Search Mathematical Formulas by Mathematical Formulas

Yoshinori Hijikata, Hideki Hashimoto, and Shogo Nishida

Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, japan
{hijikata,nishida}@sys.es.osaka-u.ac.jp

Abstract. Users cannot search information by mathematical formulas as queries in existing search engines. This is because mathematical formulas are not expressed as a sequence of characters. Some formulas are expressed in a complex structure like fractional numbers and index numbers. We present a search engine for MathML objects using the structure of mathematical formulas. The system makes the inverted indices by using the DOM structure of the MathML object. We proposed three types of index. One type is constructed from some paths of the DOM structure and expressed in XPath. The other type is constructed by encoding the nodes in the same level in DOM structure. The third type is a hybrid method of them. This paper describes the experiment conducted to study the effectiveness of those indices.

Keywords: MathML, search engine, mathematical formula, index, inverted file.

1 Introduction

The number of documents including mathematical formulas on the Web is increasing. However, people cannot search web documents by a mathematical formula as a search query in current search engines on the Web. The current search engines allow us to use a sequence of characters for a search query. Mathematical formulas are not necessarily expressed in a sequence of characters. They are sometimes expressed in a structure like fractional numbers and index numbers. For an example, xy and x^y have the same expressions if they are expressed in a sequence of characters. Both of them include x and y and they are arranged in the same order. However the meaning of xy and the meaning of x^y are totally different. Therefore people cannot search mathematical formulas by using a search engine which only uses the kind of characters and the order of them. If we want to search mathematical formulas, the system must consider their mathematical structures.

MathML (Mathematical Markup Language) [2] is being introduced worldwide as a standard to express mathematical formulas on the Web. People can express the structure of the mathematical formula by exploiting the hierarchical structure of XML. Nakanishi et al. have proposed a search method of mathematical formulas for MathML objects [3]. Its search type is the similarity search. It uses the vector space model, one of the popular search methods in the field of information retrieval. However the vector space model only has the information about the existence of characters and mathematical symbols. It does not express the structure of mathematical formulas. If two different formulas include similar characters and symbols, they are listed in the same search result. Furthermore, the search performance (speed) of the vector space model drastically becomes slow if the number of objects (documents) becomes large. The inverted file (or inverted index) is the better search method especially for commercial search engines. As adopted in major search engines like Google and Yahoo!, the inverted file provides a fast search performance.

In our research, we propose a search engine of mathematical formulas. We make an inverted index by using the DOM (one of the API to access the XML document [4]) structure of MathML of objects. We propose three types of indexing method of MathML objects. The first uses the vertical structure of the DOM structure. The second uses the horizontal structure of the DOM structure. The last uses the hybrid structure of the above. We conduct an experiment to evaluate the these indexing methods.

2 Overview of the Math Search Engine

Figure 1 shows the system structure of the proposed search engine of mathematical formulas. The system consists of two sub systems. One of them crawls the Web and collects web documents including MathML objects, and then it stores them with the search indices in an inverted file. The other system sends a query to the inverted file and outputs the search results when it receives the query from a user.

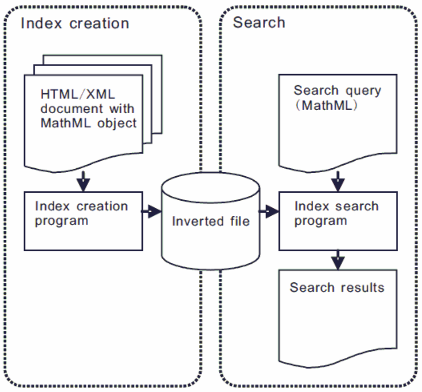
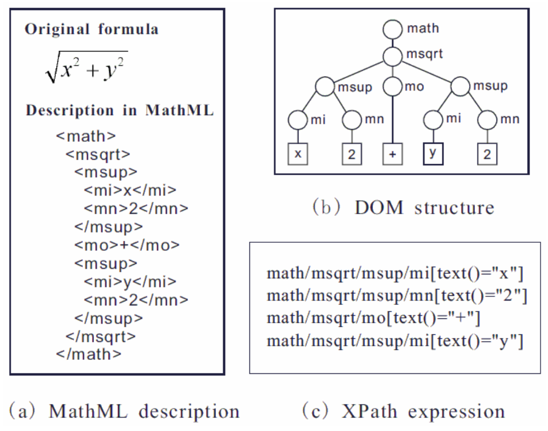


