Mathematical Extension of Full Text Search Engine Indexer

Jozef Mišutka, Leo Galamboš
Department of Software Engineering
Charles University in Prague
Ke Karlovu 3, 121 16 Prague, Czech Republic
Email: jmisutka@gmail.com

***Abstract*—The world of mathematical knowledge on the WWW has grown enormously. Despite the clear importance of a mathematical search engine this research field had been abandoned until very recently. Although, currently available full text search engines can be used on these documents too, they are deficient in almost all cases. They cannot handle structured mathematical text and mathematical operations. Many problems are the result of the mathematical nature. By applying axioms, equal transformations, and by using different notation each formula can be expressed in numerous ways. Ambiguous searches like "sin" or "a" would return documents containing sine function and the English noun sin or documents containing variable a and indefinite article a. Moreover, mathematical operators and special notation cannot be expressed in their query languages. In this work, we address these issues and present a technique how to index real-world scientific documents containing mathematical notation by exploiting the current state-of-art of full text search engines. Our approach has several advantages over existing solutions. It is primarily intended for documents on the WWW, which are mostly semantically poor, and offers an extensible level of mathematical awareness supporting also similarity searches. Furthermore, it is designed as an extension and therefore any full text search engine can easily adopt it. The experiments over two real-world document sets showed that the performance is highly dependent on several features of the mathematical search engine.**

***Index Terms*—mathematical search engine; mathematical indexing; formula indexing; mathematical similarity search.**

I. Introduction

Most scientific papers are published electronically. Moreover, great efforts have been made to digitise mathematical knowledge in the last years. The result is that the number of scientific documents is growing rapidly and the necessity to search in these documents has increased significantly. Although, currently available full text search engines can be used on these documents too, they are deficient in almost all cases. They cannot handle structured mathematical text and mathematical operations. Ambiguous searches like "sin" or "a" would return documents containing sine function and the English noun sin or documents containing variable a and indefinite article a. Moreover, mathematical operators and special notation cannot be expressed in their query languages.

A better approach is to use the uniform style of scientific papers. It has been exploited by several search engines (http://scholar.google.com, http://citeseer.ist.psu.edu). It is applicable only for simple textual searching but with special features like citation list of a document and related key documents. When focusing on the mathematics, similar techniques used in theorem provers and proof checkers can be applied but they have several disadvantages. The most important disadvantages are applicability, performance and zero fault tolerance - similar searching is not possible.

The vast majority of real world mathematical document formats (PDF, PS, MathML, HTML) does not contain enough semantic information to interpret the mathematical symbols unambiguously, even when the source code of the document (TEX, LATEX) is available [1], [2]. With these observations in mind, neither pure mathematical nor pure textual approach to mathematical searching is satisfactory.

We can combine both approaches together, exploiting the advantages, by extending the most used and successful search engine capable of indexing and searching the WWW to index mathematical notation. We propose a solution based on full text search engine. It is primarily intended for real-world scientific documents which are mostly semantically poor. But it still offers an extensible level of mathematical awareness supporting also similarity searches.

These key features have been identified to build an applicable, user friendly mathematical search engine: 1) enabling search for scientific notations, 2) support for raw text queries, 3) search for mathematical notation should be limited to mathematical content (e.g. "sin" problem), 4) support for basic mathematical operations to enable formula matching other than exact hits e.g. similarity searching, pattern matching, subformula searching, and 5) results should be ordered by ranking functions extended for mathematical content.

Construction of a mathematical search engine requires these issues to be addressed: 1) extraction of mathematical notation from documents, 2) storing mathematical notation, and 3) extending ranking functions. Extracting a mathematical formula includes identification and conversion to supported, at least syntactically equivalent, form. Mathematical text is highly structured and symbolic. For that reason it cannot be expressed without a special markup language designed for mathematics. Identification of mathematical formulae is minimised to identification of mathematical markup language. The conversion to supported form must be tolerant because the meaning of symbols is context dependent and there is usually little semantic information to use. Mathematical formulae $a\,b$

