Wildcards in Math Search, Implementation Issues

Moody E. Altamimi
Department of Computer Science
The George Washington University
Washington, DC, 20052, USA
maltamimi@gmail.com

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

Abstract

Math search is a new area of research with many enabling technologies but also many challenges. Some of the enabling technologies include XML, XPath, XQuery, and MathML. Some of the challenges involve enabling search systems to recognize mathematical symbols and structures. Several math search projects have made considerable progress in meeting those challenges. One of the remaining challenges is the creation and implementation of a math query language that enables the general users to express their information needs intuitively yet precisely. A new query language that extends the current standard query syntax has been developed. An important feature of the math query language is the introduction of a powerful set of wildcards that are deemed important for math search. These wildcards provide for more precise structural search and multi-levels of abstraction. This paper will present three sets of wildcards and discuss their implementation details.

1 INTRODUCTION

The need to facilitate scientific information exchange between researchers has resulted in the creation of a growing number of specialized mathematical library projects around the world. These projects aim to make scientific literature available on the Web. With the increasing online availability of electronic documents that contain mathematical expressions, the ability to find relevant information has become increasingly important. Yet, support for searching for mathematical expressions is only in its infancy.

The development of math search capabilities is a new area of research with many technical challenges. Some of the challenges involve enabling search systems to recognize mathematical symbols and structures. Several math search projects have made considerable progress in meeting those challenges. Several research projects on math search have addressed many of the issues and challenges in math search. Notable among those math-search projects are the math search of the DLMF project at NIST [1], and the math search system of Design Science [2].

The query languages assumed or implemented in those systems follow primarily the same syntax as standard text search. That syntax consists mainly of Boolean query operators (i.e., "and", "or", and "not") and phrase operators. Phrase queries are important in math search since math expressions and fragments of expression are meant to be sequences of **consecutive** terms, that is, phrases. The standard syntax however, provides for a limited use of wildcards, namely, "?" and "*". The first stands for one arbitrary character inside a keyword, and the second stands for zero or more arbitrary characters inside a keyword.

Such wildcard syntax is severely limiting in math search. For example, there is a need for additional wildcards to stand for any number of rows or columns when searching for a matrix, and for wildcards to specify subparts of a mathematical expression. Also, if a user does not care what certain terms are (such as variable names) but cares that two or more of those terms are identical, the standard query syntax is inadequate to express such a need.

A new query language that extends the current standard query syntax has been developed [13]. The language describes the user's information needs by allowing the authoring of different types of queries and allowing the use of an expanded set of wildcards. The expanded set of wildcards will enable science and math users to specify their information needs in a more precise way to guarantee that the matches are more relevant to their needs. In addition to character-level wildcards, two new sets of wildcards are introduced. The new sets handle search at the parsed tree level of a mathematical structure. They will allow the user greater levels of expression.

This paper focuses on the powerful set of wildcards proposed by the query language in [13] and describes their implementation algorithms. The implementation of the language maps queries written in that language into XPath/XQuery queries [3, 4]. It is assumed that the math content is in Content MathML [5]. The justification for this assumption is based on current technological advances and expected future practices. For example, many conversion tools already exist for converting LaTeX to MathML, such as the Rice University tool for conversion to Content MathML [6], and Bruce Miller's LaTeXML and associated software [7], which convert from LaTeX to a special XML syntax that includes presentation MathML and some content mark up. Furthermore, as the

math authoring community becomes more comfortable with MathML and, more importantly, becomes more convinced of the need for and benefits of Content MathML, more conversion tools and authoring tools that yield Content MathML will become available and more dominantly used.

2 BACKGROUND AND RELATED WORK

This section surveys work related to equation-based math search systems and the user query languages they offer. Mainly query languages developed for the DLMF project, Design Science search system, and Mathematica, will be described and particular emphasis will be on their limited wildcard capabilities. The section also describes briefly our proposed query language in [13] to put the different sets of wildcards in context.

DLMF and Mathdex

Youssef et al. [8] developed the first generation of an equation-based math search system as part of the Digital Library of Mathematical Functions [1] (DLMF) project at NIST. The DLMF project provides an online source of mathematical content such as formulas and graphs, and allows for the search and retrieval of that content [8]. The mathematical content of DLMF, originally in LaTeX, is converted to html and xhtml using the LaTeXML markup language and software tool developed at NIST. Youssef, who is developing the search system for DLMF, opted for an evolutionary approach, building on the existing text search technology. As a result, the query language syntax is almost identical to text search syntax, with the added power of recognizing mathematical symbols and structures to a great extent.

