An Extensive Math Query Language

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

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

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. This paper will present such a language and detail its features. The new math query language offers an alternative way to describe mathematical expressions that is more consistent and less ambiguous than conventional mathematical notation. In addition, the language goes beyond the Boolean and proximity query syntax found in standard text search systems. It defines a powerful set of wildcards that are deemed important for math search. These wildcards provide for more precise structural search and multi-levels of abstractions.

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 resolved 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 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, it is not capable of expressing an ellipsis. 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.

This paper proposes a new query language that extends the current standard query syntax. 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 proposed math query language 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.

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.

2.1 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…)"; 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.

2.2 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 structural search and wildcards, and thus suffers the same limitations of text-based search systems mentioned above.

3 THE PROPOSED QUERY LANGUAGE

Mathematical notation can be ambiguous and cumbersome to type. The math query language presented in this paper 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. Furthermore, 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.

The following subsections will describe the characteristics of the query language

3.1 Characteristics of the Language

In the case of operators and special characters that do not have a corresponding key on the keyboard, the user is not required to use additional software. In those cases, the language provides different symbols and keywords that are part of the English alphabet. For example, in the expression a • b, the • symbol is represented as = = =. Another example is when the mean of an identifier x that is commonly denoted by $\overline{X}$ is represented using the

"mean" keyword as mean(x), or simply "X bar" if $\overline{X}$ refers to an arbitrary variable name.

Common two-dimensional configurations of mathematical notation are represented in a linear form. The superscript and subscript layouts will be captured through the use of symbols to identify these structures. The "^" symbol and the "_" symbol are used to explicitly denote superscript and subscript layout, respectively. Binding operators [17] like integration, differentiation and summation, for example, will be made linear through the use of keywords to denote these operators in addition to the use of parentheses to group their arguments. Function parameters (as opposed to function variables) will be represented using subscript and superscript notation. Structures such as matrices will have a linear representation through the use of the "matrix" keyword in combination with parenthesis where "," and ";" symbols are used to separate columns and rows. The linearization of the two-dimensional layout, in addition to defining keywords that correspond to symbols that do not exist on keyboards, allow for the implementation of an input module that is easy to use.

For effective processing, ambiguity and inconsistencies inherent in common mathematical notation are resolved by developing a language where the focus is on capturing the underlying conceptual structures rather than the abstract notational structures of an expression. The syntax requires the user to explicitly define relations between notations that are otherwise implicit or dependent on context and knowledge of the problem domain. To do so, functions and operators that are omitted for conciseness are assigned symbols or keywords. For example, when representing g(x+y) as the multiplication of an identifier g with the sum of the identifiers x and y, the user is required to use * to indicate the multiplication operation; otherwise, g will be considered as a function applied at x+y. Also, when parentheses are used to represent elements of a set or a matrix, the user is required to use keywords like *set* and *matrix* in addition to parentheses.

Generally, to minimize ambiguity, the following additional rules must be obeyed:

- When different formats are used to represent the same concept, the user is restricted to one form. Handling mathematical equivalences is out of the scope of this research. For example, when representing the square root of x $\sqrt{x}$, x^(1/2) is interpreted differently and will not mean the square root of x, but root^2[x] will. Another example, 1/x and x^-1 will be interpreted differently.
- Some functions do not require the use of parentheses when their argument is simple. For example, sin x does not require parentheses but sin(x+1) does. The math query language in this case will require the use of parentheses to enclose the argument regardless of how simple it is to maintain a more consistent syntax.
- The @ symbol will be used to declare function application. As a result, (a, b, c) for example, will be parsed as a triplet while @(a, b) will be parsed as arguments of a function.

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, and the rest of this section introduces formally the new features of the query language.

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
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

3.2 New Capabilities and Features

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.

Three sets of wildcards are introduced to help capture user needs in the area of math search. The first set of wildcards handles search at the character level. Their use is similar to wildcards in the area of text search to facilitate keyword search. The second and third sets handle search at the parsed tree level of a mathematical structure. These two sets are math specific and allow the user greater levels of expression.

Character Level (keyword search): At this level, two wildcards are used: "$" and "?". Their use is similar to the use of * and ? in text search respectively. The processing of wildcards at the character level will be 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
**$** symbol when used alone, the meaning of it is overloaded to mean any arbitrary single term, which can be a variable, a number, a matrix name, a function name, an operator name, a set name, and so on. For example, f($) 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.
--- stands for 0 or more terms that form a sub-expression. For example, f(x,---) matches a function f with either one argument x or two arguments with the first one being x and the other argument being arbitrary. As a result, f(x), f(x, y), f(x, y^2) will all be retrieved. Note: --- will also be used to support structural search, this will be discussed later.
-- 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.

Sequence-level and parse tree-level wildcards: The two wildcards in this set will match for 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 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
.. 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 vectors and 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)
… 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.

