



Hermitech Laboratory

Address: Chapaeva str., 7, Zhytomyr, 10029, Ukraine
Web: <http://www.mmlsoft.com>
E-mail: info@mmlsoft.com

FORMULATOR v3.9

MathML Weaver

User Manual

Published By

Hermitech Laboratory
7, Chapaeva str.,
Zhytomyr, 10029
Ukraine

E-mail: info@mmlsoft.com

Web: <http://www.mmlsoft.com>

Copyright © 2003-2009 by Hermitech Laboratory. All rights reserved.

Brief Table of Contents

About Formulator 3.9 MathML Weaver	7
List of supported features	11
Getting Started	16
Basic Concepts of Editing	18
Tutorial: MathML Presentation Markup with Formulator	32
Tutorial: MathML Content Markup with Formulator	53
Tutorial: Formulator ActiveX Control	141
Working with Mathematical Web Pages in Formulator	146
Formulator Conformance with MathML 2.0	150
Formulator Conformance with MathML 2.0: Test Suite	159
Appendix A. Keyboard Shortcuts	186

Detailed Table of Contents

About Formulator 3.9 MathML Weaver	7
Products List of Formulator 3.9 MathML Suite	8
Formulator 3.9 MathML Weaver License Agreement	8
List of supported features	11
Comprehensive collection of mathematical symbols and templates.....	11
Common standards conformance	13
WYSIWYG-style editing.....	13
Data import/export	14
Web browsers compliance	14
Cooperating with other editors	15
Getting Started.....	16
Installing Formulator	16
Formulator Setup.....	16
Removing Formulator	16
Moving Formulator to a different folder	17
Start menu items.....	17
Checking for the latest version and Feedback	17
Basic Concepts of Editing.....	18
Main Window Elements.....	18
Modes of Mathematical Expressions Editing.....	19
Status Bar	22
Styles	23
Sizes	23
Menus.....	24
File menu	24
Edit menu	25
View menu	26
Style menu.....	27
Size menu.....	28
Options menu	29
Window menu	30
Help menu	30
Tutorial: MathML Presentation Markup with Formulator	32
Token Elements	32
Fractions and roots.....	35
Scripts and Limits.....	37
Enclose Expression Inside Notation	38
Tables and Matrices	40
Style Change.....	46
Error Message	52
Tutorial: MathML Content Markup with Formulator	53
General Principles of Editing and Navigation.....	53
Token Elements: externally defined symbol (csymbol).....	64
Token Elements: numbers (cn) and identifiers (ci).....	69
Basic Content Elements: apply	75

Basic Content Elements: functions and operators	82
Basic Content Elements: invisible and transparent elements	91
Arithmetic, Algebra, Logic and Relations	99
Sequences and Series	111
Constructor elements: theory of sets and linear algebra	115
Calculus and Vector Calculus: Integral and Differentiation.....	121
Partial Differentiation	126
Statistics	131
Read-only elements.....	133
Combining Presentation and Content Markup	134
Tutorial: Formulator ActiveX Control.....	141
Working with Mathematical Web Pages in Formulator.....	146
Exporting expressions to XHTML web pages with the MathML markup	146
Viewing XHTML web pages with the MathML markup	146
Formulator Conformance with MathML 2.0	150
Implemented Elements and Attributes	150
General.....	150
Presentation: Token Elements	150
Presentation: General Layout	151
Presentation: Scripts and Limits.....	151
Presentation: Tables and Matrices	152
Presentation: Enlivening Expressions	152
Content: Token Elements	152
Content: Relations	152
Content: Sequences and Series	153
Content: Basic Content Elements	153
Content: Theory of Sets	153
Content: Arithmetic, Algebra and Logic.....	154
Content: Linear Algebra	155
Content: Calculus and Vector Calculus	155
Content: Constants and Symbol Elements.....	156
Content: Elementary Functions.....	156
Content: Statistics	157
Content: Semantic Mapping Elements	157
Notes	158
Formulator Conformance with MathML 2.0: Test Suite.....	159
General.....	159
Math	159
GenAttribs	159
Presentation.....	160
TokenElements	160
GeneralLayout	162
ScriptsAndLimits.....	165
TablesAndMatrices	166
DynamicExpressions.....	167

Content.....	167
TokenElements	167
BasicContentElements	168
ArithmeticAlgebraLogic.....	169
Relations	171
Calculus	172
TheoryOfSets.....	172
SequencesAndSeries	173
ElementaryFunctions	174
Statistics	176
LinearAlgebra	176
SemanticMappingElements	177
ConstantsAndSymbols	177
Characters.....	178
EntityNames.....	178
NumericRefs.....	178
UTF8	179
Error Handling.....	180
BadAttribs.....	180
BadChildren	180
BadEntities.....	180
BadTags.....	180
NumChildren.....	181
Torture Tests	181
Size.....	181
Complexity	181
Topics	181
EmbellishedOp	181
LargeOp	181
LineBreak.....	182
Nesting.....	183
Primes.....	183
Accents	183
StretchyChars.....	183
WhiteSpace	184
Notes	185
Appendix A. Keyboard Shortcuts.....	186
Formatting.....	186
Menu Commands.....	186
Navigation and Selection	187
Toolbar Commands.....	187

About Formulator 3.9 MathML Weaver

Formulator is a powerful interactive mathematical expressions editor. It uses WYSIWYG-style editing and allows creating mathematical equations through simple point-and-click techniques. Formulator is the intelligent software, aware of presentation or semantics face of mathematics. Advanced editing features are available by means of MathML authoring techniques which help Formulator to implement effective solutions for a wide audience of software developers, educators, students, STM publishers, etc. Among the distinguishing features of Formulator is component versions support (in the form of ActiveX Control or Automation Server) and extensive support of MathML 2.0 (Presentation, Content and Mixed markups) through easy-to-use graphical interface.

Exporting functionality of Formulator is extremely useful when a user wants to share you math documents. With the help of Formulator users have a freedom to publish their work in so different ways as MathML plain text fragments, XHTML documents with MathML inlined, raster images (BMP, GIF, JPG, PNG), vector graphics (EMF). Moreover, by means of standards MS Windows mechanisms, Formulator can work with other editors, word processors, presentation programs, HTML and XML-authoring tools and many other types of software, to create equations for tests and class materials, Web pages, research papers, etc.

Formulator is visually oriented tool. It proposes intuitive and easy-to-use ways to create and edit both Presentation and Content MathML markups. Comprehensive collection of mathematical symbols and templates of Formulator is useful both to users having publishing needs and to those who is oriented on analysis of expressions meaning. For each basic mathematical construct, Formulator provides a template containing graphics and edit boxes. There are many template groups and mathematical operators, including fractions, radicals, sums, matrices, various types of brackets and braces, etc. Equations can be easily created by inserting mathematical templates and filling in their edit boxes. Users can insert templates into the edit boxes of other templates, and in that way complex hierarchical formula can be built up.

Field of applications for Formulator is really wide, since it comprises both a standalone application edition, run as a separate program, and a component edition, that is, an ActiveX control, that can be incorporated into any ActiveX container. When using the Formulator ActiveX Control this revolutionary functionality makes it simple to develop new software products, keenly aware of the mathematical typesetting and semantics rules. This helps to dramatically accelerates application development in computer-aided education, computer algebra systems, authoring tools, and many other applications for mathematics, science, business, etc.

Products List of Formulator 3.9 MathML Suite

Hermitech Laboratory provides a range of different solutions based on the interactive mathematical expressions editor. This set of products is known as *Formulator 3.9 MathML Suite*:

- mathematical expressions editing (based on MathML 2.0 standard)
 - standalone edition – *Formulator MathML Weaver (freeware)*
 - component editions – *Formulator ActiveX Control*
- mathematical expressions rendering API
 - *Formulator 3.9 API*
- plug-ins (for Internet Explorer)
 - *Formulator MathML IE Performer (freeware)*
- support of mathematics learning activities
 - *Formulator Tarsia (freeware)*

Formulator 3.9 MathML Weaver License Agreement

Please read the following terms and conditions before installing, executing and using standalone application edition of Formulator 3.9 MathML Weaver ("Software") provided by Hermitech, Laboratory of Mathematical and Modeling Software, Ukraine ("Hermitech Laboratory"). By installing and using the Software You indicate Your acceptance of the terms and conditions set in this License Agreement (the "License Agreement").

1. COMMERCIAL VERSION LICENSE

You may NOT use the Software under this License Agreement for commercial purposes. Commercial users can obtain a license from Hermitech Laboratory by sending an order by e-mail, using the following address: info@mmlsoft.com.

2. NON-COMMERCIAL VERSION LICENSE

2.1. Qualification for a Non-Commercial Version License.

To qualify for a Non-Commercial Version License, You must: (1) use the Software for non-commercial purposes as defined herein and be a Non-Commercial Entity as defined herein, or (2) be an University User as defined herein.

The term "Non-Commercial Entity" is limited to the following:

- (1) University or other educational institutions (such as pre-schools, elementary schools, middle or junior high schools, high schools, and community or junior colleges), non-profit organizations (such as public libraries, charities, and other organizations created for the promotion of social welfare).
- (2) "University Users" who use the Software for professional (occupational) duties (such as preparing of educational materials for later use in class) or personal use, and other individual users who use the Software for personal non-commercial purposes (such as hobby, recreational, or educational purposes). The term "University Users" is limited to students, faculty members,

researchers, administrators, support staff, and employees of a university or other educational institution when acting in this capacity.

If You do not qualify for a Non-Commercial Version License, then You should discontinue the downloading or installation process and purchase a Commercial Use License, or obtain an Evaluation License, or request Hermitech Laboratory for elucidation by sending an e-mail using the following address: info@mmlsoft.com.

For the avoidance of doubt, the following are considered examples of commercial uses of the Software:

- (1) bundling or integrating the Software with any hardware product or another software product for commercial use;
- (2) use at or for a commercial enterprise;
- (3) use for financial gain, personal or otherwise.

2.2. License Grant. As long as you comply with the terms of this License Agreement, Hermitech Laboratory grants to You a FREE OF CHARGE, limited, non-exclusive, non-transferable license to use the Software for NON-COMMERCIAL PURPOSES, without right to sub-license.

2.3. Intellectual Property Ownership, Copyright Protection.

This Software, any images and documentation incorporated into the Software are the intellectual property of and are owned by Hermitech Laboratory.

This license is not a sale of the Software or any copy of the Software. The Software contains valuable trade secrets of Hermitech Laboratory. All worldwide ownership of and all rights, titles and interests in and to the Software, and all copies and portions of the Software, including without limitation, all intellectual property rights therein and thereto, are and will remain exclusively with Hermitech Laboratory.

The Software is protected, among other ways, by international copyright laws. All rights not expressly granted herein are retained by Hermitech Laboratory.

2.4. Public Access (University Networks).

If You are a University User, as long as you comply with the terms of this License Agreement, You are hereby licensed to place a copy of the Software onto a network of Your University (or Your other educational institution) for use by other University Users.

2.5. Restrictions.

You may not modify, translate, reverse engineer, decompile, disassemble (except to the extent applicable laws prohibit such restriction), bundle with another software product or create derivative works based on the Software, rent, lease, transfer, license or otherwise transfer rights to the Software, or remove any proprietary notices, licenses, displays, installation procedures or labels on or in the Software, whether set forth in files or on media.

2.6. Disclaimer of Warranty.

You expressly agree that the use of this Software is at Your own risk. The Software is provided on an "AS IS" basis, without warranty of any kind, including without limitation the warranties that it is free of defects and errors, fit for a particular purpose, or non-infringing. The information and services provided by the Software and/or Hermitech Laboratory are similarly provided on an "AS IS" basis, without warranty of any kind. The accuracy and reliability of any information content or services provided by the Software and or Hermitech Laboratory should be independently verified by You as the user prior to making purchase decisions and or any other decisions based on such information content and services.

2.7. Limitation of Liability. To the maximum extent permitted by law, in no event will neither Hermitech Laboratory nor anyone else involved in the creation, production, or delivery of this Software be liable for any damages arising from the use of or inability to use the Software, including, without limitation, damages to users' systems and/or Software and/or data, computer failure or malfunction, performance delays, and all other damages or losses.

List of supported features

Formulator 3.9 MathML Weaver provides a range of solutions for people seeking to create materials containing mathematical notation or to develop new software that is acquainted with presentation or content side of mathematics.

Formulator comprises both a standalone application edition, that can be run as a separate program, and a component edition, that is, an ActiveX Control (or Automation Server), that can be incorporated into any ActiveX container (or Active Document container). The component edition is the perfect way for software developers to insert mathematics rendering/editing/processing functionality in their applications.

New software products take over from Formulator's component all its competence in math and easy-to-use graphical interface, thus becoming keenly aware of the mathematical typesetting and semantics rules. Professional support of MathML format in Formulator accelerates application development in such various topics, as computer-aided education (distance learning, electronic textbooks and other classroom materials); automated creation of attractive reports; computer algebra systems; authoring, training, publishing tools (both for web and desktop-oriented), and many other applications for mathematics, science, business, economics, etc.

Educators in different fields of knowledge can make use of Formulator to create tests and other classroom materials containing mathematic formulas. Formulator supports Content markup of MathML and so proposes an easy way of automatic processing mathematics, expanding abilities of students and educators to better represent, encode, and reuse mathematical applications and contexts.

STM publishers can use Formulator to allow better and quicker interaction in the field of expressing and approaching ideas. Mathematical notations are forming a significant part of the common communication process and must be computer-aided as well. By using the Presentation markup of MathML Formulator provides an easy way to put dynamic math pages on the Web and to support desktop publishing applications.

There is a tendency to store STM publishing content expressed in XML. Since MathML is the XML-based language for mathematics, it is an inviting decision to represent STM content using MathML. When the MathML coded expressions must be edited, Formulator is really a good choice.

Comprehensive collection of mathematical symbols and templates

Formulator has a vocabulary of mathematical operators with more than five hundreds of individually set records. In addition, there are several hundreds of mathematical symbols which can be treated as operators with common properties and rendered using preinstalled fonts of the operation system.

Except of solitary symbols, there are a lot of mathematical templates in Formulator. Each of these templates represents a form with graphics and empty slots. By inserting other mathematical templates and symbols into empty slots hierarchical formulas can be built up.

Mathematical templates in Formulator provide dual facilities to edit formulas. In accordance with the MathML approach to mathematics coding, there are “Presentation” and “Content” templates. The first group of templates is of special interest to users having publishing needs and has a lot of presentation abilities: fractions, radicals, sums, integrals, products, matrices, various types of brackets and braces, and many other templates. The second group is oriented on mathematical semantics and is extremely useful when the meaning of the entered formula is critical.

A group of Presentation mathematical templates comprises:

- relational and logical symbols
- spaces templates
- operator symbols
- arrow symbols
- set theory symbols
- special constants
- miscellaneous symbols
- Greek characters (lowercase)
- Greek characters (uppercase)
- differentiation templates
- fence templates
- fraction and radical templates
- subscript and superscript templates
- summation templates
- integral templates
- underbar and overbar templates
- labeled arrow templates
- products and set theory templates
- table templates
- box templates

A group of Content mathematical templates comprises:

- token elements
- basic content elements
- piecewise declaration templates
- arithmetic operators
- algebra operators
- logic operators
- maximum and minimum templates
- relations templates
- calculus and vector calculus templates
- theory of sets templates
- sequences and series templates
- common trigonometric functions
- common hyperbolic functions
- common exponential functions

- statistics templates
- linear algebra templates
- semantics templates
- constant and symbol elements
- qualifier elements templates

Common standards conformance

Formulator creates mathematical expressions in accordance with the following W3C Recommendations:

- [XML 1.0](#)
- [MathML 2.0](#)

The Mathematical Markup Language, or MathML, is an XML application for describing mathematical notation and holding both its presentation and content. MathML is designed to provide the encoding of mathematical information for the powerful, general software tools exchanging, processing and rendering mathematical data. Formulator uses MathML as a main intermediate language while editing formulas and so proposes to a user a rich functionality of operations with MathML, working with Presentation, Content and Mixed markups.

By supporting MathML Formulator brings several additional benefits to users, including publicity of the standard, easiness of sharing mathematical data, equal opportunity to use the standard for both presentation and computation motives. Finally, MathML is based on XML and brings all the advantages of XML.

This collection of documentation includes materials which cover conformance to MathML 2.0 standard in great details. Please see “Formulator MathML 2.0 Conformance” and “Formulator MathML 2.0 Test Suite” for the further discussion.

WYSIWYG-style editing

Formulator is intuitive and visually oriented tool; its easy-to-use graphical interface is built according to modern conventional techniques. Formulator allows you to create mathematical equations through simple point-and-click techniques, but it also provides some kind of intelligent behavior, in the sense that it knows mathematical typesetting rules and elements of semantics of mathematics (e.g., automatically resizing square root signs and parentheses to fit their contents, inserting appropriately sized spaces around mathematical operators and relational symbols, etc.).

There are many customizable features for mathematical expressions editing that makes this process comfortable to different users with different needs:

- “zoom” feature allows equations to be magnified so that small details such as hats, primes, subscripts and superscripts can easily be seen;
- “nested view” can be used to better see the structure of mathematical equations;
- by using font and character styles, a user can quickly change the appearance of characters;

- users can allow Formulator to select font size automatically or manually apply needed type of size;
- users can execute some Formulator operations directly from the keyboard via shortcuts;
- customizable built-in toolbars make it easier to access predefined MathML symbols and templates (both for Presentation and Content markups, and for their combination, that is called “Mixed markup”).

Data import/export

The most powerful resource to export is MathML representation of formulas. Formulator is closely connected with this approved standard and so any importing/exporting actions for MathML are naturally integrated into routine editing process, namely, in operations with the Clipboard and file open/save commands.

Another way to communicate in mathematics provides such means of export as converting formulas into graphic files and publishing mathematics on web.

The first feature is easy to use with the corresponding menu commands. Abilities of graphics export are rich for various purposes. The resulting files can be generated in a vector format (Windows Enhanced Metafile) or in several raster formats (BMP, GIF, JPG, PNG).

Ability to publish mathematics on web is provided by following to W3C recommendation to have web pages written using XHTML with the MathML markup inlined. This feature is available from the menu command or by means of an additional mode of editing mathematical expressions (the “XHTML” option). The resulting web page can be easily viewed with modern popular internet browsers, either using additional plug-ins (e.g., Hermitech's own plug-in for Internet Explorer, known also as “Formulator MathML IE Performer”, which is based on Formulator), or by browser's native methods.

Web browsers compliance

Expressions created using Formulator may be saved as images which can be published on the web, or alternatively, Formulator can export XHTML documents containing MathML expressions to display mathematical notation within a web page. By using the XSLT stylesheet for MathML provided by the W3C Math Working Group, the resulting XHTML documents can be viewed with the following web browsers:

- MS Windows:
 - Internet Explorer (5.5 and greater) with Hermitech's plug-in (MathML IE Performer)
 - Internet Explorer (5.5 and greater) with MathPlayer plug-in
 - Internet Explorer (5.0 and greater) with Techexplorer plug-in
 - Mozilla

- Netscape 7.0
 - Netscape 6.1 with Techexplorer plug-in
- Macintosh:
 - Mozilla
 - Internet Explorer (5.0 and greater) with Techexplorer plug-in
- Linux/Unix:
 - Mozilla
 - Netscape 7.0
 - Netscape 6.1 with Techexplorer plug-in

Cooperating with other editors

The easiest way to communicate with other applications is to import/export formulas in plain MathML text using simple operations with the Clipboard. For example, thus you can use Formulator with your favorite HTML or XML editor. But there are additional means of integrating mathematic expressions created by Formulator into external word processors (e.g., Microsoft Word). Just using Drag-and-Drop or Copy/Paste commands, a user can insert formulas into the word processor and further edit them using Formulator by double-clicking on them in the document. When static image of a formula is needed, then a user can export expressions to vector or raster graphics and use them on the web or in editors.

Getting Started

Installing Formulator

To install Formulator, finish all non-essential running applications, and then simply run the Formulator Setup program:

- If you received Formulator on a CDROM, Windows will probably run Setup automatically when you insert the CD. If it doesn't, just use the Run command to run SETUP.EXE in the Formulator folder of the CD.
- If you downloaded Formulator electronically, run the self-extracting executable that you saved on your hard disk. Any temporary files will be automatically deleted when Setup is finished.

To complete the installation, just follow the instructions in Setup's dialogs.

Formulator Setup

This dialog allows you to install Formulator. Use this dialog to install Formulator to the folder shown in the Destination Folder box.

Destination Folder:

Use this area to change the folder on your hard disk where Setup will install most of Formulator's files. Use the Browse button to navigate to a folder. Setup will also install fonts and system files into appropriate system folders on your hard disk.

OK

Click OK to begin the installation process.

Exit

Click Exit to terminate Setup and abort the installation.

Removing Formulator

To remove Formulator, do one of the following:

- Choose "Uninstall Formulator" from the Formulator sub-menu in Windows Start menu.
- Choose Control Panel from the Windows Start menu, double-click Add/Remove Programs, choose Formulator from the list of removable applications.

This will run removing process for the Formulator application. Follow the instructions presented in dialogs.

Moving Formulator to a different folder

If you decide to move Formulator and its components to a different folder on your hard disk, DO NOT just move the files using the Windows Explorer. This will not properly update Formulator and OLE's entries in the Windows Registry and Formulator will not work properly. We recommend removing Formulator and then re-installing it in the new location.

See also sections "Removing Formulator" and "Installing Formulator".

Start menu items

When Formulator is installed, a Formulator sub-menu is added to the Windows Start menu. It contains the following items:

Formulator

Runs the Formulator application.

Formulator Help

Opens the help file at its Table of Contents.

Formulator User Manual

Opens the Portable Document Format (PDF) version of the Formulator User Manual.

Uninstall Formulator

Removes the Formulator application.

Checking for the latest version and Feedback

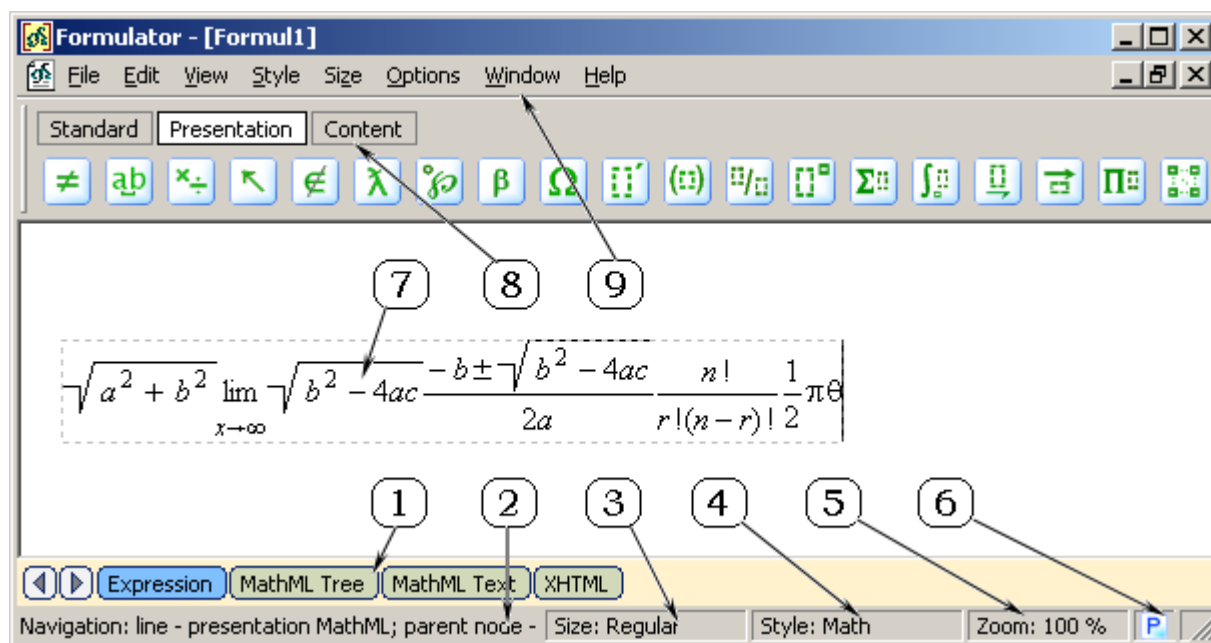
Please use your Web browser to access information about the latest version and editions of Formulator via the Internet on our site: <http://www.mmlsoft.com>.

Formulator has been designed and developed with a care about needs of end users. We always enjoy hearing from our users, so if you have a bug to report, a new feature to request, or anything else to say, please let us know. Please send your messages to support@mmlsoft.com.

Basic Concepts of Editing

Main Window Elements

The following picture shows general layout of the main windows of Formulator.



Legend:

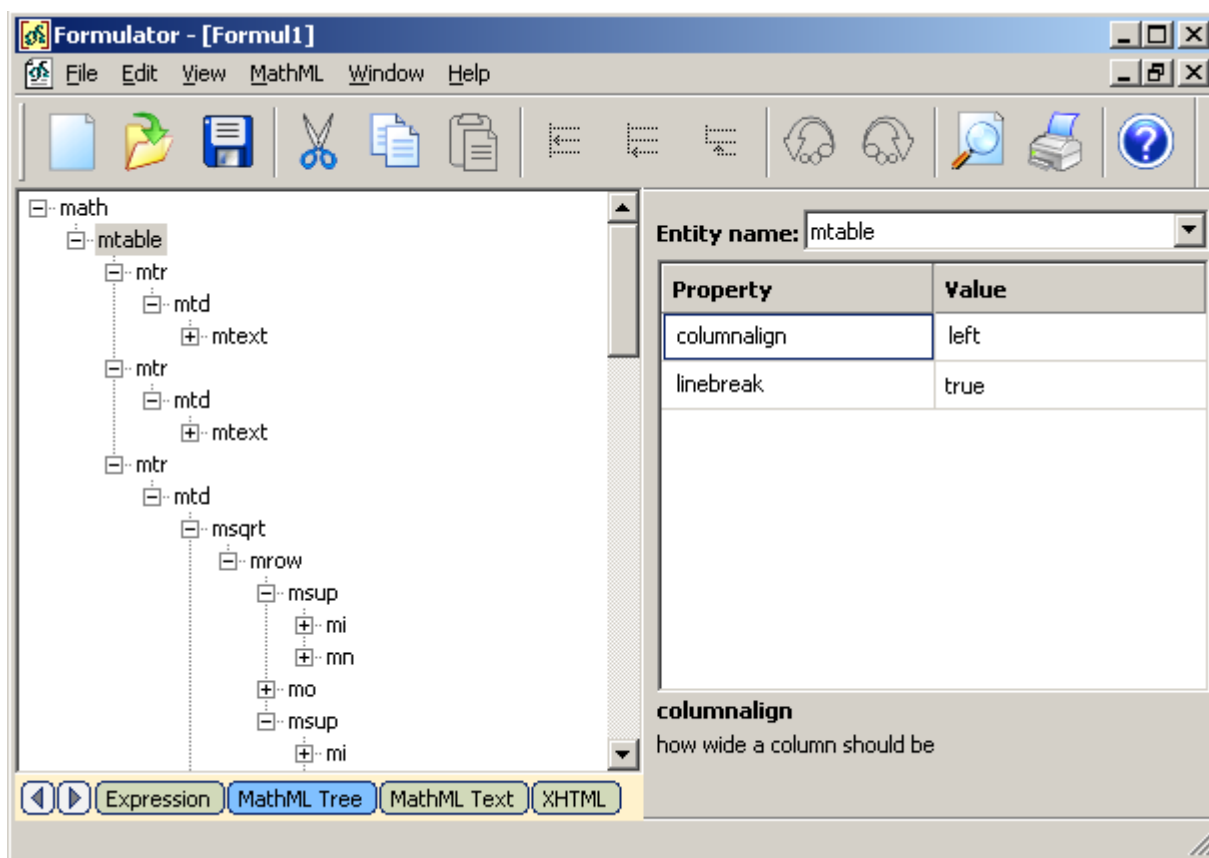
- 1 Bottom navigation bar (shows available modes of mathematical expressions editing).
- 2 Expression navigation information (shows type of MathML markup under the cursor – Presentation of Content, corresponding types of the current formula node and its parent node).
- 3 Indication of the size for the current formula node.
- 4 Indication of the style for the current formula node.
- 5 Current zoom factor.
- 6 Status icons (current input mode – Presentation of Content; mode of invisible symbols displaying and warning about need/possibility to refresh the view of the current document).
- 7 Document area.
- 8 Tabbed toolbar (can be switched to the common set of toolbars).
- 9 System menu.

