

MathEX: A Direct-Manipulation Structural Editor for Compound XML Documents

Samuel S. Dooley
Integre Technical Publishing Co., Inc.
sam@integretechpub.com

Abstract

This paper describes how MathEX addresses a number of well-known issues with direct-manipulation structural user interfaces for mathematics. The solutions described here allow MathEX to support a diverse collection of XML vocabularies for mathematics, including Content and Presentation MathML, XHTML+MathML compound documents, and OpenMath semantic markup. This paper illustrates how MathEX allows an application author to tailor the user experience within a specific XML vocabulary, within a specific document, or within a specific collection of elements within a document, without the need for a modal interface. MathEX allows an author to customize the notational presentation, keyboard editing, structural palettes, and other properties of each operator in a document, whether or not the operator is known to the editor in advance. MathEX is implemented in Java, and may be used as a stand-alone application, as a web page applet, or as a Swing component within another application.

1 Introduction

Direct-manipulation editors for mathematical expressions appeared over twenty years ago, shortly after windowing systems with point-and-click user interfaces [1, 2, 3, 4, 5]. Over the next several years, user interfaces for general-purpose computer algebra systems, as well as more specialized mathematical applications, began to include direct-manipulation expression editors [6, 7, 8]. The amount of effort required to develop such an interface, or to port one from one operating system to another, limited general acceptance of these interfaces to major software vendors, and motivated research into standard protocols between front-end user interfaces, responsible for defining the user experience, and computational engines, responsible for producing a desired result [9, 10].

Partly as a way to specify these protocols, and partly as a response to the World Wide Web, mathematical software vendors came together over ten years ago to define a standard encoding for presentational and structural information for mathematics, the Mathematical Markup Language (MathML) [11]. MathML provides a formal separation between presentation markup elements and content

markup elements, and its introduction created an opportunity for the development of general-purpose, content-based, direct-manipulation structural editors for mathematical expressions [20]. This separation between structure and presentation became a driving force behind a number of XML web standards, most visibly XSL Transformations and XSL Formatting Objects [12, 13].

A second driving force behind XML web standards was the realization by a number of user communities, including the mathematical community, that web browsers could not support a broad range of specialized markup vocabularies. Specifications such as XML Namespaces [15], XML Schema [14], XHTML Modularization [16], and the Compound Document Framework [17], when taken together, provide a representational model for document profiles composed of a collection of document types; yet this model does not address major issues involved in authoring compound documents for mathematics.

MathEX [21] began as a research project to develop a content-based, direct-manipulation structural editor for mathematics, originally known as the IBM MathML Expression Editor [22, 23, 24]. Initially, this editor used the IBM techexplorer Hypermedia Browser [25] as its core rendering engine. After the MathML portion of the techexplorer rendering engine was reimplemented in Java, the current implementation of MathEX was realized as a stand-alone Java application, a Java applet, and a Swing component.

This paper describes how MathEX addresses a number of well-known issues with direct-manipulation structural user interfaces for mathematics. The solutions described here allow MathEX to support a diverse collection of XML vocabularies for mathematics, including Content and Presentation MathML, XHTML+MathML compound documents, and OpenMath semantic markup. This paper illustrates how MathEX allows an application author to tailor the user experience within a specific XML vocabulary, within a specific document, or within a specific collection of elements within a document, without the need for a modal interface. MathEX allows an author to customize the notational presentation, keyboard editing, structural palettes, and other properties of each operator in a document, whether or not the operator is known to the editor in advance.

2 Direct Manipulation Issues

MathEX addresses a number of well-known issues with direct-manipulation structural user interfaces for mathematics, including flexible coordination between presentational and structural representations of mathematical expressions, detailed configuration of interactive editing behaviors, and transparent extensibility to new content operators. This section describes the MathEX solutions for these issues and how these solutions provide significant advantages to a direct-manipulation user interface for mathematics.

2.1 Representation

A direct-manipulation editor that operates using only the presentational form for an expression cannot capture the computational structure of an expression in an unambiguous fashion. A direct-manipulation editor that operates using only the structural form for an expression cannot provide the kind of natural visual interaction that users expect. To resolve this trade-off, MathEX maintains both a structural form and a presentational form for each expression as it is created. This dual representation allows a user to interact with the visual presentation of an expression, while simultaneously creating the underlying structural form for the expression. Each user interaction in MathEX operates within this framework—presentational aspects operate in terms of the presentational form, while structural aspects operate in terms of the content form.

