

# Equivalence Detection Using Parse-tree Normalization for Math Search

Mohammed Shatnawi  
Department of Computer Info. Systems  
Jordan University of Science and Tech.  
Jordan-Irbid (22110)-P.O.Box (3030)  
mshatnawi@just.edu.jo

Abdou Youssef  
Department of Computer Science  
The George Washington University  
Washington, DC 20052  
ayoussef@gwu.edu

## Abstract

*In recent years, efforts have begun to put math contents on the Web. As for other types of Web information, search capabilities should be provided to enable users to find what they need because without the ability to search the data for specific items, the data is useless. Conventional (i.e. text-based or even multimedia-based) search engines fall short of providing math-search capabilities. Preliminary efforts to create math-search systems have started, and many of the issues and the challenges for building such systems have been identified. One of the more difficult challenges is the detection of mathematical equivalence between expression in users' queries and expressions in math contents.*

*The purpose of this research is to develop techniques and algorithm for equivalence-detection based math search. In particular, this research aims to explore some proposed normalization rules, then to develop a general way that can be utilized to transform both the repository contents and users' input expressions into a unified normalized form.*

## 1.1 Introduction

Finding needed information on the Web is not easy to achieve with a high degree of accuracy. Information retrieval systems have been designed to help users locate and retrieve their requests on the Web. Information retrieval systems are composed of some algorithms that try to make the search and retrieval of the requested information as accurate and fast as possible.

Among all of these, "the text aspect has been the only data type that lends itself to a full functional

processing" [1]. Many algorithms that work together trying to refine text search have achieved a good level of maturity. Unfortunately those search engines did not achieve the same progress in terms of mathematical expression as a separate distinguished type of text

The major obstacle to math search in current text search systems is that those systems do not differentiate between a user query that contains a mathematical expression and any other query that contains text term. Therefore, they process mathematical expressions as other texts, regardless of its nature of being well-structured and having properties that make it different from other forms of text.

## 1.2 Accessing Math Expressions on the Web

There are many items that contain mathematical expressions in their content. Unfortunately, many of these items can not be accessed and retrieved by current search engines for the following reasons:

- Virtually all searches are text-based [14], thus, unless we have an agreed upon technique that is understood by both users and search engines, a user needs to know the best search terms and the best way to write a query to be used in searching for any mathematical expression.
- The same expression can be rewritten in many different but equivalent ways (e.g.  $1/x$  and  $x^{-1}$ ) [13] [12].
- Text-based search engines do not consider the syntax of a mathematical expression as one of its main features [13] [12].
- The way used to search for equivalent text terms (i.e. thesaurus to search for synonyms) is not feasible for searching for an equivalent mathematical expression [13].

### 1.2.1 Equivalence and Inconsistency Equivalence.

One major problem in being able to retrieve relevant items is the inconsistency between the author's vocabulary and the user's vocabulary. Therefore, the user may search for a term the author does not provide. This problem has been studied in text search and there are some proposed solutions such as searching for the synonyms during the search process using thesaurus lookup. A similar problem exists when you search for a mathematical expression because the term  $y+x$  is the same as  $x+y$  mathematically, and  $0.5$  is the same as  $1/2$ . This problem adds another obstacle that makes the current search engines fail in retrieving items that contain mathematical expressions. More precisely, the same mathematical expression can be represented in many (sometimes infinite) numbers of ways; thus, it is not feasible to use a thesaurus structure to search for all equivalent expressions.

Even if the current search engines are equipped with tools to enhance their ability in retrieving items that contain a certain type of a mathematical expression, they will still fail in retrieving the documents that contain variants of that mathematical expression. Therefore, there is a need for a way to retrieve the documents that contain not only the expression itself but also the expression's equivalent forms.

### 1.2.2 Syntax Interpretation

Another important reason that makes current search engines fail in retrieving mathematical expressions is that search engines do not understand mathematical structures but they well-understand text because a word in an unstructured text is simply a word with no data type definition and no conceptual definition.

Mathematical expressions are well structured and the structure itself holds their correct interpretations.

### 1.3 Relation with Theorem Proving Systems

Generally, in theorem proving we want to verify whether some statement (the conjecture) is a logical consequence of a set of statements (e.g. axioms and hypotheses) [5]. In particular, theorem proving concerns itself with proving whether two given mathematical expressions are equivalent.

