Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 859157, 5 pages
http://dx.doi.org/10.1155/2014/859157

Research Article

A Novel Mathematical Formula for Retrieval Algorithm

Yuping Qin,[1] Hamid Reza Karimi,[2] Aihua Zhang,[1] and Qiangkui Leng[3]

[1] *College of Engineering, Bohai University, Jinzhou 121013, China*
[2] *Department of Engineering, Faculty of Engineering and Science, The University of Agder, 4898 Grimstad, Norway*
[3] *College of Computer Science, Beijing University of Technology, Beijing 100124, China*

Correspondence should be addressed to Yuping Qin; qlq888888@sina.com

Received 6 January 2014; Accepted 23 January 2014; Published 26 March 2014

Academic Editor: Xudong Zhao

A method is proposed to retrieve mathematical formula in LaTeX documents. Firstly, we represent the retrieved mathematical formula by binary tree according to its LaTeX description, normalize the structure of the binary tree, and obtain the structure code and then search the mathematical formula table that is named by the structure code and the formula elements of the first two levels of the binary tree in the mathematical formula database. If the table exists, then we search the normalizing variable name preorder traversing sequence of the binary tree in the table and display the document information that contain the mathematical formula. The experimental results show that the algorithm realizes the retrieval of mathematical formula in LaTeX documents and has higher retrieval precision and faster retrieval speed.

1. Introduction

With the rapid development of the internet and digital libraries, more and more documents that contain mathematical formulas are stored on the computer. In order to share and communicate these documents quickly, online retrieval for mathematical formulas has attracted much attention and has become an important research area.

The retrieval technology for text already is relatively mature [1–7]. However, how to effectively retrieve mathematical formulas in documents is still an ongoing research issue [8]. And some control ideas, such as data driven [9–13] and system switch [14–17], have also been employed for this. Lee and Wang [18] presented a system of mathematical formula reorganization, but this system cannot handle multiline mathematical formulas, as well as more complex single-line ones. Fateman et al. [19] designed a system of mathematical formula reorganization, but the system can only reorganize integral tables with fixed format. Zanibbi et al. [20–22] proposed methods that can achieve good results for scanned images of the formulas and support automatic evaluation of recognition performance. Nonetheless, the methods cannot analyze the expression with two or more modifiers. MatheReader [23] can recognize more kinds of mathematical expressions; however, it still does not reach the degree of practical application.

The description methods of mathematical formulas mainly include MathML, LaTeX, and image. Among them, LaTeX has been widely used to edit scientific papers, books, files, dissertations, manuscripts, personal letters, and a variety of complex symbolic formulas. In addition, other format documents can be easily converted to LaTeX format. Therefore, a method is proposed to retrieve mathematical formula in LaTeX documents.

The rest of the paper is organized as follows. Section 2 gives the binary tree description of mathematical formula. Section 3 introduces the design of database. Section 4 describes our mathematical formula retrieval method in detail. Experimental results are presented in Section 5. Conclusion is outlined in Section 6.

2. Binary Tree Representation of Mathematical Formula

2.1. Construction of Binary Tree. Due to the noticeable structural feature, a complicated mathematical formula in LaTeX form can be divided into multiple subexpressions and then

Table 1: Data structure of a binary tree.

Field	Data type	Meaning
Formula element	String	Operator, variable, or constant
Category	String	OPS (satisfying the commutative law), VAR (variable), OPU (not satisfying the commutative law), and CON (constant)
Priority	Integer	Operator priority (the larger the value is, the higher the priority is; maximum machine number if priority is for variables and constants)
Combination	String	LR (left-right), UD (up-down), and SG (single)
Node height	Integer	The height of the binary tree that regards current node as root
Structure code	String	Structure code of current node = structure code of its left child + node height + its structure code of right child

each subexpression can be divided into much smaller ones. We repeat the procedure until no collapsible component is left. The final subexpressions are called formula elements.

The operator has three operands, such as "$\sum$," which has a close relationship with its top region, bottom region, and right region. We combine it with the right subexpression by adding an operator "link."

We traverse the formula element string with "link" from left to right to generate the priority list of formula elements and then the binary tree representation of a mathematical formula can be obtained according to its structural feature and the priority list. The data structure of the binary tree is given in Table 1.

We use recursion approach to get the binary tree representation of a formula element. Root, the lowest priority element, is first created and then we create the left subtree according to the elements before the root element in the formula element string. Accordingly, the right subtree can be created by the elements after the root element in the formula element string.

For each node, its element category and combination can be determined by the formula element. The height of each node can be calculated by the following:

$$H(\text{node}) = H_r > H_l ? \quad H_r : H_l + 1, \tag{1}$$

where $H(\text{node})$ is the height of node, H_l is the height of left child of node, and H_r is the height of right child of node.