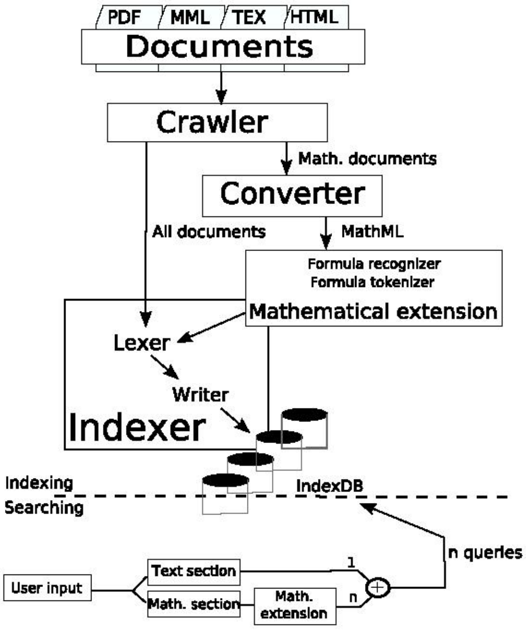

Figure 1. Architecture of an arbitrary full text search engine

resp. $a(b+c)$ are very likely to be the shorter form of $a*b$ resp. $a*(b+c)$ but on the other hand Π can be either the constant or a function representing permutation. We solve this problem by defining exactly one meaning for each symbol and by using simple heuristics. Storing mathematical notation involves several problems. One of the most important paradigms used to solve these problems is that the input is not stored only once but is augmented and stored in various different synonyms - it is the opposite of stemming[1]. By applying *transformation* and *generalisation* rules together with an *ordering* algorithm on the input formula the synonyms are created and problems connected with the mathematical nature are minimised. This technique also enables similarity searching. Furthermore, to make the search engine more applicable, simple text is separated from the mathematical notation. The ranking functions used by full text search engines can be left unchanged for the simple textual part but has to be changed for the mathematical part.

The searching phase of a full text search engine has to be extended too. User input is separated to simple text query and mathematical query. Afterwards, the mathematical query is processed using the same algorithm as in the index part. The algorithm produces N representations which are appended to the simple text query creating N search queries ordered by similarity. Then, the queries are executed. Later queries have higher possibility to match an indexed formula.

The number of scientific documents on the WWW is extensive and growing rapidly. The success of full text search engines has shown that almost without any semantic information satisfactory search results can be produced. The new technique of indexing structured text using full text search engine is the major contribution of this paper. Because it is designed as an extension every full text search engine can easily adopt it (Fig. 1). For the evaluation purposes Egothor v2 was used as the basis and was extended to a mathematical search engine.

The rest of this paper is organised as follows. Section II summarises all known solutions to mathematical searching and is put in context with related work. In Section III we describe our solution and its features. Section IV includes experimental results obtained by a prototype implementation. It shows how performance is effected when changing properties of the search engines. Conclusion and future directions are discussed in Section V.

II. Related Work

To the best of our knowledge, there is very little prior work dealing with mathematical searching especially with indexing of mathematical notation. Miller and Youssef [3] describe a basic approach to mathematical searching. They propose a simple method how to transform structured mathematical notation to simple linear text. Their work was used in the construction of a search engine capable of indexing mathematical notation - MathDex[2]. The search engine relies on similarity but cannot handle mathematical operations directly.

Michael Kohlhase and Ioan Sÿucan [4] built a mathematical search engine - MathWebSearch[3] - based on substitution tree indexing. It does not support textual queries but has a stronger mathematical basis.

Both mentioned mathematical search engines have been in parallel development with our solution [5].

III. Indexing Mathematical formulae

Full text indexing can be imagined as a description of an arbitrary input by textual words - tokens. When the input consists of simple words the identity function can be used to produce tokens. However, mathematical formulae are highly structured without a general canonical form. By applying axioms, equal transformations, and by using different notation each formula can be expressed in numerous ways. Furthermore, many formulae hold only in special theories with special requirements.

Linearisation, transformation rules, generalisation rules and ordering algorithm simplify the complex and highly symbolic mathematical structures into linear structures with well defined symbols.