MathEX obtains several advantages from this dual representation. Keyboard bindings operate directly on the content form to provide a natural and unambiguous linear input model, while presenting a high-quality two-dimensional presentational form with each keystroke. Lexical error checking is eliminated, as each editing operation creates only well-formed content expressions. The visual presentation of an expression defines cursor navigation, with an explicit visual indication of the underlying content structure of the current expression. Visual palettes of editing operations are defined in terms of the transformation to be applied to the content form, and are displayed in terms of the presentational form corresponding to the transformation. While each of these objectives can be accomplished in other ways, the flexibility of the MathEX dual representation is exemplified by its use as a foundation for a configuration framework for content operators, as described in the following paragraphs.

2.2 Configuration

MathEX describes each content operator by a collection of properties that specify interactive behaviors associated with the operator, and encodes these properties into operator declarations that define, for the MathEX interface, the editing behavior of a content operator. Every aspect of the presentational and structural behavior of a content operator is determined by these operator declarations.

Layout declarations provide transformation rules that convert a content form to a presentation form. Editing declarations provide transformations from a predecessor content form to a successor form, and provide key bindings that incrementally elaborate a content form. Additional properties define an operator precedence relation that affects when the presentational form will insert parentheses, and how far up the content structure an editing transformation will apply, to produce the expected presentational and structural forms from a sequence of keyboard operations. Other properties carry specialized collections of presentational or editing conventions, such as for additive, n-ary, or array operators. Any conventional interaction in MathEX, whether presentational or structural, is captured by a configuration declaration that specifies which content operators participate in the conventional behavior. Many of these con-

ventional interactions affect both presentational and structural aspects of an operator, or the relationship between the presentational and structural forms for an expression. The MathEX dual representation allows these declarations to be made explicit, and to be associated with a content operator in a completely data-driven fashion.

MathEX provides a standard configuration file that contains a default collection of content operator declarations, and that defines the usual behavior of the editor. An application developer may use the MathEX API to create additional declarations, to remove existing declarations, or to replace the entire collection with another configuration. A developer can thus create an entirely new behavior for the editor, with respect to any content operator, whether or not the operator is known to the editor in advance.

2.3 Extensibility

Because every conventional interaction that MathEX provides is controlled by a configuration declaration, new content operators may participate in predefined behaviors with the same status as the MathML standard content operators. An end user may define a custom content operator in MathEX simply by applying a new function symbol to a collection of arguments, and a default collection of operator declarations will apply to the new operator until other declarations, that apply specifically to the operator, override the default behavior.

The extent of the MathEX operator declaration framework means that the introduction of a new content operator can be as ordinary, or as elaborate, as necessary for a particular application. As a result, full customization of a new content operator is primarily the responsibility, not of the end user, but of the application developer, who will use the MathEX API to create a custom configuration that defines the editing behavior of the new content operator. More extensive user interfaces to assist the application developer in writing MathEX operator declarations are under investigation.

3 XML Vocabularies

MathEX supports a diverse collection of XML vocabularies for mathematics, including Content and Presentation MathML, XHTML+MathML compound documents, and OpenMath semantic markup. Each of these document profiles places its own unique demands on the authoring user interface, and illustrates the challenges involved in the compound document authoring task. The following paragraphs describe how MathEX addresses the unique aspects of these various XML vocabularies.

3.1 Content and Presentation MathML

MathML is unusual in that it provides markup elements that represent both presentational and structural information, as distinct language subsets within a

single namespace. This design reflects the fact that mathematical expressions, more than other vocabularies, employ extensive interactions between the structural and presentational properties of an expression. MathML thus presents a special challenge to a direct-manipulation editor because each interaction initiated by the user may have consequences for both the presentation and the structure of an expression.