Modes of Mathematical Expressions Editing

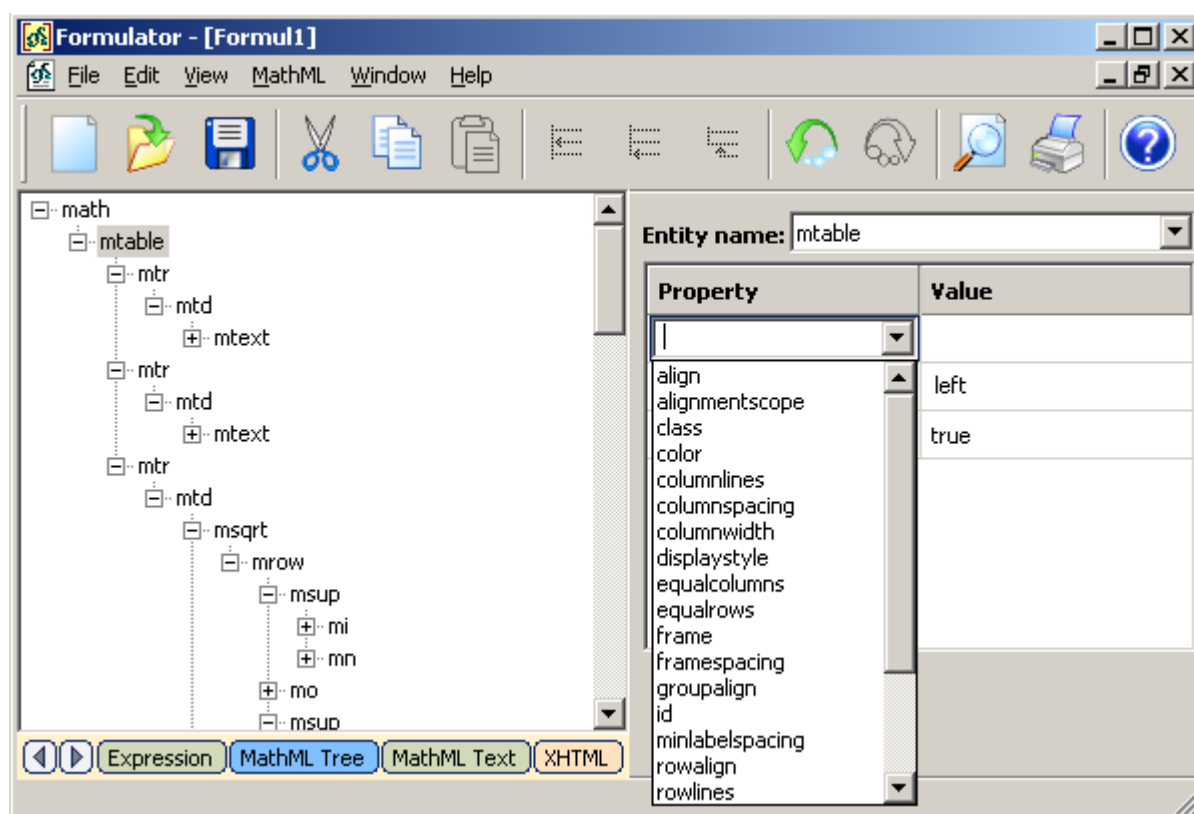
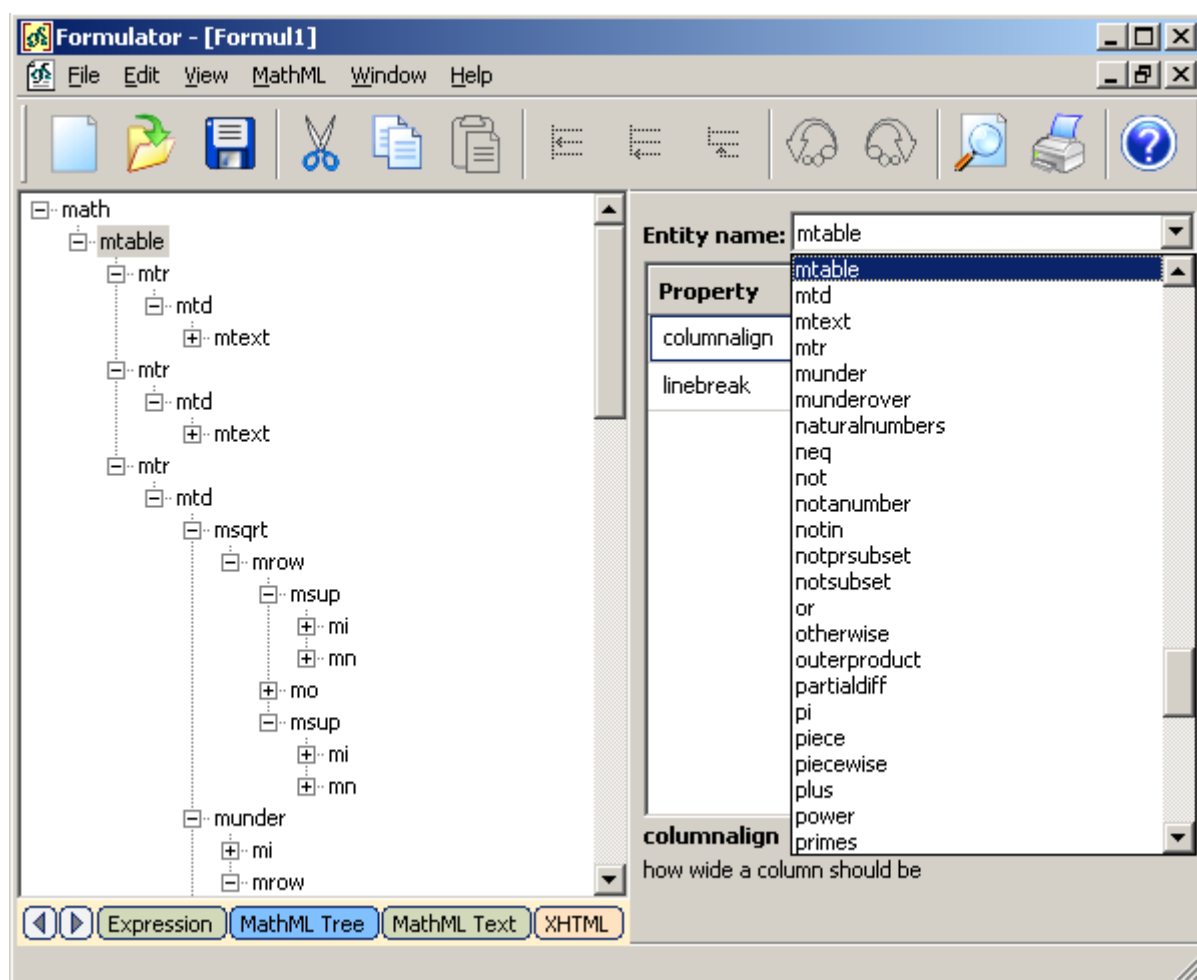
When using the full version of Formulator a user benefits from several modes of editing mathematical expressions. The list of such modes is given at the bottom of a document and includes “Expression”, “MathML Tree”, “MathML Text” and “XHTML” options. Formulator Express supports only the main option (“Expression”); other modes are unavailable.

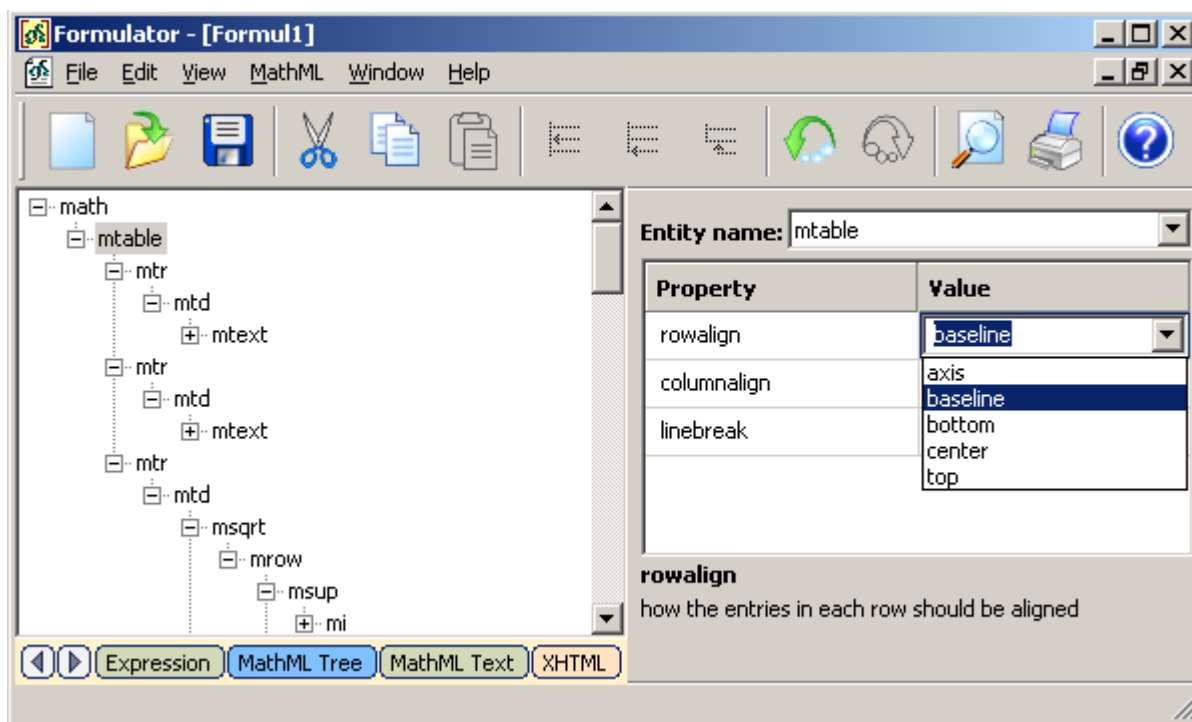
The first mode of editing presents usual WYSIWYG technology and allows creating mathematical equations through simple point-and-click techniques. The above considered figure of the main windows shows exactly this kind of editing mode.

The next mode is “MathML Tree”. It allows to build up mathematical expression as a MathML tree or to do a fine tuning of MathML tags and attributes for the before created formula. The document area consists of two parts. The left is for manipulation with structure of the MathML tree and the right side is for editing the current MathML node (selected on the left side).



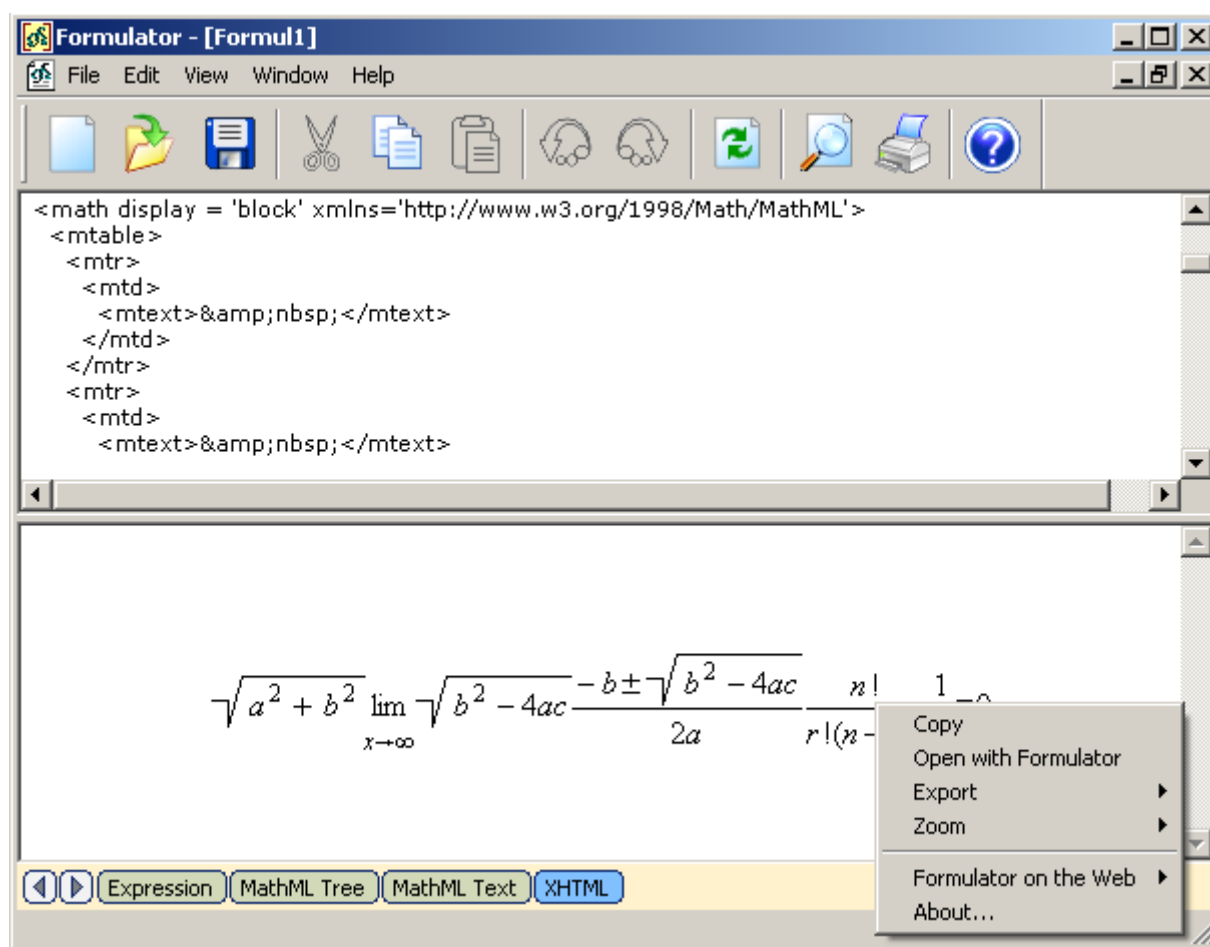
The right document area proposes smart editing of tree, making use of knowledge about elements supporting currently in MathML 2.0. The following figures show how to use predefined lists of tags and attributes. Note also the brief description of the current element at the bottom of the right document side.





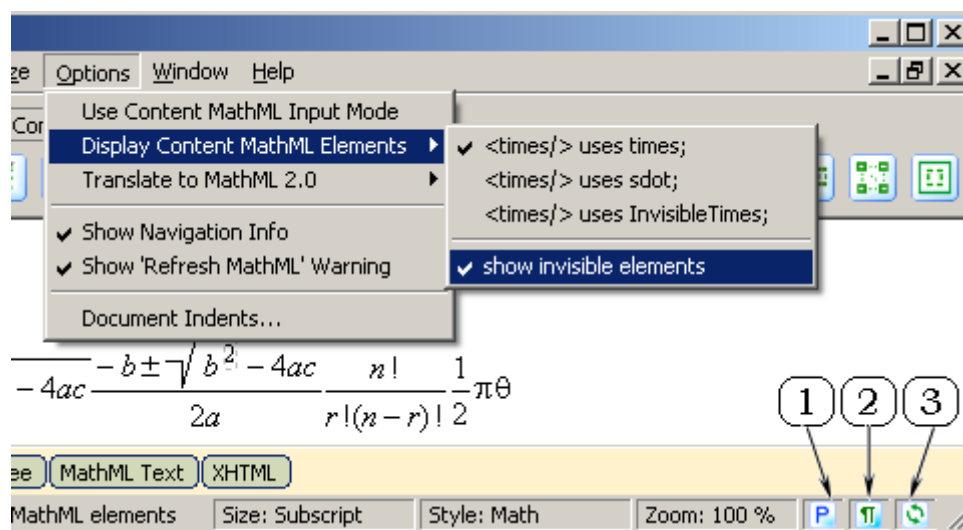
The mode of “MathML Text” is generally repeats the previous mode, but is implemented as a simple text editor (for example, Notepad). It allows to interfere in visual editing results on-the-fly by switching from the “Expression” view, changing values or attributes and switching back to the main editing mode. Sometimes this kind of editing mode can do more for the understanding of the internal document structure than others.

The last editing mode shows how the current mathematical document will be viewed after publishing on web. Ability to publish mathematics on web is provided in Formulator 3.9 MathML Weaver by following to W3C recommendation to have web pages written using XHTML with the MathML markup inlined. We use for playing mathematics on web our own plug-in for Internet Explorer, known also as “Formulator MathML IE Performer”. It is based on Formulator and so can be easily inserted into the main body of MathML Weaver when a user switches to the “XHTML” editing mode. This feature allows to make use of the menu shortcuts of the MathML IE Performer (see the following figure).



Status Bar

The “Expression” editing mode uses rich information status bar, consisting of the expression navigation information (can be switched of by using Options menu items); indication of the size and style for the current formula, indication of the current zoom factor and status icons, which are shown on the next figure.



Legend:

- 1 Indicating the current input mode ('P' is for Presentation Markup and 'C' is for Content Markup). It should be used along with the Option menu item "Use Content MathML Input Mode". Selecting Content MathML Input Mode leads to inserting of Content MathML mathematical templates when a user presses a sign of the corresponding operation.
- 2 Mode of invisible symbols displaying. The red color of the icon shows that some formula elements (for example, `<declare>`, `<bvar>`, etc.) are invisible; otherwise MathML Weaver forces rendering of such elements. This can be useful for the visual editing of some auxiliary MathML elements. Surely, a user can switches to another editing mode and do needed alterations by hand, but WYSIWYG technology is more appropriate in this case.
- 3 Warning about need/possibility to refresh the view of the current document. It displays notification to a user when the current document needs to be refreshed through MathML. This feature is needed when changing of an option influences all or part of text presentation. E.g., it could be the case of using another symbol for `<times/>` element of the Content markup, or beautifying of such a formula in Content markup that uses additional slots for inputting `<bvar>` elements.

Styles

Each character in a Formulator equation can be directly assigned a specific font and character style or can be of one of eleven styles. Each style is defined as a combination of a font and character style. By changing the style of a text fragment, a user can quickly define its appearance and behavior rules and by changing the definition of a style, a user can quickly change the appearance of all the characters that use it.

The 11 styles available in Formulator are Text, Variable, Function, Greek, Vector-Matrix, Number, Fixed, Operator, Extra-Math, User 1, and User 2. Formulator assigns styles to certain kinds of characters automatically, based on its knowledge of mathematics and typesetting conventions. This intelligent assignment of styles is a useful feature of Formulator which can significantly simplify your work. Assigning style Math to a set of characters a user can explicitly define this intelligent behavior to be a rule for these characters.

Sizes

Formulator provides default font sizes for each edit box in a mathematical expression. There are four reserved cases of changing font size:

- regular text;
- large symbols (like sums, products, integrals, etc.);
- subscript/superscript text;
- subscript/superscript nested in the subscript/superscript text (subscript/superscript of the next level).

You can allow Formulator to select font size automatically or manually apply needed type of size using items if the “Size” menu.

If you think that text should be of particular size, there is an option to assign explicitly any font size between 5 and 127 points using the “Other...” item of the “Size” menu. Similarly, items “Smaller” and “Larger” allow to change current font size by 1 point.

Default values of reserved types of font size can be changed using dialog box “Define size” (the “Define...” item of the “Size” menu).

Menus

File menu

- **New (Ctrl+N)**

Opens a new, empty, equation window. This window will be untitled until you give it a name when you save it as a file using the Save or Save As commands on this menu.

- **Open (Ctrl+O)**

Opens an existing Formulator equation file from disk, and displays it in a new window. When you choose the Open command, Formulator displays the standard Open dialog box, which lets you change drives and directories until you find the file you need, and then open it either by double-clicking or by using the OK button. If you worked on a Formulator file recently, you may be able to open it more quickly by choosing its name from the bottom of the File menu.

- **Close (Ctrl+F4)**

Closes the equation window. If you have made changes to the equation in this window, a dialog box will appear asking you if you want to save these changes. You can also close a window by clicking the Close Box in the upper-right corner of the window.

- **Save (Ctrl+S)**

Saves the current version of the equation that you’re working on. If your equation is untitled, a Save As dialog box will appear so that you can choose a name for it. Once an equation has been named and saved on disk, using the Save command again will replace the previous version with the new one. If you want to keep the previous version in addition to the new one, use the Save As command described below.

- **Save As**

Preserves the current version of the equation that you’re working on by saving it on disk. You use the Save As command, rather than the Save command, when you want to save an untitled equation, or when you want to save an equation under a new name, in a different directory, or in a different file format.

- **Print (Ctrl+P)**

Prints the equations contained in the currently active Formulator window.

- **Print Preview**

Displays equations contained in the currently active Formulator window the way they will be printed.

- **Print Setup**

Sets up the printing parameters.

- **Export**

Exports the current version of the equation that you're working on by saving it on disk in a graphic file or a web-page.

- **Exit (Alt+F4)**

Terminates the Formulator (or Formulator Server) program. If you have made changes to equations in open Formulator windows, you will be asked if you want to save those changes.

- **(Most-recently-used files)**

Up to four Formulator equation files that you recently worked on are listed at the bottom of the File menu. You can re-open any of these files just by choosing its name from this menu. This is simply a convenient way to open a file, saving you the step of locating the file on disk.

Edit menu

- **Undo (Ctrl+Z)**

Undoes the last command as shown in the text of the menu item. Every command can be undone, back to the point at which the window was opened. This item will be grayed out when there are no more commands to be undone.

- **Redo (Ctrl+Y)**

Redoes the most recent Undo command. Every Undone command can be redone.

- **Cut (Ctrl+X)**

Copies whatever is currently selected to the Clipboard, and deletes it from the equation. Equation is placed on the clipboard in two formats: binary and textual MathML format. Depending on the application that is accessing to this data, one of these copies will be used automatically.

- **Copy (Ctrl+C)**

Copies whatever is currently selected to the Clipboard. Equation is placed on the clipboard in two formats: binary and textual MathML format. Depending on the

application that is accessing to this data, one of these copies will be used automatically.

- Paste (Ctrl+V)

Inserts the contents of the Clipboard into the equation at the insertion point, or replaces whatever is selected with the contents of the Clipboard. For this command to be successful, the Clipboard must contain a Formulator equation or a text. If the text doesn't meets requirements of the MathML format, it will be inserted literally.

- Select All (Ctrl+A)

Selects the entire equation, including portions that may be outside the bounds of the window. This command is useful for subsequently copying the equation to the Clipboard for transfer to a word processing document, or before using the Backspace or Delete keys to delete the entire contents of the equation window.

- Refresh All Through MathML

Exports the entire document to MathML and imports it back. This feature is needed when changing of an option influences all or part of text presentation. E.g., it could be the case of using another symbol for `<times/>` element of the Content markup, or beautifying of such a formula in Content markup that uses additional slots for inputting `<bvar/>` elements.

- Insert Image...

Opens an existing image and inserts it in the current document. When exporting formulas into MathML, Formulator considers images as an external resource (a reference to a graphics file).

- Insert MathML Element...

Creates a new MathML element using Properties dialog, that lets to select existing MathML name for a new tag and to insert/edit/delete attributes and values in intuitive and visually oriented manner.

- MathML Element Properties...

Edits existing MathML element using Properties dialog, that lets to select a MathML name for the tag and to insert/edit/delete attributes and values in intuitive and visually oriented manner.

View menu

- Change Layout

This command switches modes of displaying the toolbar. Available modes are tabbed toolbar (Standard, Presentation and Content pages) and conventional toolbar (each page is presented by a separate toolbar that can be floating or docked).

- Toolbars

This pop-up menu comprises items which toggle the display of each page of the tabbed toolbar or show/hide separates floating/docked toolbars. A checkmark next to this menu item indicates that toolbar is currently displayed.

- Tabbed toolbar

This command toggles the display of the entire tabbed toolbar. A checkmark next to this menu item indicates that toolbar is currently displayed.

- Status Bar

This command toggles the display of the status bar. A checkmark next to this menu item indicates that the status bar is currently displayed.

- Zoom

This item displays the Zoom menu which allows you to change the viewing scale.

- Show Nesting (Ctrl+Shift+N)

This command toggles the equation display between normal viewing mode and nesting mode where the you can see the hierarchical structure of your equations.

- Show Read-Only (Ctrl+Shift+R)

This command toggles the equation display between normal viewing mode and “show read-only nodes” mode where the you can see highlighted that nodes which can't be changed.

- Line Spacing...

This command displays a dialog box that changes default value of distance between neighbour lines.

- Refresh All Through MathML

This command exports the entire document to MathML and imports it back. The need for this command appears when the current document rendering differs from normative for the sake of comfortable editing. E.g., it can be the case of visual editing of 'bvar' element which should be invisible when printing. Another example is changing of presentation equivalent for some content elements.

Style menu

- Math

The Math style is the default style for typing mathematics. When the current style is Math, Formulator assigns styles to certain kinds of characters automatically, based on its knowledge of mathematics and typesetting conventions. E.g., if Formulator recognizes a sequence of alphabetic characters as a standard operator (from MathML operator dictionary), it will use the Operator style.

- Text (Ctrl+Shift+E)

Use the Text style when you want to type a plain text (e.g., in a natural language) instead of in math.

- Function (Ctrl+Shift+F)

Use this command to create a function name.

- Variable (Ctrl+Shift+V)

Use this command to assign the Variable style to characters.

- Greek (Ctrl+Shift+G)

This command allows you to use the keyboard to type Greek letters. Once this style is active, the letters you type will be given the Greek style. You can also enter a single Greek letter by using the Ctrl+G one-shot keyboard shortcut or by using the Greek symbol palette.

- Vector-Matrix (Ctrl+Shift+B)

Mathematical vectors and matrices are given a bold character style. Use this command to assign the Vector-Matrix style to selected text or subsequently typed characters.

- Fixed (Ctrl+Shift+X)

Use this command to assign the Fixed (monospace) style to characters.

- Operator (Ctrl+Shift+O)

Use this command to assign the Operator style to characters which are not recognized as a standard operator.

- User 1 (Ctrl+Shift+U)

- User 2 (Ctrl+Alt+Shift+U)

The User 1 and User 2 styles may be used any way you like. Use the Define Styles dialog to assign a font and character style to each.

- Other style...

This command brings up the Other Style dialog, allowing you to assign a font and character style and color to selected text or subsequently typed characters.

- Define style...

This command brings up the Define Styles dialog, allowing you to change the font and character style and color assigned to each style.

Size menu

- Large Symbol

Use this command to assign the Large Symbol type of font size to characters.

- Regular

Use this command to assign the Regular type of font size to characters.

- Subscript

Use this command to assign the Subscript type of font size to characters.

- Sub-Subscript

Use this command to assign the Sub-Subscript type of font size to characters.

- Other...

This command brings up the Other Size dialog, allowing you to assign a font size to selected text or subsequently typed characters.

- Smaller (Alt+<)

This command decrease a font size of selected text by 1 point.

- Larger (Alt+>)


This command increase a font size of selected text by 1 point.

- Define

This command brings up the Define Size dialog, allowing you to change default characters sizes (regular, subscripts, etc.).

Options menu

- Use Content MathML Input Mode

Switches between Content and Presentation MathML Input Mode (the last one is used by default). Selecting Content MathML Input Mode leads to inserting of Content MathML mathematical templates when a user presses a sign of the corresponding operation. E.g., pressing '+' in the Presentation MathML Input Mode leads to inserting of the `<mo>` element (Presentation markup); in the Content MathML Input Mode such an action inserts a mathematical template () for the `<apply>` element with the operator element `<plus/>`.

- Display Content MathML Elements

The first three options of this pop-up menu item are for optional rendering of the `<times/>` element (Content markup). It can be rendered using such presentation elements as: `"×"`, `"⋅"`, and `"⁢"`.

The next option ('show invisible elements') forces rendering of invisible Content MathML elements, namely: 'declare', 'momentabout', 'annotation-xml' and in some expressions 'bvar'.

- Translate to MathML 2.0

The first three options of this pop-up menu item are for selecting whether a namespace will be used while converting mathematical expression to MathML.

The next two options toggles rendering of the `<math>` element in XHTML between 'block' and 'inline' modes.

The next option ('use symbol name for Unicode characters') defines whether Formulator should try to find the equivalent entity name for Unicode characters while converting expressions into MathML (otherwise the corresponding number will be used).

- Show Navigation Info

Defines whether Formulator should display navigation information about the current and the parent node in the Status Bar.

- Show 'Refresh MathML' Warning

Defines whether Formulator should display notification to a user when the current document needs to be refreshed through MathML. This feature is needed when changing of an option influences all or part of text presentation. E.g., it could be the case of using another symbol for `<times/>` element of the Content markup, or beautifying of such a formula in Content markup that uses additional slots for inputting `<bvar>` elements.

- Document Indents...

This item selects indents values for a document.

Window menu

- New Window

Opens a new, empty, equation window. This window will be untitled until you give it a name when you save it as a file using the Save or Save As commands on this menu.

- Cascade

Rearranges all open equation so they overlap in a cascade.

- Tile

Tiles all open equation windows vertically.

- Arrange Icons

Arranges the icons of the windows you have minimized, neatly at the bottom left of the window.

- (open windows)

The titles of all open equation windows are listed at the bottom of the Window menu. You can bring any of these windows to the front just by choosing its name from this menu.

Help menu

This menu contains commands that bring up Formulator's online help, or find out about the Formulator application (e.g. its version number).

- Contents and Index (F1)

This command displays the table-of-contents for Formulator's online help.

- Tip of a Day

This command displays the 'Tip of a Day' dialog.

- Formulator on the Web

This pop-up menu includes items for quick open in the default internet browser or e-mail client of URL's, connected with Formulator software (home and ordering page for the standalone and component version of the product, feedback address, downloads page).

- Registration

This command opens the Registration dialog that allows to enter the purchased registration code.

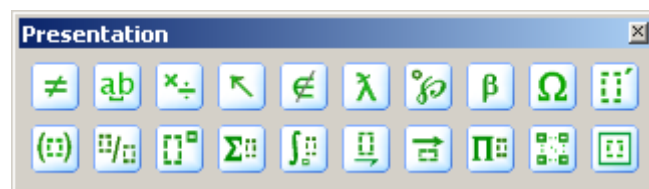
- About Formulator...

This command displays Formulator's About box showing you:

- The version of the Formulator application you are currently using;
- What kind of version (Evaluation or Registered) you are working with;
- Hermitech Laboratory contact information.