For example, for mathematical formula $(\sum_{i=1}^{10} a^i + x \times y \times z) \times (x \times y + y \times z)$, its LaTeX form is (\sum_[i=1]^[10]a^i+x\times y\times z)\times(x\times y+y \times z). The corresponding binary tree representation is given in Figure 1.

2.2. Normalization Processing. Due to the fact that some operators satisfy the commutative law, that is, for these operands, one can exchange them randomly for constituting different mathematical expressions; the meanings of these expressions are identical. But it is worth noting that the structures of the corresponding binary trees are likely to be different. Hence, the normalization must be done for differently structural but identically meaningful binary trees. We traverse the binary tree in preorder, if the category of the formula element is OPS and the height of left child is higher than that of right child, then exchanging the left subtree and right subtree of the node. Figure 2 shows the normalized binary tree corresponding to Figure 1.

After normalizing the binary tree, the structure code of every node can be generated by traversing the binary tree in postorder. The structure code of node "node" can be obtained according to the following:

$$C(\text{node}) = \begin{cases} C_l(\text{code}) + H(\text{code}) + C_r(\text{code}) & \text{nonleft node} \\ 1 & \text{left node,} \end{cases} \tag{2}$$

where $C_l(\text{node})$ is the structure code of left child of node and $C_r(\text{node})$ is the structure code of right child of node.

Note that variable names of mathematical expression are independent of the formula meaning. For a given structure binary tree, we can get its corresponding sequence of the formula elements according to given traversal order. To make the sequence unique, we still need to normalize all the variable names in the sequence. The normalization approach is to use a fixed set of variable names to successively replace each formula element labeled "VAR" in the formula element sequence.

3. Database Designing

Retrieval database of mathematical formulas contains two kinds of tables: one is document information table and the other is formula information table. Their structures are given in Tables 2 and 3. Naming rule for the formula information table is described as follows:

$$C(\text{root}) + E(\text{root}) + E(\text{left child of root}) + E(\text{right child of root}), \tag{3}$$

where $E(\text{root})$ is the formula element of root, E (left child of root) is the formula element of left child of root, and E(right child of root) is the formula element of right child of root.

Mathematical formulas with the same information, including structure code, formula element of root, formula element of the left child, and element of the right child, are stored in a table.

4. Retrieval Algorithm

For the retrieved mathematical formula, we create the corresponding binary tree representation by its LaTeX format, obtain the structure code after normalizing the structure of binary trees, and then search the formula information table

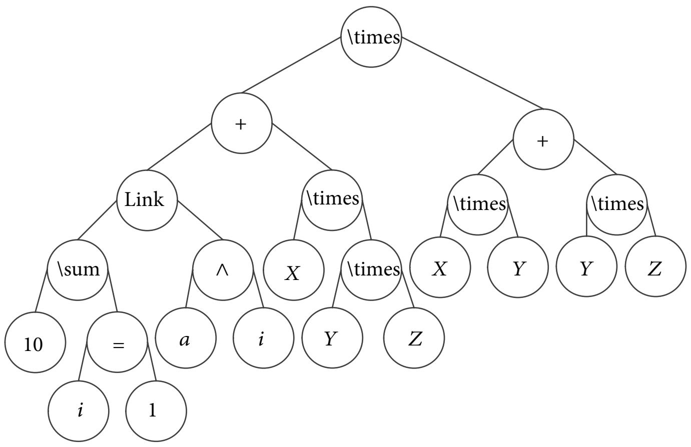

FIGURE 1: Binary tree representation of a mathematical formula.

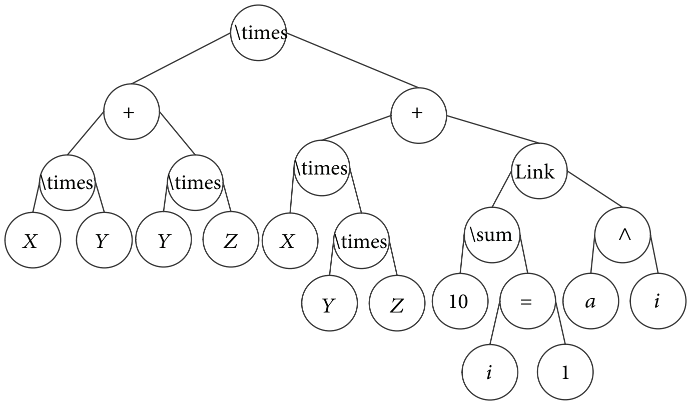

FIGURE 2: Normalized binary tree.

TABLE 2: Structure of document information table.