MathEX addresses this duality in its user interface by maintaining both structural and presentational representations for each expression, as described in Section 2. Since the user interface has access to both representations, MathEX is able to convert a potential liability—that is, the need to maintain multiple representations for an expression—into an opportunity, namely, the ability to enrich the user experience by accurately capturing the presentational and structural aspects of each interaction. By further extending this duality with an operator declaration framework, the duality of the MathEX representation yields a collection of control points that an application developer may use to tailor the user experience for a wide range of mathematical applications, some of which are illustrated in Section 4.

The extensive nature of the interactions between content and presentation MathML markup provides a robust setting for exploring representational issues for a direct-manipulation editor. The design decisions needed to manage the relationships between content and presentation markup in MathML provide parallels to design decisions that are needed for other XML vocabularies. The MathEX solutions to these issues for mathematical markup provide a basis for extending the editor to address these and related issues for other namespaces.

3.2 XHTML+MathML Compound Documents

XHTML [18] provides markup elements that represent textual information contained in a document. A document profile for XHTML+MathML compound documents provides a framework for including mathematical information into textual documents. However, this document profile imposes new requirements on the authoring interface, since typical editing conventions for text are much different than for math. Authoring interfaces for these documents, such as the equation editor within Microsoft Word, often provide a modal environment composed of two applications or components working together, where one component provides a mode optimized for editing text, and another component provides a mode optimized for editing math.

One objective of the MathEX operator declaration framework is to adapt the MathEX editing component to a broad range of editing applications. While previous experience has demonstrated that the MathEX operator declarations are sufficient to encompass a wide range of mathematical operators and their notational and interactive behaviors, extension of these operator declarations to nonmathematical structural input is ongoing. Early investigations have demonstrated that a basic textual editor for XHTML presentational markup can be configured using the MathEX operator declarations, although a number of implementation issues remain, including rendering optimizations appropriate for

<pre> <apply> <divide/> <apply> <plus/> <ci>x</ci> <cn>1</cn> </apply> <cn>3</cn> </apply> </pre>	$\frac{x+1}{3}$	<pre> <OMA> <OMS cd="arith1" name="divide"/> <OMA> <OMS cd="arith1" name="plus"/> <OMV name="x"/> <OMI>1</OMI> </OMA> <OMI>3</OMI> </OMA> </pre>
---	-----------------	--

Figure 1: MathML and OpenMath Markup in MathEX

textual content, selection models for range and multiple selection that accommodate text and math, and more flexible user interaction models for collections of attribute value assignments.

By extending the operator declaration framework to encompass XHTML elements, MathEX is able to provide a seamless transition in editing behavior across namespace boundaries, and to accommodate the differences in expectations in editing behavior for textual and mathematical content. To illustrate these capabilities, Section 4 includes a user interface customization example that involves editing conventions for multiple content types in the same document.

3.3 OpenMath Semantic Markup

OpenMath [19] provides structural elements that are similar to content MathML elements, as well as additional elements that define aspects of the semantic behavior of structural operators in the context of one or more content dictionaries. A user interface with the capability to preserve the distinction between the structural and semantic properties of an operator has the potential to realize a degree of flexibility similar to what MathEX obtains by the separation of structural and presentational information for mathematical expressions.

The operator declaration framework in MathEX is sufficient to capture the structural aspects of OpenMath markup, with each OpenMath symbol defined as a custom operator in a MathEX custom configuration. It is then straightforward to extend the custom configuration to allow a user to edit mathematical expressions using the same key bindings, menus, and palettes used to create Content MathML, and yet have the editor produce OpenMath expressions instead. At the level of the user interface, the end user need not know whether the underlying representation is Content MathML or OpenMath: the interactive experience is the same. (Figure 1)

A truly semantic user interface for OpenMath should provide operations that leverage the semantic knowledge contained in the content dictionaries to provide new kinds of interactions that enrich the user experience. In the same way that a structural editor provides operations that preserve the structural integrity of a mathematical expression, a semantic user interface should provide operations that preserve important semantic properties of the expressions in a document. Examples of such properties could include equality and order relationships in

$$3 \blacksquare 4 \blacksquare 2 \blacksquare 6 = 16$$

Figure 2: Directed Exploration Exercise in MathEX

a domain, or logical relationships such as equivalence and implication. While these types of operations have been outside the scope of the MathEX interface, the MathEX API allows external applications to be aware of these relationships and to implement editing operations that respect them on an equal footing with the structural editing operations that are more common in editing applications.