Tutorial: MathML Presentation Markup with Formulator


Mathematical templates in Formulator provide dual facilities to edit formulas. In accordance with the MathML approach to mathematics coding, there are “Presentation” and “Content” templates. The first group of templates is of special interest to users having publishing needs and has a lot of presentation abilities: fractions, radicals, sums, integrals, products, matrices, various types of brackets and braces, and many other templates. The second group is oriented on mathematical semantics and is extremely useful when the meaning of the entered formula is critical.



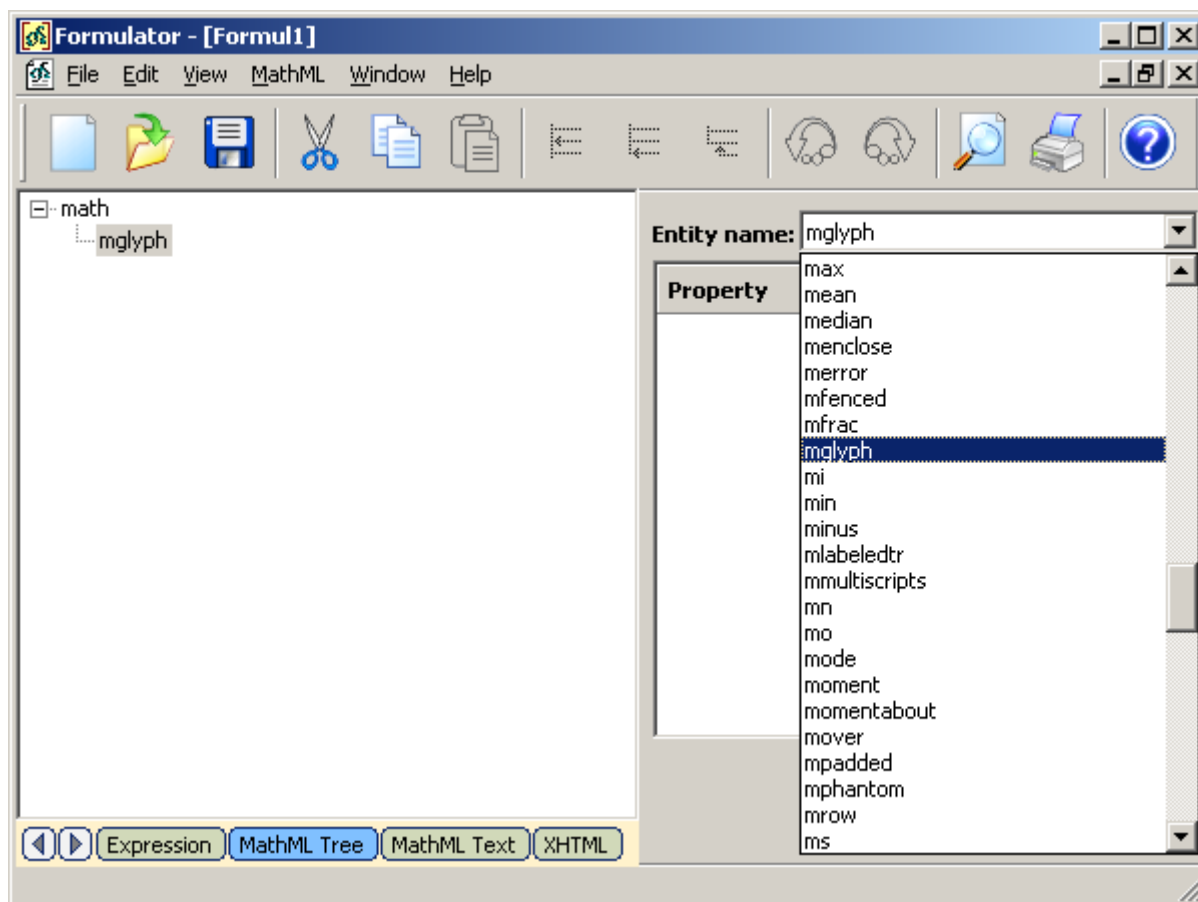
A group of Presentation mathematical templates comprises:

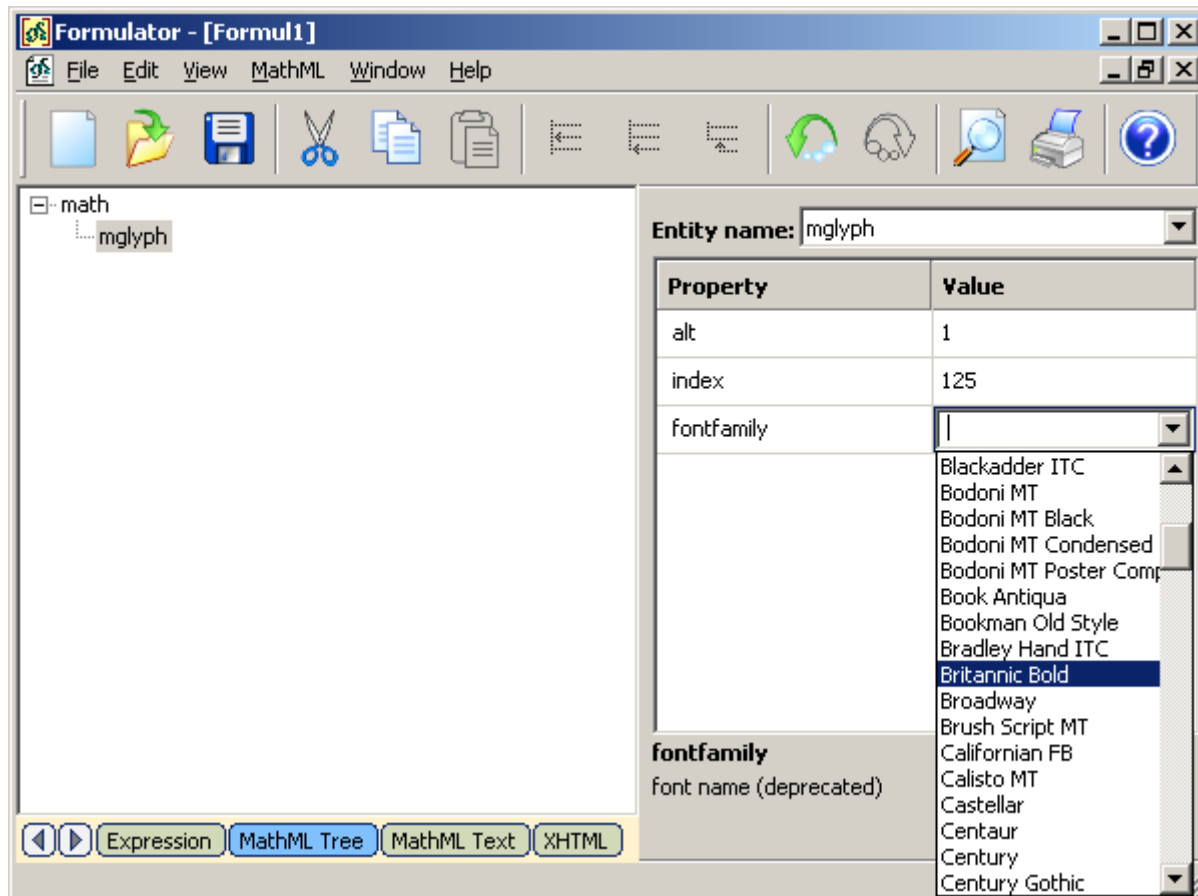
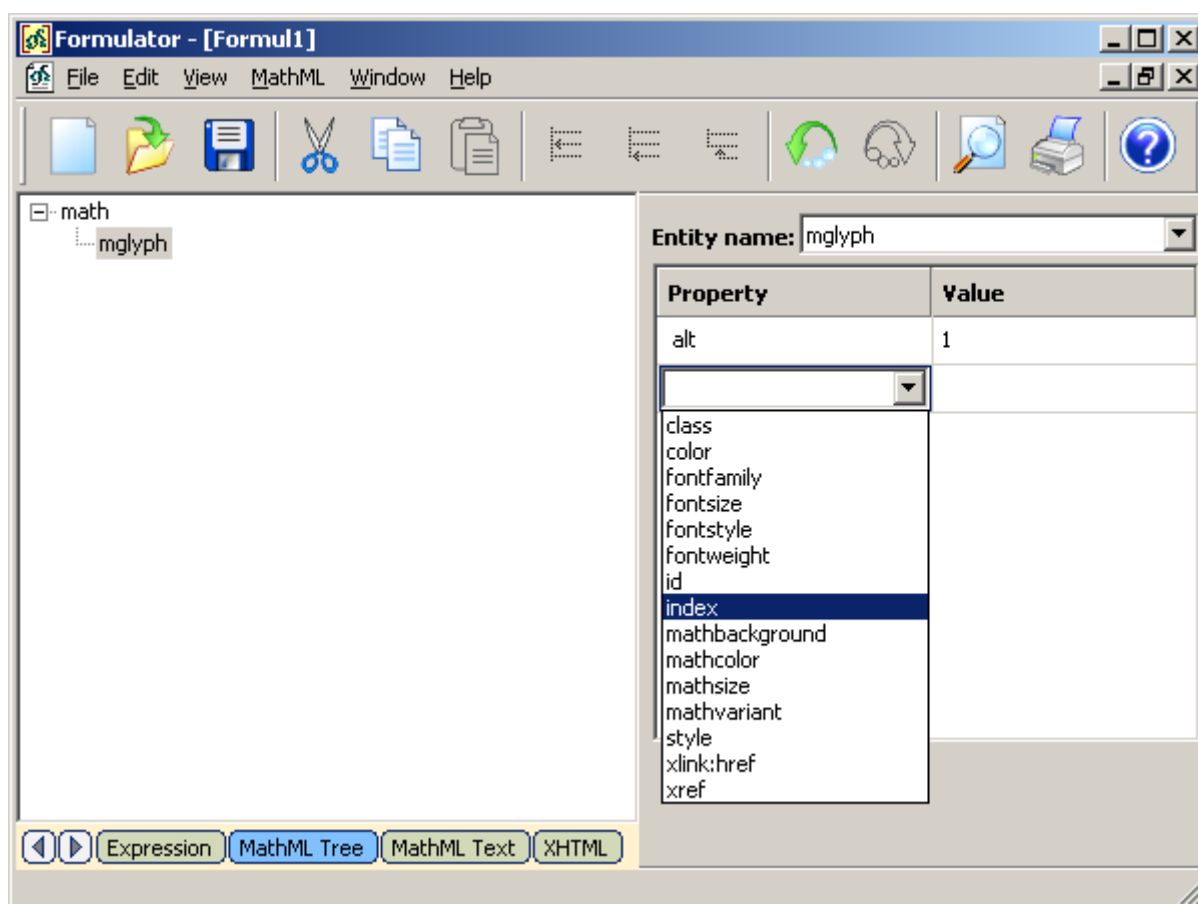
- relational and logical symbols
- spaces templates
- operator symbols
- arrow symbols
- set theory symbols
- special constants
- miscellaneous symbols
- Greek characters (lowercase)
- Greek characters (uppercase)
- differentiation templates
- fence templates
- fraction and radical templates
- subscript and superscript templates
- summation templates
- integral templates
- underbar and overbar templates
- labeled arrow templates
- products and set theory templates
- table templates
- box templates

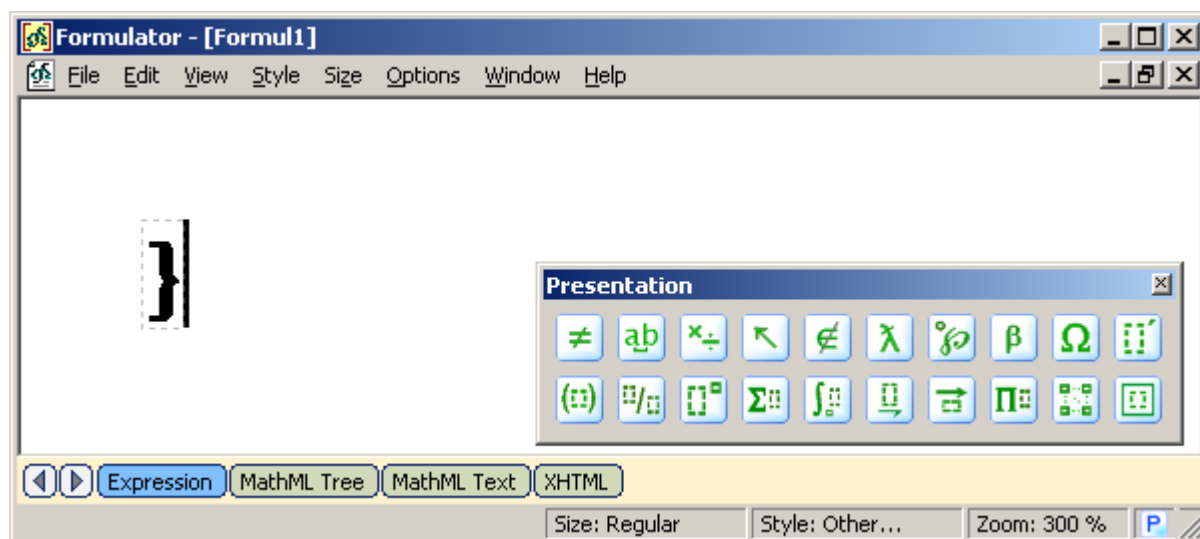
Token Elements

These elements can be just typed and their kind will be determined automatically if the current style is ‘Math’. If a user exactly knows what kind of token elements is needed then a Style menu item can be used to specify the kind (mi, mo, mn). Besides, there is a toolbar  for “mo” equivalents of space elements.

There are token elements that neither will be detected automatically when typing, nor have a special toolbar (ms, mspace, mglyph). They can be edited manually, using “MathML Tree” or “MathML Text” options. See the next figures for the explanation.




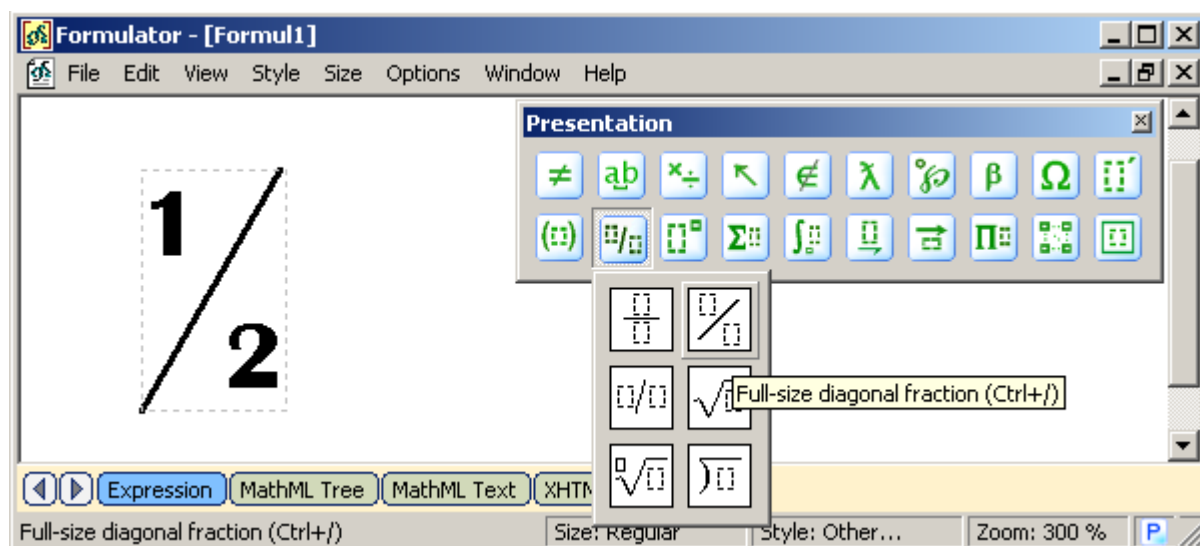


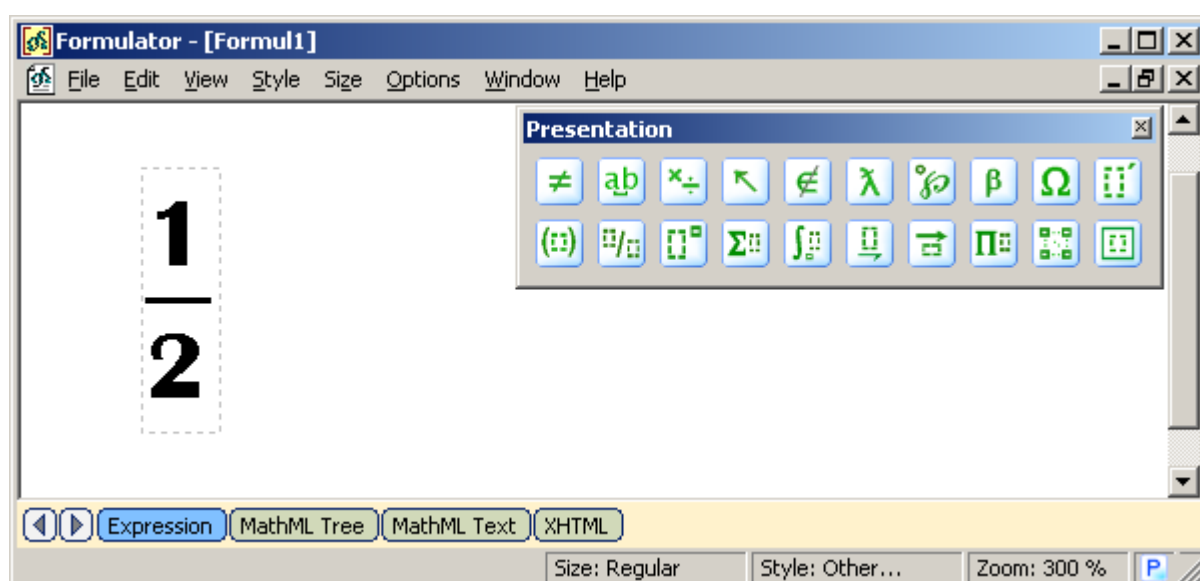
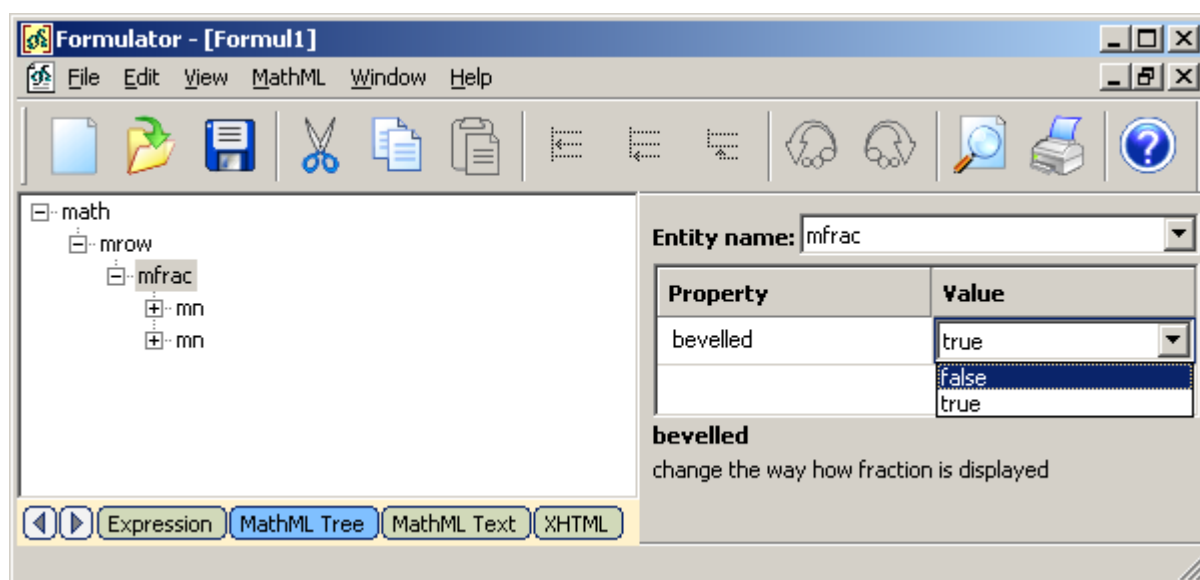


Fractions and roots

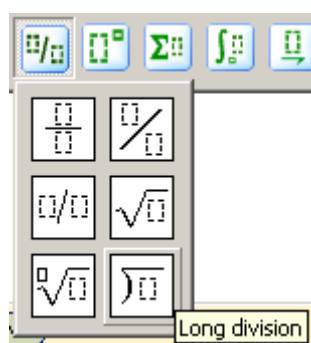
The “mfrac” element is used for fractions; the “msqrt” element is used for square roots, while the “mroot” element is used to draw radicals with indices.

Fractions and roots can be inserted via  toolbar. There are several buttons which take into account different available forms of fences and roots. Fine tuning or specific editing features can be accessed via the “MathML Tree” and “MathML Text” options. For example, the following figures show how to insert the “bevelled” version of a fraction and then convert it to a simple variant using “MathML Tree” editing mode.





Note that the button for the long division is also placed into this toolbar, but used another scheme of MathML encoding, the so-called, “enclosing notation” (menclose).

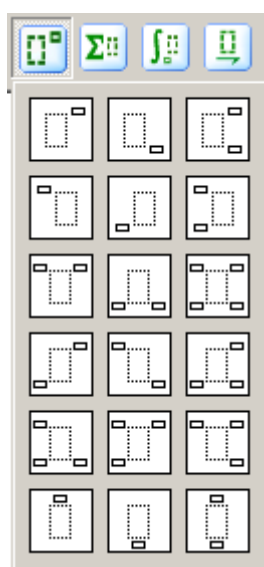


Scripts and Limits

These are the elements of the “Scripts and Limits” group:

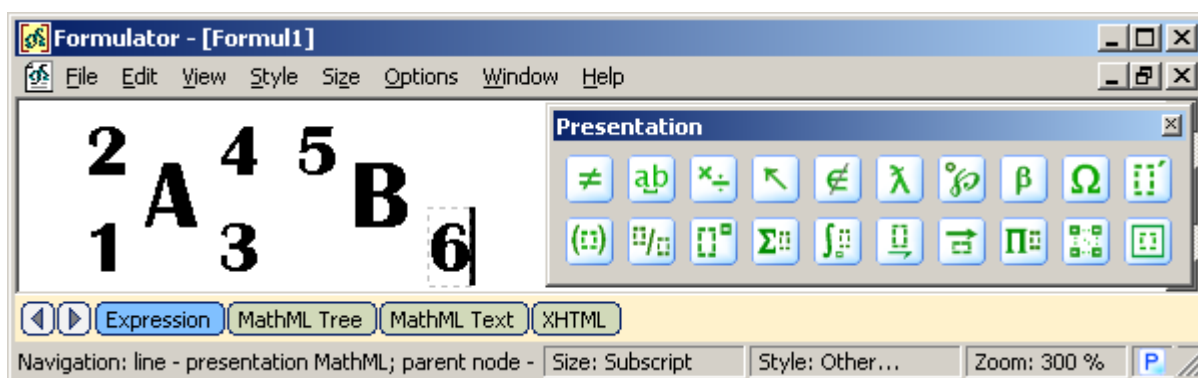
<code>msub</code>	attach a subscript to a base
<code>msup</code>	attach a superscript to a base
<code>msubsup</code>	attach a subscript-superscript pair to a base
<code>munder</code>	attach an underscript to a base
<code>mover</code>	attach an overscript to a base
<code>munderover</code>	attach an underscript-overscript pair to a base
<code>mmultiscripts</code>	attach prescripts and tensor indices to a base

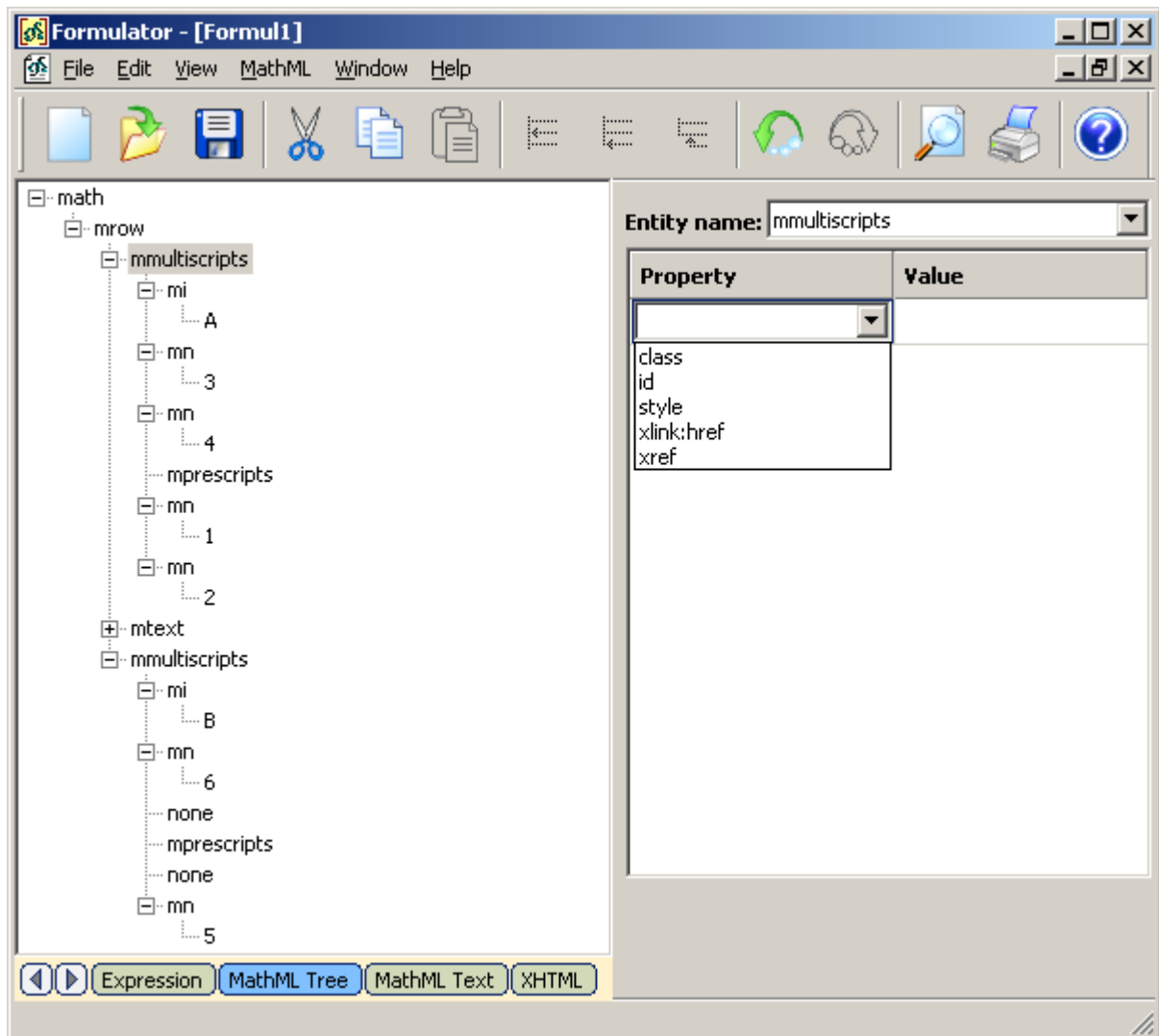
MathML Weaver works with script and limit schemata via  toolbar:



Simple elements of this schemata are implemented as partial cases (the first buttons line – `msub`, `msup`, `msubsup`; the last buttons line – `munder`, `mover`, `munderover`), but the most powerful MathML element here is “`mmultiscripts`”, since it is able to encode all the complicated parent-child slots relations of the script and limit schemata.

The next two figures show how the “`mmultiscripts`” forms the equivalent MathML tree for the different cases of scripts. Note application of the `<none/>` and `<mprescripts/>` elements.







