is appropriate or not. The ultimate goal of search procedures is normally to retrieve exactly the content that is required.

In this respect searching for mathematical content is not different from searching for any other content. But, the major problem is how to analyze the available content and how to describe it in a format that can be utilized by search engines.

In the remainder of this document we discuss and evaluate a number of approaches to searching mathematical content ranging from basic textual methods through to semantics-driven techniques.

## 2 Fulltext Search

Fulltext search is the simplest method. For mathematical content this method is especially limited since important parts of the content consist of formulas. To be of practical use, it must be easy for the user to enter the content to be searched for. But where formulas are involved, this is not straightforward, although there are a few tools – computer algebra systems and L<sup>A</sup>T<sub>E</sub>X shells – which can help, they are yet to be integrated into the interfaces for searching mathematical content.

Even if the user were capable of entering the description of a formula in L<sup>A</sup>T<sub>E</sub>X – the language mostly commonly used by authors of mathematical documents – the corresponding content may not be found since the author may have expressed the formula differently, for example using his own user-defined L<sup>A</sup>T<sub>E</sub>X macro. Also for special characters like German umlauts there are several possibilities for annotation in L<sup>A</sup>T<sub>E</sub>X.

In order to support fully even this uncomfortable approach to fulltext search, the encoding of formulas and special characters must be standardized, macros must be expanded and search requests have to be translated accordingly – for example translating umlauts into their standard encoding. The implication is that it is necessary to preprocess documents before they can be searched.

In some document representations, fulltext search for formulas becomes impossible, when for example formulas are represented as images. This technique is used by popular converters from L<sup>A</sup>T<sub>E</sub>X to HTML, such as TeX4Ht and LeTeX2HTML.

Although it has many inadequacies, fulltext search is not entirely useless. The reason is that, in mathematical documents knowledge is frequently expressed as a mixture of text and formulas such as:

“If  $A$  and  $B$  are groups, then  $A \otimes B \otimes A$  is a group too.”

In this example, fulltext search could reveal that the sentence talks about groups but it could not readily detect the more important fact about the construction (denoted by  $\otimes$ ).

## 3 Semi-textual search

Fulltext search behaves better when the structure of the information is relatively rigid. For instance, `grep` or similar tools have proved to be very effective in the realm of computer science, where the structure of textual documents (typically a program, or the output of a program) is much more regular than that of the everyday mathematical language.

An interesting experiment in building a semantic based retrieval system integrated with `grep` has been conducted as part of the Mizar project [2]. Mizar is a long-term project aimed at building a comprehensive library of formal mathematical knowledge (Mizar Mathematical Library – MML) expressed by means of a suitable, semi-natural language. The grammar of this language is sufficiently flexible to overcome most of the usual and annoying limitations of formal mathematics; on the other it is still sufficiently rigid to allow significant textual queries. The proposed language is based on a wide set of primitive operations (returning

for each article the set of concepts introduced in it, or for each mathematical item the set of all other items which are referenced in it, or which use it, just to give a few examples). Typical operations over lists, such as union, merge, filter and so on allow the construction of complex compound queries. The main filtering operation is still textual, directly calling `grep`.

The work on semantic based retrieval in MML has only recently started. The importance of sophisticated tools for browsing and searching has become apparent in some sub-projects within Mizar which involve many authors and many articles. The current, still experimental, implementation turned out to be very helpful, providing searching functionalities impossible with a purely textual search. A problem of the language, at least in its present version, is that it is clearly geared toward advanced Mizar users.

## 4 Semantic Search for Terms

Unlike text from other fields, mathematical text has a standard semantics agreed by most practitioners. It is therefore possible to improve the search possibilities by taking advantage of this standard semantics. `OMDoc` specifies a standard way for supporting this and *MBase* [9] is a specific database supporting the semantic retrieval of terms.

The basic idea of `OMDoc` is to specify the way in which mathematical text is augmented by a description of its meaning. Potentially it becomes even possible to omit the text and to generate it on demand from the semantic description. Writing these semantic descriptions is not an easy task and has so far been accomplished only for a small set of sample documents. The required expertise is not available for most working mathematicians and supporting tools are still at an early stage of their development.