4 Application Illustrations

MathEX provides a collection of control points that an application author may use to tailor the user experience within a specific XML vocabulary, within a specific document, or within a specific collection of elements within a document, without the need for a modal interface. This section describes application examples that illustrate the flexibility of the operator declaration framework and the kinds of interaction models that can be realized using the MathEX user interface. By embedding MathEX as a Swing component and using the MathEX API, application designers can incorporate these mathematical interaction models into a wide range of scientific, technical, and educational applications.

4.1 Directed Exploration

The ability to restrict the scope of a declaration to a specific document allows MathEX to define nontraditional user interfaces for particular applications such as directed explorations of mathematical concepts. An author for an interactive textbook, for example, could define one such specialized user interface to guide student focus on strategic choices for one exercise, and an entirely different user interface for the next exercise, where the choice of the user interface can be made to reinforce specific pedagogical goals.

Consider the following exercise. A student is presented with an equation, involving only constants and operators, but where the operator symbols have been removed from the equation. The student is then challenged to select operator symbols to satisfy the equation. A reasonable user interface for such an example would present the equation with placeholder elements in place of the operators, and the remaining constants in the appropriate places. Several alternative means can then be provided to allow the student to supply the missing operators, including keyboard bindings, mouse actions, or menu or palette selections. (Figure 2)

The author can realize this example in MathEX using a collection of custom operator declarations to define the presentation of the problem statement and the keyboard, mouse, menu, and palette actions that allow the student to select

Name	Amount \$	Name	Amount DM
Alpha	\$0.02	Alpha	0,02 DM
Beta	\$8,192.01	Beta	8.192,01 DM
Gamma	\$1,002,003.04	Gamma	1.002.003,04 DM

Figure 3: Localized Currency Interface in MathEX

the operator. Any additional programming is then limited to the selection of the constants and operators in the problem statement and in the evaluation of the student response, not to the implementation of the user interface used to collect the student response. Since much of the development time for a new application is spent in the implementation of the user interface, the MathEX operator declaration framework results in a significant reduction of the amount of effort needed to create these custom interactions.

4.2 Localization

Software developers expend enormous resources adapting their applications to regional user interface conventions, including choice of language, reading order, and various other features. Even for information as mundane as currency amounts, there are at least five dimensions that define the presentation of a currency amount for a particular locale: currency symbol, currency symbol position, thousands separator, decimal separator, and number of digits past the decimal point.

Within a mathematical editor, these conventions affect not only how a currency amount is presented, but also the behavior of key bindings for the digits and separators—bindings which apply only to currency amounts, and not necessarily to other text in a document.

As a simple example, consider the presentation of a two-column table, where the first column is to contain customer names, and the second column is to contain currency amounts. When entering a customer name, MathEX operator declarations allow the end user to enter text in the linear form natural for text. When entering a currency amount, MathEX operator declarations that apply only to the second column of the table allow the end user to enter the currency amount in the linear form natural for the currency amount, and in the currency format natural for a particular locale. Regardless of the locale, the structure of the end result in the document is a tagged currency amount that is presented in the appropriate currency format. (Figure 3)

The ability to define the scope of a declaration in terms of a collection of elements in a document allows MathEX to combine multiple data types within a single document, where the user interacts with each data type using editing conventions that are most appropriate for that data type.

5 Conclusions

5.1 Compound Documents

The literature describes a growing number of successful direct-manipulation user interfaces for specific mathematical applications [26, 27, 28, 29]. The strengths and weaknesses of these interfaces have been extensively investigated, and are well-understood by the mathematical software community. Standard markup languages for mathematics promise to broaden the reach of mathematical software into diverse fields of science, technology, and education. However, these applications require authoring tools that can integrate mathematical markup into other markup vocabularies. Compound XML documents provide a framework for combining content markup from several domain-specific vocabularies, including mathematics, into a single document. However, the compound document authoring task poses special problems for the user interface, including how to customize the user experience for each vocabulary, and how to integrate these diverse user experiences into a unified whole. Solutions for these issues, insofar as they relate to mathematical content, promise greater insight into the more general compound document authoring problem, in much the same way that markup solutions for mathematics helped to frame early discussions toward markup solutions for other web standards.