Enclose Expression Inside Notation

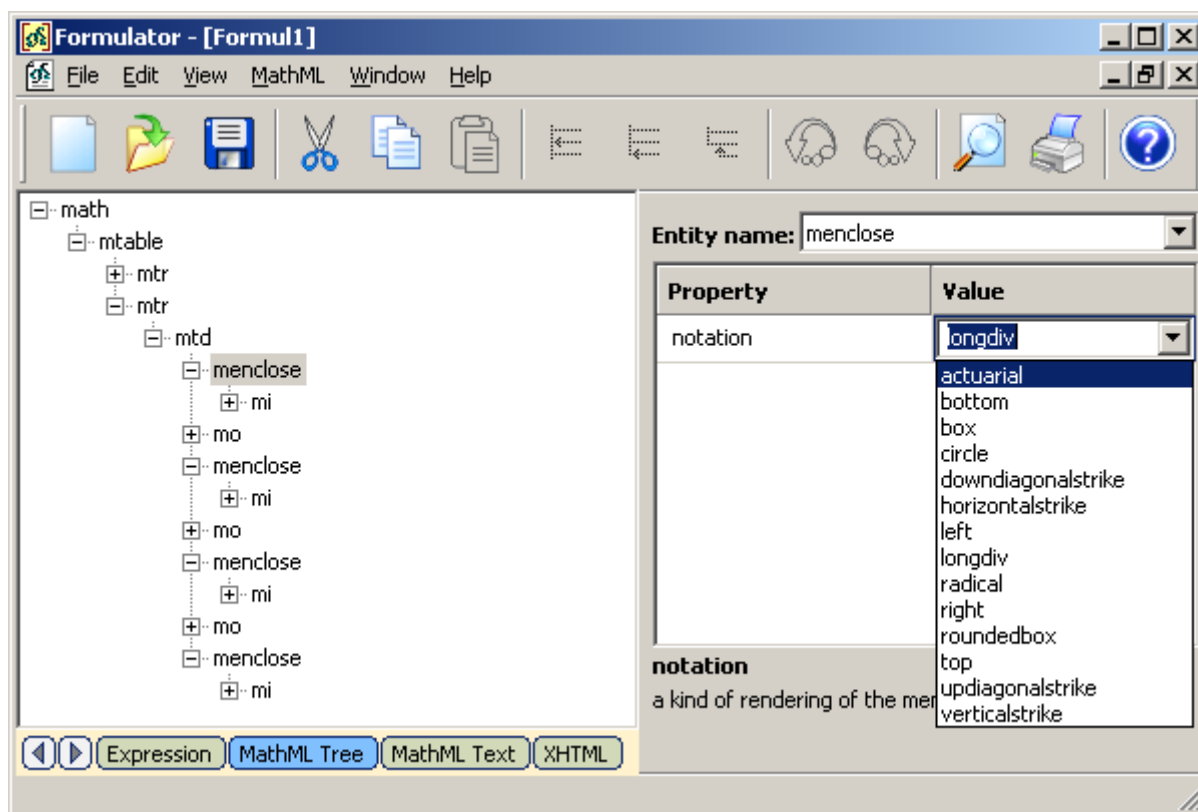
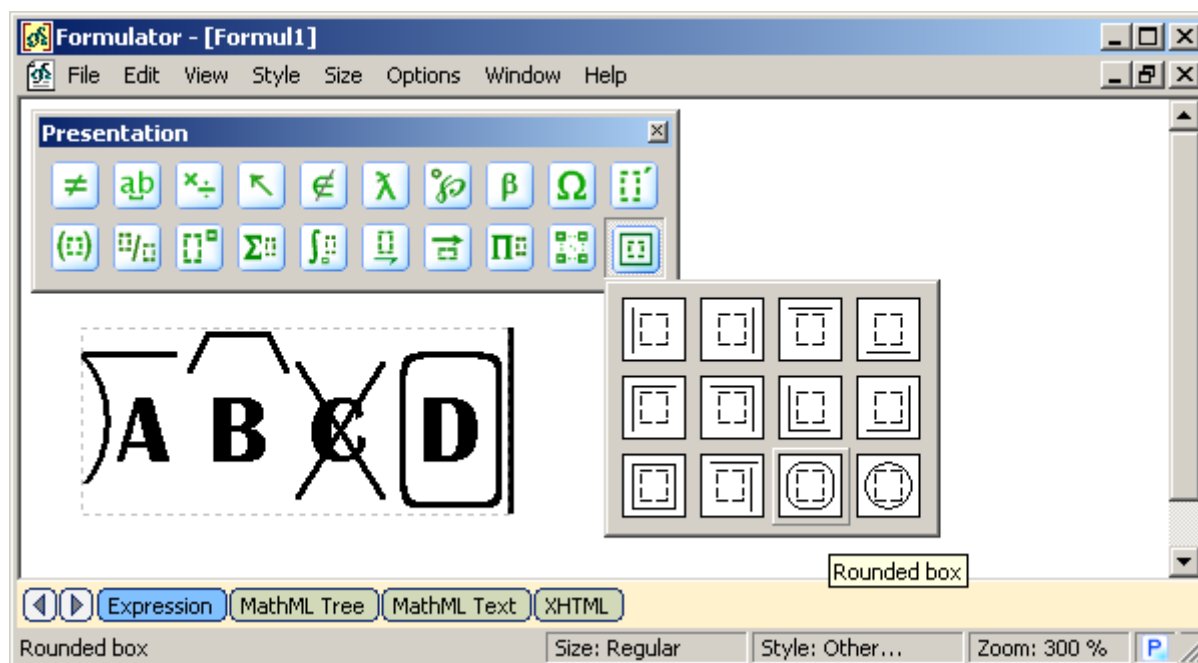
The “menclose” element renders its content inside the enclosing notation specified by its notation attribute. According to the W3C Recommendation (Mathematical Markup Language (MathML) Version 2.0), “the values allowed for notation are open-ended”. So, that is a place for handling encoding variants which are not directly specified in MathML 2.0.

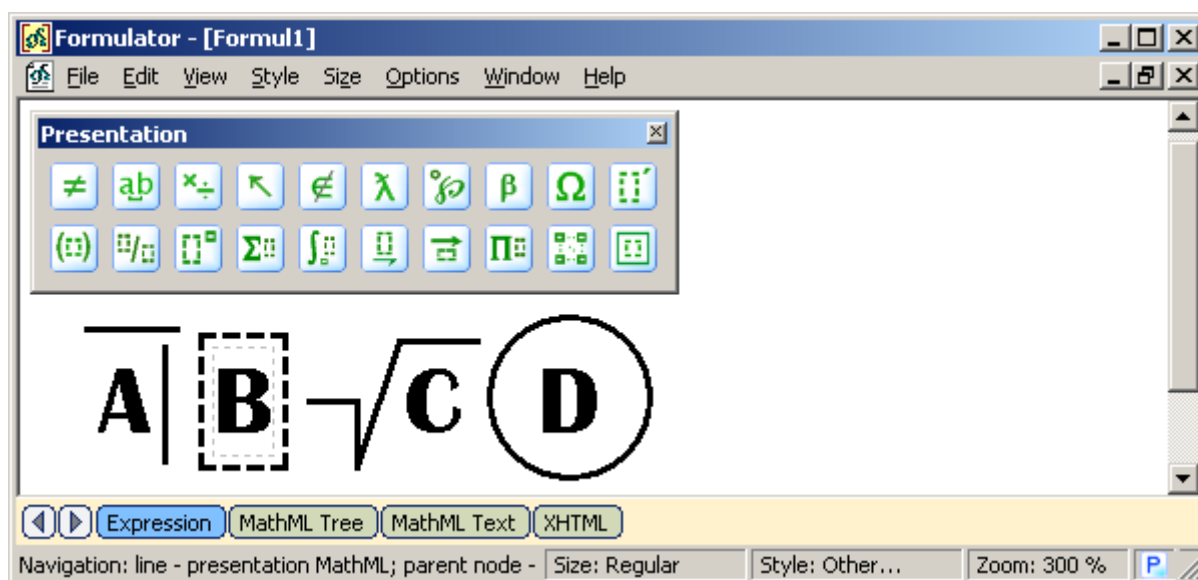
MathML Weaver benefits from the “menclose” element by implementing both the proposed by W3C values of the “notation” attribute (longdiv, actuarial, radical, box, roundedbox, circle, left, right, top, bottom, updiagonalstrike, downdiagonalstrike, verticalstrike, horizontalstrike) and its own values (joint-status, top-left, top-right, bottom-left, bottom-right, box-dashed). There are several toolbars for having “menclose” element, since buttons are grouped according more to their sense than to their MathML equivalent. A user can find “menclose” elements in such toolbars as:

- ✓ fractions and radicals ();
- ✓ underbars and overbars ();


✓ boxes ()

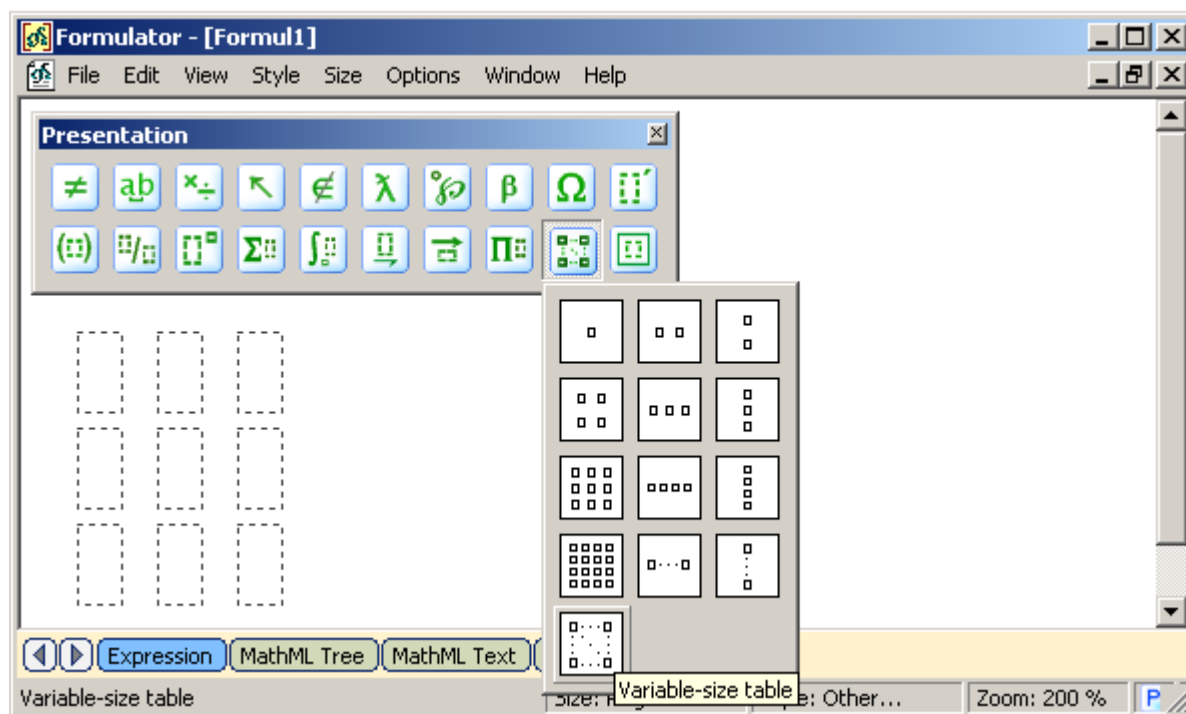
See the following figures for example of using “menclose” elements. We insert several formulas using “menclose” elements and then via WYSIWYG-style editing on the “MathML Tree” page we change values of the “notation” attribute. Results are shown by switching again on the “Expression” page.



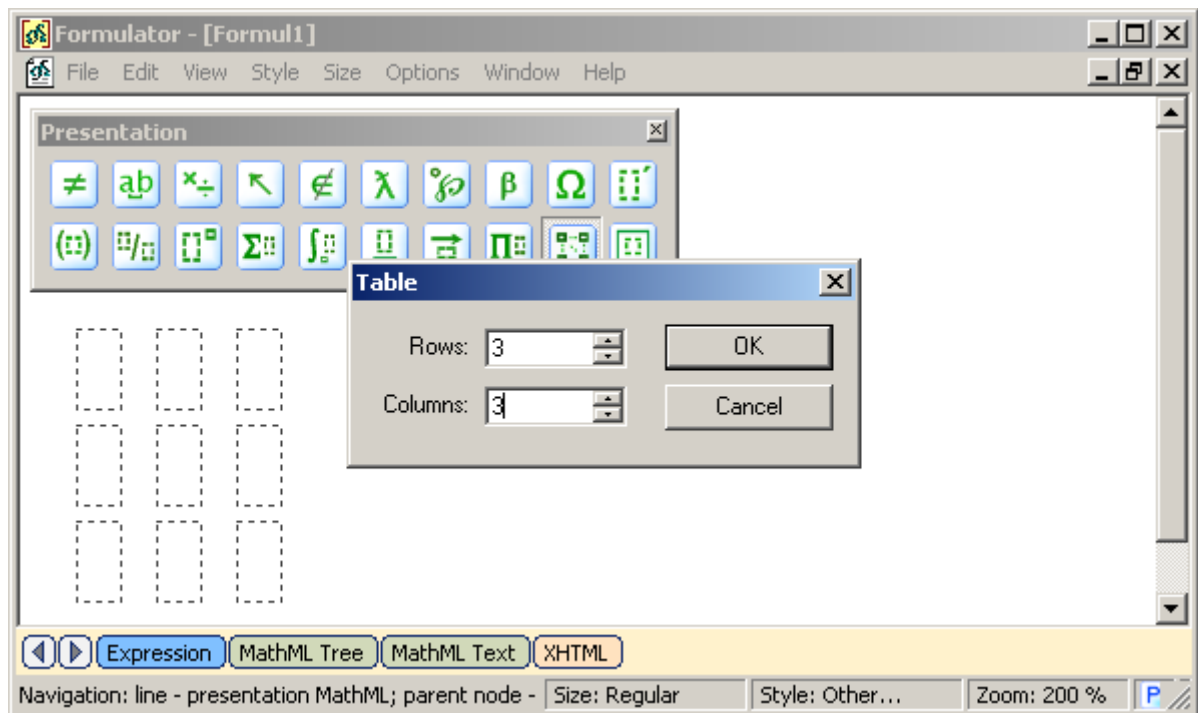


Tables and Matrices

A matrix or table is specified using the “mtable” element. It can be accessed via  toolbar:

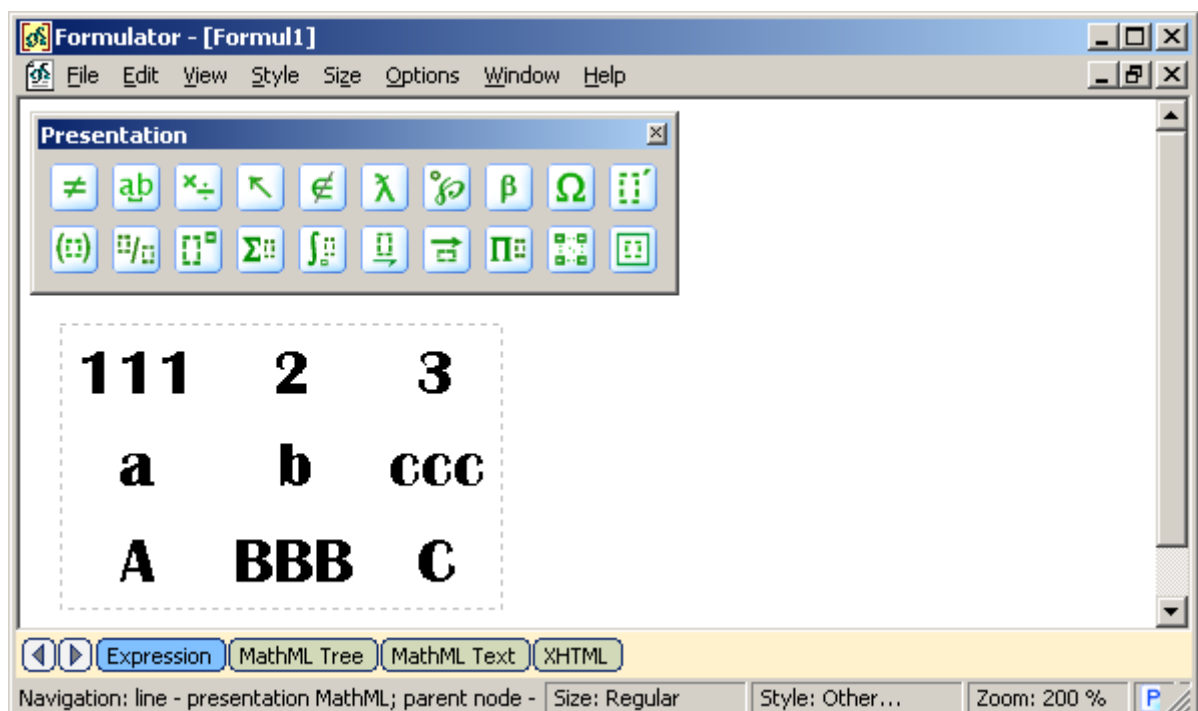


There are 12 buttons for simple cases of a matrix and one button that requests a user to enter the number of rows and columns:



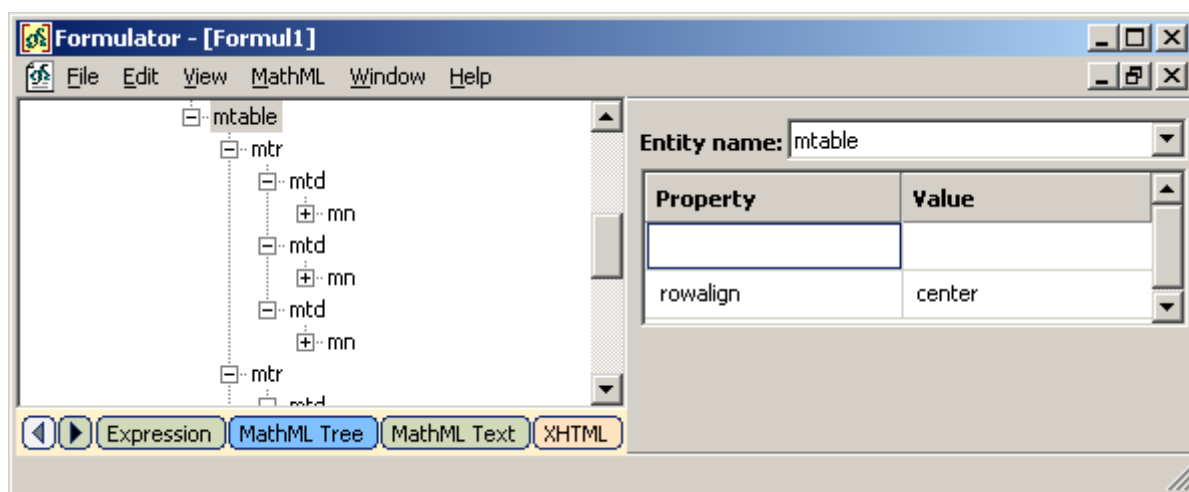
After inserting of a table or a matrix a user still has a chance to change its appearance via fine tuning of attributes for the “mtable”, “mtr” and “mtd” elements. The next example shows how to alter alignments and framing of all the table and of some of its cells.

1. Enter a table and fill it with values.

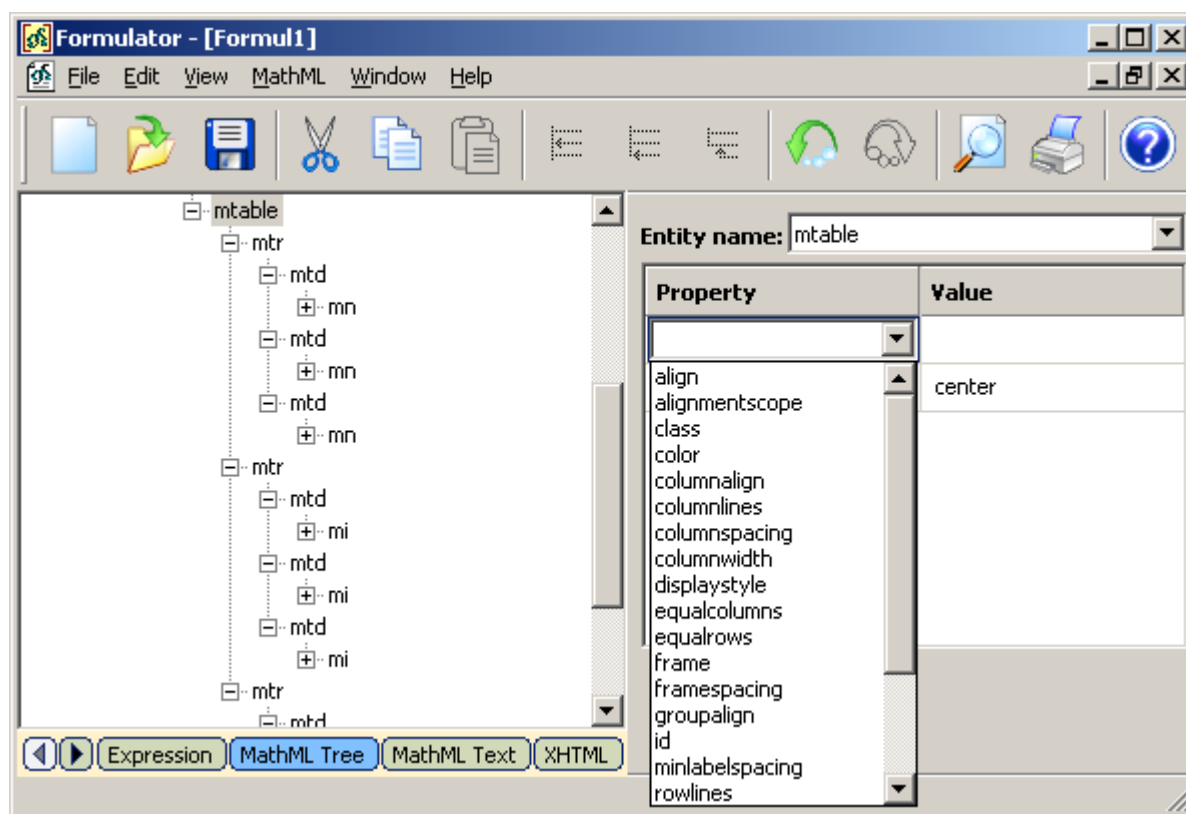


2. Switch to the “MathML Tree”, select “mtable” element on the left side of the document (note that in multiline expressions we don’t need the *outer* “mtable” element that is used for encoding several lines as rows of a table). Now the right side indicates current attributes of the “mtable” element.

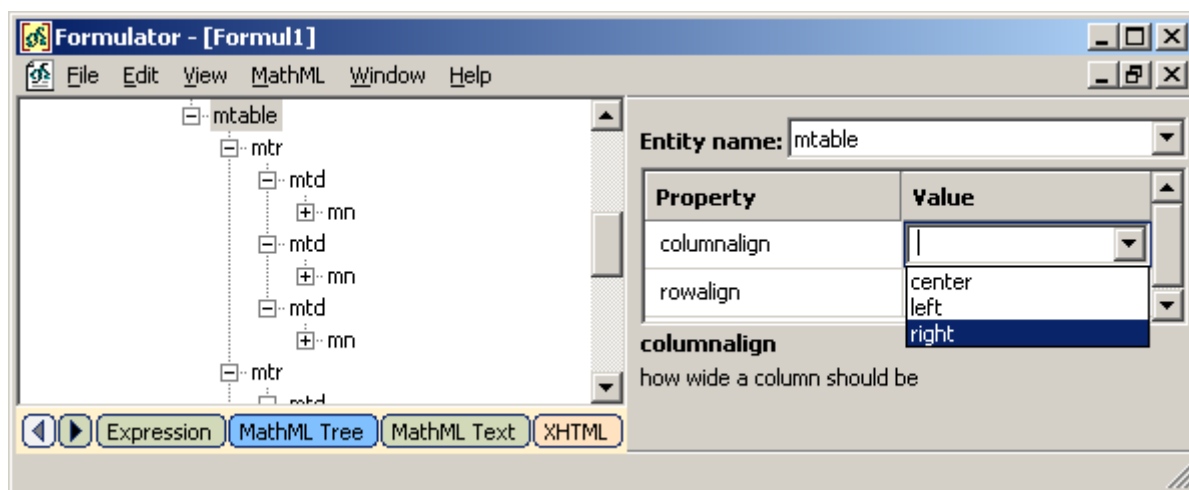
Click on the right side of the document to give it focus; then press the “Insert” button on the “Property-Value” pane. This will insert a new empty line for the attribute.



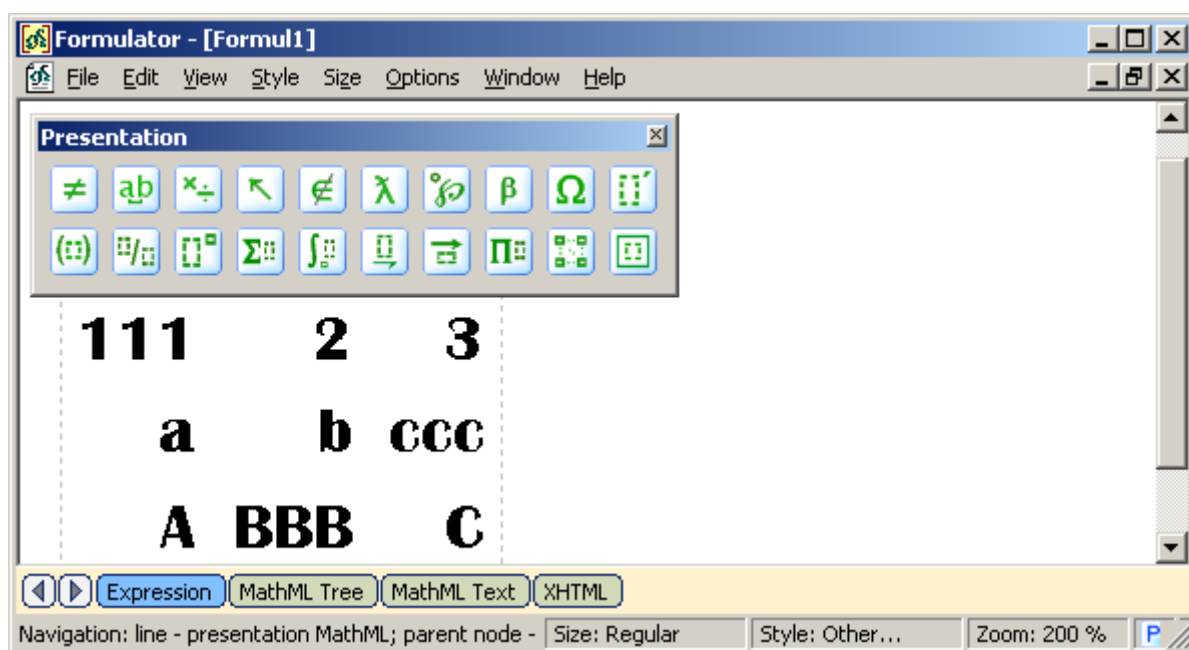
3. Click on the blank field of newly added attribute and use the drop list to get the list of all predefined attributes which are proper for the “mtable” element.



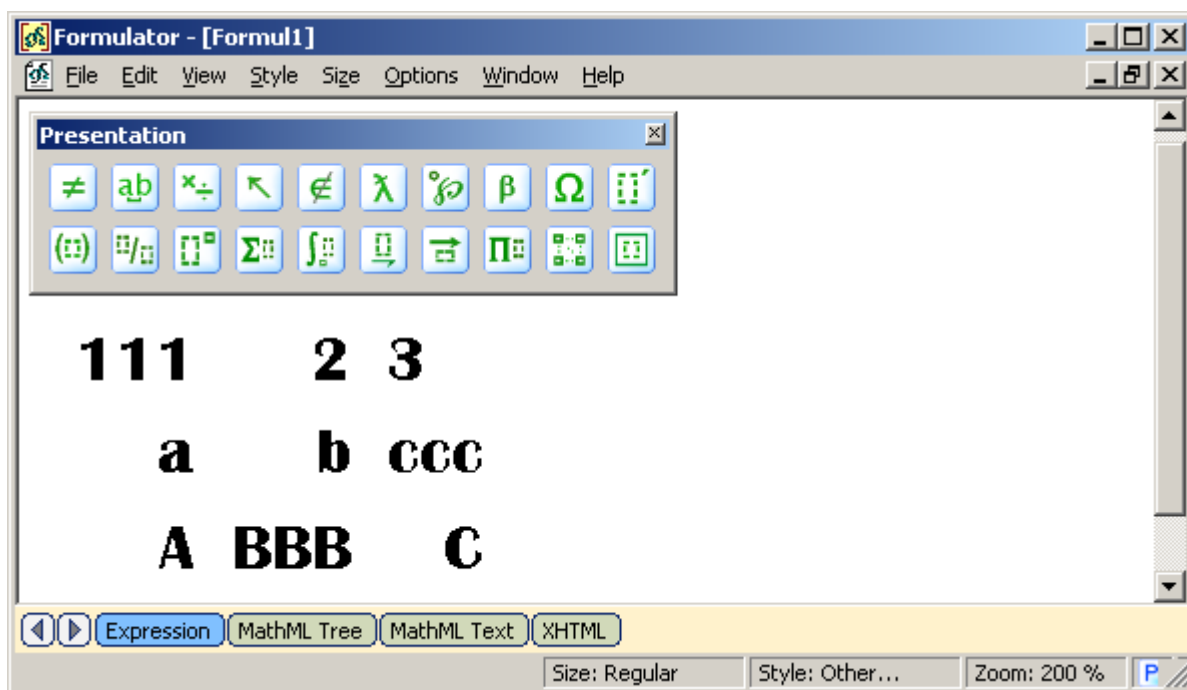
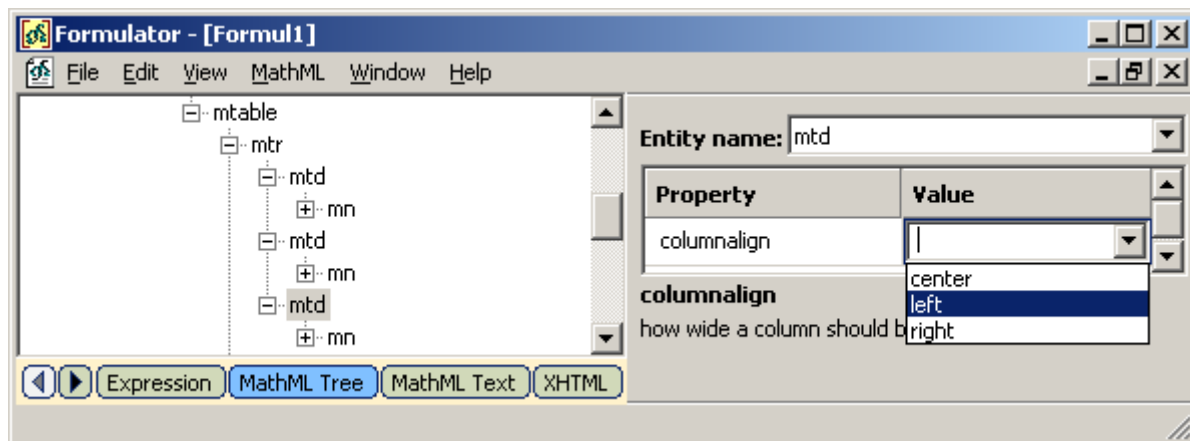
4. Select the “columnalign” attribute in the list; then place cursor to the “Value” column of the “Property-Value” pane and use the drop list to get the list of all predefined values which are proper for the “columnalign” attribute. Change the value of the attribute to the value of “right”.