However, once established, the availability of the semantic annotation opens up a new class of possibilities for searching mathematical content. For example, it becomes possible to search for a symbol, say  $\otimes$ , and to distinguish between its use as the denotation for a cartesian product of groups from its use for vector product.

Note that for the user the problem of easy query entry remains and is further complicated by the need to specify the context in which the query may be satisfied. A particular situation where this can be made simpler for the user is when the search can be related to some current content, that is to determine the contextual constraint from the query origin context.

In this case the user may copy the subject of interest into a search template and the copy mechanism can ensure that all relevant semantic information is transferred as well. When search is restricted to selected content objects only, these selected objects may be presented in a document as links. A click on such a link can then initiate a search for specific related content. The *ActiveMath* system, developed at DFKI in Saarbrücken, uses this technique to link each occurrence of a symbol in a document to its definition.

Besides searching for well-defined formulas, adding semantic annotation offers the possibility of retrieving instances of templates. For example a search for “ $X \otimes Y \otimes X$ ”, where  $X$  and  $Y$  are variables (wildcards) in the template, will retrieve  $A \otimes B \otimes A$  as well as  $A \otimes (B \otimes C) \otimes A$  and  $B \otimes A \otimes B$  but not  $A \otimes B \otimes C$ .

Note however that the exact behaviour of this semantic template search depends on the standardization of the search semantics and that the user has to be aware of it. For example if the standard semantics of  $X \otimes Y \otimes Z$  is  $(X \otimes Y) \otimes Z$ , a search for “ $X \otimes Y \otimes X$ ” will find  $(A \otimes B) \otimes A$  but not  $A \otimes (B \otimes A)$  though both expressions are semantically equivalent in the case that  $\otimes$  is associative. To take account of this, it would be necessary either to add also all semantically equivalent annotations (which is infeasible because there might be infinitely many equivalent forms) or the search engine has to test similar located content on-the-fly for semantic equivalence. The integration of a computer algebra system into a search engine may provide a way to realize this, but resource demands will be considerable.

It may in fact even depend on the purpose of the search whether it is desirable to retrieve semantically equivalent content. For a beginner who still has to learn that the operation  $\otimes$  is commutative it may not be desirable to retrieve  $B \otimes A^2$  when searching for “ $X \otimes Y \otimes X$ ” while an experienced user would welcome it if the search engine allows for several variants of search requests. Currently there is not enough practical experience to decide how a user interface should be designed to make the large variety of possible search variants enabled by semantic search to the user.

## 5 Fast filtering of the search space

It is not uncommon to require unification in order to process the sort of semantic queries in which we are interested. As such, we are making demands which conventional database query languages cannot meet. A workaround is therefore to apply a unification process externally on a set of candidate matches and it becomes essential to be able to filter the data coming out of the repository very rapidly in order to produce a reasonably small set of candidates.

To solve this problem, the HELM group in Bologna has proposed an approach [6] based on four distinct phases:

- *Data-mining*: a small set of metadata is automatically computed from each theorem or definition in the library, providing an approximation of its actual content.
- *Pattern compilation*: when the user submits a semantic query, this is translated into a set of constraints over the metadata and then compiled into a low-level query.
- *Filtering*: the low-level query is executed over the database of metadata, obtaining a set of candidates.
- *Matching*: the actual semantic query is iterated over all candidates, obtaining a precise result set.

If the filtering operation is both quick and correct (in the sense that no good candidates are dropped), the previous approach achieves both accuracy and performance. Moreover, it is extremely modular: extending the metadata set we improve the accuracy of the approximation, and are typically able to cope with more complex semantic queries.

The Metadata used by the HELM group are essentially relations expressing that a given mathematical item refers to another item at a given position. Moreover, an extremely simple set of crucial position descriptors has been identified, covering a very large set of queries of common interest.

A typical unification query is translated into a two sets of constraints, providing a lower and an upper bound for the candidates. In particular, all candidates must be more instantiated than what is required to satisfy the so called *must* constraints, but less instantiated than required to meet the *only* constraints. The precise generation of must and only constraints depends on the particular form of unification in which the user is interested.