Separator Symbols In argument lists, sequences, and in matrices, different kinds of separators are used. Therefore, the query language has to provide explicit symbols for separators, with definite semantics. The next table defines the separators used in our query language.

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 ("…" and "..") 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 the 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 the exponent. While _{…n…} is a query that looks for n to be in a subscript.

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 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 $ wildcard alone is inadequate to express that particular math search need. Consequently, a new syntax is added as specified 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. Not only is the user being separated from a particular literal meaning but the user can now specify if are arbitrary are identical or independent within the same expression.

The third level abstraction is enabling 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 $ creates a fourth level of abstraction. 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 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 fifth level of abstraction. 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.

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 software support that is currently underway by Sexton and Sorge of the University of Birmingham [16]. They are currently working on the development of algorithms for the analysis of "abstract matrices". This term defines a common class of matrices where underspecified parts are denoted using the ellipses symbol (a series of three dots) [16].

4 LIMITATION OF THE QUERY LANGUAGE

The query language specified in the previous section is meant to be a major extension to the standard query languages in use in current math search systems. It allows users to be more precise in specifying their math search needs. However, the language still has a number of limitations, which can be addressed in future implementations. The following is list of some of the limitations

- The specified math query language has a syntax that requires the user to be specific about what he/she intend when searching to resolve ambiguities. However, it puts a burden on the user to express all the specificity. In the future, this language can act as an intermediary language to a more flexible (allows for ambiguities) language where the burden of interpreting what is intended falls more on the system than on the user.
- The query language does not cover every area of math. There are specialized areas of math that haven't been incorporated into the language but could be added in the future. The language is limited to the search for mathematical constructs such as formulas and equations and doesn't handle formal mathematics such as theorems and proofs.
- The user query must be grammatically correct. Part of the parsing effort is to validate the syntax of the language at this time. Future versions should be more syntax-forgiving.
- Currently, 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.
- We limited tree-level wildcards to single levels of the tree. That is, the wildcards 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 subset 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.
- The parser doesn't handle the following mathematical equivalences:

– Sets and Lists: The list/set of elements in the document can be defined explicitly in the set/list container elements. In this case, the corresponding syntax in the user query language will be done using List[expr, expr,…] and Set[expr, expr,…], respectively. In MathML, the list/set elements can also be defined implicitly using bound variables and a condition element restricting their domain. In the documents, both formats can be used but their logical equivalences will not be considered at this time. In addition, the MathML set constructor element has a type attribute. This attribute can be present in the documents but will not be used as a search criterion on the query side at this time. Lists differ from sets in that there is an explicit order to the elements. The attribute that specifies the order in a list but will not be used as a search criterion on the query side at this time. The assumption is that elements appear in the document in a lexicographic order even if they're numeric.

– Sum and product operators: the following equivalence sum(a,b,c) vs. a+b+c is outside the scope of this research. The MathML encodings in this case are different but equivalent. At this time, the user has to be specific as to whether they mean the sum operator or the use of the plus sign when performing a search because the sum construct in MathML has a different meaning than the plus operator.

– The equivalent representation of numbers using different forms, for example, 1/3 vs. 0.333 vs. 3^-1 will not be supported at this time.

– Equivalences due to distributive laws for logical and set theory operations will not be supported at this time.

5 CONCLUSION AND FUTURE WORK

In this paper, a new query language for math search was defined. The language goes beyond the Boolean and proximity query syntax found in standard text search systems. It defines a powerful set of wildcards that provide for more precise structural search and multi-levels of abstractions. Although implementations techniques for the query language were not discussed due to space limitations, we have developed algorithms for mapping the constructs of the query language to XPath/XQuery queries. It is assumed that the content is encoded in MathML.

Naturally, the language has many 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, while at the same lightening the query-formulation burden on the user by making the query processing system do more work. Further into the future, both subjective and objective performance evaluation of the query language should be conducted to determine user satisfaction and measure improvements in precision and recall.

6 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] M. Kohlhase and A. Franke, "MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems," *Journal of Symbolic Computation*, 32(4): p. 365-402, Academic Press, 2001. [publisher is acquired by Elsevier, The Netherlands]

[12] Helm: Hypertextual Electronic Library of Mathematics. http://helm.cs.unibo.it/.

[13] Monet: Mathematics on the Net. http://monet.nag.co.uk.

[14] MoWGLI: Mathematics on the Web: Get It by Logics and Interfaces. http://mowgli.cs.unibo.it/.

[15] A. Asperti and S. Zacchiroli, "Searching Mathematics on the Web: State of the Art and Future Developments," In: *Proceedings of New Developments in Electronic Publishing of Mathematics*, p. 9-18, Edited by FIZ Karlsruhe, 2004.

[16] 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.

[17] 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.