A. *Extraction of Mathematical Formulae*

After analysing several formats suitable for mathematical indexing (MathML, LaTeX, TeX, XML, PDF, PS, HTML) capable of describing either visual presentation of mathematics or semantic meaning, MathML was chosen as the primary supported format. The goal of MathML, developed by the W3C consortium, is to enable mathematics to be served, received, and processed on the WWW just as HTML has

[1] Stemming is the process where inflected or derived words are reduced to their root form.

[2] http://www.mathdex.com:8080/mathdex/search

[3] http://search.mathweb.org/

enabled this functionality for text. MathML value lies in the ability to encode both mathematical visualisation (Presentation MathML) and semantic (Content MathML). It can be embedded in HTML and all other formats can be converted to MathML. It must be noted that the majority of input documents does not contain enough semantic information and must be expressed in Presentation MathML. PDF format is by far the most used document format of scientific documents and special emphasis has been made to index it. The Infty Project [6] enables the conversion from PDF to MathML.

The indexing algorithm of a full text search engine must use a recognition technique responsible for analysing parts of the text and parse them correctly to words, sentences etc. An analogous technique must be also used in a mathematical search engine and is called formula recogniser. If a document contains mathematical notation it is converted to MathML and delegated to the mathematical extension. The extension finds starting element of a mathematical notation according to the MathML specification and the formula recogniser parses it to a tree-like structure which is marked as mathematical. There is an important difference between parsing Content and Presentation MathML. Content MathML contains information whether an element is a number, a constant, a variable or any other type but the Presentation MathML does not. To remedy this important deficiency several simple heuristics are applied together with another paradigm. Mathematical symbols are context dependent. The solution is to choose one meaning and operate with the formulae identically in indexing and in searching phase. We can improve this technique by indexing semantically rich formulae as they would not contain any semantic information. Each ambiguous symbol is converted to its normal form and predefined semantic meaning, for example π, Π, Pi, pi to function π. The best specification of normal forms depends on the document set we are indexing and is a subject to evaluation. The advantage is that the user does not have to provide semantic information about the input. Precision of retrieved documents is decreased but the ranking function can be used to minimise the loss.

B. Storing Mathematical Formulae

Common indexing algorithm works only with single linear words whereas mathematical formulae can be structured in more levels. To adapt the structured notation for sequential indexer linearisation must be performed. Exponents, numerators, denominators, arguments and other structured fragments are labelled and then represented sequentially. A very simple but important step must be performed before parsing the formula. Symbols that do not have a representation in a standard encoding have to be converted to words (e.g. $\int$ is stored as int). The advantage of using infix form to express mathematical notation is readability but the disadvantage is the necessity of parentheses. The proposed search engine uses postfix notation also called reverse polish notation. This technique has two main advantages. The first one is that it does not need parentheses and the second one is that it enables one type of similarity searching as illustrates the following example. The formula $(a + b) - (c + d)$ is converted to $ab + cd + -$, let's assume that formula tokens are $ab+$ and $cd+$. The resulting index database contains three words in this order: $ab+$, $cd+$, $-$. This allows to search for the subformulae ($ab+$, $cd+$) without knowing the operator between them.

Augmentation algorithm

The main disadvantage of a full text search engine indexing structured data is clear. It can only search for documents containing specified words. This is a problem because mathematical formulae can be expressed in many equivalent ways. The most important problems arising from the mathematical nature are: 1) no commonly used mathematical format nor unitary notation ($1/x = \frac{1}{x} = (x)^{-1}$, $\pi = \Pi = Pi$, $xy = x * y, ...$), 2) symbol meaning dependent on context, 3) no canonical form ($1 + 1 + a = a + 2$, $sin^2x = 1 - cos^2x$), 4) structured text ($e^{\frac{x+1}{x-1}}$), 5) many mathematical structures with different axioms.