Mathdex [2] is a web-based search engine developed by Design Science [9] as part of an NSF grant to facilitate equation-based search. Mathdex indexes not only LaTeX but also Presentation MathML, and it crawls the Web looking for Math contents and indexing them. Like the DLMF search, Mathdex follows an evolutionary approach by utilizing text search technology.

Even though text search technology has reached a high level of maturity, it cannot fully capture all of the characteristics inherent in mathematical content. As a result, the query language developed for the DLMF project has limited expressive power when the user is trying to look for patterns within mathematical structures. For example, if a user wants to look for an expression where "x^2+1" is somewhere in the denominator without caring exactly in the denominator, a user should be able to write the following query "/(…x^2+1…)" using the ... wildcard; unfortunately, this is not possible using the DLMF search system or Mathdex. As another example, consider the situation where a user wishes to specify a query that contains cos^2 x+sin^2 x and indicate that the variable x could be any other symbol, the best that can be done currently is to write cos^2 $+sin^2 $, where "$" is a wildcard that stands for any arbitrary string of characters; this, however, fails to enforce that the two wildcards must stand for the same variable name.

Mathematica Search

Wolfram research offers a large online repository of mathematical functions and formulas [10] encoded in different formats: Mathematica's standard format, MathML and ASCII. Mathematica offers an experimental search tool that allows the user to specify the terms (functions, numbers, constants, operations) in the query, using drop down menus. The user can also specify options to filter the search results based on function types (elementary functions only or integer functions only). The use of drop down menus allows the user to create more complex queries by using Boolean AND and OR. The user is able, however, to search for Mathematica patterns using the Mathematica language, which eliminates ambiguities inherent in mathematical notation. The language covers a wider range of mathematics than our proposed math user query language. However, it only offers basic support of structural search and wildcards, and thus suffers the same limitations of text-based search systems mentioned above.

An Extensive Math Query Language

Mathematical notation can be ambiguous and cumbersome to type. The math query language presented in [13] offers an alternative way to describe mathematical expressions that are more consistent and less ambiguous than conventional mathematical notation. The syntax is intuitive and covers notation that is commonly used when possible. However, the language is by no means complete. The language focuses on mathematical notation that is commonly used and supported by Content MathML 2.0 in the areas of arithmetic, algebra, calculus, etc. In addition, syntax to handle notation that Youssef and others didn't cover but will benefit the math search community is developed for matrices, ordinary and partial differential equations, integration, and function composition. More importantly, a more comprehensive set of wildcard symbols is introduced, which will enable the creation of more complex queries that specify subparts of an expression, and which will provide for more support of abstraction and structured search. The end result is an ASCII language that makes unambiguous use of symbols and has a well-defined grammar.

As mentioned earlier, there are additional features and syntax that are possible with the XML-query approach but not easily implementable by text-IR approach. Table 1 illustrates that point with a few examples. For a more detailed description of the characteristics of our proposed query language, please refer to [13].

Table 1. User math queries in different areas of math

Type	Query Example and Explanation
Matrices	Matrix[;;...a,b...] Look for a,b somewhere in the third row of a matrix. In the context of matrices, the comma (,) is used to separate columns while the semi-colon(;) is used to separate rows.
Partial Differentiation	D_{x^2,y^3}(expr) Look for the fifth partial derivative of an expression for 2 times with respect to x and 3 times with respect to y
Function Composition	f\og Look for f composed with g

3 WILDCARDS

Wildcards in the area of text search such as "*" and "?" stand for multiple characters and a single character within a single keyword. As argued earlier, these wildcards are very inadequate, and new wildcards are needed. For example, there is a need for additional wildcards to stand for any number of rows or columns when searching for a matrix, and for wildcards to specify subparts of a mathematical expression.

Two new sets of wildcards are introduced to help capture user needs in the area of math search: term-level wildcards and sequence-level wildcards. These two sets are discussed next; for completeness, the commonly used wildcards are described first.

Character Level (keyword search): This is the commonly used set of wildcards: "$" and "?". The wildcard "$" is the same as the more familiar "*", but it is used instead of "*" because the latter is used as the symbol for multiplication. The processing of wildcards at the character level is done through pattern matching algorithms.