5.2 MathEX Solutions

MathEX provides an extensible basis for user interfaces for a broad range of content-aware mathematical applications. The extensibility of the MathEX user interface—provided by the dual representation of presentational and structural forms and by the the content operator declaration framework—allows MathEX to address a number of well-known issues with direct-manipulation structural user interfaces, and to support a diverse collection of XML vocabularies, including Content and Presentation MathML, XHTML+MathML compound documents, and OpenMath semantic markup. The operator declaration framework provides a collection of control points that an application author may use to tailor the user experience within a specific XML vocabulary, within a specific document, or within a specific collection of elements within a document, without the need for a modal interface. These solutions allow MathEX to provide a customizable user interface for authoring compound XML documents that include structural, as well as presentational, mathematical markup. These compound documents, when combined with custom user interfaces that define their behavior, provide an intuitive interactive experience for the end user, and leverage the rich structural information provided by content mathematical markup.

References

- [1] SMITH, C. J. AND SOIFFER, N. M. MathScribe: A User Interface for Computer Algebra Systems. In CHAR, B. W., ed. *Proceedings of the 1986 Sym-*

- posium on Symbolic and Algebraic Computation*. New York: ACM Press, pp. 7–12.
- [2] TEKTRONIX, INC. *MathScribe User's Manual*. Tektronix, Inc., 1988.
 - [3] SOIFFER, N. M. *The Design of a User Interface for Computer Algebra Systems*. Ph.D. thesis, Computer Science Division, EECS Department, University of California, Berkeley, 1991.
 - [4] BONADIO, A. AND WARREN, E. *Theorist: Reference Manual*. Waterloo, Ontario: Prescience Corporation, 1987.
 - [5] PARACOMP, INC. *Milo: The Math Processor for the Macintosh (User's Guide)*. Paracomp, Inc., 1988.
 - [6] WOLFRAM, S. *The Mathematica Book*, fourth ed. Cambridge, UK: Cambridge University Press, 1999.
 - [7] CHAR, B. W., GEDDES, K. O., GONNET, G. H., LEONG, B. L., MONAGAN, M., AND WATT, S. M. *Maple V Language Reference Manual*. New York: Springer-Verlag, 1991.
 - [8] MATHSOFT, INC. *Mathcad: User's Guide with Reference Manual*. Cambridge, MA: MathSoft, Inc., 2001.
 - [9] LEONG, B. L. Iris: Design of a User Interface Program for Symbolic Algebra. In CHAR, B. W., ed. *Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation*. New York: ACM Press, pp. 1–6.
 - [10] PURTILO, J. Applications of a Software Interconnection System in Mathematical Problem Solving Environments. In CHAR, B. W., ed. *Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation*. New York: ACM Press, pp. 16–23.
 - [11] CARLISLE, D., ION, P., MINER, R., AND POPPELIER, N. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). W3C Recommendation REC-MathML2-20031021, W3C, 21 October 2003.
<http://www.w3.org/TR/2003/REC-MathML2-20031021>.
 - [12] CLARK, J. XSL Transformations (XSLT) Version 1.0. W3C Recommendation REC-xslt-19991116, W3C, 16 November 1999.
<http://www.w3.org/TR/1999/REC-xslt-19991116>.
 - [13] BERGLUND, A. Extensible Stylesheet Language (XSL) Version 1.1. W3C Recommendation REC-xsl11-20061205, W3C, 05 December 2006.
<http://www.w3.org/TR/2006/REC-xsl11-20061205>.
 - [14] FALLSIDE, D. C. AND WALMSLEY, P. XML Schema Part 0: Primer (Second Edition). W3C Recommendation REC-xmlschema-0-20041028, W3C, 28 October 2004.
<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028>.