Fig. 1. Overview of our proposed system

Web documents which this search engine targets are HTML documents or XML documents which include mathematical formulas expressed in MathML like Figure 2-(a). <msqrt> represents the root, <msup> represents the square, <mi> represents the variable number, <mn> represents the numerical number, <mo> represents the mathematical operator.

When the system obtains a new web document, it retrieves each MathML object from the document. Then it acquires the DOM structure of the MathML object (see Figure 2-(b)). It generates indices by using a specific structure in the DOM structure. For giving an example of indices, we use a path in the DOM structure. XPath [5] are usually used for expressing a path in the DOM structure. Figure 2-(c) represents examples of XPath for all the paths of the MathML object in Figure 2-(a).



```
<math>
 <msqrt>
  <msup>
   <mi>x</mi>
   <mn>2</mn>
  </msup>
  <mo>+</mo>
  <msup>
   <mi>y</mi>
   <mn>2</mn>
  </msup>
 </msqrt>
</math>
```

```
math/msqrt/msup/mi[text()="x"]
math/msqrt/msup/mn[text()="2"]
math/msqrt/mo[text()="+"]
math/msqrt/msup/mi[text()="y"]
```

Fig. 2. An example of MathML description, dom and XPath expression

It stores the ID of the document with these indices as keys. In this step, the system does not include the following tags in the indices because they are not relating to the content of the formula: <mrow>, <mstyle>, <semantics>, <annotation>. Note that <mrow> represents the horizontal location of the formula at the time of rendering, <mstyle> represents the rendering attribute like color and font, <semantics> and <annotation> represents the additional semantic annotation to the formula. Table 1 shows an example of inverted indices. The first row shows that the document 1, 52, 70, 271, ... include the MathML object which has a fraction including the symbol " δ ."

Table 1. Examples of inverted index

XPath description	List of document ID
/math/mfrac/mo[text()=”\delta”]	1,52,70,271,,,
/math/msqrt/mfrac/mn[text()=”1”]	2,16,55,102,,,
/math/mfrac/mfrac/mi[text()=”\pi”]	22,93,181,,,

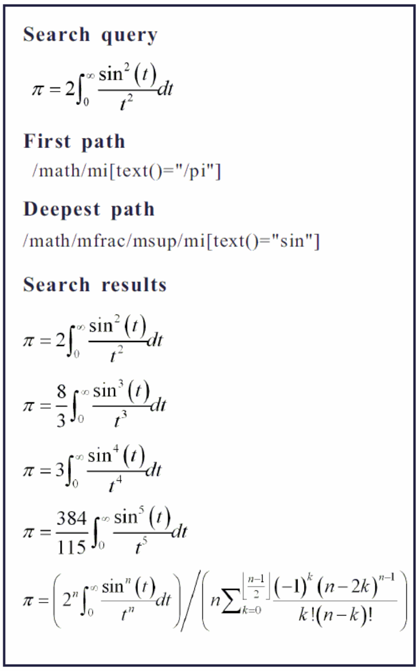
The system acquires the DOM structure of the given MathML object of the search query and acquires a key which is expressed as a specific path (Details are explained in the later section) in XPath. Then it acquires the document IDs whose indices match the key from the inverted file. For realizing a partial-match search, it just outputs these web document. For realizing an exact match, it checks all the paths (from the root node to the all leaf nodes) between the MathML object of the search query and the MathML objects in the acquired web document. It only outputs the web document in the search results which has a MathML object whose all paths match to all the paths of the MathML object of the search query.