Term Level (tree search): From the previous set, the $ can be used to stand for a single arbitrary term but this is not sufficient; wildcards that stand for a sequence of terms are needed. The following wildcards are introduced that allow the user to search for a sub-tree in the tree structure of mathematical expressions.

Table 2. Term-level wildcards

Symbol and its meaning
• **The wildcard "$":** when used alone, it stands for any arbitrary single term, which can be a variable, a number, a matrix name, a function name, an operator name, a set name, and so on. For example, f($) matches a function f with exactly one argument that is a single term. As a result, f(x) will be retrieved but not f(x^2). Another example, $ + 2 is a query that will retrieve 3+2 as well as x+2 but not (3*4)+2 since 3*4 is not a single term but a group of terms that form a sub-expression.
• **The wildcard "- -":** stands for one or more terms. This means something must exist, either a single term or a sub-expression. For example, f(x, --) will match a function f with exactly two arguments where the first is x and the second can be a single term or an expression. As a result, f(x, y) and f(x, y^2) will be retrieved but not f(x). Note, on the other hand, that f(x, $) matches functions f with two arguments where the second must be a single term. Another example, matrix(--,x+1) will look for x+1 in the last column in matrices that have exactly two columns. While matrix(--) matches any matrix with one row and one column irrespective of the nature of that entry. Another example, --+-- matches sin+cos, A+B, and matrix(a,b; c,d)+ matrix(1,2; 3,4) among others. Note: -- will also be used to support structural search, which will be discussed later.

Processing these two wildcards is relatively straightforward. When encountering $ the search will be for a term that is either an identifier or numeric value. When encountering -- the search is for the "existence" of a sub-expression (sub tree in the expression tree).

Sequence-level and parse tree-level wildcards: The two wildcards in this set will match zero-or-more, or one-or-more members in a sequence, or more generally, for zero-or-more/one-or-more nodes at the same level in a parse tree. A new wildcard that searches for exactly one member will not be considered since "--" from the previous set of wildcards can stand for a single member. (Note that a sequence can be a vector, a sequence of arguments of a function, a series, and such.) The two wildcards are … and .. described and illustrated in the next table.

The following example explains the need for wildcards in this set. If the content is f(x+5, 2+y+z, z^2-5*z) and the user wants to write a query that looks for a function f where the last argument is z^2-5*z and not really know the exact number of arguments, the previous set of wildcards will only allow for f(--,z^2-5*z). That query will not retrieve the content since "--" will stand for a single argument. Clearly there is a need for a new set of wildcards that allows the user to specify more than one argument. When the user enters f(.., z^2-5*z), the content will

be retrieved because the search will be for function f with at least two arguments where the last one is z^2-5*z.

Table 3. Sequence-level and parse tree-level wildcards

Symbol and its meaning
• **The wildcard "..":** stands for one or more consecutive members in a sequence, or one or more consecutive nodes at the same level in a parse tree. Examples: matrix(;..,x+1,..) will search for x+1 in the second row of the matrix irrespective of its column location. matrix(..) will search for matrices with a single column. matrix(a+--+b,c+--+d,..,15) will retrieve this vector matrix(a+f(2,3)+b, c+4+d, 12, 13, 15)
• **The wildcard "…":** stands for zero or more consecutive members in a sequence, or one or more consecutive nodes at the same level in a parse tree. Examples: f(x,…) matches a function f with at least one argument x, such as f(x,y), f(x,z^2,y) and f(x). matrix[;…,x+1,…] matches any matrix where x+1 is in the second row irrespective of its column location.

Algorithm for processing sequence-level and parse tree-level wildcards:
The following are the steps needed when processing a sequence of members. For example, matrix(a+--+b, c+--+d,.., 15) or f(x, .. , y , z , …)

When creating an XPath expression to search for a sequence of members, the search will be for those members at certain positions as specified in the user query. In the algorithm below, "position" is the specific position where the member should be in the math expression tree for that document to be retrieved.