The management of must and only constraints typically requires set comparison operations that are not available in traditional relational query languages. For this reason, a new (RDF-based) query language has also been designed and implemented [5, 7].

An extensive statistical analysis has been performed over the library of the Coq Proof Assistant [1], in order to validate the metadata model, to measure its scalability and to test its discriminatory power.

The experiments have been quite successful, and the approach looks extremely promising, but it remains to be seen whether the approach can be effectively applied in the context of automatic theorem proving.

## 6 Semantic Search for Knowledge

It is not only formulas (terms) but also many natural language sentences in mathematical documents that have a precise meaning. Let us consider again the sentence “if  $A$  and  $B$  are groups, then  $A \otimes B \otimes A$  is a group too”. This can be formalized as

$$\text{group}(A) \wedge \text{group}(B) \Rightarrow \text{group}(A \otimes B \otimes A).$$

Thus searching for the template  $X \Rightarrow \text{group}(Y \otimes Z \otimes U)$  where  $X, Y, Z, U$  are variables in the template would yield the sample sentence in the database that  $A$  and  $B$  being groups is a sufficient condition for  $A \otimes B \otimes A$  being a group.

But, this search would fail if the semantics of the sentence were annotated in the equivalent form “If  $A$  is a group and  $A \otimes B \otimes A$  is not a group, then  $B$  is not a group”

$$\text{group}(A) \wedge \neg \text{group}(A \otimes B \otimes A) \Rightarrow \neg \text{group}(B).$$

A traditional (but expensive) approach [4] to this problem is to identify statements up to isomorphisms (that is, essentially, statements having a same, suitable normal form). The theory of type isomorphism is very

interesting and quite entangled, but unfortunately also quite expensive to implement. Moreover, in general, it is not expressive enough to capture other interesting cases, such as the simple expansion of a definition: just consider for instance the two logically equivalent sentences (a)  $x$  is positive and (b)  $x > 0$ .

In 1997 the *ILF* system developed at the Humboldt University Berlin within the DFG focus programme “Deduction” was used to handle this problem in the following way.

First of all the objective of the search was reinterpreted. It was no longer the goal to find out whether a certain sentence could be retrieved from the knowledge base exactly as entered. Rather the goal became to find out whether the formula entered was a *semantic consequence* of the knowledge that was in the database.

To decide this, first the sentence entered<sup>1</sup> by the user was analysed to determine the symbols it contained. Secondly, an ordinary database request was made to retrieve all content using only these symbols and possibly a related set of auxiliary symbols such as the equality symbol. Thirdly, two automatic theorem provers were launched in competition with one another given the goal of proving the phrase entered by the user from the retrieved formulas in a limited time (30 seconds in this experiment).

If one of the provers was successful, the generated proof was analyzed and all retrieved sentences that had actually been used in the proof of the conjecture of the user were returned. In this case the user received the information that their conjecture can be easily proved from these retrieved formulas. Of course, the special case that the formula entered was exactly like or a logical variant of one in the database is handled by this procedure as well.

If no prover generated a proof in the limited available time, the user was informed that the search was not successful but that several related formulas were found in the database so that the user could examine them and attempt to verify the conjecture.

The purpose of using two theorem provers in these experiments was just to evaluate the power of different provers for these search tasks. The experiments used the strongest first order theorem provers available at that time on a database of 100 formalized theorems from the Mizar library. Though the search was highly flexible and fairly tolerant to variations in user input it was apparent that it would not scale up with current automated provers. The main reason for this is that current provers, being tailored for first order logic, do not make efficient use of the type information contained in the formalized knowledge. Another problem is that the search space grows exponentially with the number of candidate formulas that can be used in the proof. Here more refined preprocessing and semantic filtering techniques will have to be developed. Also modifying the task so that only a single formula equivalent to the user’s conjecture has to be found could provide higher efficiency by parallelizing the resulting proof tasks.

It should be also noted that by using first order provers it is not possible to search against a template where one of the variables stands for a complete sub-formula and is thus a higher order variable. There are higher order provers who can handle such requests, but the search space they have to handle will be still more complex.

## 7 Key Phrase Search

All search procedures discussed so far check only for the occurrence of some mathematical content. They are not able to evaluate the relevance of the occurrences found. For example a user searching for the cartesian product symbol  $\otimes$  may find hundreds of places where this symbol is used as an auxiliary operator, though he was possibly only interested in seeing its definition and its most basic properties.