3 Experiment

We think that using only one path for an index is not enough for narrowing down the search result. However, if we make an index from all the paths of the MathML object, the inverted file becomes large and many MathML objects are associated to one index. Therefore, we decided to select two paths from all the paths in the MathML object and generate an index by combining these paths. We built a search engine which introduces the above indexing method and conducted an experiment for the evaluation.

Table 2. Examples of created inverted index in the experiment

Description combining two XPath descriptions	List of document ID
/math/mi[text()=”cot”],/math/mover/ mi[text()=”\infty”]	47206,47208, 47276,,,
/math/munderover/mo[text()=”\Sigma”], /math/msup/mfrac/mn[text()=”1”]	850,1211,1379,,,
/math/mo[text()=”\int”], /math/mfrac/msqrt/msup/mi[text()=”sin”]	18679,39824,39839,,,



Search query

$$\pi = 2\int_0^{\infty} \frac{\sin^2(t)}{t^2} dt$$

First path

/math/mi[text()="/pi"]

Deepest path

/math/mfrac/msup/mi[text()="sin"]

Search results

$$\pi = 2\int_0^{\infty} \frac{\sin^2(t)}{t^2} dt$$

$$\pi = \frac{8}{3}\int_0^{\infty} \frac{\sin^3(t)}{t^3} dt$$

$$\pi = 3\int_0^{\infty} \frac{\sin^4(t)}{t^4} dt$$

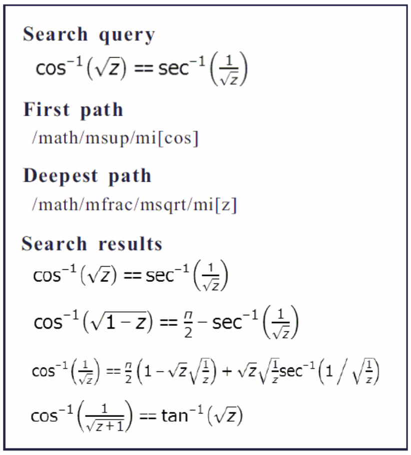
$$\pi = \frac{384}{115}\int_0^{\infty} \frac{\sin^5(t)}{t^5} dt$$

$$\pi = \left(2^n\int_0^{\infty} \frac{\sin^n(t)}{t^n} dt\right) \Big/ \left(n\sum_{k=0}^{\left\lfloor\frac{n-1}{2}\right\rfloor} \frac{(-1)^k (n-2k)^{n-1}}{k!(n-k)!}\right)$$

Fig. 3. Example (1) of the search result in the experiment

We used 87000 MathML objects provided by WOLFRAM RESEARCH [6]. We extracted all the paths in the DOM structure of each MathML object. We selected the first path and the deepest path and combined them as an index. By including the first path in the index, we can represent how the formula started (e.g. The formula expresses the definition or the equation). We think that the deepest path shows the most characteristic part in the formula. Table 2 shows examples of the created inverted file. The format is just a pair of the first path and the deepest path (expressed in XPath and separated by a commna). We selected one MathML object from the 87000 MathML objects and searched other MathML objects by our system. The search is conducted in the way of partial-match search. Figure 3 and 4 are examples.

In Figure 3, the XPath of the first path of the search query represents that the formula starts with π and includes the power of $\sin$. We can see that the search results include the formula with the square of \$\sin\$ and the cube of $\sin$. Also we can see that it includes the generalized formula that represents the n-power of $\sin$. The indexing methods using the first path and the deepest path can realize a similarity search by keeping the same structure with the search query. Figure 4 is an example with bad results. In Figure 4, the XPath of the first path of the search query represents that the formula starts with the square of \$cos\$. Actually, all the search results include the square (inverse) of \$cos\$. However, the whole mathematical structure of the third search result is different from the search query. The XPath of the deepest path of the search query represents that the formula includes the fraction with the square root of \$z\$. The third search result includes this structure, but there are two terms in the right-hand member. This is quite different from the search query. We notice that using the vertical structure in the DOM tree cannot solve this type of problem.