Our problem is different from theorem proving in that we have only one mathematical expression as

input and we need to find its many equivalent forms as our output after applying rules of equivalence that have been fed to the system based on predefined Grammar of Equivalence Rules (GER) on the input expression.

### 1.4 Objectives

The objective of this research is to design and implement an effective and reliable technique that transforms a user input expression into a unique normalized form. This form will be used in searching for a mathematical expression in a way that takes into account its unique properties. The way that expressions are stored in the searchable database must be compliant with the way normalized expressions are interpreted.

### 1.5 Definition of Normalization

Normalization is a sequence of transformations that transforms an original expression form one algebraic/structural form into an equivalent one. According to this definition we divided the normalization into two types, algebraic and structural normalization.

In algebraic normalization, the process of normalization is done on the expression in its algebraic form. For example, the expression  $z+y+x$  will be normalized into  $x+y+z$ .

In structural normalization, the expression's parse tree structure will change after normalizing the mathematical expression. For this reason, we call it structural normalization.

### 1.6 Significant Contributions

This research makes two significant contributions to the field of math search.

- Introduction of a new approach for addressing the math equivalence and detection techniques.
- Development of a completely new method to discover different equivalent math expressions and map all of them into one normalized form.

## 2. A Mathematical Expression Parser (MEP)

The first step of our work is to create a Mathematical Expression Parser (MEP), which creates a parse tree for a certain mathematical expression.

### 2.1 Equivalence Detection and Normalization (EDN)

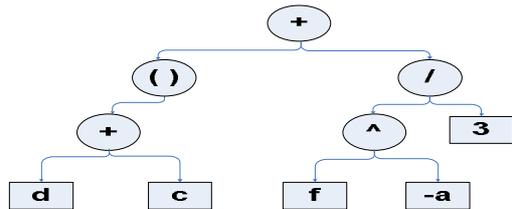
The equivalence detection and normalization is the most important part of our work. Indeed it is the core of our research. The EDN aims to transform the expression tree that we have created earlier using MEP into a normalized tree. This tree is equivalent to the original tree but it is an agreed upon representation, based on some rules, to facilitate the search process.

In the first part of this research and for better understanding of our research we propose four fixed rules that can be applied to the tree that results from the MEP. Therefore, after applying them as needed, we shall be able to get the final normalized tree form. After that, the last normalized tree is used for comparison and matching during the search process.

## 2.2 Group Removal Rule

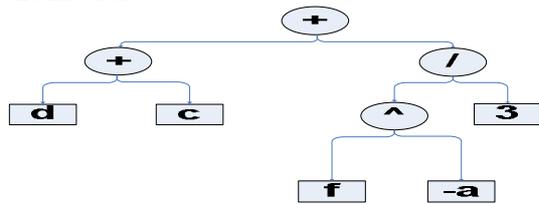
A mathematical expression is grouped if it appears between left and right parentheses.

It is obvious that the above expression can be transformed to the following expression tree using MEP: Figure 1: Tree representation of  $(d+c)+f^a/3$  before rule one



The parse tree after applying this rule is depicted in figure 2.

Figure 2: Tree representation of  $(d+c)+f^a/3$  after the first rule



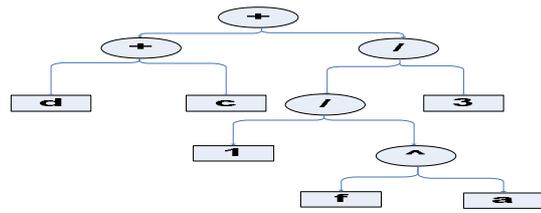
## 2.3 to the Negative Power Rule

The previous example has "to the negative power" sub expression (i.e.  $f^{-a}$ ). This part can be transformed to an equivalent expression by using the following mathematical rule:

- $x^{-y}$  is equivalent to  $1/x^y$

Therefore, according to this rule, the previous expression should be transformed to  $d+c+1/f^a/3$

Figure 3: Tree representation for  $(d+c)+f^a/3$  after the second rule

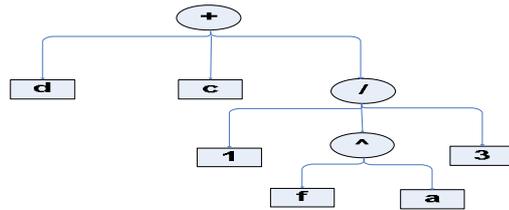


## 2.4 Tree Height Compression

In this section we will follow the same procedure of decreasing the height of the tree by applying the rule of Height Compression. This rule works as follows:

All the similar parent nodes that are descending from the same node are combined with lowest level parent node. Therefore, the leaves will be children of that common node given that the parent of each of those leaves will not change but their level will be changed after applying this rule. All of the above can be illustrated more by applying this rule on our example in figure 5. Therefore, the tree now would look like the following tree:

Figure 4: Tree representation for  $(d+c)+f^a/3$  after the third rule



## 2.5 Tree Reorder Rule

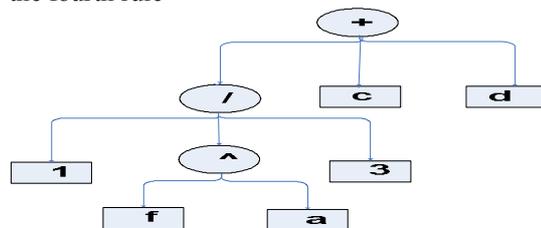
Sorting or reordering the leaves is done by following a user defined rule of reordering. For example, we proposed our defined rule, which is:

Numbers < Alphabetic (string, character)  
< Operations (\*, +) < Grouped Parenthesis

Since we proposed the above rule, this does not mean that other users can not propose their own rule. But we have to apply the same proposed rule consistently on both the user query and the searchable database.

After applying this rule, the expression tree looks like the following:

Figure 5: Tree representation for  $(d+c)+f^a/3$  after the fourth rule



### 3. Grammar of Equivalence Rules

A Grammar is used to generate an infinite set of valid mathematical equivalence rules (e.g.  $x^2$  mathematically equivalent to  $1/x^2$ ). The grammar rules will impose some desired structure on the equivalence rules; the system administrator should follow this structure in order to add a valid mathematical equivalence rule to the GER. Moreover, our system should not accept an invalid rule (i.e. a rule that does not comply with the grammar rules).

#### 3.1 Syntax of GER Rules

The basic syntax for the rules in GER would be in the form of:

$E: E$  ( $E$  is a non terminal symbol which represents a mathematical expression)

The left hand side of the ":" operator is the expression before applying certain rule of equivalence, while the right hand side of the ":" operator is the expression after applying certain rule of equivalence.

#### 3.2 GER's Grammar: Formal Definition

The grammar that we will start explaining is a Context Free Grammar (CFG), where every production rule is in the form  $V \rightarrow w$  where  $V$  is non-terminal symbol and  $w$  is a string consisting of terminals and/or non-terminals.

Our grammar  $G$  is a quadruple  $(T, N, S, R)$ , where:

$T$  is a finite set of terminal symbols,

$N$  is a finite set of non-terminal symbols,

$S$  is a unique starting symbol.

$R$  is a finite set of productions of the form  $\alpha \rightarrow \beta$ , where  $\alpha$  and  $\beta$  are strings of non-terminals and terminals.

We build the following grammar to start with for our normalization system, we use the notational shorthand "|", which can be read as "or", to represent multiple production rules within a single line:

$G = \{T, N, S, R\}$   
 $T = \{0, 1, 2... 9, -, \$, \#, \cdot, /, \wedge, \text{^}\}$ ,  
 $N = \{S, E, T, F, B, D\}$ ,  
 $R = \{S \rightarrow E: E,$   
 $E \rightarrow E+T \mid E-T \mid T, T \rightarrow T * F \mid T / F \mid T \wedge F \mid F,$   
 $F \rightarrow (E) \mid B, (i.e. B stands for basic term)$   
 $B \rightarrow \$D \mid -\$D \mid \#D \mid 0 \mid 1 \mid -1, D \rightarrow 0...9 \mid DD\}$

Our normalization system will be built based on the above basic grammar. Based on that grammar one can expand it to include many valid mathematical equivalence rules.

#### 3.3 Tree Compression Rule's Grammar

This rule has a distinct grammar in order to make it easier to add such those rules (i.e. tree compression rule for different operators). Tree Compression rule goes under the structural normalization; the grammar for this rule will be as follows:

First of all, the general rule format that is discussed earlier is a little different form this rule format, which should be specified as follows:

$E: G$

Here, in the tree compression rule, the structure of the tree is going to be changed after applying any form of the above rule. Therefore the right sub expression (i.e. the one after applying this rule) has a different structure from the left sub expression.