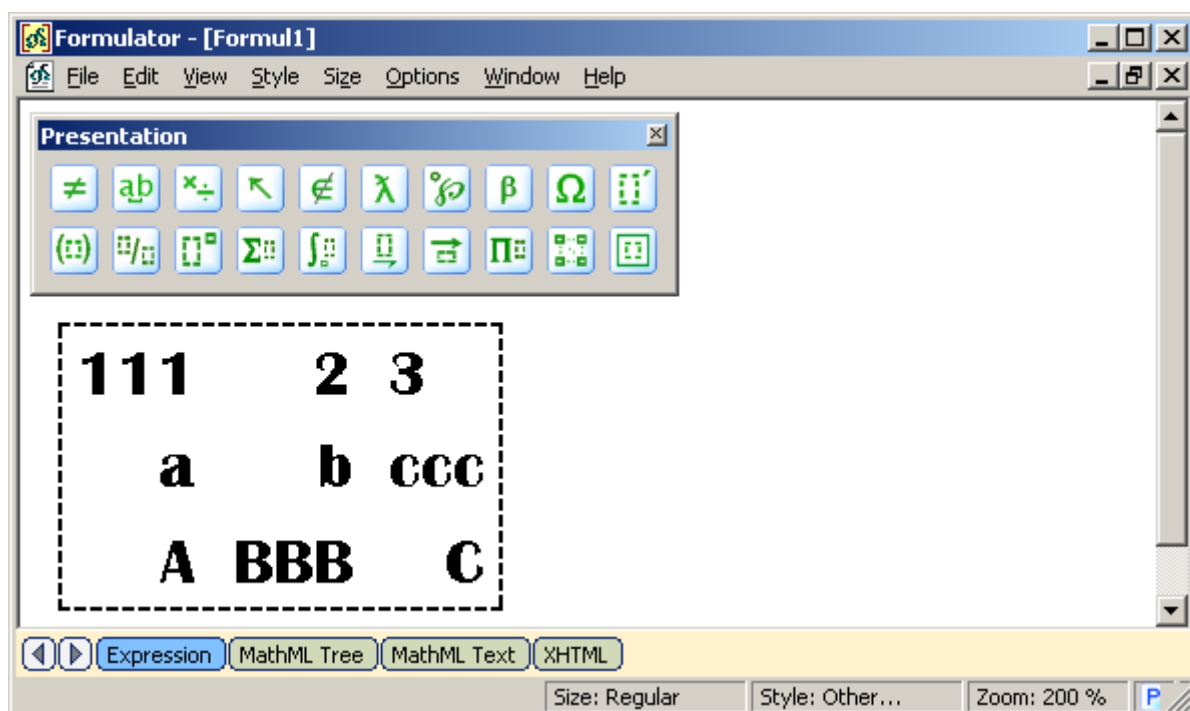
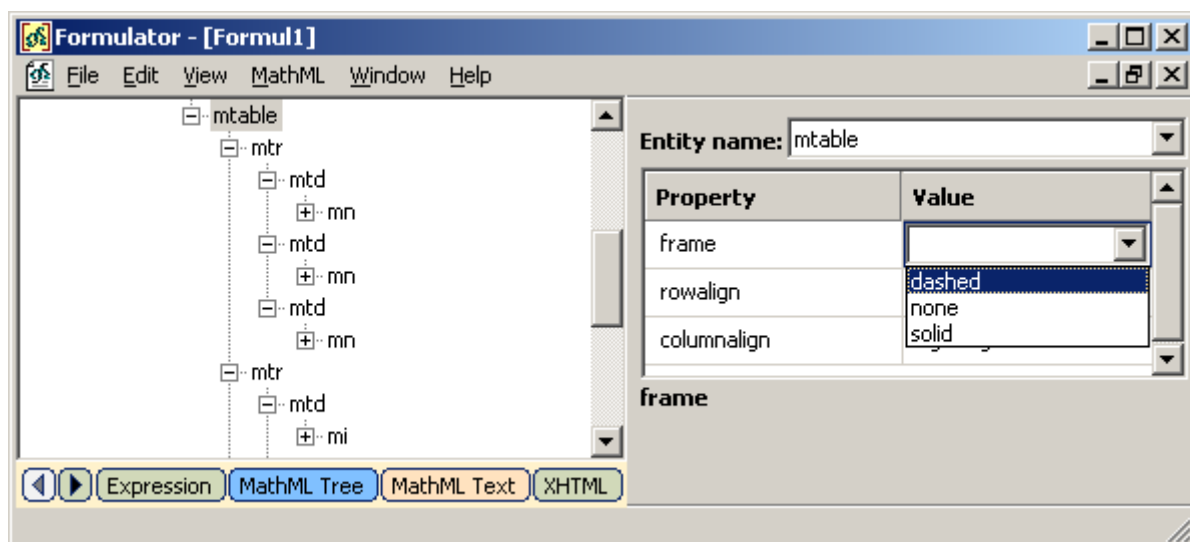
5. See how alignment of the whole table is changed.

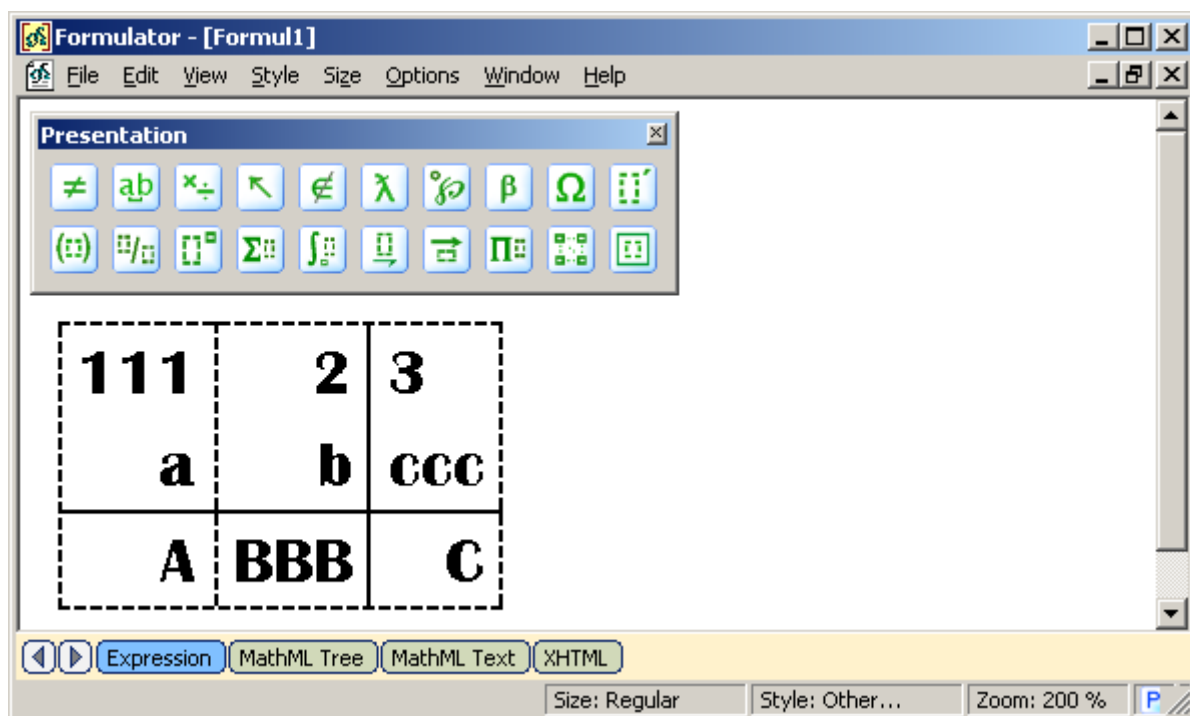
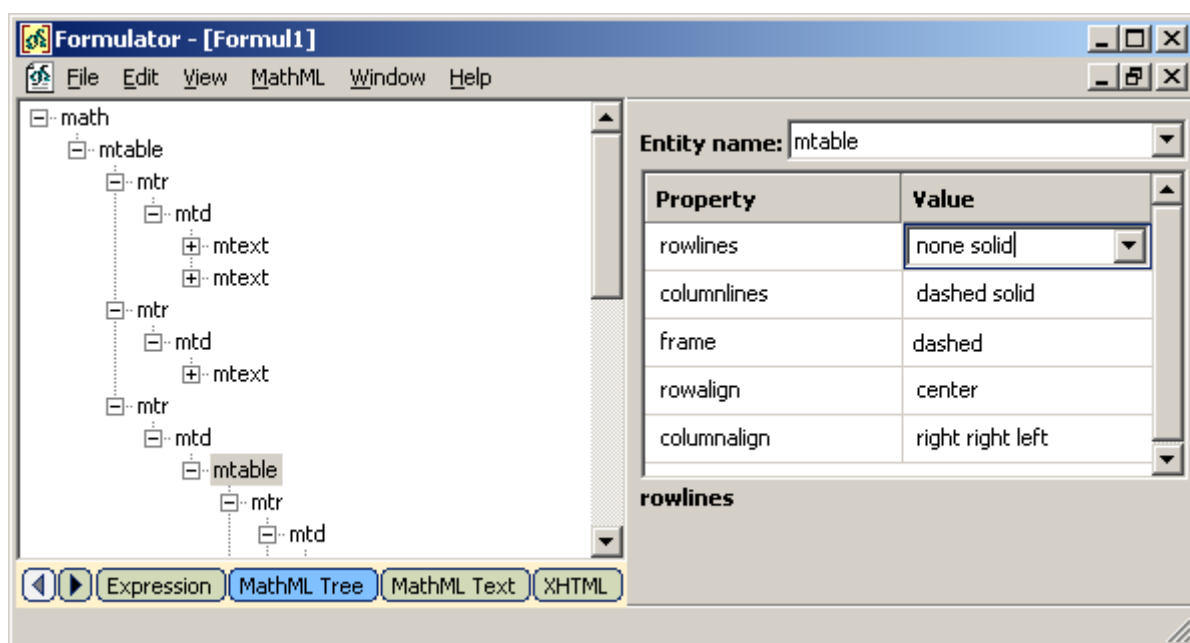


6. Using similar operations change alignment individually for the cell (1, 3) of the table.



7. Switch to “MathML Tree” to change the frame of the table; switch back to see the results.

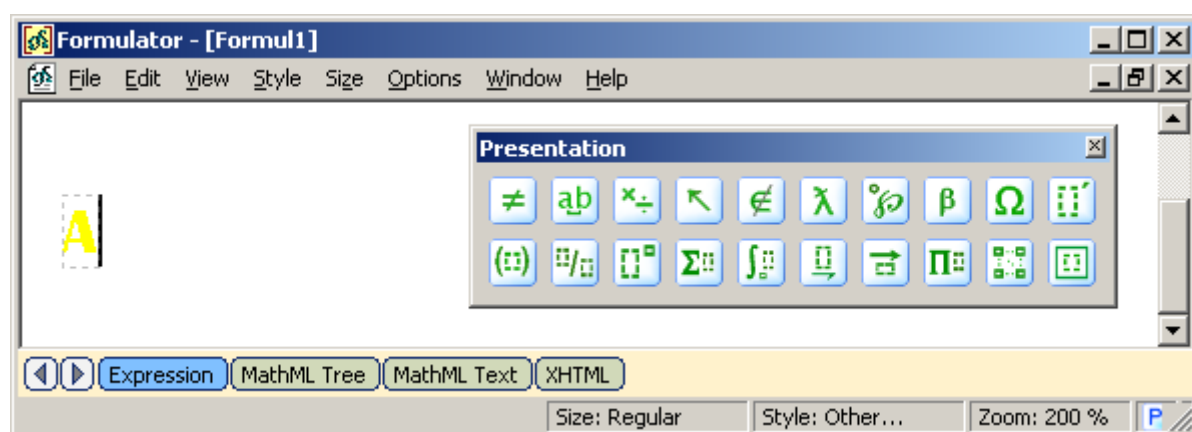
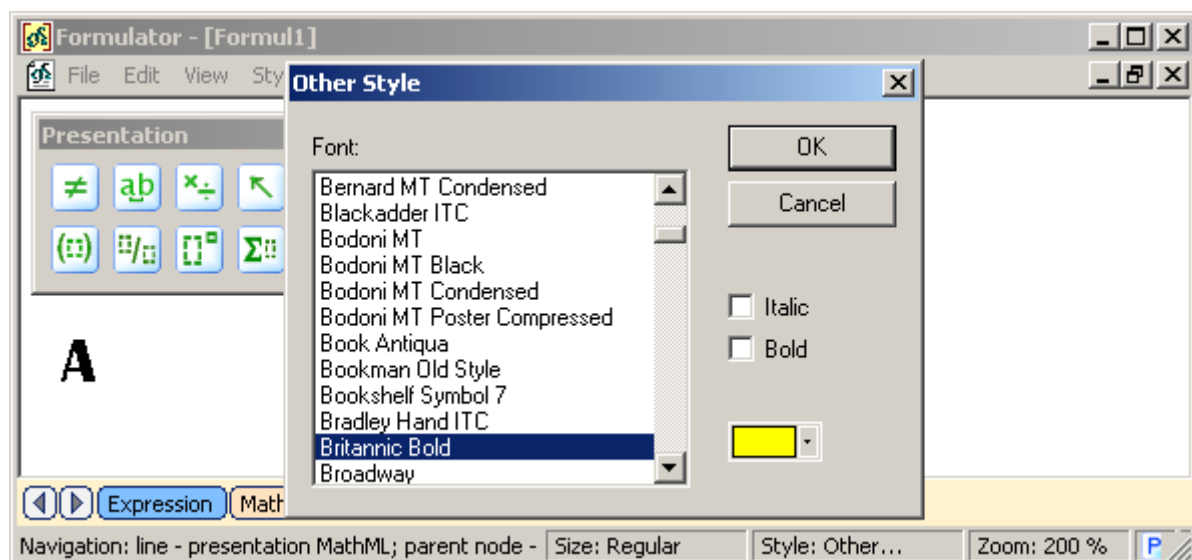




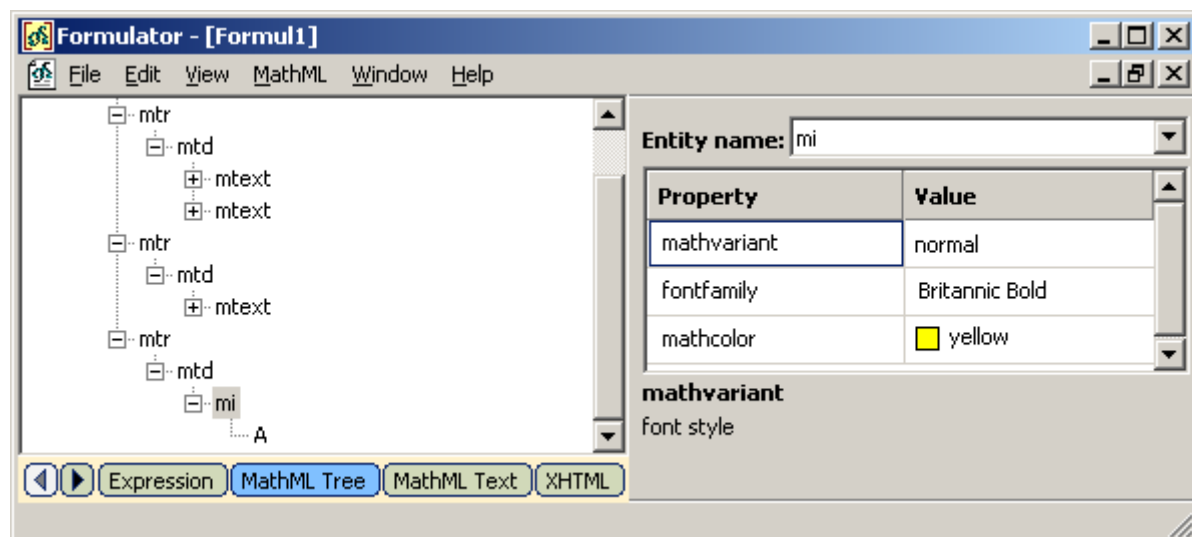
Style Change

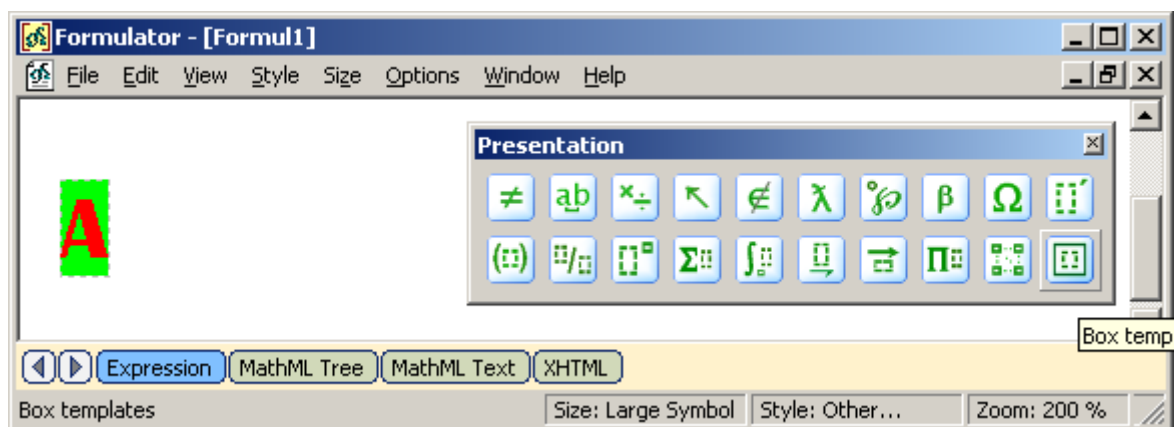
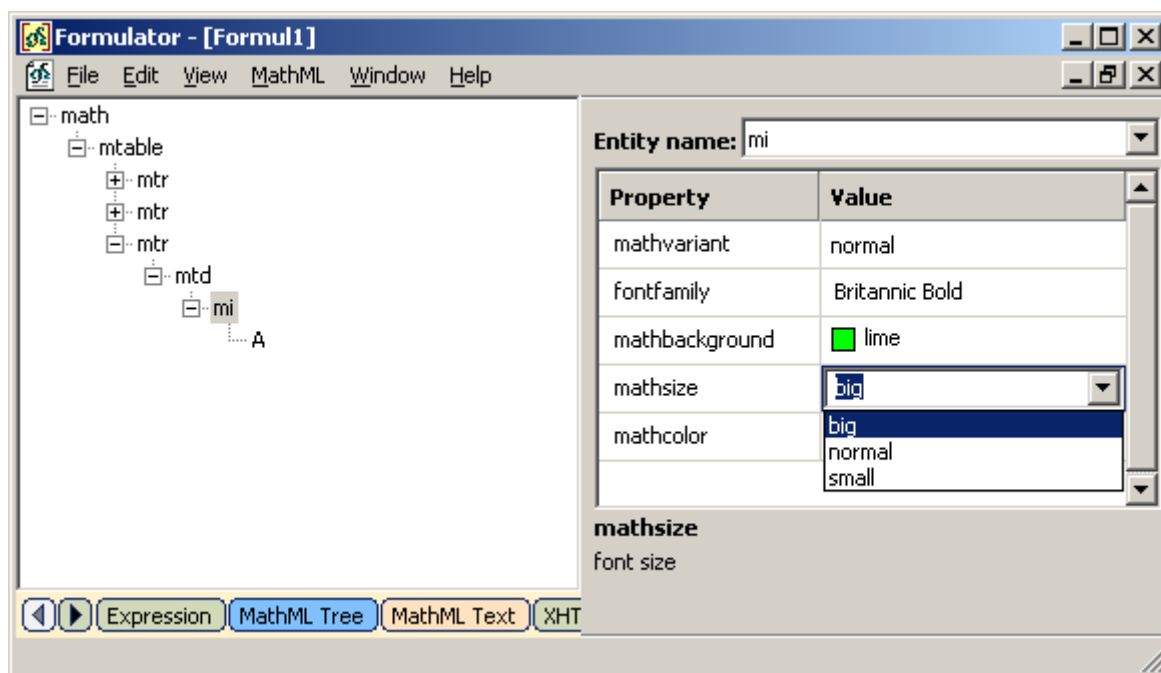
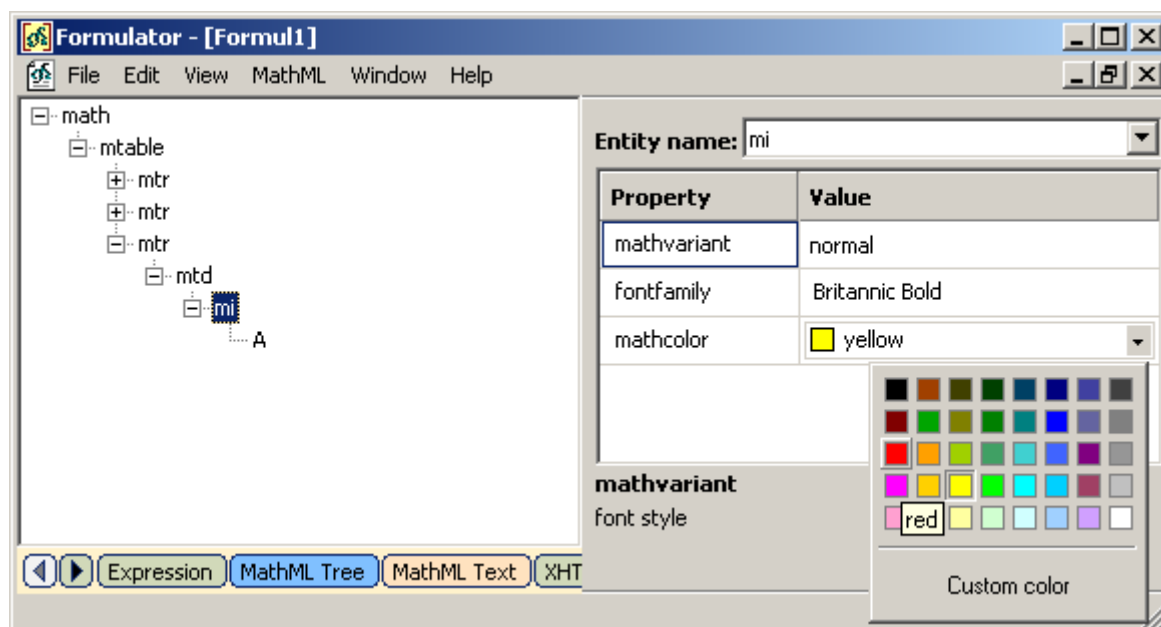
There are two ways to deal with “mstyle” element and style attributes in Formulator MathML Weaver.

The first way is implicitly changing of style of formula elements by using menu Style and Size.



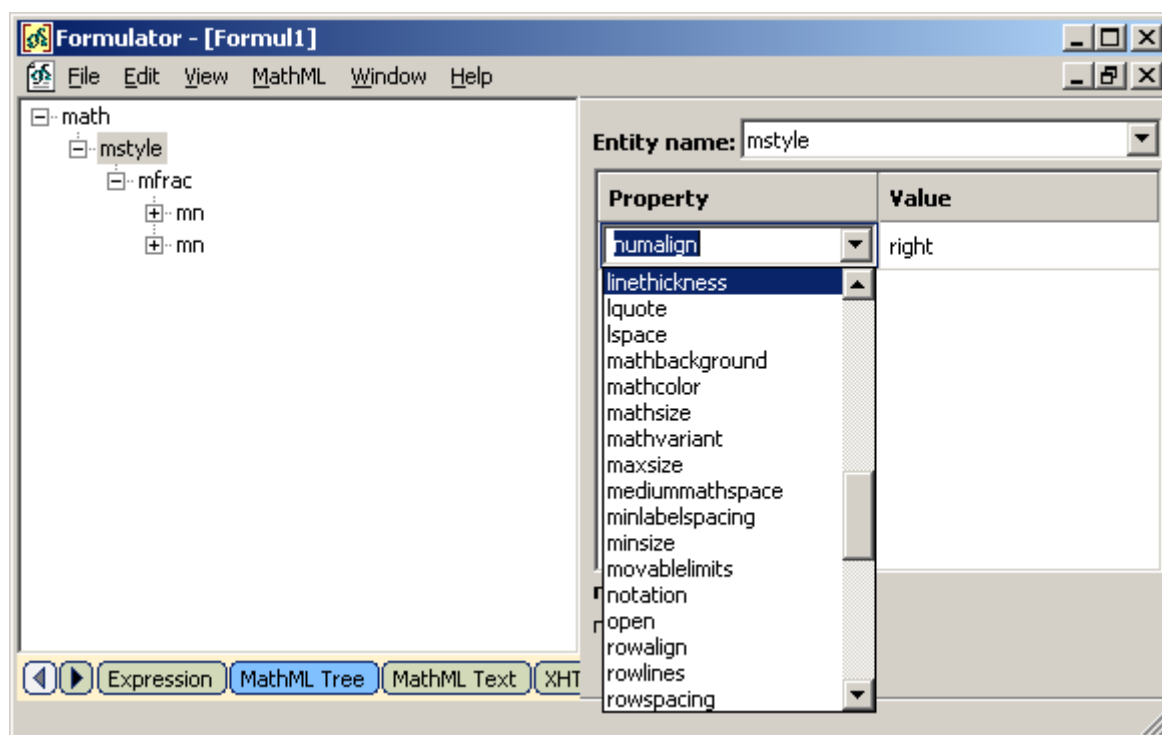
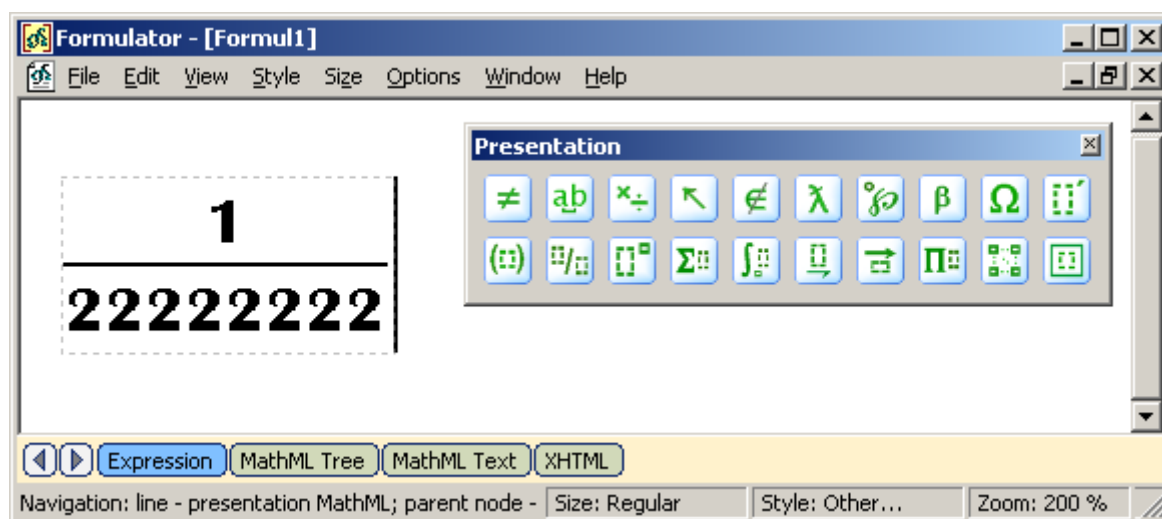
Then we can switch to the “MathML Tree” page and proceed with editing via WYSIWYG-style actions.