1. if the sequence starts with a non-wildcard member:
   - create sub-XPath expression for that member setting its position to start position
   - until a wildcard is reached, create a sub-XPath expression for each following member setting their position relative to previous position
2. if the first member is a wildcard:
   - if .. set pos' to position+1
   - if ... set pos' to position
3. if other members follow
   - create sub-XPath expression for that member at position > pos'
   - until a wildcard or end of sequence is reached, create a sub-XPath expression for each following member so that it immediately follows the previous member
   - if end of sequence is reached, restrict the position of the last member to be the last member of the sequence
   - if wildcard is reached go to step 2

Pos' is the new position value that will be created depending on the wildcard that is being processed. The ".." wildcard mean that at least one member must exist in that position. This means that the incremental value will be 1. In the case of the "…" because it stands for zero or more members in that sequence, pos' can be the current position. Capturing pos' is important since members that follow the wildcard will have an XPath expression that looks for members where their position is greater than pos'

Separator Symbols In argument lists, sequences, and in matrices, different kinds of separators are used. The query language provides explicit symbols for separators, with definite semantics. The next table defines three separator symbols that when used with wildcards, support for structural search becomes possible.

Table 4. Seperator symbols

Symbol	**Meaning**
,	Is used to separate entries in a sequence, or arguments of a function. In the context of matrices, it is used to separate columns.
;	Separates rows in the context of matrices. For example, matrix[;;] matches any three-row matrix, regardless of its content.
@	Indicates function application. For example, @(x+1) matches any function where its argument is x+1.

Support for Structural Search Through the use of wildcards and separator symbols, the math query language offers support for structural search. This is done by allowing the user to specify where in an expression a term or phrase must occur, and to specify part of the expression and not the whole expression. Support for structural search gives the user math query more expressive power.

- The @ symbol directs the search to arguments of a function. For example, the query @(x, y) will look for a function where the arguments are x and y.
- The semi colon directs the search in the rows of a matrix.
- The comma directs the search in arguments of a function or entries of a column of a matrix.

- The use of the dot symbols ("…") in combination with other symbols helps the user specify where in the structure to look. The dot symbols can be used to surround an expression to search for subparts. For example, /(…x+1…) will search for x+1 somewhere in a denominator. In this example, the query is looking for x+1 where several terms can precede it and several terms can follow it. In essence, x+1 is part of a subexpression that forms the denominator. If the query is (x+1)/, the search will be for x+1 as the numerator. As another example, ^(…n…) is a query that looks for n as part of an exponent. While _{…n…} is a query that looks for n to be in a subscript.

Algorithm for processing the use of the … symbol to support structural search:

When encountering an expression exp' that is surrounded by the … symbols, an XPath expression will be created to search for the exp' as a descendent of the overall expression's tree structure.

Support for Different Levels of Abstraction The $ symbol when used alone stands for any arbitrary term, whether a numeric value or a variable name, and of whatever data type. This creates the first level of abstraction in the query language by allowing the user to step away from the literal specification of the term. Multiple occurrences of $ in a query in this case will stand for arbitrary terms that may or may not be the same. For example, the query $^2+$^2=20 makes no distinction between the arbitrary terms and can be matched by x^2+y^2=20, by x^2+x^2=20, and 2^2+4^2=20.

However, if the user wishes to search for patterns that match cos^2 $ + sin^2 $ with the additional constraint that the two arbitrary terms intended by the two occurrences of "$" must be equal, the use of cos^2 $ + sin^2 $ will be wrong because it matches cos^2 x + sin^2 y . Clearly, the use of the $ wildcard alone is inadequate to express that particular math search need. Consequently, a new syntax is added as specified in the following grammar rule:

WildcardTerm **::= $[1|2|3|...][‘n’|‘v’][‘R’|‘Z’|‘Q’|‘C’|‘P’|‘F’]**

Here is the explanation of this syntax. The syntax $ followed by a number (e.g., $1 $2, $3, etc.) designates any arbitrary term, whether a numeric value or a variable name, and of whatever data type. If $1 occurs twice in a query, that means that the two occurrences stand for the same number or the same identifier. If $1 ad $2 occur in a query, then they stand for arbitrary tokens that need not be the same. Referring back to the previous example, cos^2 $1 + sin^2 $1 will be used instead where it is now clear that the user is searching for an expression where the terms are arbitrary but equal. In essence, this syntax supports a second level of abstraction (or specificity, to be precise). Not only is the user being separated from a particular literal meaning but the user can now specify if the arbitrary symbols are identical or independent within the same expression.