To fully exploit the full text search engine and minimise the main disadvantages the indexed formula is not represented only by one word (or ordered sequences of words) but by several words (or ordered sequences of words). This technique is called augmentation. The first representation is an ordered textual representation of the input formula itself. Next representation is always created by applying some of generalisation and transformation rules to the last representation. More generalised representations of mathematical formulae are computed in several iterations and indexed afterwards. The generalisation is based on a set of transformation rules. In each step one or more transformation rules are used. Some assumptions are made on the underlying mathematical model which is simplified in each step of the algorithm. A representation from later iteration would match more formulae because it is simplified. It is clear that the first iteration is the most important and naturally should be ranked the highest. Augmentation does not solve the unique canonical form problem completely, but it can reduce the possibility that two equivalent formulae do not match. To store all of the possible representations is clearly impossible because unique canonical form of mathematical formulae does not exist. Due to the performance, the maximum number of representation is set to a fixed number.

From the mathematical point of view, in this stage a function Q is constructed. The domain is the space of all mathematical formulae (F) and the range is $F^N := F_1 \times F_2 \times, ..., \times F_N$. Simply speaking, it produces N (N is a predefined constant dependent on the generalisation and transformation rules) formulae $f_1, f_2, ..., f_N$ for one input formula. The function Q can be defined: $Q : F \to F^N$, $Q(f) = [f_1, ..., f_N]$. The function Q must satisfy one requirement about its domain F^N: for all valid i, F_{i+1} is a generalisation of F_i. The algorithm which produces the output of function Q is called generalisation algorithm. Different representations do not need to be strictly mathematically equal but the advantage is that more and more formulae are mapped to the generalised formula in each step. One important feature is that there can be more than one function Q with different specialisations. The only limitation is the number of important formulae in one document which is usually negligible comparing to the number of textual words. This is a list of transformation rules used in our implementation:

1) *Partial evaluation*
   $7 + a + 5 \rightarrow 12 + a$ ($7 + a + 5$ is converted to $12 + a$)
2) *Approximate numerical constants*
   $5.82 \doteq 6$
3) *Remove brackets using distributivity*
   $a * (b + c) \rightarrow a * b + b * c$
4) *Multiply tokens:*
   $\frac{a+b}{2} * \Pi \rightarrow \frac{\Pi a + \Pi b}{2}$
5) *Assign each numerator its own denominator*
   $\frac{\Pi a + \Pi b}{2} \rightarrow \frac{\Pi a}{2} + \frac{\Pi b}{2}$
6) *Replace constants with* const *symbol*
   $74 + a^2 + b^2 \rightarrow const + a^{const} + b^{const}$
   or $\rightarrow const + a^2 + b^2$
7) *Replace unknown constants, variables with* id *symbol*
   $a^2 - b^2 + 2bc \rightarrow id_1^2 - id_1^2 + 2id_1id_2$
   or $\rightarrow id_1^2 - id_2^2 + 2id_1id_2$
   or $\rightarrow id_1^2 - id_2^2 + 2id_2id_3$

The list of the rules applicable to a formula is not complete but contains rules solving the most common mathematical problems. Any other rule can be simply added. The number of text representations of one formula is not limited and depends on the goal of the search engine. The last transformation is the most severe one because there is no global numbering, however, local numbering in a token is possible.

Another problem is that mathematically equivalent formulae with the same but permuted operators and operands would be considered as different when compared letter by letter. Ordering algorithm guarantees that two mathematically equal formulae with the same but permuted operands have the same canonical representation and that two similar (but not equal) formulae have a similar (but not equal) unique representations. This can be guaranteed because of the simplifications and assumptions made on the underlying mathematical apparatus. One definition of formulae similarity is that the more operations they have in common the more similar they are. To get the best search results the token ordering must be done with regard to the similarity of operations not to the similarity of characters or values. The ordering does not use simple string comparison, if not necessary, but a rather more complex algorithm [5].

There are several common token types - word, number, date, etc. To prevent the ambiguous word search, resulting from collisions between mathematical tokens and simple textual tokens, each mathematical token is marked with a special type.

Many scientific fields use formulae (physics, mathematics, computer science, medicine, chemistry, etc.) to describe various processes. Many formulae are sound and valid only in specific mathematical structures. In the simplest design, instead of distinguishing between them, all structures are generalised into single one in which these basic and most common axioms hold: 1) *commutativity*, 2) *associativity*, and 3) *distributivity*.