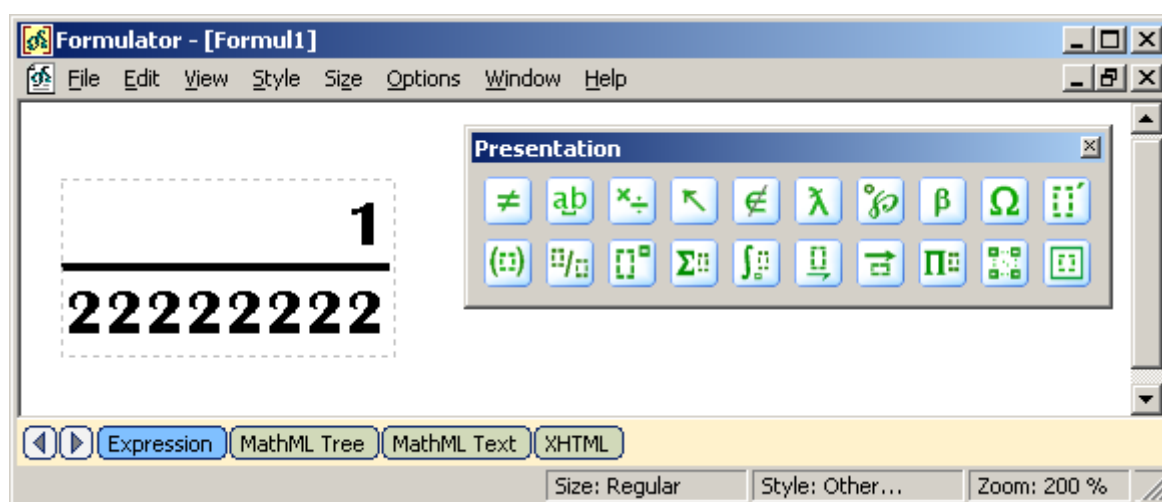
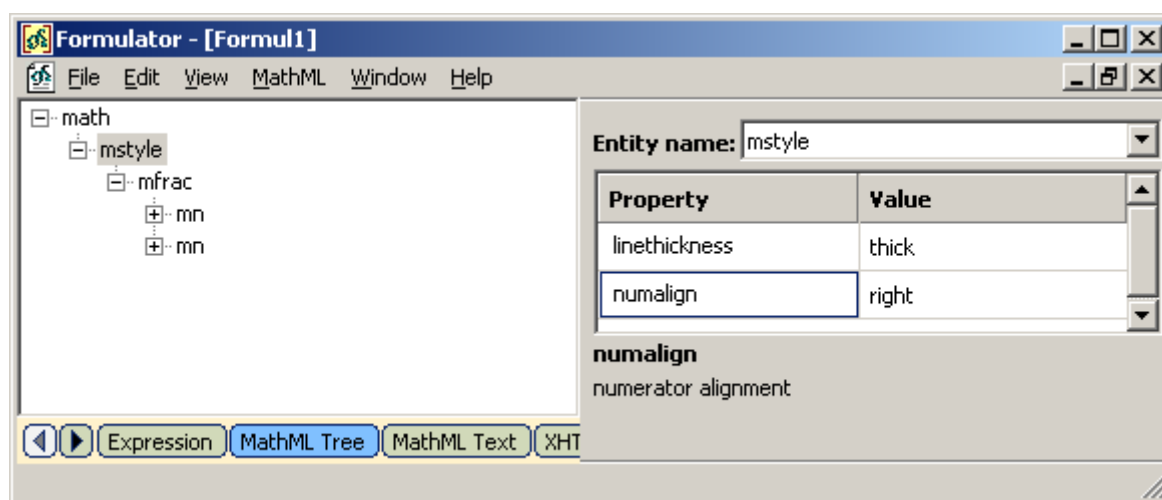




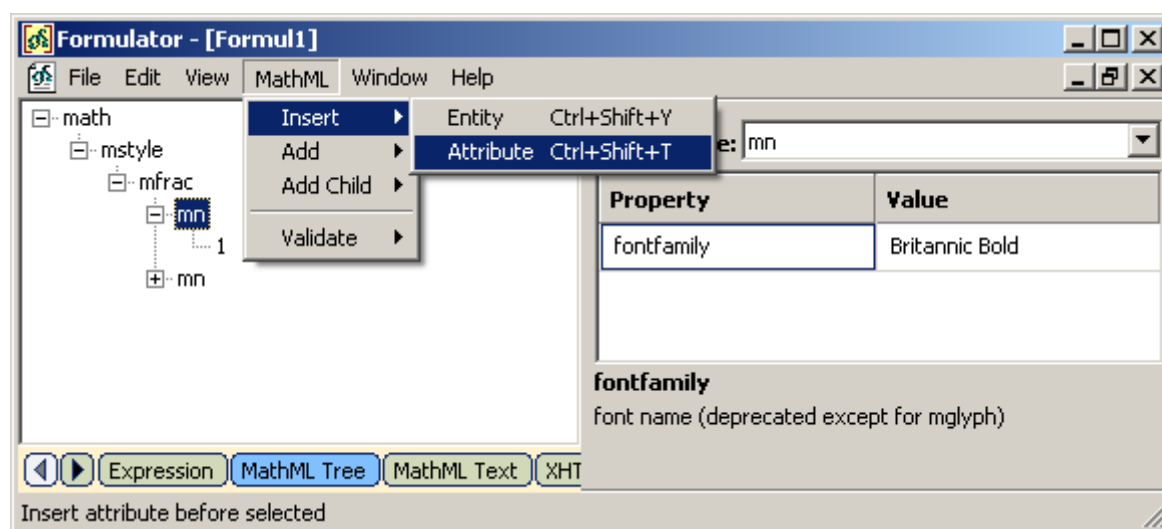
The second way of style changing is to insert the “mstyle” element on the “MathML Tree” page and to edit it using operations with nodes of the tree. Note that the “mstyle” element is inserted as an empty invisible “mrow” element that carries its own set of attributes.

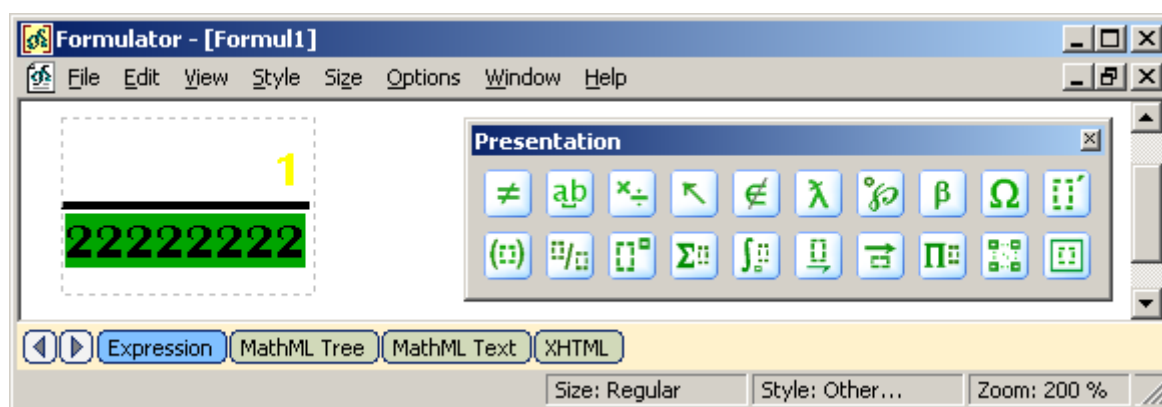
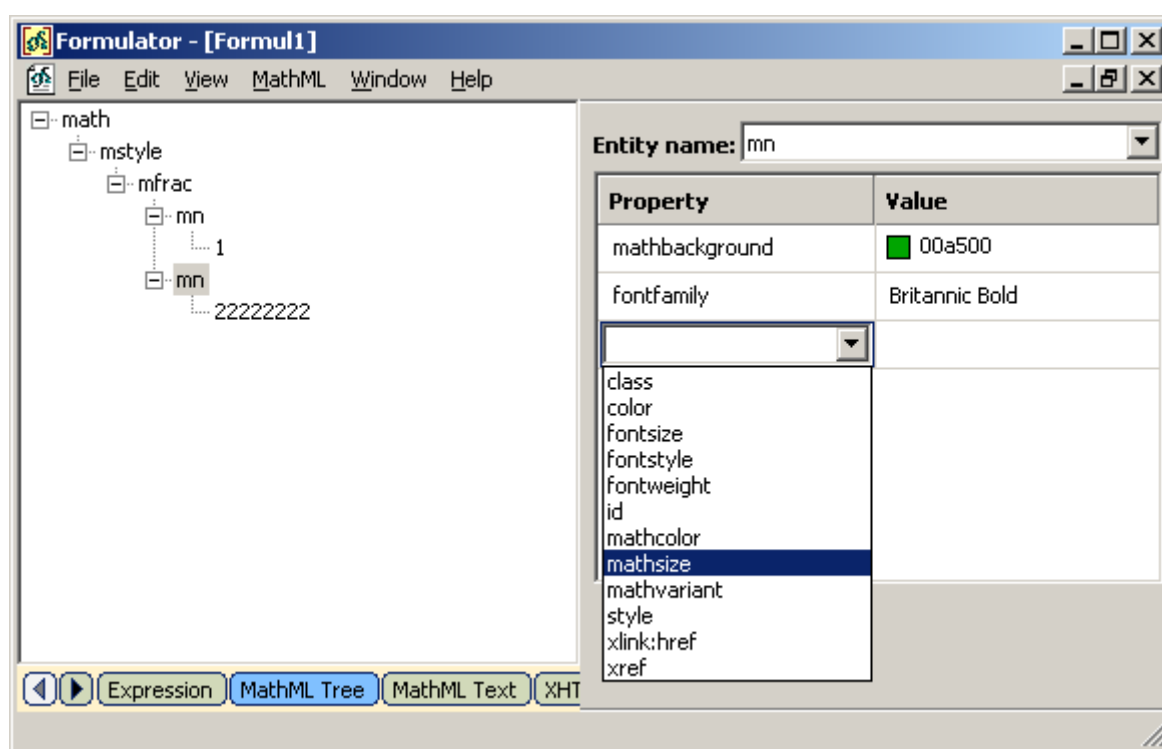
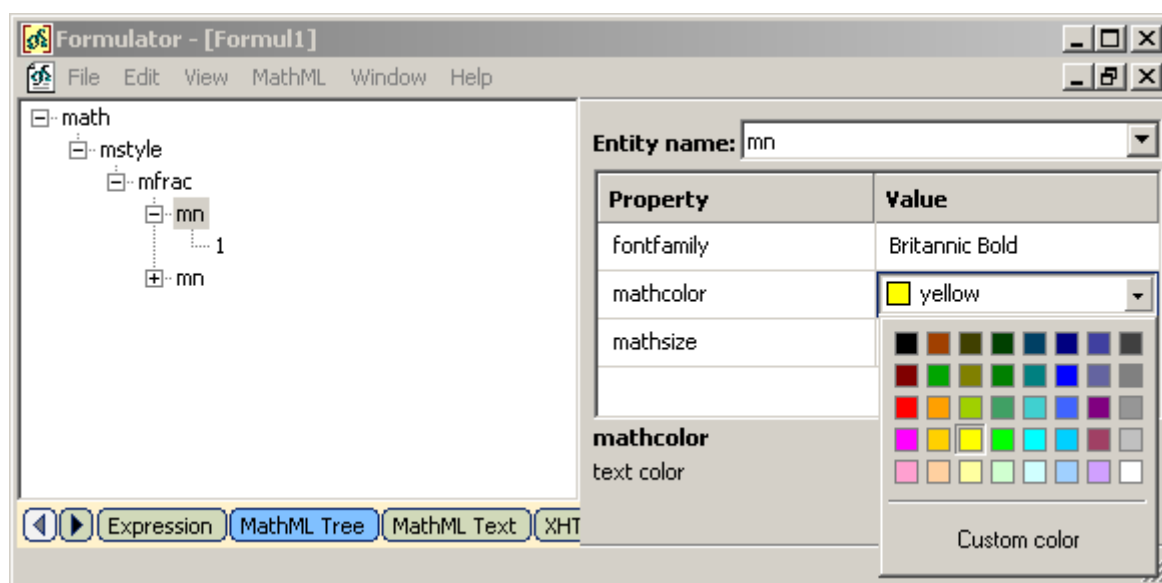
1. The next example shows how to change alignment of the fraction elements using attributes of the “mstyle” element.





2. The next example shows how to change appearance of token elements by editing attributes, related to a font.

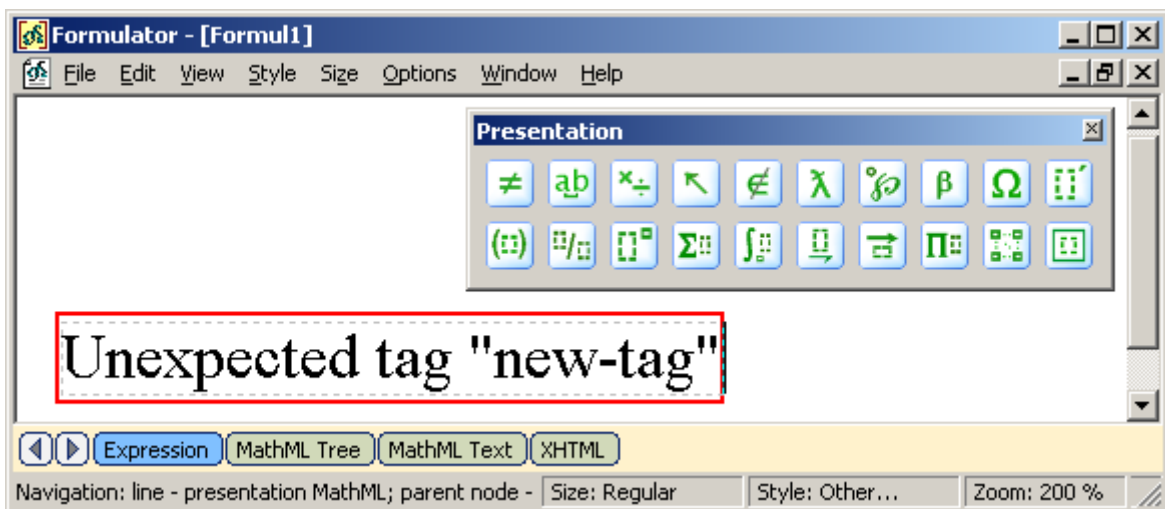
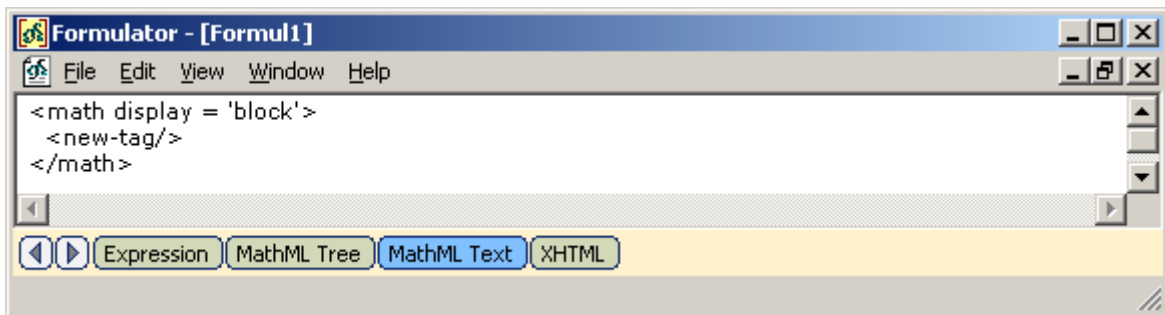




Error Message

A user can enter a new tag name on the “MathML Text” page, since it is just a text editor addition to MathML Weaver. This case should be treated as a user error, because the editor don’t know how to deal with unknown elements.

In order to get user know about errors Formulator MathML Weaver uses the “merror” element. In the next example a user enters unknown tag name “new-tag” and gets a message about error in MathML 2.0 notation.



Tutorial: MathML Content Markup with Formulator

Presentation elements describe visual two-dimensional hierarchical forms and thus give more or less precise instructions how to render and how edit mathematical constructs. Content Markup closely follows the semantic structure of the mathematical objects. This is quite the challenging approach to coding mathematics for such a WYSIWYG-style mathematical editor, as Formulator 3.9 MathML Weaver.

As an answer to this problem MathML Weaver implements a technology of internal wrapping of Content Markup expressions in Presentation Markup nodes. The main tool for this is an empty frame element that encodes a new hierarchical relations inside of Content Markup expression and carries all the specific information about final names and values of tags and attributes. This allows to edit Content Markup expressions in WYSIWYG-style, but at the same to establish explicit connections between mathematical structures and their mathematical meanings.

Please note that, as a rule, a user should not care about wrapping of Content Markup expressions in Presentation Markup nodes. From the point of view of end-user, there is just a way in MathML Weaver to insert and to edit visually a semantic kind of MathML notation. The only reason why we raise this question in the manual is that bearing in mind of such an peculiarity of the Content Markup editing implementation can be useful for understanding of the behavior of MathML Weaver in some complicated cases.

General Principles of Editing and Navigation

MathML Weaver proposes two ways to create Content Markup elements. The first is most general, since it supposes to make use of mathematical toolbars:

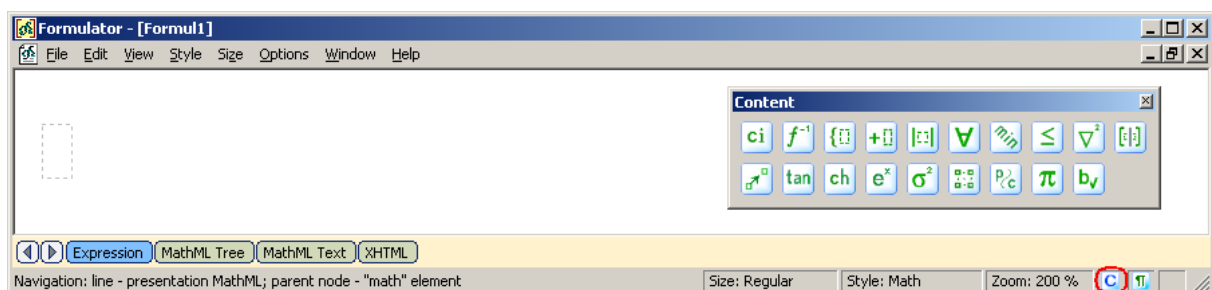
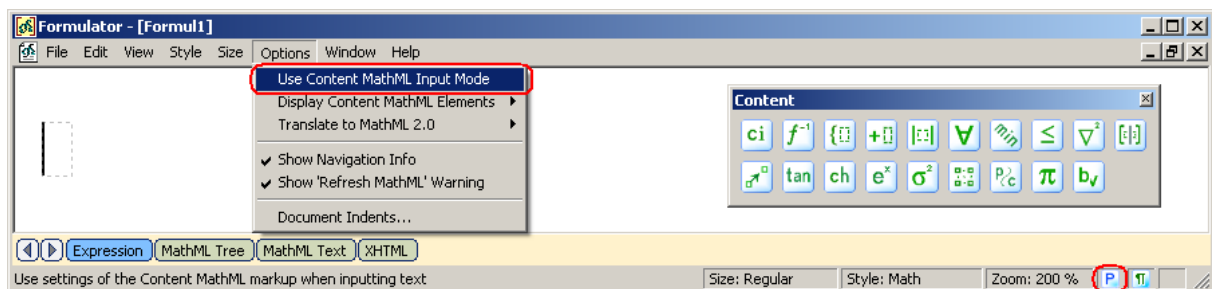


A group of Content mathematical templates comprises:

- token elements
- basic content elements
- piecewise declaration templates
- arithmetic operators
- algebra operators
- logic operators
- maximum and minimum templates
- relations templates
- calculus and vector calculus templates
- theory of sets templates
- sequences and series templates
- common trigonometric functions

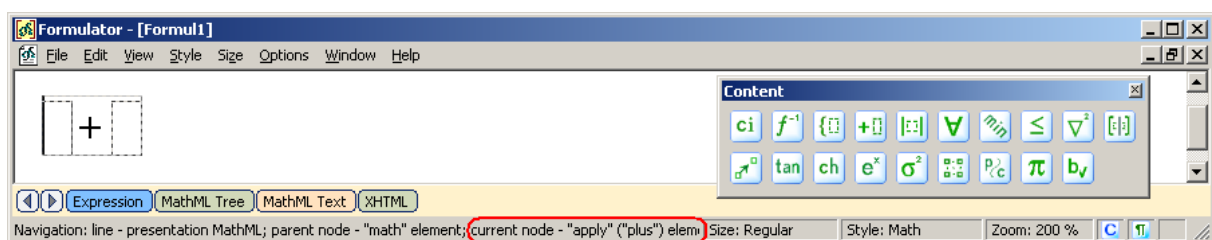
- common hyperbolic functions
- common exponential functions
- statistics templates
- linear algebra templates
- semantics templates
- constant and symbol elements
- qualifier elements templates

The second way to insert Content Markup elements is connected with an option to switch input between Content and Presentation MathML input modes (the last one is used by default):



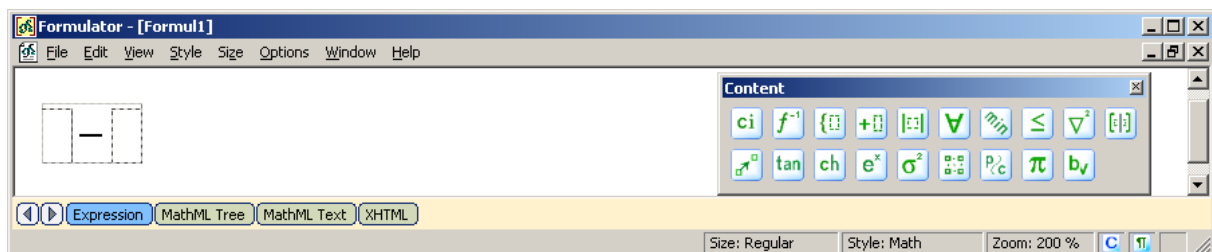
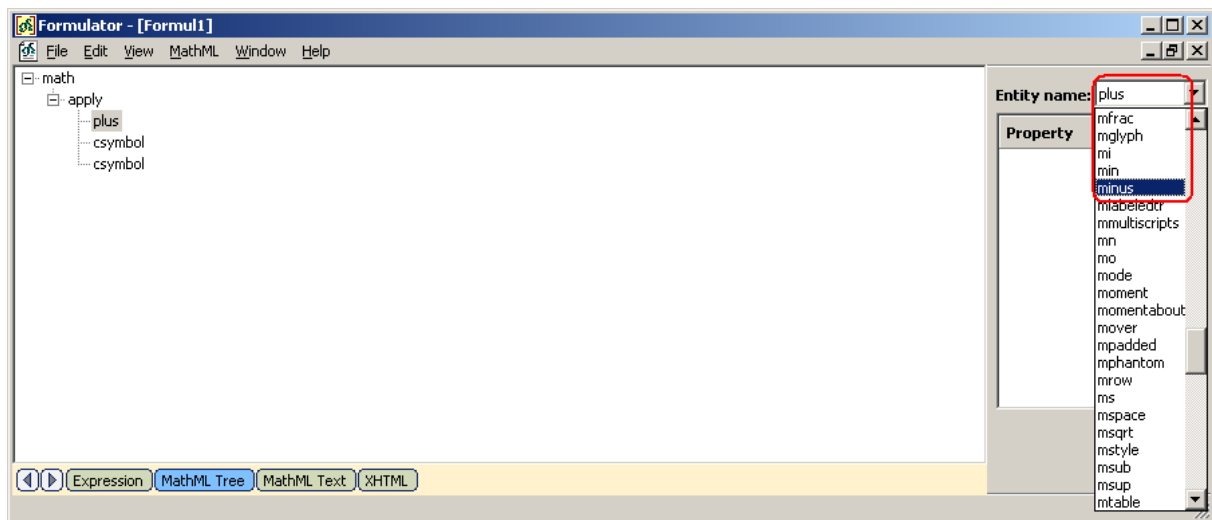
Selecting Content MathML Input Mode leads to inserting of Content MathML mathematical templates when a user presses a sign of the corresponding operation. E.g., pressing '+' in the Content MathML Input Mode inserts a mathematical template $\boxed{} + \boxed{}$ for the <apply> element with the operator element <plus/>. The next example shows how to work with this mode in more details.

1. After switching to the Content MathML Input Mode type '+' sign (make sure that Status Icon in the right bottom corner of MathML Weaver indicates). We now see the <apply> element inserted. Then we can use the "Show Nesting" command and navigation arrows to investigate how this Content Markup element is composed.



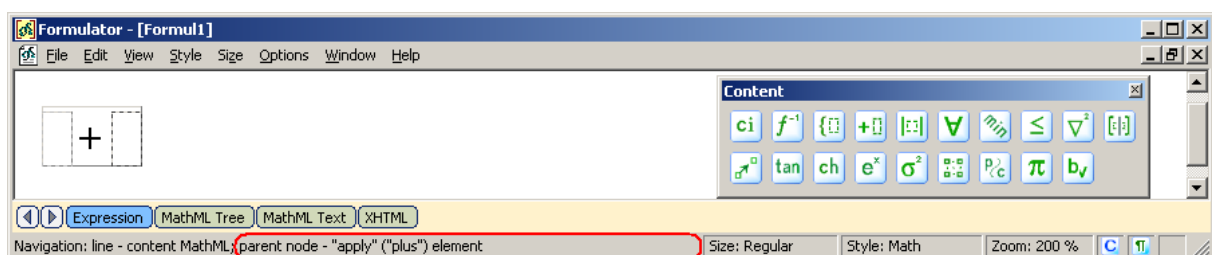
A grey rectangle around the $+$ form and tips in the status bar suggests that in the document there is one frame node, marked as an “apply” element and internally encoded as an additional nesting “mrow” element.

After pressing the Right arrow we consequently get to the input slots of this “apply” form. ‘+’ sign cannot be edited on this stage, since it is an integral part of the expression. For the sake of flexibility the type of operation still can be easily changed using the “MathML Tree” page:

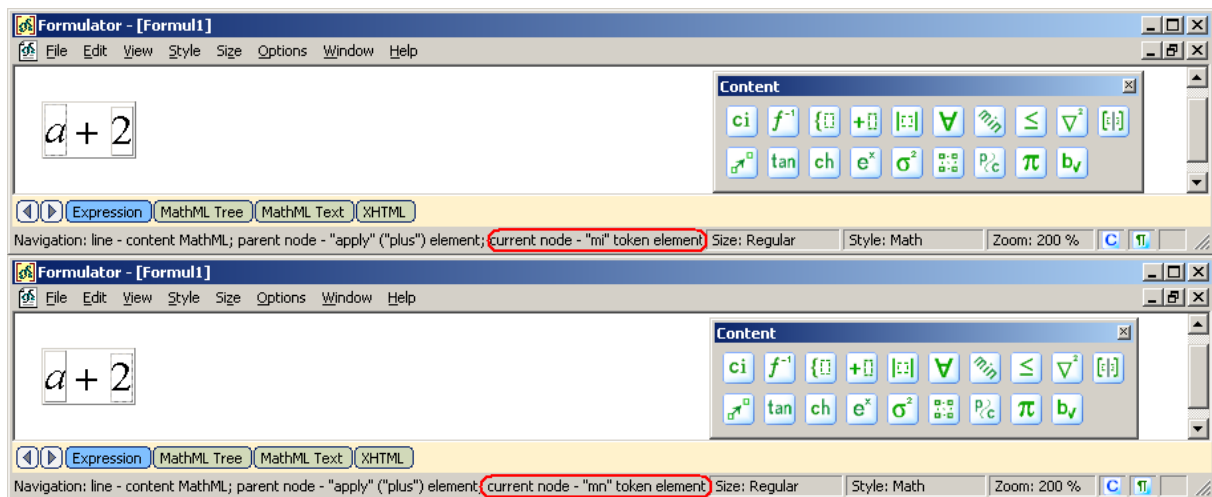


2. Now we can enter arguments of the just created “apply” element. The most important thing about input slots of this visual form is that their kinds can be automatically detected by MathML Weaver after a user types some text or utilizes Formulator’s mathematical toolbars.

Initially, these input slots don’t have any predefined kind (associated Content or Presentation Markup element). It can be easily discovered if we look on the navigation information bar; there is a record for the parent node, but the current node record is undefined yet.

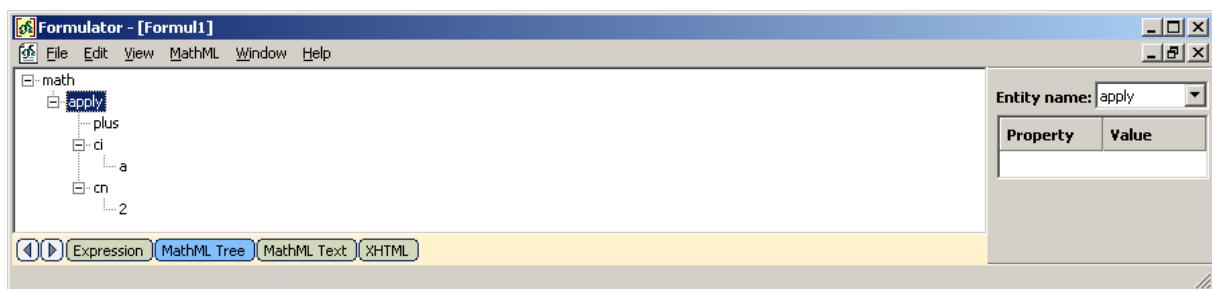


Type character 'a' in the first slot, press the Right arrow and type '2' in the second slot:

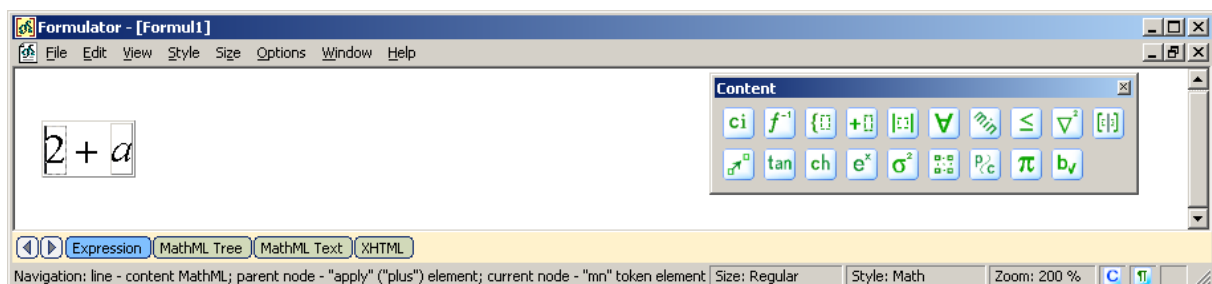


Note that on this stage 'a' and '2' are still marked as presentation elements ("mi", "mn"). They become Content elements only *after* they will be converted from the internal document representation to MathML 2.0 text.