The third level of abstraction-specificity is the enabling of the user to specify if the term is a numeric arbitrary value or an arbitrary variable name by attaching the appropriate symbol to the $. The syntax $n stands for an arbitrary term that is a numeric value such as 2, 4, 3.14, and so on, while $v stands for an arbitrary term that is a variable such as a, x, y, etc. For example, the query $n+$n^2+$n^3 is matched by 2+2^2+2^3 and by 4+5^2+7^2, but is not matched by x+3^2+y^3 because x and y are not numeric tokens.

Combing different parts of the rule with the $ is possible and has the combined effect. In this case, the user is able to specify if arbitrary numeric values or variable names need to be identical or independent in a query. For example, if you need to specify a query with three or more arbitrary numerical terms, and two of which must be identical, and the third is not necessarily identical to the other two, the user then uses $1n and $2n (and so on) to stand for potentially different numeric terms. The query $1n+$1n^2+$2n^3 describes that request and is matched by 2+2^2+15^3 and by 2+2^2+2^3; but it is not matched by 4+5^2+7^3 because 4 and 5 are not identical numerical tokens.

If the user wishes to specify the data type of the term, then the appropriate symbol from {**R, Z, Q, C, P, F**} is appended. This syntax creates support for the yet another level of abstraction-specificity. For example, $nZ stands for any arbitrary integer numerical token, $vC stands for any arbitrary identifier of Complex data type, $1vR and $2vR stand for any two arbitrary identifiers of type real.

Content MathML does allow its numeric and identifier elements to have more data types than those offered by our Query Language. But for purposes of this research, the data type symbols will be limited to: **R** for real, **Z** for integer, **Q** for rational, **C** for complex, **P** for complex-polar, and F for function.

Algorithm for processing different levels of abstraction-specificity:

"singleToken" will be a term that satisfies the wildcard rule designed for the support of different levels of abstraction.

1 if singleToken is $ and does not contain 'n' or 'v'
   - create sub-XPath expression that search for an item that is either a numeric value or an identifier

2 else if singleToken contains 'n' or 'v'
   - if 'n' create sub-XPath expression that search for an item that is a numeric value
   - if 'v' create sub-XPath expression that search for an item that is an identifier

3 if singleToken contains 'R' 'Z' 'Q' 'C' 'P' or 'F'
   - add a condition to the sub-XPath expression to search for a particular type according to the symbol used

4 if singleToken contains a number
   - retrieve item
   - store singleToken, item pair

After execution of XPath/XQuery query the "singleToken, item" pairs need to be analyzed:

5 if any two pairs have identical singleToken values, the items must be identical; if not, the mathematical expression retrieved is not an accurate result

6 if any two pairs have different singleToken values, the items must not be identical; if not, the mathematical expression retrieved is not an accurate result

A Note on Ellipsis The ellipsis (…), is an important mathematical symbol that stands for implicit patterns in sequences, series and matrices. We refer to the ellipsis symbol that is commonly used in mathematics as *pattern-bound* ellipsis. For example: in the sequence 2, 4, 6, …, 2n the … stands for even numbers. While in the series $1 + x/1! + x^2/2! + x^3/3! + \ldots$, the … stands for the pattern $x^n/n!$.

Our math query language, however, offers limited support for this symbol. The ellipsis is not assigned a hidden pattern and no attempt is made to determine the hidden pattern will be made during the processing of the user search query. The symbol is used as a generic search term at the sequence level to stand for any node that doesn't need to resemble in pattern the following or previous members in that sequence. The ellipsis in this case is referred to as *unbounded.*

Unbounded ellipses are used in the math query language horizontally between separator symbols in the context of matrices. When used between commas it indicates zero or more rows. But when used between semi-colons its use is similar to the use of the vertical ellipsis symbol and stands for zero or more rows. The language doesn't support the diagonal or anti-diagonal unbounded use of the ellipses in the context of matrices at this time.

In future extensions of the language, we will introduce an elaboration of that syntax that will enable the user to specify that the matches comply with the implied pattern of the ellipses. This will be possible with the algorithmic support that is currently underway for recognizing the patterns implicit in ellipses in math expression, such as that by Sexton and Sorge of the University of Birmingham [11]. They are currently working on the development of algorithms for the analysis of "abstract matrices". This term defines a common class of matrices where under-specified parts are denoted using the ellipses symbol (a series of three dots) [11].