One of the most important but less obvious problems is the question of what exactly is atomic information that can be searched for in a scientific search engine. In a simple full text search engine the smallest information we can search for is a word. The grain of a formula should be its reasonably big fragment - subformula. Formula tokenizer decides what the atomic information unit is and this has a great impact on the performance. When the tokens are small the possibility of two being equal is higher and therefore index database is smaller. The evaluation of different formula tokenizer can be found in Section IV.

C. Ranking function

Each word in a document has a weight which indicates the relevance to the document. Usually words in titles are ranked higher than words in body of a document because they are considered to be more important. There are many heuristics which adjust the relevance of words in the relation to the document. Because the mathematical search engine produces words they should be ranked too. If two documents contain the same mathematical formula ($\doteq$) but both match with a different representation of the formula ($f_i \doteq f_j$ but $i \neq j$, let $i < j$) then document containing f_i must have a higher rank. Let R be a ranking function ranking word regarding to a document then the requirement for the ranking of the formula words can be written: $R(f_1) \geq R(f_2) \geq ... \geq R(f_N)$. The ranking algorithm must take into account that later representations are less similar to the original formula.

IV. Experimental evaluation

We conducted several experiments on two real-world document sets to evaluate and describe the important characteristics of the proposed technique. We have not included tests regarding precision and recall since there is no accepted evaluation metric designed specifically for mathematical searching making the results little informative [5].

Because mathematical formulae are very complex non linear structures more simplifications must be introduced. To keep the indexing algorithm simple, equations are considered as two formulae with equal operator between them. Constraints and variables of known operators are not considered ($\int_0^\infty x^2dx \rightarrow \int x^2$). Matrix is converted to a set of formulae. These simplifications are not based on any known limitations but are introduced to allow the mathematical search engine to be simpler and faster.

A. Setup and Document Set

The prototype implementation stores every formula in five representations. It uses generalisation and transformation rules described in Section III-B. The purpose of the first representation is to allow mathematically most precise match. The second representation is mathematically equal to the original formula. First two representations have relatively high rank because they are very similar to the original formula. The third representation is created by applying more complex

TABLE I
CNX AND ARXIV CHARACTERISTICS

Document set	#Docs	#Formulae	Aver. #Formulae
CNX	421	32306	76.74
ARXIV	1915	852388	445.11

TABLE II
CHARACTERISTICS OF DIFFERENT FORMULA TOKENIZERS

ARXIV	#Words	#Avg. Word p. Form.	#Repr.	#Diff. Repr.
all	4261940	1.00	4261940	2121729
alls	4628917	1.09	4261940	2121234
com	11410081	2.68	4261940	2121463
coms	20801243	4.88	4261940	2120545
num	34182695	8.02	4261940	2120319
const	32214217	7.86	4261940	2120319
CNX	#Words	#Avg. Word p. Form.	#Repr.	#Diff. Repr.
all	161530	1.00	161530	78630
alls	175081	1.08	161530	78603
com	401102	2.48	161530	78626
coms	752047	4.65	161530	78566
num	1345213	8.33	161530	78518
const	1275287	7.90	161530	78518

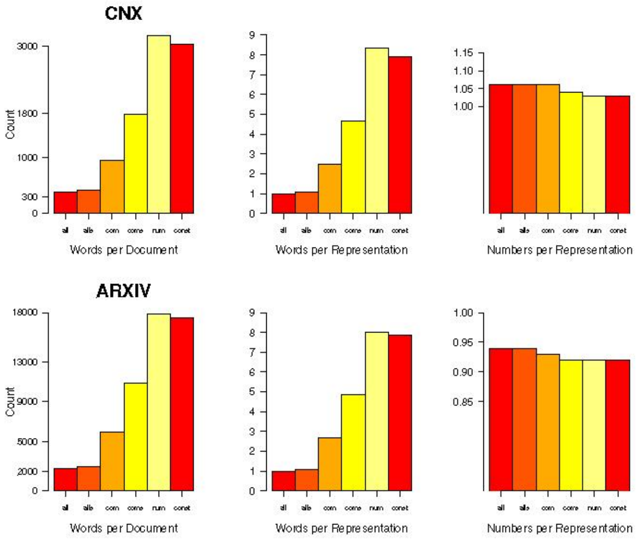

Figure 2. Average characteristics of document sets

transformations rules. The main goal is to reduce the number of possible representations mapping very common variations to a single one. The fourth representation uses the idea that constants are not important in many parts of mathematics. The last one limits the similarity to the operations and number of operands rather than to the symbol meaning.