The grammar for this rule is explained as follows:

$T = \{0, 1, 2... 9, \$, -\}$ ,

$N = \{S, E, T, F, B, G, K, D\}$ ,

$R = \{S \rightarrow E: E \mid E: G, E \rightarrow E+T \mid E-T \mid T,$

$T \rightarrow T * F \mid T / F \mid T \wedge F \mid F, F \rightarrow (E) \mid B, B \rightarrow \$D \mid -\$D$

$D \rightarrow 0...9 \mid DD, G \rightarrow +GK \mid -GK \mid /GK \mid \wedge GK \mid (G) \mid K \mid KK, K \rightarrow B$

Based on the above grammar, the system administrator can use to add the following tree compression rules:

#### 3.4 Generic Normalization

The normalization system that is built based on GER is termed *generic normalization*. Based on GER the system administrator should be able to add any valid mathematical equivalence rules, our normalization system should be able to detect equivalence for those added rules. We have developed algorithms that detect equivalence for any added rule that conforms to the grammar; any added rule to the generic normalization system is derived from a general principle in which a rule is admissible if and only if there is a corresponding transformation on the parse-tree [9].

Our universal normalization algorithm's idea is straight forward and based on the idea of pattern matching. The algorithm scans the input expression looking for a match with the left hand side of a rule so a rule can be applied on that expression (i.e. or sub-expression).

#### 3.5 Rule Validation

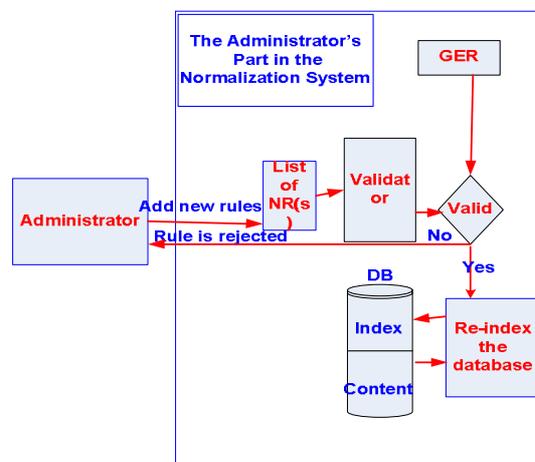
Our normalization system should not allow the system administrator to add an equivalence rule that does not comply with the specified format. In other words, if a rule does not comply with the format that we have specified earlier and/or does not comply

with the above GER grammar, the system rejects the rule.

The system administrator should have the basic mathematical knowledge in order to avoid adding an invalid mathematical equivalence rule, or the system administrator should have a trusted mathematical reference to refer to, in order to verify the equivalence rule correctness.

The Validator is a component of our normalization system that is responsible for validating the correctness of any equivalence rules that is added by a system administrator. The Validator verifies if the added rule is compliant with the GER grammar format. This validation process is done using a compiler compiler such as javaCC.

Figure 6: General Normalization System Based on GER (Administrator's Part)



#### 4. Performance Analysis

Measuring the performance of any newly developed system is required to evaluate its effectiveness and to compare it with other systems.

The major problem in measuring the performance of math search systems is the lack of any math query benchmark because this area is relatively new. In the absence of an agreed upon query benchmark, the performance of our normalization system is based on the searchable database content. Therefore, the result of a certain search using the same set of normalization rules on two different database contents results in two different outcomes [3].

The main goal of the normalization system is to increase the number of true hits when a user searches for a math expression. Therefore, after applying a set of normalization rules on both; a certain type of database content and a user math query, this process will result in new math expressions which will not be founded without applying that set of normalization rules. The following examples will clarify the above concepts.

Suppose the database content has the following math expression:

$$(a^b)^c + k^{-g}$$

And the user searches for  $a^{(b*c)} + 1/k^g$  or the user searches for part of the previous expression (*i.e.*  $a^{(b*c)}$  or  $1/k^g$ ). In this case, without applying any kind of normalization rules, the searches does not retrieve the expression  $(a^b)^c + k^{-g}$  since this expression does not match the user request or the search may retrieve many irrelevant items before it retrieves the relevant one. The search engine uses the techniques for text retrieval and probability of occurrences (*i.e.* a, b, c, k, and g may not achieve the required threshold to be retrieved as a result of the user search).