Field name	Type	Content
PAPER_ID	Text	Document code
PAPER NAME	Text	Document name
PAPER_IFO	Text	Document information

TABLE 3: Structure of formula information table.

Field name	Type	Content
FORMULA_ID	Text	Formula code
PAPER_ID	Text	Document code
NORMALIZED	Text	Normalized formula element sequence

named by structure code and the formula elements of the first two layers of the binary tree in the formula database. If the table exists, we find the preorder traversing sequence of the binary tree in the table. The retrieval algorithm is described in detail as follows.

Step 1. For a candidate testing LaTeX document, extract all mathematical formulas to get a retrieved formula set *Formula* = $\{f_1, f_2, \ldots, f_n\}$ and go to Step 2.

Step 2. If *Formula* is nonempty, then take out a formula f_i from *Formula*, create its binary tree representation, and normalize structure of the binary tree to get binary tree T_i. Traverse T_i in preorder and normalize variable names to get traversing sequence L_1 and go to Step 3; else, go to Step 8.

Step 3. Calculate the structure code of root according to (2). Let T_name be C(root) + E(root) + E(left child of root) + E(right child of root); go to Step 4.

Step 4. Search the data table named T_name in the formula database. If the table exists, go to Step 5; else, go to Step 7.

Step 5. $L = \{L_1\}$. For each nonleft node, if its element category is OPS and the heights of left child and right child are identical, exchange its left and right subtrees. Traverse the tree in preorder and normalize variable names to get the corresponding traversal sequence. If the sequence is not existing in L, then add the sequence to L. Finally, get formula element sequence set $L = \{L_1, L_2, \ldots, L_m\}$ and go to Step 6.

Step 6. Search the formula element sequence that is the same as L_i ($i = 1, 2, \ldots, m$) in the table. If it exists, output the document information containing formula f_i; else, go to Step 7.

Table 4: The way of modifying mathematical formula.

1	2	3	4
Not modifying	Modifying variable name	Exchanging operands that the operator satisfies commutative law	Modifying variable name and exchanging operands that the operator satisfies commutative law

Step 7. If the element category of the root is OPS and heights of its left child and right child are identical, exchange E(left child of root) and E(right child of root) in T_name and go to Step 3; else, go to Step 2.

Step 8. End.

5. Experimental Results

To verify the effectiveness of the proposed method on different types of mathematical formulas, we collect 1138 different mathematical formulas from 500 pressed research papers written in English and Chinese. We represent every mathematical formula by binary tree according to its LaTeX description, normalize the structure of the binary tree, and obtain the structure code. We save the preorder traversing sequence of normalizing variable name to the formula information table that is named by the structure code and the formula elements of the first two levels of the binary tree. We Save these documents information to the document information table at the same time.

The computational experiments were done on a Pentium 2.0 G with 2.0 MB memory, Windows XP SP3, and ACCESS 2007. The precision, recall, and F_1 values are used to evaluate the retrieval performance of the algorithm:

$$\begin{aligned} P &= \frac{A}{A+C}, \\ R &= \frac{A}{A+B}, \\ F_1 &= \frac{P \times R \times 2}{P+R}, \end{aligned} \tag{4}$$

where A is the number of mathematical formulas retrieved correctly in retrieval results, B is the number of mathematical formulas that should be retrieved but do not appear in retrieval results, and C is the number of mathematical formulas that should not be retrieved but appear in retrieval results.

To verify the performance of the proposed method, some mathematical formulas are modified according to Table 4.

In experiments, retrievals are done 2016 times; the average precision is 96.35%, the average recall is 95.38%, the average F_1 value is 96.86%, and the retrieval time is 378 ms.

The experimental results show that the proposed method obtains high retrieval accuracy. The key reasons are that the method realized semantic retrieval. If the semantic of retrieved mathematical formula is the same as the destination mathematical formula, then the corresponding structure of binary tree is uniform after normalizing the structure of the binary tree. Even if the destination mathematical formula exits in more than one binary tree representation, after normalizing variable names, at least one preorder traversing sequence of binary tree is the same as retrieved mathematical formula. The retrieval speed of the proposed approach is fast. The key reasons are that the method searches the table named by the structure code and the formula elements of the first two levels of the binary tree. If the table exists in the mathematical formula database, then to search the preorder traversing sequence of the retrieved mathematical formula in the table.