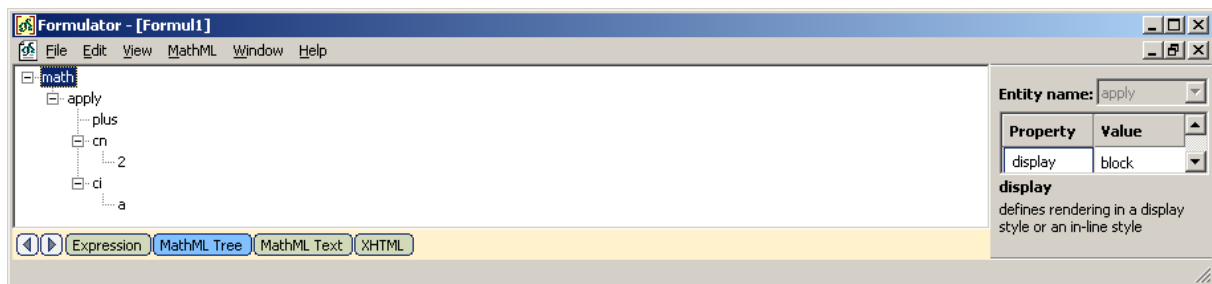
Switch to the "MathML Tree" page and make certain that the "apply" element is created correctly.



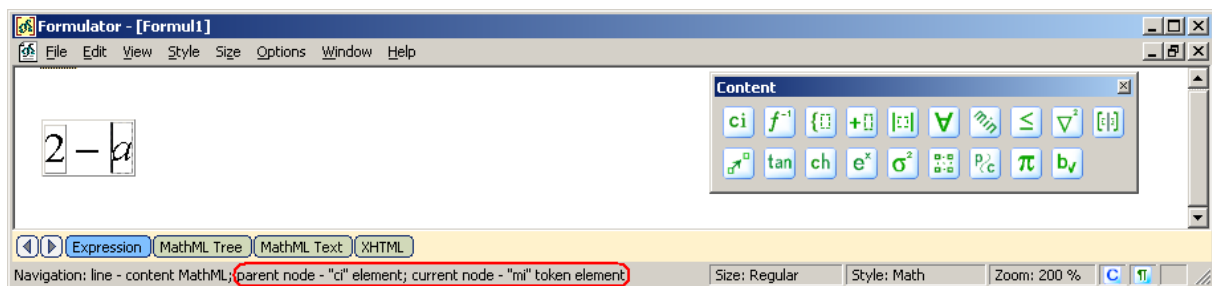
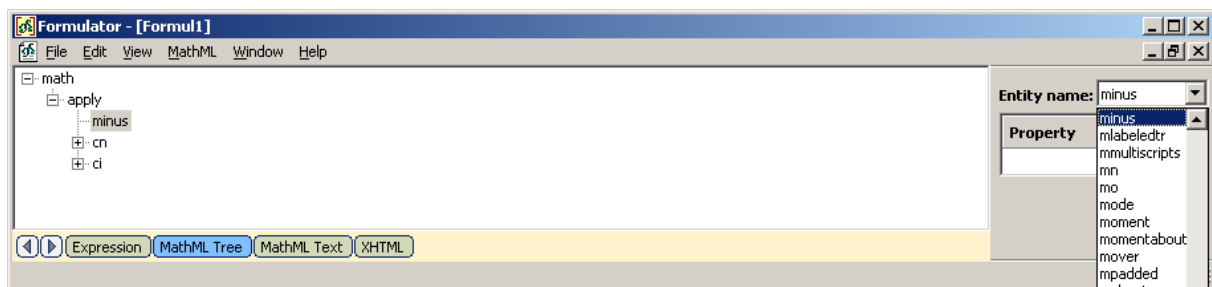
Now switch back to the "Expression" page and exchange 'a' and '2' in their input slots.



The “MathML Tree” page shows that input slots still work as automatically detected by their content areas, because the first slot now turns to be “cn” element and the second slot became the “ci” element.

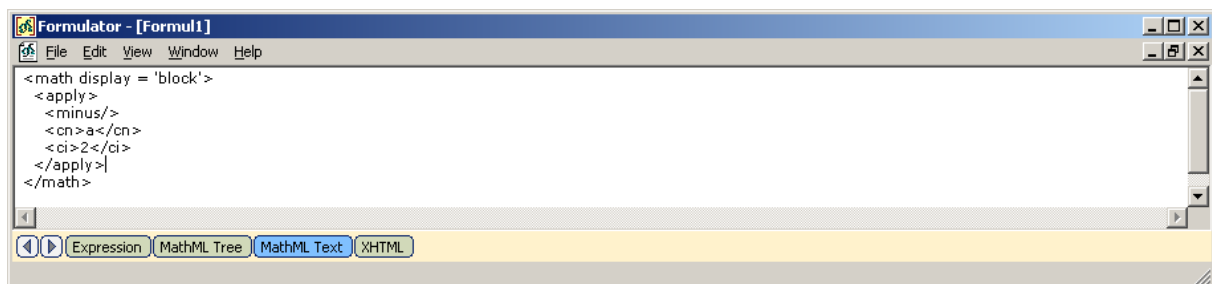
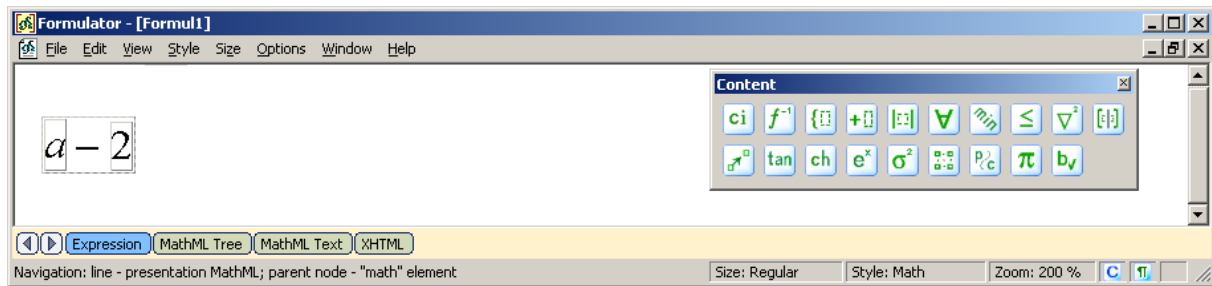


3. This smart behavior of the input slots of the “apply” form lasts for all the time of WYSIWYG-style editing, but stops when a user manually interferes in the editing by means of the “MathML Tree” or “MathML Text” page. To see this, switch to the “MathML Tree” and change type of the operator from <plus/> to <minus/>:



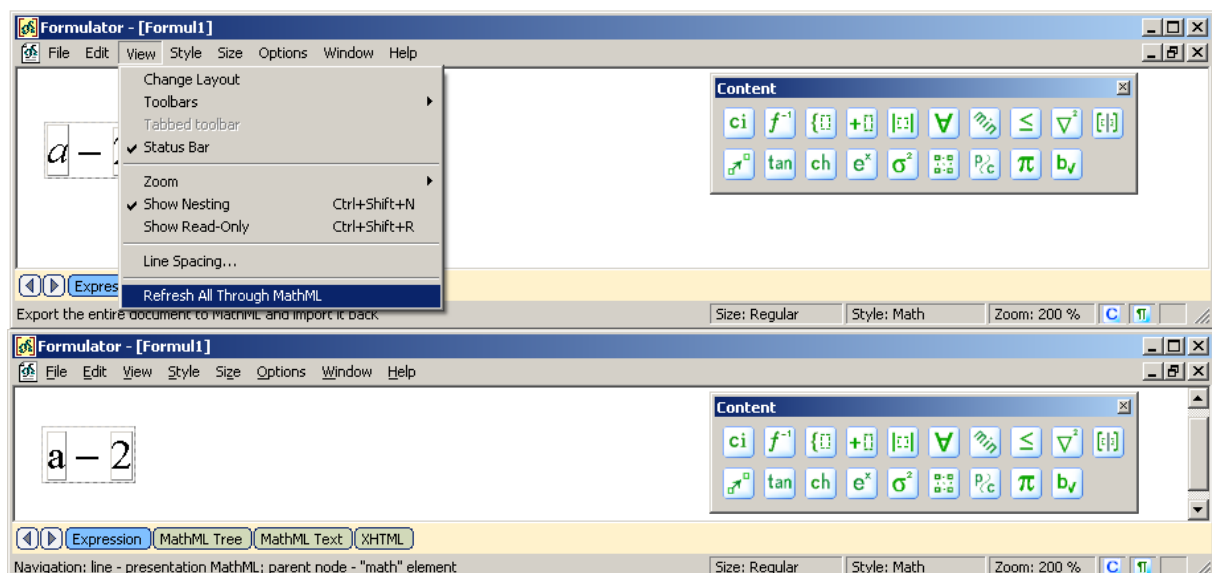
At first glance nothing changed, but take a look to the navigation information bar (red circled area). Now We have a rigid Content Markup structure, composed of the “apply” element with a leaf of the “ci” element. The internal presentation of the “ci” element is formed of the “mi” element, but it is of less importance, since in the MathML 2.0 text there will be no internal presentation remains.

To make sure that a new structure of the expression is fixed try to repeat our trick with exchanging 'a' and '2' in their input slots.



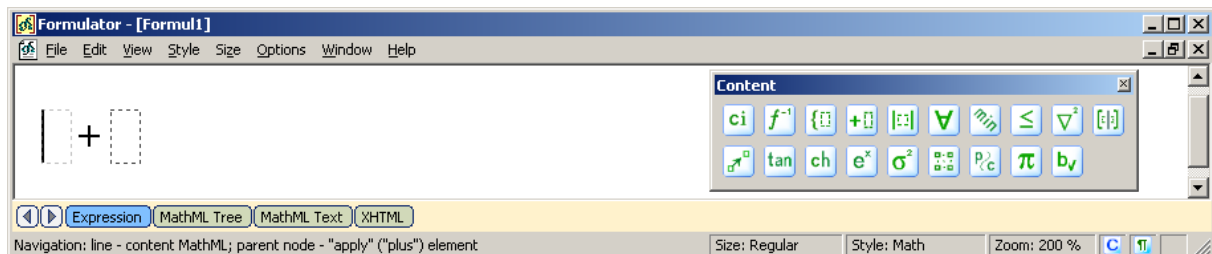
Now 'a' is considered as a number token element "cn" and '2' is considered as an identifier element "ci", because in the *rigid structured document* input slots have already known their kind of Content MathML elements and no automatic detection was provided.

In this example we show another typical feature of the way that MathML Weaver deals with Content Markup. Since 'a' is situated in the "cn" element, it should not be rendered as italic. But MathML Weaver doesn't interfere into a user's work in such a case, but proposes instead a short way to bring the formula's image to conformity with its semantics. Choose the "Refresh All Through MathML" command from the View menu and see how the entire look of the mathematical formula is changed.

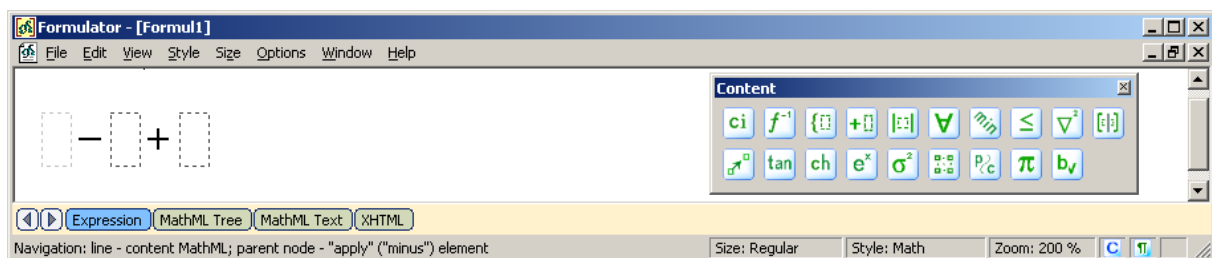


3. Input slots of the created “apply” form can be filled as well with more complicated expressions than tokens. See, for example, how to create mathematical expression with several arithmetical operations.

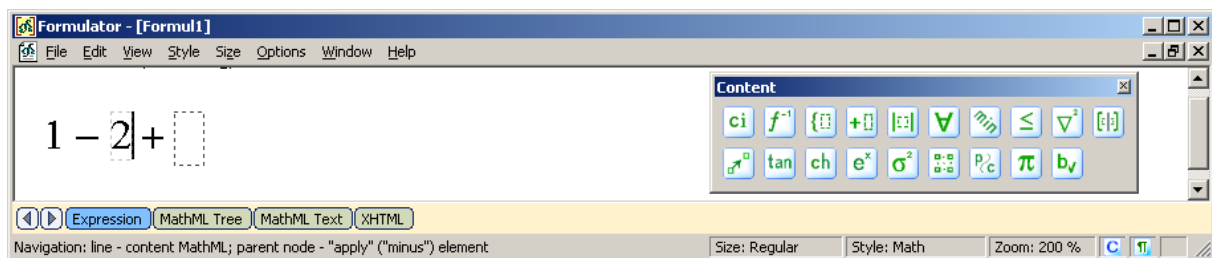
Press the ‘+’ sign.



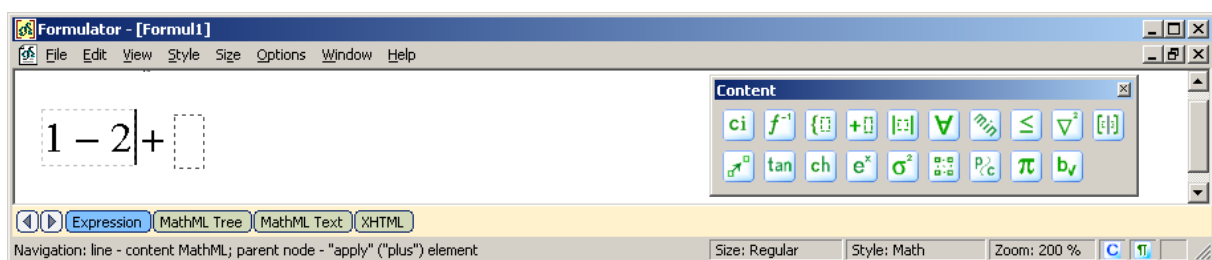
Press the ‘-’ sign.



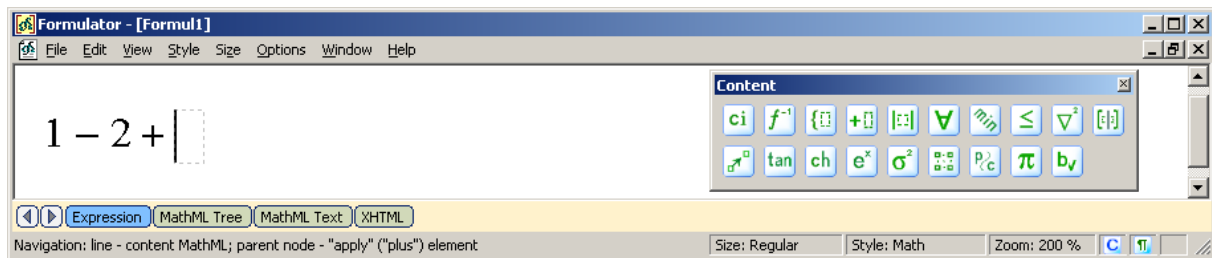
Press ‘1’, the Right arrow, ‘2’.



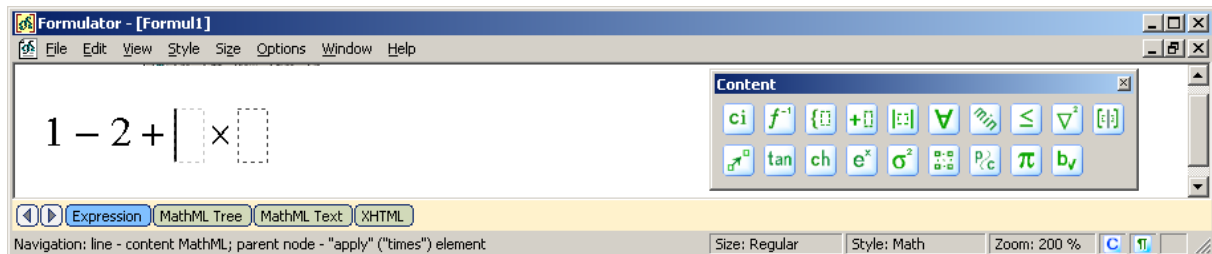
Press the Right arrow. Now be careful with editing actions, because dashed selection around 1 - 2 form indicates that we now on the hierarchical level of the “apply” element with <minus/> operator inside.



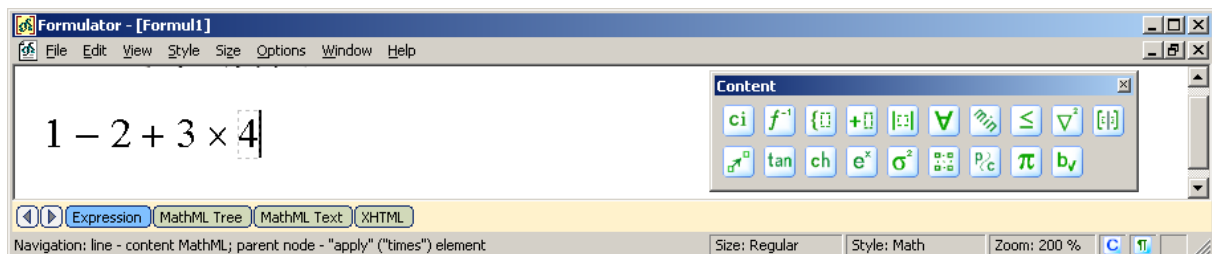
To continue editing the formula we need to place cursor to an empty input slot on the right side of the formula, so, just press the Right arrow once more.



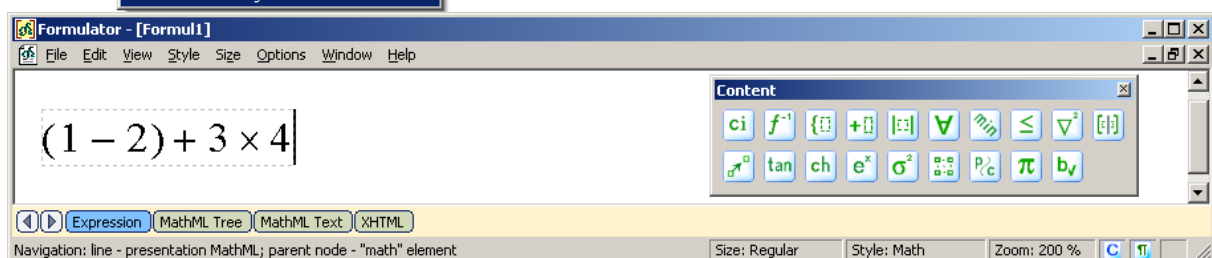
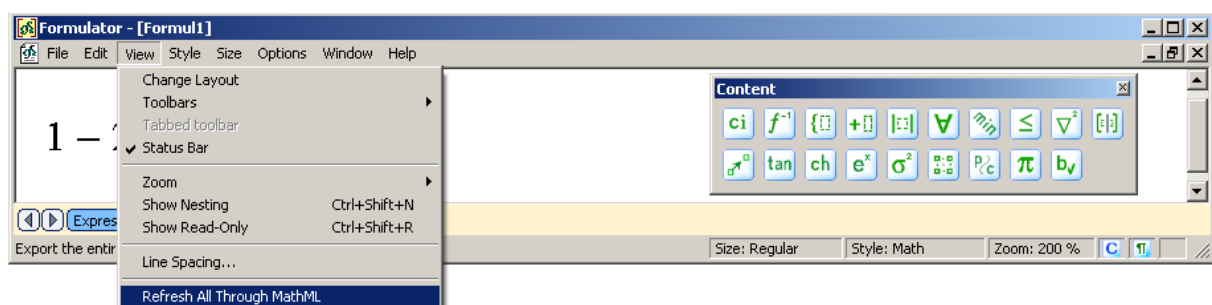
Press '*'.



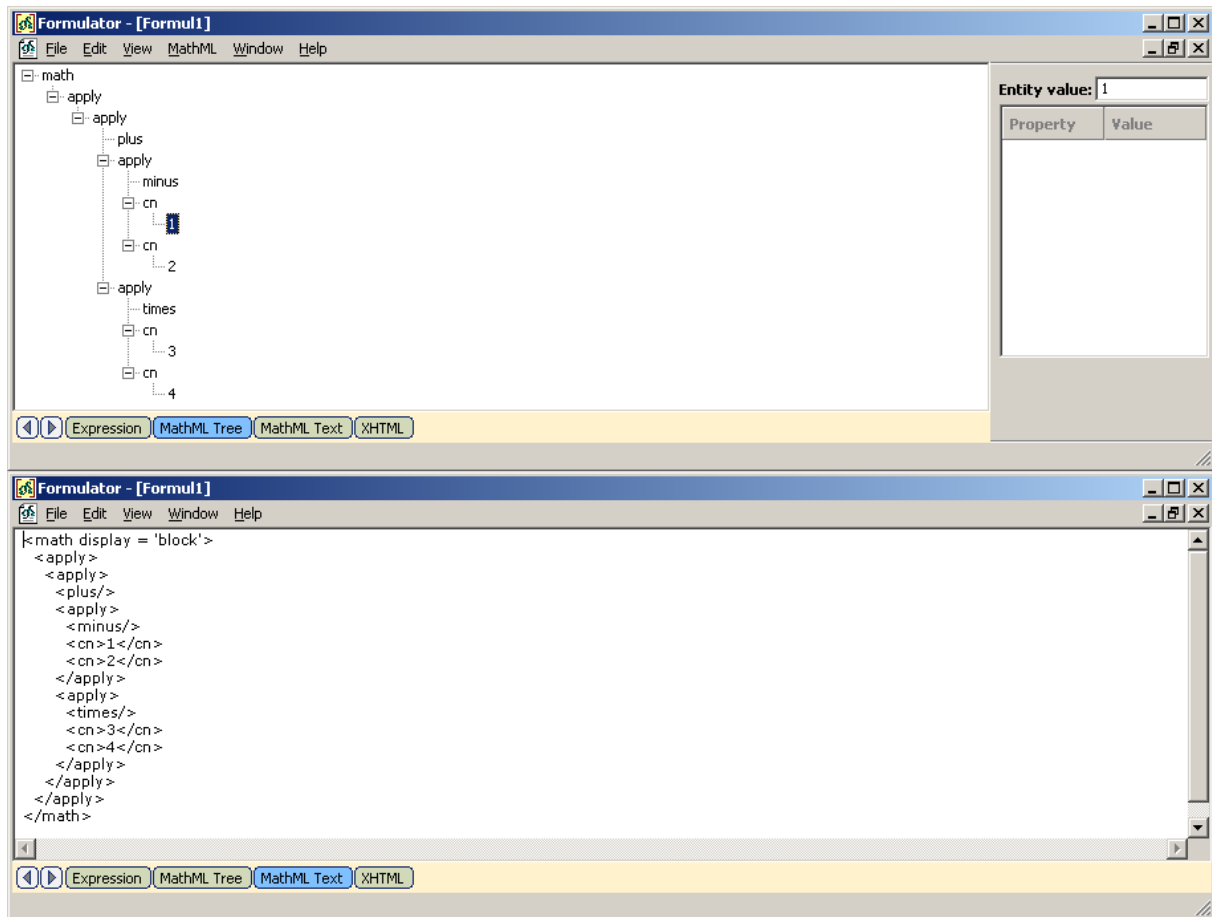
Press '3', the Right arrow, '4'.



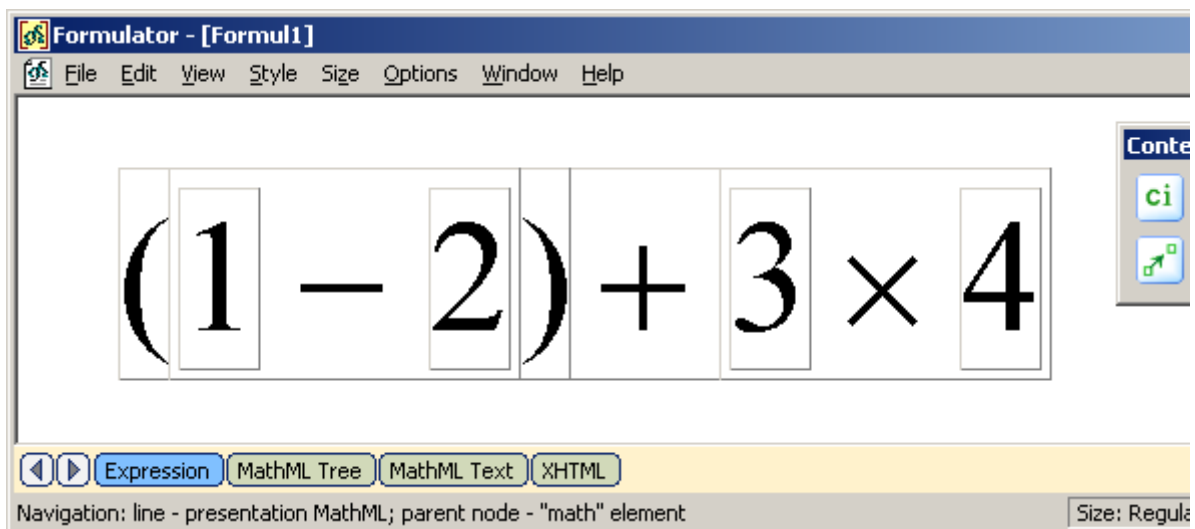
The needed formula is created but lacks of proper appearance, because MathML Weaver has no chance yet to apply its precedence rules and to calculate how to draw this Content Markup formula. Choose the “Refresh All Through MathML” command and see how the image of the mathematical formula is changed.



After we have created a mathematical formula we can check how it is encoded in MathML 2.0 format using the “MathML Tree” or “MathML Text” pages.

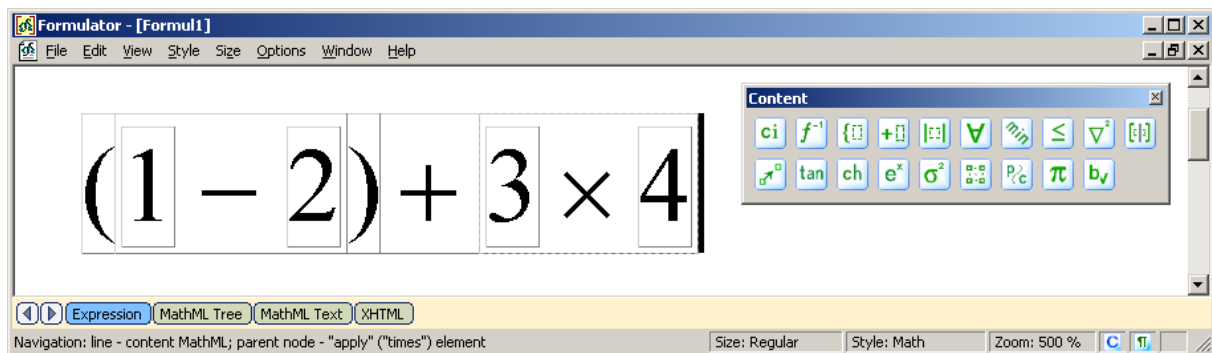


4. Navigation through the newly created formula is not absolutely evident until we think about hierarchical structure of the Content Markup expression. Choose the “Show Nesting” command (by pressing Ctrl+Shift+N or from the “View” menu); then choose a new greater scale factor by pressing Ctrl+5 or from the “Zoom” submenu of the “View” menu.

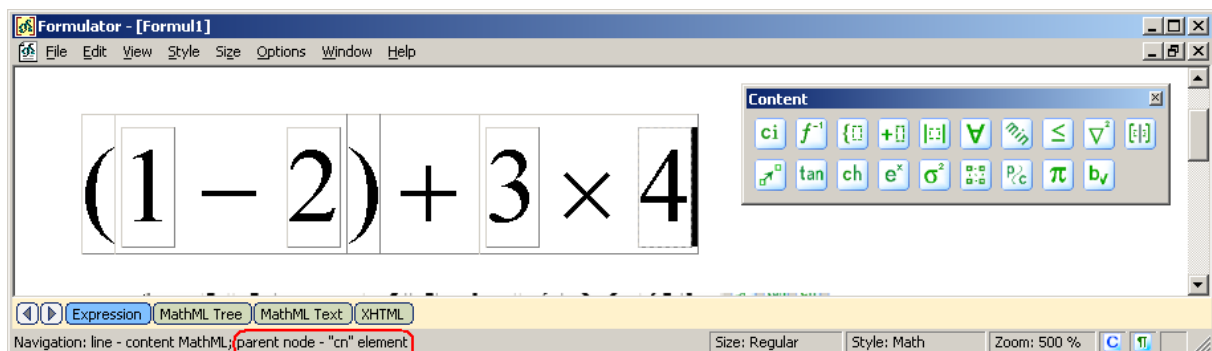


Now we can clearly see that the structure of the expression is not simple and we cannot expect the navigation to be as plain as in the Presentation Markup case. The higher level “apply” element has two child “apply” elements, and each of them in its turn has child leaves, encoded with token elements.