We used two different document sets for our experiments. Connections[4] (referred to as *CNX* in the remainder) is a site where educational material is shared. It is organised into modules, in other words small pieces of knowledge, which are further organised into courses, books etc. The modules are displayed using embedded MathML with both Presentation and Content MathML included. There are around 500 modules in *CNX* Mathematics and Statistics section. *CNX* document set was obtained by partially mirroring its site in July 2007. This set also includes documents not containing mathematical formulae totalling 99 MB with 421 scientific documents.

The second document set is based on the arXiv document database[5]. The KWARC Research group has been trying to transform scientific knowledge captured in the arXiv document database into MathML. We have downloaded almost 500MB of documents but used only transformed ones totalling 252MB with 1915 mathematical documents. This document set will be referred to as *ARXIV*.

The characteristics of both document sets are illustrated in Tab. I.

Granularity of mathematical formula

The granularity of formulae has an impact on index database size, speed and mainly on applicability. This section provides comparison of different formula tokenizers. A tokenizer inspects a formula and decides whether it is small enough to be atomic or is split into more parts.

There are 6 different tokenizers for both document sets. The common used, *com*, accepts subformula with small depth difference and entity count, *coms* accepts only really simple ones, *all* accepts all formulae, *alls* accepts all nodes with neither index nor exponent, *num* accepts only numbers, and *const* accepts only constants. The characteristics of the tokenizers are illustrated in Tab. II.

The first column shows the total number of words. Tokenizers producing longer words like *all* or *alls* produce significantly less words than the other ones. The average word count shows that each formula can be represented in 4.2 ordered words in average. The number of representations is the same for all tokenizers as they do not dependent on the tokenizers itself. Despite the significant difference between tokenizers the number of different representations is very small.

Average characteristics of both document sets are shown in Fig. 2. It is interesting to note that two completely different document sets have similar characteristics. The first graph shows expected behaviour of formula tokenizers. The stricter a formula tokenizer is the more words it produces. The biggest word count is almost 8 times the size of the smallest one in both document sets. Second graph shows the average number of words per representation. Tokenizers *num* and *const* produce many words. This has two implications: 1) it is difficult to reasonably define what is a similar subformula and what is not, and 2) higher word count in one formula can cause performance problems. The last graph in each document set shows the number of numerical constants per one formula representation.

Size of the index database

It is important to note that the size of index database includes the whole index directory including inverted index, indexed meta-data, term occurrences, indexed normal text, index-sequential file for improving performance etc.