Search query

$\cos^{-1}(\sqrt{z}) == \sec^{-1}\left(\frac{1}{\sqrt{z}}\right)$

First path

/math/msup/mi[cos]

Deepest path

/math/mfrac/msqrt/mi[z]

Search results

$\cos^{-1}(\sqrt{z}) == \sec^{-1}\left(\frac{1}{\sqrt{z}}\right)$

$\cos^{-1}(\sqrt{1-z}) == \frac{\pi}{2} - \sec^{-1}\left(\frac{1}{\sqrt{z}}\right)$

$\cos^{-1}\left(\frac{1}{\sqrt{z}}\right) == \frac{\pi}{2}\left(1 - \sqrt{z}\sqrt{\frac{1}{z}}\right) + \sqrt{z}\sqrt{\frac{1}{z}}\sec^{-1}\left(1 / \sqrt{\frac{1}{z}}\right)$

$\cos^{-1}\left(\frac{1}{\sqrt{z+1}}\right) == \tan^{-1}(\sqrt{z})$

Fig. 4. Example (2) of the search result in the experiment

4 Incorporation of Breadth-First Search

We propose an indexing method of MathML object using the horizontal structure of the DOM tree. In detail, we use the tag names of the sibling nodes in one level of the

DOM tree (After here, "breadth-first search index (BFS index"). We compare the BFS index with the index which use the first path and the deepest path explained in the previous section (After here, "depth-first search index (DFS index)"). Ideally, BFS index represents the whole structure of the mathematical formula.

Figure 5 represents the sibling nodes in the same level in the DOM tree. In this experiment, we use the level which includes more than three nodes for the first time when checking the number of nodes in each level from the root level. In Figure 5, we use the sibling nodes for indexing this formula in the level surrounded by the red dashed line. Table 3 shows examples of BFS index.

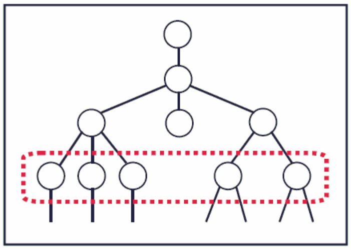
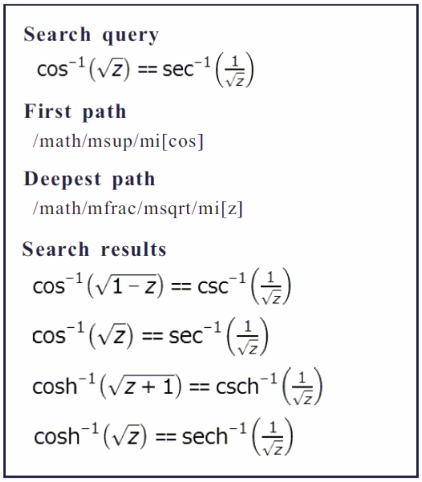


Fig. 5. Sibling nodes to be used for making a BFS index

Table 3. Examples of created BFS index

Description of sibling nodes	List of document ID
<msubsup><mo><mfrac><mfrac>	1,29,33,204,,,
<mfrac><mo><mfrac><mo><mrow>	2,8,93,181,,,
<msup>Mmrow><mo><mrow>	3,64,226,,,



Search query

$\cos^{-1}(\sqrt{z}) == \sec^{-1}\left(\frac{1}{\sqrt{z}}\right)$

First path

/math/msup/mi[cos]

Deepest path

/math/mfrac/msqrt/mi[z]