6. Conclusion

Based on the binary tree representation of mathematical formula, a mathematical formula retrieval method for LaTeX documents is introduced in this paper. Experimental results show that the algorithm not only realizes semantic retrieval of mathematical formula but also has higher retrieval precision and faster retrieval speed. The results achieved in the offline retrieval promise the proposed method will work in the online case as well. The disadvantage of the existing retrieval system is that it cannot retrieve mathematical formula in LaTeX documents when it is solved. How to retrieve mathematical formula in PDF documents and WORD documents would be our research work in future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This study is partly supported by the National Natural Science Foundation of China (no. 61304149), the Polish-Norwegian Research Programme (Project no. Pol-Nor/200957/47/2013), the Natural Science Foundation of Liaoning Province in China (no. 201202003), and the Program for New Century Excellent Talents in University (no. NCET-11-1005).

References

[1] T. C. Hoad and J. Zobel, "Methods for identifying versioned and plagiarized documents," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 3, pp. 203–215, 2003.

[2] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe, "Collection statistics for fast duplicate document detection," *ACM Transactions on Information Systems*, vol. 20, no. 2, pp. 171–191, 2002.

[3] J. Zobel and A. Moffat, "Exploring the similarity space," *ACM SIGIR Forum*, vol. 32, no. 1, pp. 18–34, 1998.

[4] A. Si, H. V. Leong, and R. W. Lau, "CHECK: a document plagiarism detection system," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 70–77, 1997.

[5] J.-P. Bao, J.-Y. Shen, X.-D. Liu, and Q.-B. Song, "Survey on natural language text copy detection," *Journal of Software*, vol. 14, no. 10, pp. 1753–1760, 2003.

[6] J.-J. Zhao and X.-G. Hu, "A way to judge plagiarism in academic papers based on word-frequency statistics of paragraphs," *Computer Technology and Development*, vol. 19, pp. 231–233, 2009.

[7] N. Kang, A. Gelbukh, and S. Han, "PPChecker: plagiarism pattern checker in document copy detection," in *Text, Speech and Dialogue*, vol. 4188 of *Lecture Notes in Computer Science*, pp. 661–667, 2006.

[8] Y.-S. Guo, L. Huang, C.-P. Liu, and X. Jiang, "An automatic mathematical expression understanding system," in *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR '07)*, pp. 719–723, Parana, Brazil, September 2007.

[9] S. Yin, S. X. Ding, A. H. A. Sari, and H. Hao, "Data-driven monitoring for stochastic systems and its application on batch process," *International Journal of Systems Science*, vol. 44, no. 7, pp. 1366–1376, 2013.

[10] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.

[11] S. Yin, H. Luo, and S. Ding, "Real-time implementation of fault-tolerant control systems with performance optimization," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 5, pp. 2402–2411, 2013.

[12] S. Yin, G. Wang, and H. R. Karimi, "Data-driven design of robust fault detection system for wind turbines," *Mechatronics*, 2013.

[13] S. Yin, X. Yang, and H. R. Karimi, "Data-driven adaptive observer for fault diagnosis," *Mathematical Problems in Engineering*, vol. 2012, Article ID 832836, 21 pages, 2012.

[14] X. Zhao, X. Liu, S. Yin, and H. Li, "Improved results on stability of continuous-time switched positive linear systems," *Automatica*, 2013.

[15] X. Zhao, P. Shi, and L. Zhang, "Asynchronously switched control of a class of slowly switched linear systems," *Systems and Control Letters*, vol. 61, no. 12, pp. 1151–1156, 2012.

[16] X. Zhao, L. Zhang, and P. Shi, "Stability of a class of switched positive linear time-delay systems," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 5, pp. 578–589, 2013.

[17] X. Zhao, L. Zhang, P. Shi, and H. Karimi, "Robust control of continuous-time systems with state-dependent uncertainties and its application to electronic circuits," *IEEE Transactions on Industrial Electronics*, 2013.

[18] H.-J. Lee and J.-S. Wang, "Design of a mathematical expression recognition system," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, vol. 2, pp. 1084–1087, Montreal, Canada, August 1995.

[19] R. J. Fateman, T. Tokuyasu, B. P. Berman, and N. Mitchell, "Optical character recognition and parsing of typeset mathematics," *Journal of Visual Communication and Image Representation*, vol. 7, no. 1, pp. 2–15, 1996.

[20] R. Zanibbi, D. Blostein, and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455–1467, 2002.

[21] D. Martín-Albo, V. Romero, and E. Vidal, "An experimental study of pruning techniques in handwritten text recognition systems," in *Pattern Recognition and Image Analysis*, pp. 559–566, Springer, New York, NY, USA, 2013.

[22] H. M. Twaakyondo and M. Okamoto, "Structure analysis and recognition of mathematical expressions," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 430–437, Montreal, Canada, August 1995.

[23] J. M. Jin, H. Y. Jiang, and Q. R. Wang, "Mathematical expression recognition system: MatheReader," *Chinese Journal of Computers*, vol. 29, no. 11, pp. 2018–2026, 2006.