[4] http://cnx.org/

[5] http://arxiv.org/

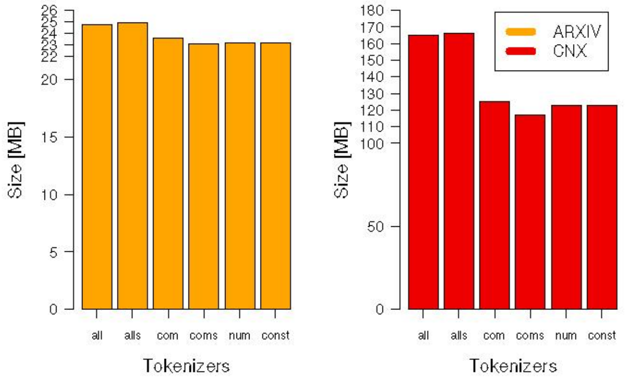

Figure 3. Index database size using different formula tokenizers

The comparison of the index database size of different formula tokenizers is in Fig. 3. The largest index database were produced by *all* and *alls* formula tokenizers because they accept all resp. almost all formulae making the words representing a formula very long. This lowers the probability of two equal words in the database. Tokenizers *num* and *const* accept only formulae with one entity so the probability of two words produced by these tokenizers is higher. The index database size of *com* is slightly above *num* and *const* as expected. Tokenizer *coms* is a special case. The size of the inverted index is the smallest using *num* tokenizer and the size of word occurrences is the smallest using *all*. The index database includes both files together with several others. The average size of all files is produced by *coms* making the index database the smallest. The difference between the largest index database and the smallest is 30% in *ARXIV* and 7% in *CNX*. The size difference between inverted indexes is 39% in *ARXIV* and 7% in *CNX*.

PDF support

One of the most important goals of our solution is applicability. Because most scientific papers are published in PDF format great emphasis was put on the ability to index it. The Infty converter can produce MathML from PDF documents. Despite the fact that a product version of Infty converter is available, it cannot be applied on a larger collection of documents. One of the problems encountered is the speed of the conversion. It takes tens of seconds to convert single PDF file and is greatly depends on the number of pages. On the other hand, the drawback is notable only during the first processing of a document set because PDF files are rarely updated comparing to HTML content. Another problem is accuracy. The newer version outperformed the older one in the number of recognised characters but on the other hand, single character was sometimes converted to a set of characters (e.g. M was converted to IVI).

V. Conclusion and Future Works

We have developed a new technique of indexing mathematical notation. By exploiting the current state-of-art of full text searching together with the new described paradigms, searching in real-world scientific documents is possible with an extensible level of mathematical awareness supporting also similarity searching. It is different from all other known techniques because it does not try to find a user query formula by concretising it. On the contrary, it tries to find an exact match at first. When it is not successful the user query formula is generalised and the search is repeated.

Another advantage is that the techniques developed over the last several years of the existence of publicly available full text search engines are automatically inherited and ready to use. There is no need to design, implement and test a new search engine. The extension character of the proposal allows every full text search to adopt it easily.

The prototype implementation of the proposed solution has proven useful when gathering user and document statistics. The results show that the probability of finding a formulae is strongly dependant on the correct symbol recognition during the indexing phase. The fine granularity has not only influenced the usability from the user point of view but also the size of the index database. The difference can be significant and must be taken into consideration.

This work shows that there are many possibilities how to address the problem of mathematical searching opening several questions for future research. How useful is this method? This cannot be answered without exhaustive user evaluation. Another question is tightly connected with the first one. How can we evaluate existing mathematical search engines? There is no widely accepted research about the characteristics and user expectations of such an engine. Therefore, evaluation and comparison of existing solutions cannot be done for now. It would be also interesting to find out whether the proposed technique could be easily used on other structured data (e.g. chemical formulae). And finally, how can be advanced search operators like proximity operator used to improve the similarity searching.

Acknowledgment

Ing. Jozef Mišutka Sr. is acknowledged for software tools used during the evaluation. We are grateful to MSc. Tomáš Kotrík for his priceless comments.

References

[1] M. Yang and R. J. Fateman, ”Extracting mathematical expressions from postscript documents”, ISSAC, ACM, 2004, pp. 305-311.

[2] J. Stuber and M. van den Brand, ”Extracting Mathematical Semantics from LaTeX Documents”, LNCS 2901, Springer, Germany, 2003, pp. 160-173.

[3] B. R. Miller and A. Youssef, ”Technical Aspects of the Digital Library of Mathematical Functions”, Annals of Mathematics and Artificial Intelligence, Springer, 2003, Netherlands, pp. 121-136.

[4] M. Kohlhase and I. Sÿucan, ”A search engine for mathematical formulae”, LNCS 4120, Springer, Germany, 2006, pp. 241-253.

[5] J. Mišutka, ”Mathematical search engine”, Master thesis, Faculty of Mathematics and Physics, Charles University in Prague, 2007.

[6] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida and T. Kanahori, ”INFTY - An integrated OCR system for mathematical documents”, Proceedings of DocEng, France, 2003.