Search results

$\cos^{-1}(\sqrt{1-z}) == \csc^{-1}\left(\frac{1}{\sqrt{z}}\right)$

$\cos^{-1}(\sqrt{z}) == \sec^{-1}\left(\frac{1}{\sqrt{z}}\right)$

$\cosh^{-1}(\sqrt{z+1}) == \operatorname{csch}^{-1}\left(\frac{1}{\sqrt{z}}\right)$

$\cosh^{-1}(\sqrt{z}) == \operatorname{sech}^{-1}\left(\frac{1}{\sqrt{z}}\right)$

Fig. 6. Example (3) of the search result in the experiment

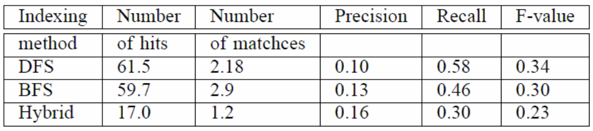
Figure 6 shows the search results of BFS index whose search query is the same as Figure 4. We can see that formulas whose whole structure is different from the search query were deleted in the results. We think that combining BFS and DFS realizes the more efficient mathematical search.

5 Comparing Three Kinds of Indexing Methods

We compare the three kinds of indexing method: (i) DFS index, (ii) BFS index and (iii) a hybrid index of DFS and BFS. We use 550 mathematical documents which we collected from the high-school mathematical teaching material provided from Ikushinsya. For creating the correct dataset of the search result, we invite 6 users to give the correct search results to the query set (55 queries).

Table 4 shows the result. DFS method outputs many number of hits and achieves the high recall. Hybrid method narrows down the search result and achieves the high precision. When we see F-values, DFS achieves the best and hybrid method ahieves the worst. When we do not have a ranking algorithm, hybrid method is the best becaues the precision is the best. However, when we have a ranking algorithm, the results will be presented in the descending order of the ranking score. In that case, DFS is the best because the probability will be down that the users miss important mathematical documents.

Table 4. Results of comparing three kinds of indexing method



Indexing	Number	Number	Precision	Recall	F-value
method	of hits	of matchces			
DFS	61.5	2.18	0.10	0.58	0.34
BFS	59.7	2.9	0.13	0.46	0.30
Hybrid	17.0	1.2	0.16	0.30	0.23

6 Conclusion

In this research, we built a search engine for the MathML object by introducing the DOM structure and the inverted file. Firstly we investigated the paths in the DOM structures of MathML objects for generating an index. We found that combining the first path and the deepest path represents the characteristics of the formula well. We conducted an experiment to see what kind of results will be produced by using this index type. The results showed that in some cases it produces the search results which include only the mathematical formula with the same structure. However, it also showed that in some cases it produces the search results in which the whole structure is different from the search query. We also propse an indexing method which uses the horizontal structure in the DOM tree (sibling nodes in the same level in the DOM tree) and the hybrid method using vertical and horizontal structure. When we conducted an experiment, we found that the hybrid method achieves the best precision and the DFS achieves the best F-value. We will work on the ranking mechanism in the future.

References

1. Kopka, H., Daly, P.W.: A Guide to LaTeX, 3rd edn. Addison-Wesley, Harlow (1999)
2. W3C: M3C Math Home (1998), `http://www.w3.org/math/`
3. Nakanishi, T., et al.: An Implementation Method of Composite Association Retrieval System for Data of Mathematical Formulas. DBSJ Letters 4(1) (2005)
4. W3C: M3C DOM Home (1998), `http://www.w3.org/dom/`
5. W3C: M3C XPath Home (1999), `http://www.w3.org/TR/xpath`
6. WOLFRAM Research: MATHML CENTRAL (2007), `http://www.mathmlcentral.com/`
7. NICT: The EDR Electronic Dictionary (1999), `http://www2.nict.go.jp/r/r312/EDR/index.html`
8. Kohno, H., et al.: Search Engine and Agent. University of Tokyo Press (2002)
9. Kita, K., et al.: Information Retrieval Algorithm. Kyoritsu Publisher (2002)