Currently there is no convincing way to find out automatically what a certain piece of content is actually describing. This is in part due to the problems of analyzing formulas which have been discussed above.

However it happens frequently that the concepts that are explained by a piece of content simply do not occur in the actual text. This phenomenon is well known also from non-mathematical texts where key phrases from an index, characterizing what a text is about, do not necessarily occur in the text itself.

---

<sup>1</sup>In these experiments the Computer Algebra System Mathematica was used as a web based front end for the *ILF* system to enter and display formulas.

Automated indexing techniques try to solve this problem by analyzing the text and comparing the words found with the words in other documents that have been manually indexed beforehand. Weak enriched thesauri as developed at CWI in Amsterdam are an example of these methods. Here key phrases are assigned based on the occurrence of specific groups of words. Other methods use statistical data.

All these methods share the problem that important information is not contained in the text but in the formulas which cannot be analyzed this way. They fall short of delivering results that are useful for the end user. [8] gives a survey of the current state of the application of weak enriched thesauri for mathematical documents and derives directions for further research from the current state. Even if automated indexing techniques are currently not usable standalone for mathematical documents, they may provide hints for manual indexing by pointing the indexer to potentially relevant key phrases from an existing thesaurus.

Mathematics has the advantage of possessing a standard subject classification, MSC 2000, that is generally agreed by the mathematical community. This classification, however, describes only rather large subtopics of Mathematics. Therefore it can be used to classify documents but it is inappropriate to retrieve content from within these documents.

Building and maintaining a standard thesaurus for Mathematics that is linked with the Mathematics Subject Classification is an urgent need. Within the IST project Trial-Solution, FIZ Karlsruhe has started work towards this based on key phrases assigned by authors of 19 mathematical books and by additional key phrases assigned from within the project – this latter work is mainly done at the Mathematical Institute of the Technical University in Chemnitz. The project thesaurus has ca. 6,000 key phrases and FIZ provides support for adding new key phrases to the thesaurus in a consistent way.

Key phrase search can be more refined when it is combined with search for specific types of information. For example in order to prepare an exam, the tutor will search for exercises on a topic where the topic itself is characterized by a collection of key phrases.

Key phrases are also a simple method to find content from other documents that is related to given pieces of content. When a multilingual thesaurus is available it is also possible that a user searches for key phrases in his native language but retrieves also content in other languages which she may understand even without knowing the appropriate search phrases in that language.

Key phrases are metadata that describe content. When mathematical knowledge is to be retrieved it is, however, not only the content that matters. When it is to be used for teaching it is also useful to search for content that has been prepared for a specific audience or content for which learning takes a certain amount of time. Also technical restrictions (like Mime types) might be essential. It may be desirable to search for applications or for motivations for a certain concept.

Standard learning objects metadata schemas like those from IEEE and IMS provide the means to encode this information in a standardized way. The high effort in assigning this kind of metadata contrasts sharply with its decreased value when the content is reused in a different context. For example what was intended to be a motivating example which can be run through by a physics student with little effort might become an application for a mathematics student that requires considerable efforts to fill the gaps left in the text. The same mathematics student will however have no problems understanding the text when he returns to it later for a second view.

## 8 Exploiting Semantic Relations

The methods discussed so far retrieve mathematical content based on features assigned to individual content pieces. This is not enough when small pieces of content are to be retrieved for reuse. Pieces of content do not live in isolation. They make sense only in an environment which provides the necessary context or prerequisites: Theorems have proofs which make use of other theorems; exercises require understanding of previous theorems etc.

Modelling this network of interrelations gives the possibility of composing meaningful documents for specific purposes. Slicing Book Technology, as invented at the University Koblenz-Landau and further developed in the IST project Trial-Solution, has applied this to as much as 5,000 pages of pre-existing mathematical

texts.

It has turned out that the most relevant semantic relations between pieces of content are binary, that is they relate only two pieces of content. [3] describes the relations which have been found to be relevant in that project. The most important of them is “ $A$  references  $B$ ” with the intended meaning “In order to understand  $A$ ,  $B$  must be understood”.