4 IMPLEMENTATION MATTERS

Our search approach is based on the emerging XML-based technologies. This is a different approach than the evolutionary approach employed by Youssef for the DLMF project and by Minor for the Mathdex project. User information needs described by the query language will be processed to create an XML Query representation. The XML Query will then be executed against math documents that are encoded using a normalized form of Content MathML. The definition of what constitutes a normalized canonical form of user documents is an important issue. It is intended to resolve issues like notational equivalences as they relate to the use of Content MathML and have impacted the processing of the user math queries, but it is outside the scope of this paper.

The input to the math query processor is text. Each user query is treated as an expression and entered as a text. When the user query is parsed and broken into "terms". The terms are classified as: operators, function names, constructors names, separator symbols, grouping symbols, wildcards, numbers, constants, and identifiers.

When parsed, mathematical constructs supported by the language are classified based on how similar their Content MathML encodings are in documents; the classification is not based on how similar to user math queries. For example, functions and operators are treated the same way during the query processing phase. The Content MathML encoding represents the logical tree structures of mathematical statements in documents. The XPath expression that is generated after parsing the user math query will describe that structure. Our math query processor is currently work in progress.

5 LIMITATIONS

We limited tree-level wildcards to single levels of the tree. That is, each wildcard either stands for a node in the logical structure of a mathematical expression tree or stands for a full sub-set tree that represents a complete sub-expression. For example, in an expression x+y*z, the -- wildcard can stand for sub expression y*z resulting in x+-- query while x--*z is currently not supported since the query cannot be parsed correctly.

In addition, the language offers limited semantic support of the ellipsis symbol. Future versions can take advantage of ongoing research on determining the implied patterns of ellipses in various contexts.

These limitations can be addressed in future implementations.

6 CONCLUSION AND FUTURE WORK

In this paper, the need for more wildcards in the area of math search has been established. It defines a powerful set of wildcards that provide for more precise structural search and multi-levels of abstractions. We have developed algorithms for mapping the wildcards to their corresponding XPath/XQuery queries. It is assumed that the content is encoded in MathML.

Naturally, there are limitations, which were discussed in some detail. Future work entails lifting those limitations and providing new extensions to the language for more precision and more expressive power. Further into the future, both subjective and objective performance evaluation of our algorithms should be conducted to determine user satisfaction and measure improvements in precision and recall.

7 REFERENCES

[1] NIST, "Digital Library of Mathematical Functions (DLMF)." http://dlmf.nist.gov/.

[2] Design Science, "Mathdex." http://www.mathdex.com:8080/mathfind/search.

[3] World Wide Web Consortium, "XML Path Language (XPath) Version 2.0," 2005. http://www.w3.org/TR/xpath20/.

[4] World Wide Web Consortium, "XQuery 1.0: An XML Query Language," 2007. http://www.w3.org/TR/xquery/.

[5] World Wide Web Consortium, "Mathematical Markup Language (MathML) Version 2.0," 2003. http://www.w3.org/TR/MathML2/.

[6] C. Winstead, "Creating Connexions Content Using LyX module," Connexions Project, Rice University, 2006. http://cnx.org/content/m13238/latest/

[7] B. Miller, "DLMF, LaTeXML and some lessons learned," In: *The Evolution of Mathematical Communication in the Age of Digital Libraries, IMA "Hot Topic" Workshop, 2006.* http://www.ima.umn.edu/2006-2007/SW12.8-9.06/abstracts.html#Miller-Bruce

[8] B. R. Miller and A. Youssef, "Technical Aspects of the Digital Library of Mathematical Functions," In: *Annals of Mathematics and Artificial Intelligence*, 38(1-3): p. 121-136, Springer Netherlands, 2003.

[9] Design Science. http://www.dessci.com/en/.

[10] The Wolfram Functions Site. http://functions.wolfram.com/

[11] A. Sexton and V. Sorge, "Abstract matrices in symbolic computation Computations," In: *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, p. 318-325, ACM Press, New York, 2006.

[12] W. A. Martin, "Computer input/output of two-dimensional notations," In: *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, p. 102-103, ACM Press, New York, 1971.

[12] W. A. Martin, "Computer input/output of two-dimensional notations," In: *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, p. 102-103, ACM Press, New York, 1971.

[13] A. Youssef and M. E. Altamimi, "An Extensive Math Query Language," Due to appear in: *Proceedings of 16th International Conference on Software Engineering and Data Engineering*, ?(?): p. ?-?, 2007.