There are two normalization rules that have been accepted and added to a list of normalization rules to be applied on the database content and on the user math query. These normalization rules are:

- 1-  $(a^b)^c : a^{(b*c)}$
- 2-  $a^{(-b)} : 1/a^b$

If the above two rules were applied, the database content and the user query will be normalized according to them. Therefore, the database content will be normalized to  $a^{(b*c)} + 1/k^g$ .

In case of a complete database and enough normalization rules, the number of relevant retrieved items will be increased (*i.e.* precision will be increased). Some of the items that would not be retrieved without normalization, normalization increases the chance for such those items to be retrieved, therefore, the recall will be effected positively as well [2][1].

#### 5. Conclusion

This research shows that we have achieved some progress in searching for a mathematical expression (*e.g.*  $y+x$ ). In our research we focused more on mathematical expression search process in terms of search engines and the Web search issues.

After applying the normalization and equivalence rules, the recall and even precision of our search will be increased. Since we are transforming different equivalent mathematical expression into a common form, this common form will be compared against the searchable database, which contains the normalized form of that expression as well. According to that, the comparison process will end up finding most of the items that have the common mathematical expression [2][1].

According to the above, our research is good in terms of enhancing the mathematical expression web search process. This way of enhancing is done by using GER in which a system administrator can add

different kind of normalization rules based on the predefined grammar. This normalization system transforms a user input, which is a mathematical expression, to a normalized unique form. The latter is equivalent to the original user input. In order to transform the input expression into its normalized form the system applies a set of rules on the input expression.

## 6.Future Work

Much remains to be done. The following is a small list of possible directions of future work:

- Once digital libraries of mathematics (*e.g.* the DLMF of NIST [13]) become available and “standard” benchmark mathematical queries have been developed and accepted, it will be logical to measure the improvement in recall (and precision) that normalization brings to math search.
- Quantification of the performance improvement of each added equivalence rule.
- Testing on human subjects in various science/math communities and at various professional levels which equivalence rules are helpful and which would be confusing.
- Expanding the grammar of equivalence rule by adding more operations.
- Relevance ranking can be adjusted to reflect:
- How widely recognized is the equivalence rule that caused the matching and
- The profile of the users.

## 7. References

- [1] Kowalski, Gerald J., Maybury, Mark T. "Information Storage and Retrieval Systems: Theory and Implementation", Springer, 2nd edition, 2000.
- [2] Precision/Recall, <http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html>
- [3] Joydeep Ghosh “Performance Evaluation of Information Retrieval Systems”, <http://www.cs.utexas.edu/users/mooney/ir-course/slides/Evaluation.ppt>
- [4] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine”, Proceedings of the 7<sup>th</sup> international conference on World Wide Web, Brisbane, Australia, 1998.
- [5] Automated Theorem Proving, <http://www.cs.miami.edu/~tptp/OverviewOfATP.html>
- [6] Semantic Web, <http://www.w3.org/2001/sw/>
- [7] Thomas W. Parsons, "Introduction to Compiler Construction", Computer Science Press, W.H. Freeman and Company, 1992.

- [8] Geoffrey Weglarz "Two Worlds of Data – Unstructured and Structured", DM Review Magazine, September 2004.
- [9] Derivations and Parse Trees, <http://www.cs.nuim.ie/~jpower/Courses/parsing/node24.html>
- [10] Saracevic, T. “Relevance: A Review of and a Framework for the Thinking on the Notion”, Journal of the American Society of Information Science, 26(4), 1975, 321-343.
- [11] Baeza- Yates, R. and Ribeiro-Neto, B. “Modern information retrieval”, Reading, MA, Addison-Wesley, 1999.
- [12] Youssef, A. “Information Search And Retrieval of Mathematics Contents: Issues and Methods”, The proceeding of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005), July 20-22, 2005, Toronto, Canada.
- [13] Abdou Youssef, Bruce R. c "Technical Aspects of the Digital Library of Mathematical Functions, Annals of Mathematics and Artificial Intelligence, Volume38, pp. 121-136, 2003.
- [14] Robert Miner "Math Searching and MathML in the NSDL", presentation, 2004.