- [15] BRAY, T., HOLLANDER, D., LAYMAN, A., AND TOBIN, R. Namespaces in XML 1.0 (Second Edition). W3C Recommendation REC-xml-names-20060816, W3C, 16 August 2006.
<http://www.w3.org/TR/2006/REC-xml-names-20060816>.
- [16] ALTHEIM, M., BOUMPHREY, F., DOOLEY, S., MCCARRON, S., SCHNITZENBAUMER, S., AND WUGOFSKI, T. Modularization of XHTML. W3C Recommendation REC-xhtml-modularization-20010410, W3C, 10 April 2001.
<http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410>.
- [17] MEHRVARZ, T. AND APPELQUIST, D. Compound Document Framework 1.0 and WICD 1.0 Profiles. W3C Working Draft WD-WICD-20050915, W3C, 15 September 2005.
<http://www.w3.org/TR/2005/WD-WICD-20050915>.
- [18] W3C HTML WORKING GROUP. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation REC-xhtml1-20020801, W3C, 01 August 2002.
<http://www.w3.org/TR/2002/REC-xhtml1-20020801>.
- [19] BUSWELL, S., CAPROTTI, O., CARLISLE, D. P., DEWAR, M. C., GAËTANO, M., AND KOHLHASE, M. The OpenMath Standard, Version 2.0. The OpenMath Society, June 2004.
<http://www.openmath.org/standard/om20-2004-06-30/>.
- [20] DOOLEY, S. S. Coordinating Mathematical Content and Presentation Markup in Interactive Mathematical Documents. In GLOOR, O., ed. *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*. New York: ACM Press, pp. 54–61.
- [21] DOOLEY, S. S. *Users Guide for the Integre MathML Expression Editor, Version 1.2.1*. Albuquerque, NM: Integre Technical Publishing Co., 11 April 2005.
<http://www.integretechpub.com/docs/zed/Users/>.
- [22] DOOLEY, S. S. Bringing MathML Content and Presentation Markup to the Web with the IBM MathML Expression Editor.
<http://www.mathmlconference.org/2002/presentations/dooleyxml/>.
- [23] DOOLEY, S. S. Editing Mathematical Content and Presentation Markup in Interactive Mathematical Documents. In MORA, T., ed. *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. New York: ACM Press, pp. 55–62.
- [24] DOOLEY, S. S. Programming the IBM MathML Expression Editor for Interactive Mathematical Applications. In COHEN, A. M., GAO, X.-S., AND TAKAYAMA, N., eds. *Proceedings of the First International Congress of Mathematical Software ICMS 2002*. Singapore: World Scientific, 2002, pp. 421–431.

- [25] *Integre techexplorer Hypermedia Browser Documentation*. Albuquerque, NM: Integre Technical Publishing Co., 2003.
<http://www.integretechpub.com/docs/techexplorer/Help/>.
- [26] MARQUÈS, D., EIXARCH, R., CASANELLAS, G., MARTÍNEZ, B., AND SMITH, T. J. WIRIS OM Tools: A Semantic Formula Editor. In LIBBRECHT, P., ed. *Proceedings of the 2006 Mathematical User-Interfaces Workshop*, St Anne's Manor, Workingham, United Kingdom, 10 August 2006.
<http://www.activemath.org/~paul/MathUI06/index.html>.
- [27] LIBBRECHT, P. AND JEDNORALSKI, D. Drag-and-drop of Formulæ from a Browser. In LIBBRECHT, P., ed. *Proceedings of the 2006 Mathematical User-Interfaces Workshop*, St Anne's Manor, Workingham, United Kingdom, 10 August 2006.
<http://www.activemath.org/~paul/MathUI06/index.html>.
- [28] KUME, M., MIYAMOTO, A., KAI, H., TOMINARI, T., NODA, M.-T., AND TAMURA, Y. Mathematical Document Authoring with xfy. In LIBBRECHT, P., ed. *Proceedings of the 2006 Mathematical User-Interfaces Workshop*, St Anne's Manor, Workingham, United Kingdom, 10 August 2006.
<http://www.activemath.org/~paul/MathUI06/index.html>.
- [29] THIMBLEBY, H. AND THIMBLEBY, W. Mathematical Mathematical User Interfaces. In LIBBRECHT, P., ed. *Proceedings of the 2006 Mathematical User-Interfaces Workshop*, St Anne's Manor, Workingham, United Kingdom, 10 August 2006.
<http://www.activemath.org/~paul/MathUI06/index.html>.