These relations as well as key phrases and types of content that have been discussed before can be combined with information about the user from a user model. This can be done by an automated inference engine. This inference engine must also process a (small) set of general rules which describe the set of content slices that should be retrieved for a user with a certain objective and a certain sort of user profile. Based on this information the inference engine will derive recommendations for content to be presented. These recommendations can be further modified by the user or they can be used to launch the dynamic creation of a new document from the selected slices on the fly.

In this way search is not initiated by entering a concept to be searched for (though this as well as fulltext search is possible) but by a request to adapt a selection of interesting slices automatically for a specific use case. Also the final result of the search is not a collection of occurrences but a new document with the retrieved content that did not exist before.

The required binary relations between content slices can only rarely be assigned automatically. Explicit references can be found in the text and used for this purpose, but usually there are only few such references. Consequently, they must be assigned manually.

Experiments in the Trial-Solution project show that this can be done by experienced students at an average speed of 15 slices per hour, including the assignment of key phrases and types.

It is important to note that this effort is not done for a particular user group or usage scenario: when reuse for a new purpose or in a new context is required, it is only necessary to adapt the general rules that describe the intended search results. The relations and key phrases assigned to the content slices do not have to be revised or adapted.

Demos and products using this technology have been produced by Slicing Information Technology and are available through the company’s website at <http://www.slicing-infotech.de>.

## 9 Summary

The most widely used methods to search for mathematical content are currently fulltext search and search for key phrases. Users are used to this and appreciate the additional help that online editors offer for these types of search. Slicing Book Technology as explained in the previous section is a further step forward. It can be gradually introduced as an extension of the use of key phrases. It does not require any change of the content itself. Semantic search provides the tools for a precise search for mathematical objects, to exchange these with other systems and to evaluate them, but this type of search typically requires rewriting the content completely. Complex semantic search has to deal with the problem of dealing with semantically equivalent objects. Their handling requires further research and tools which are most likely to arise from computer algebra systems and automated theorem provers. A combination with key phrase search may additionally help solving the problem of ensuring the relevance of the mathematical objects retrieved.

## References

- [1] A.Asperti, F.Guidi, L.Padovani, C.Sacerdoti. Some statistical considerations on the Coq library. Draft, University of Bologna. 2003.
- [2] Grzegorz Bancerek, P. Rudnicki. Information Retrieval in MML. Proceedings of the second International Conference on Mathematical Knowledge Management, Bertinoro, Italy, February 2003. LNCS 2594, pp. 119–131.

- [3] I. Dahn. Management of Informal Mathematical Knowledge - Lessons Learned from the Trial-Solution Project. In: Fengshai Bai and Bernd Wegner (eds.): *Electronic Information and Communication in Mathematics*, LNCS vol. 2730, pp. 29–43, 2003
- [4] D. Delahaye. information Retrieval in a coq proof library using Type isomorphisms. In T.Coquand et al. editors, *TYPES*, LNCS 1956, pp.131-147, 200.
- [5] F. Guidi. *Searching and Retrieving in Content-Based Repositories of Formal Mathematical Knowledge*, Ph.D. Thesis in Computer Science, Technical Report UBLCS 2003-06, University of Bologna, March 2003.
- [6] F. Guidi, C. Sacerdoti Coen. Querying Distributed Digital Libraries of Mathematics. In *Calculus 2003*, Aracne Editrice S.R.L., Thérèse Hardin and Renaud Rioboo editors, ISBN 88-7999-545-6, pp. 43–57.
- [7] F. Guidi, I.Schena. A Query Language for a Metadata Framework about Mathematical Resources. Proceedings of the second International Conference on Mathematical Knowledge Management, Bertinoro, Italy, February 2003. LNCS 2594, pp. 105–118.
- [8] M. Hazewinkel. Identification clouds and automatic assignment of key phrases: lessons learned from the TRIAL SOLUTION project. Internal paper, Trial-Solution 2003.
- [9] M. Kohlhase, A. Franke. MBase: Representing Knowledge and Content for the Integration of Mathematical Software Systems. *J. Symbolic Computation*, 32(4), 2001, pp365–402.