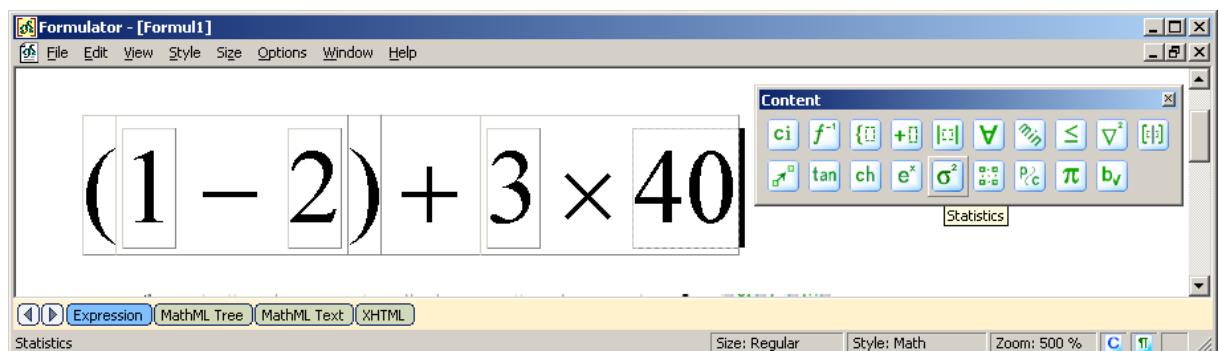
Press the mouse button on the right of the formula. This will place the cursor in the outermost right position available for editing.



Press the Left arrow. The cursor is now in a position for editing an input slot that currently has '4' in it. See the navigation information bar to make sure of this. Note that cursor height is changed also according to the current inner level of hierarchy.



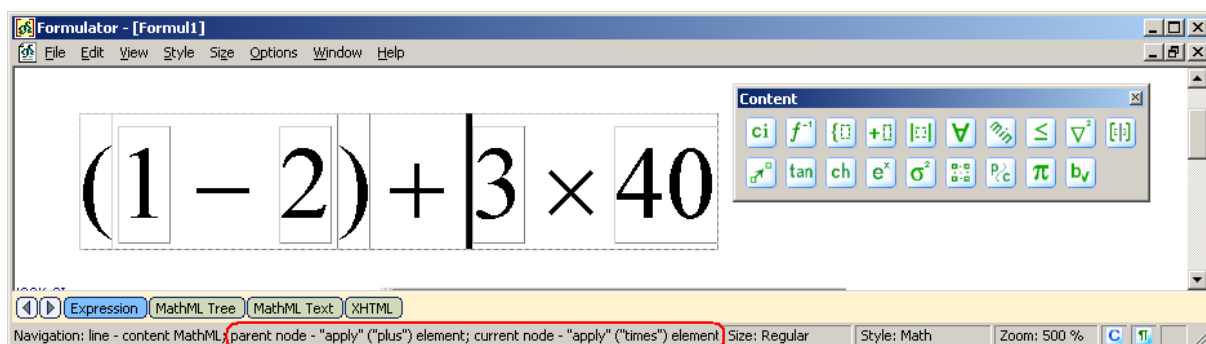
Press '0' to edit the last argument of the expression.



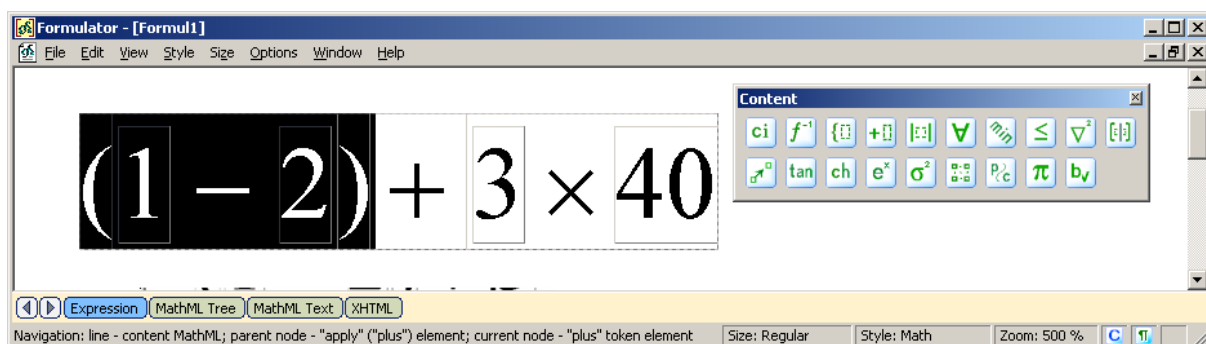
Press the Left arrow three times. We now again on the level of the `<times/>` “apply” element. In this position we could delete the last argument after pressing Delete button two times. The first pressure would select the entire input slot and mark it with black. The second pressure would confirm deletion.

In order to get back to the higher level of the `<plus/>` “apply” element we need now 5 times pressing of the Left arrow. By each pressing we go through different levels of the formula structure, where each level propose its own editing facilities. Imagine that we are navigating through the tree of Content Markup elements and this procedure becomes evident.

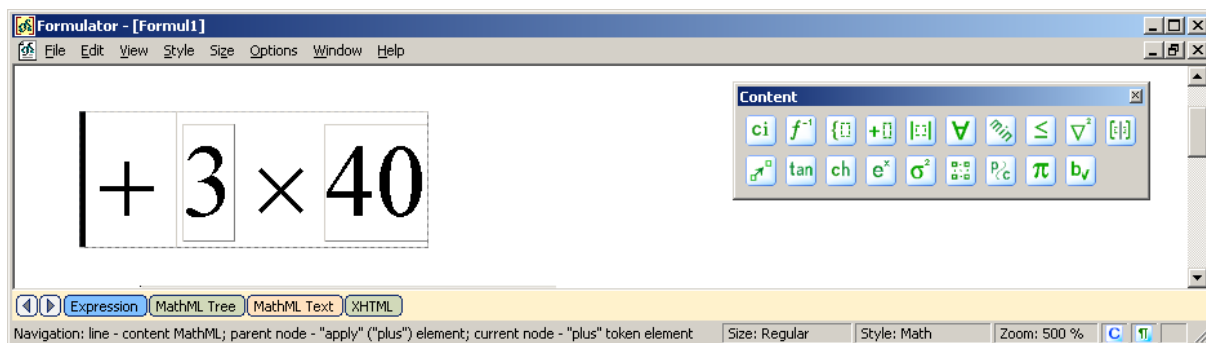
See how the navigation information bar helps to understand which level of the formula structure is currently available for editing.



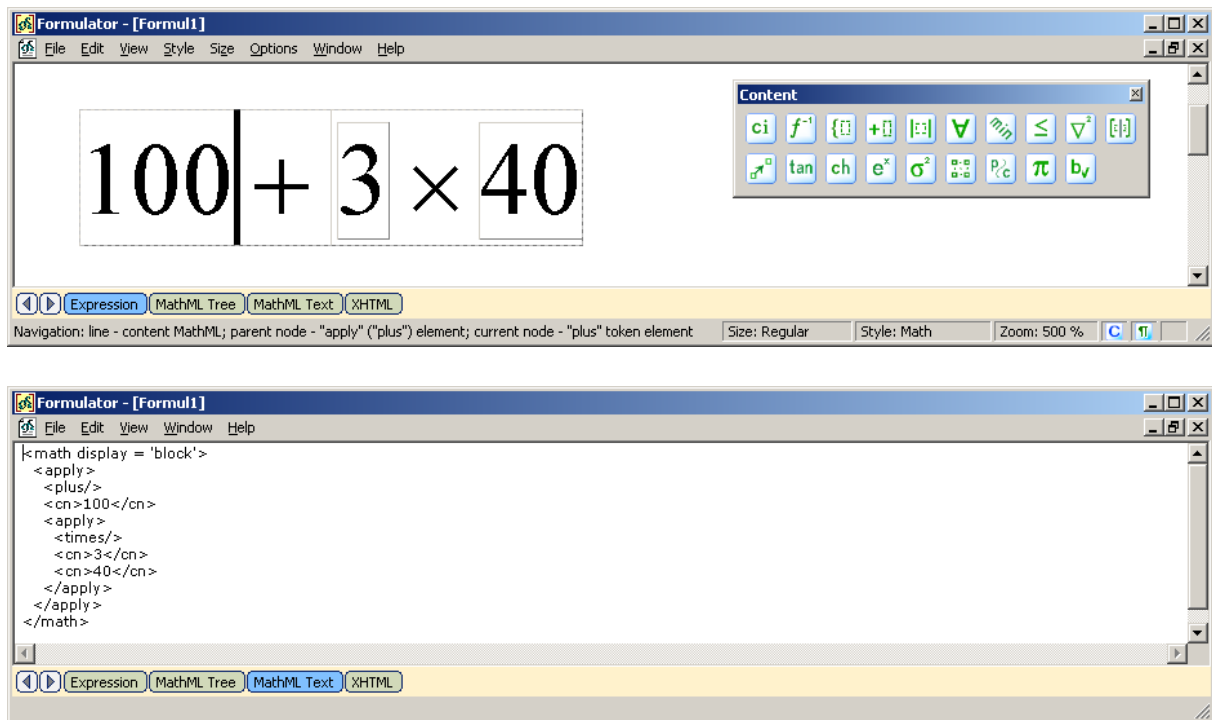
Press the Left arrow and the Backspace button.



$(1 - 2)$ form is marked for deletion. Press the Backspace button once more time.



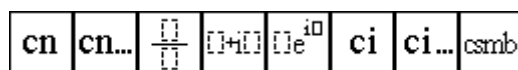
Now type number 100 and by switching to the “MathML Text” see how MathML Weaver automatically detected that while a user is situated inside of the Content Markup expression then the Content token element is needed.



Token Elements: externally defined symbol (csymbol)

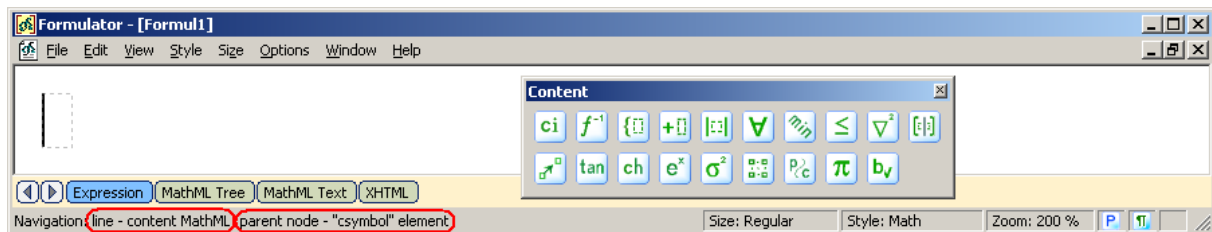
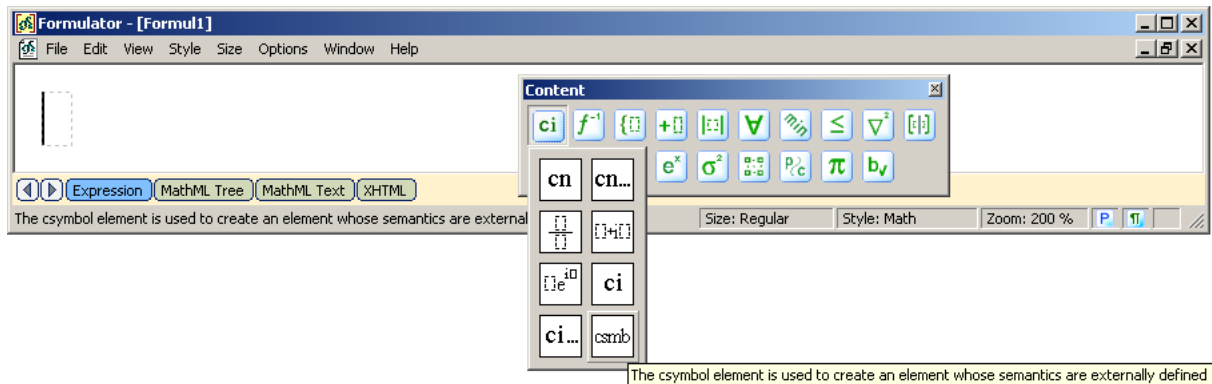
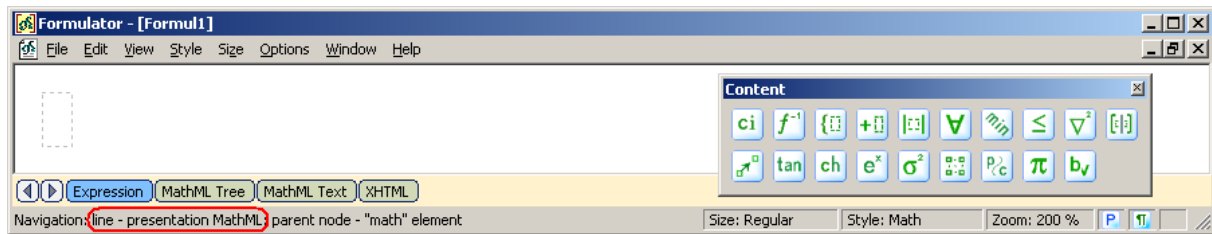
Content expression trees are built up starting from token elements. Numbers and symbols are marked by the “cn” and “ci” elements; the “csymbol” element allows to create an element whose semantics are externally defined.

Content Markup token elements are available via **ci** toolbar:

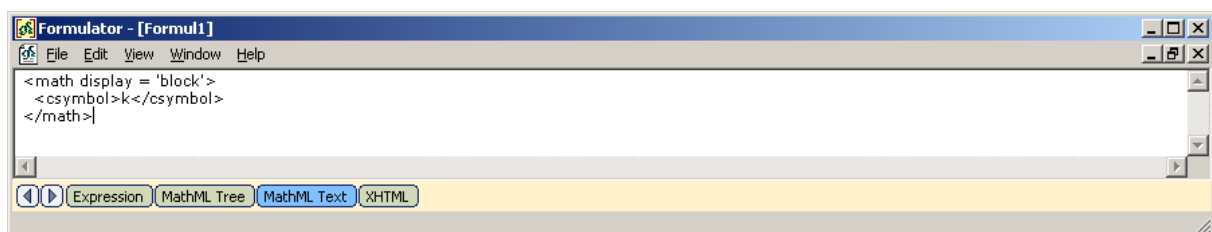
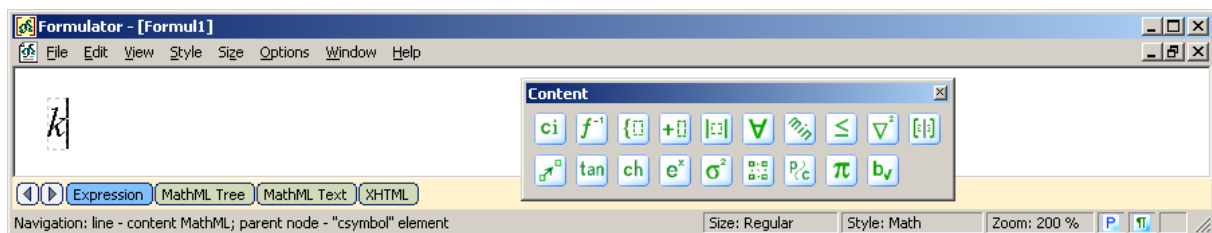


As we can see from this figure, MathML Weaver provide several buttons for a token element to account for existing options of base, type, etc. A button with ‘...’ sign on its image leads to a dialog requesting some context dependent attributes of the newly created expression (e.g., type of an identifier).

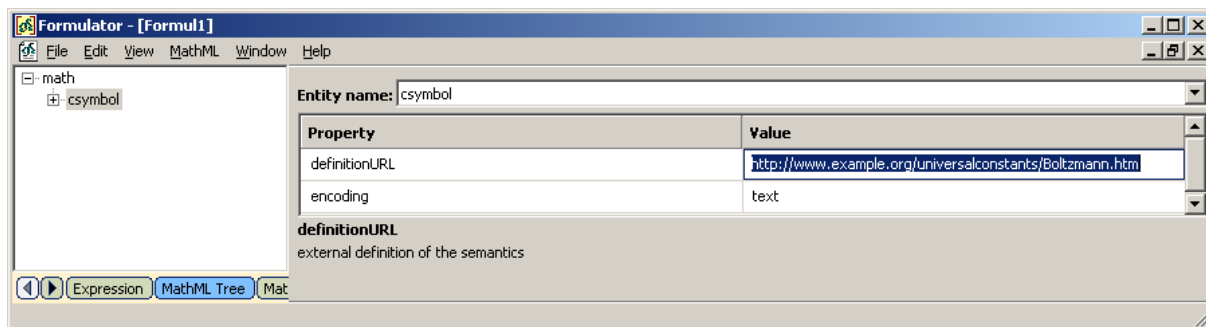
1. The simplest case of the Content token element is presented by “csymbol”. When a user presses **csmb** icon, the current empty slot is not altered visually, but the navigation information on the status bar helps to understand that we have already started to build a Content expression tree. The next tree figures shows this in details.



Now all that we type in the current empty slot will be interpreted as a content of the “csymbol” element:




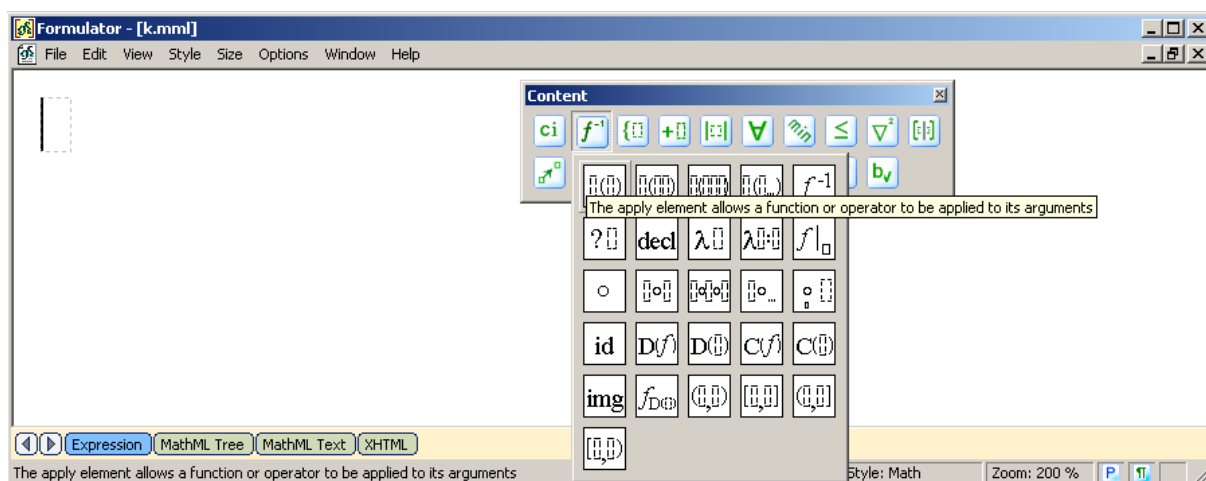
2. It is worth noting that this just created Content expression can be supplied with all the need attributes on the page of “MathML Tree”. The next figure shows how to turn this expression into the example 4.4.1.3.3 from the W3C MathML Recommendation (reference to human readable text description of Boltzmann's constant).



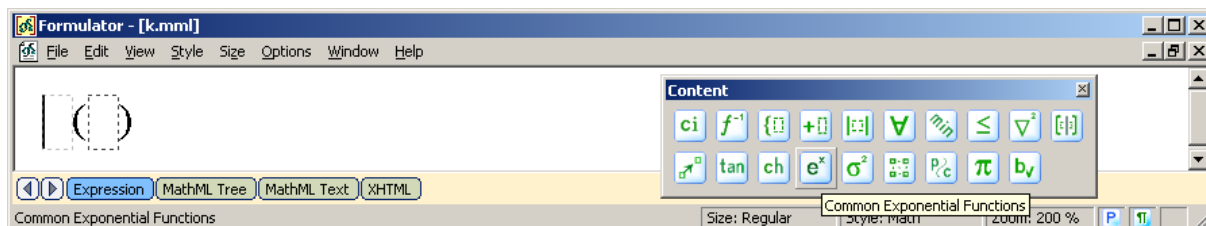
3. Another important feature of the inserted visual forms for Content Markup in MathML Weaver is that they also can carry Presentation Markup. For example, if we use a set of Presentation toolbars we can easily create another example from the section 4.4.1.3.3 of the W3C MathML Recommendation (reference to OpenMath formal syntax definition of Bessel function):

```
<!-- reference to OpenMath formal syntax definition of Bessel function -->
<apply>
  <csymbol encoding="OpenMath"
    definitionURL="http://www.openmath.org/cd/hypergeo2#BesselJ">
    <msub><mi>J</mi><mn>0</mn></msub>
  </csymbol>
  <ci>y</ci>
</apply>
```

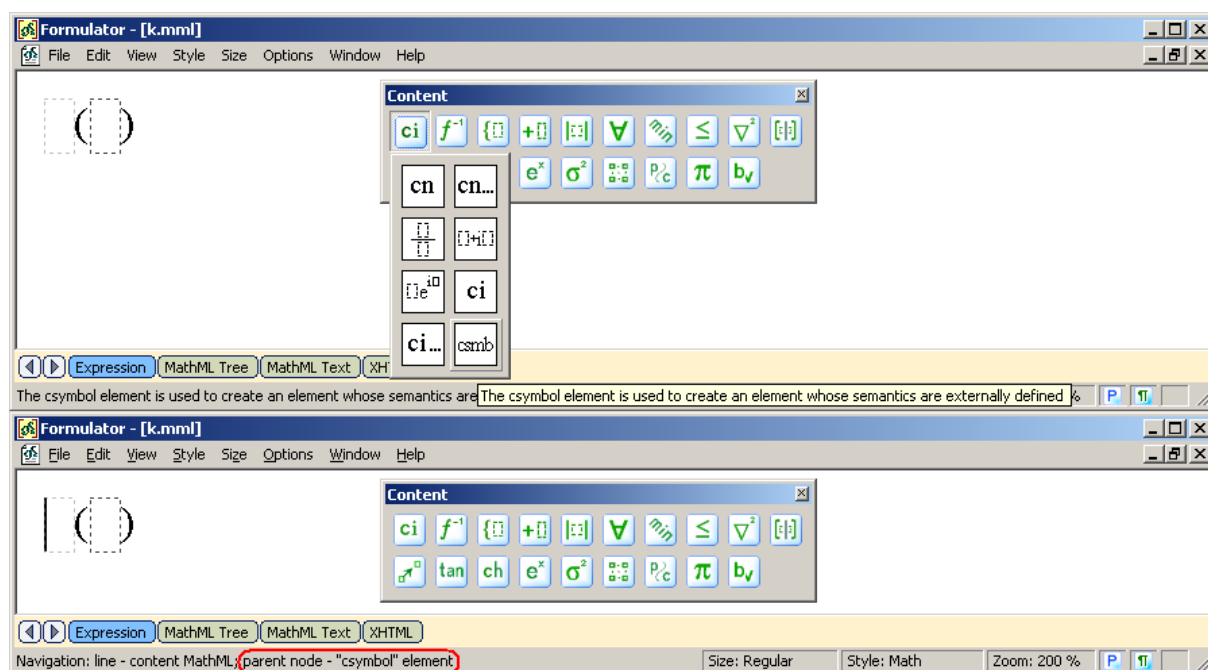
Press the  button on the Content toolbar set to get access to the group of “apply” elements of different form. Choose the first button.



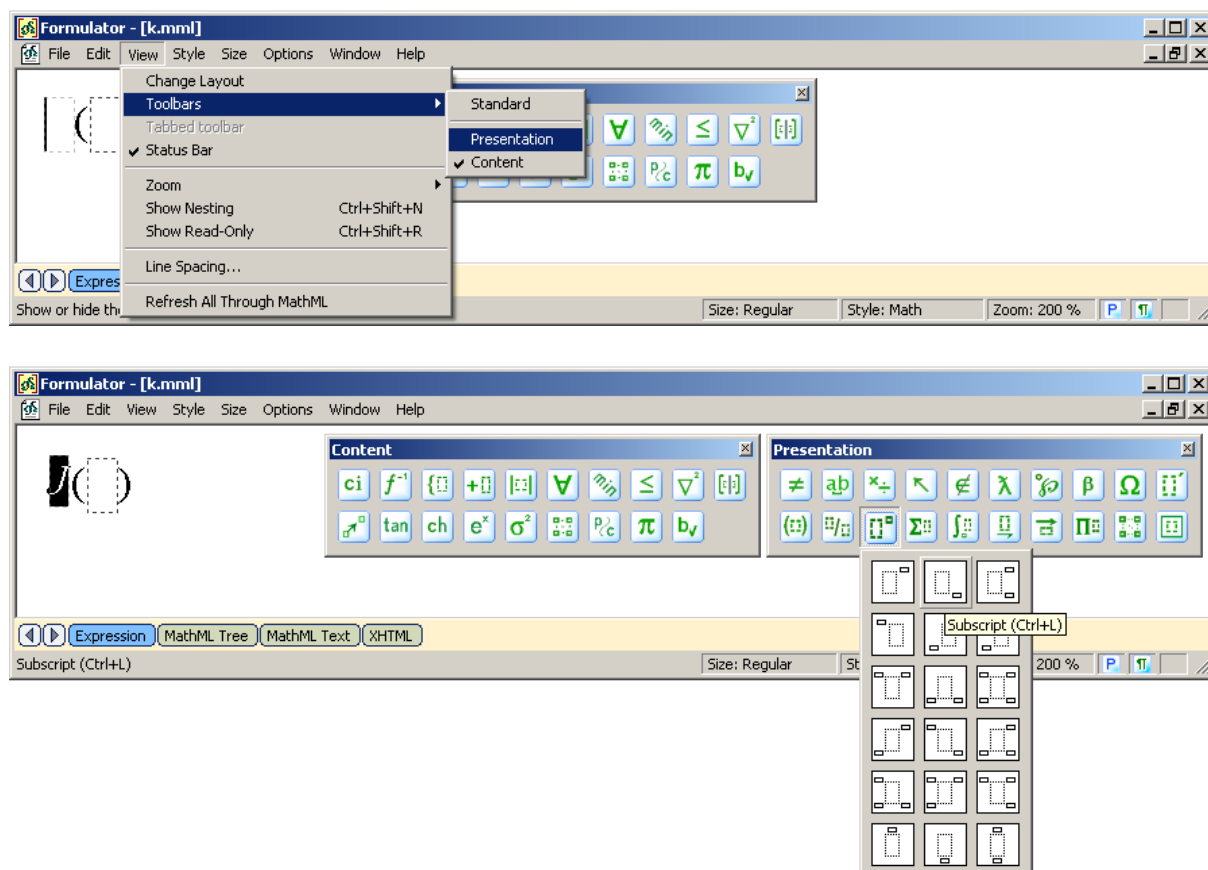
This will insert an “apply” element with one argument in its functional form.

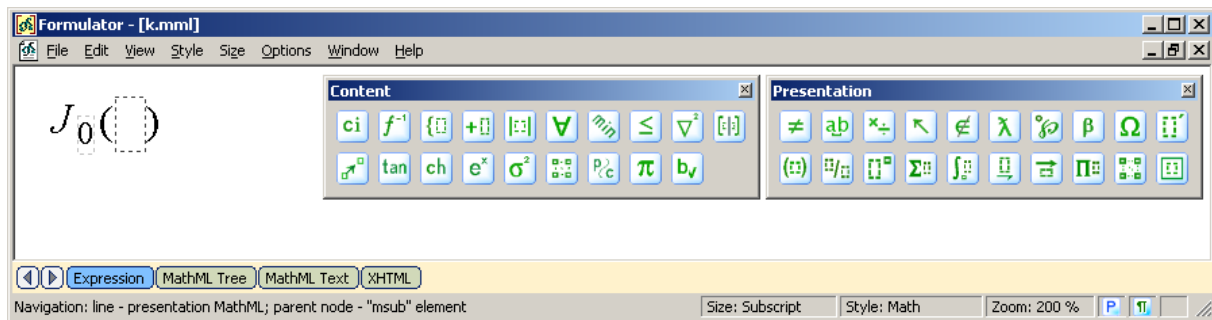


Now insert the “csymbol” element in the first input slot of the “apply” element.

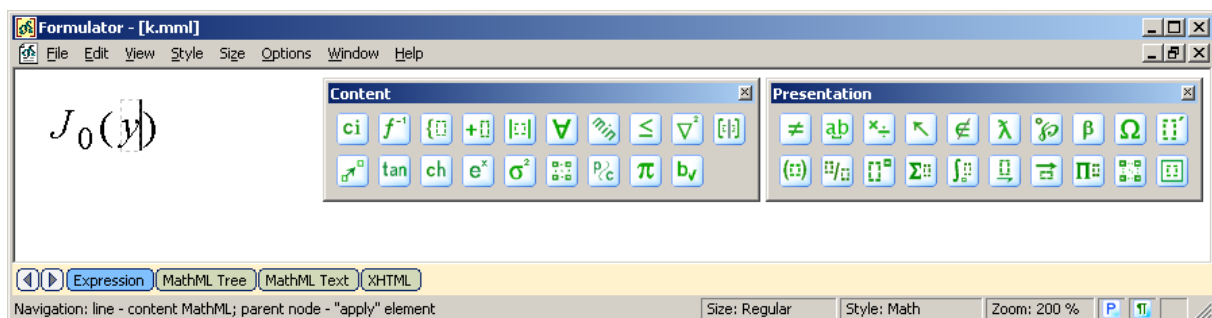


Using the Presentation toolbars set build up contents of the “csymbol” element: $\langle \text{msub} \rangle \langle \text{mi} \rangle J \langle \text{mi} \rangle \langle \text{mn} \rangle 0 \langle \text{mn} \rangle \langle \text{msub} \rangle$ (J_0).

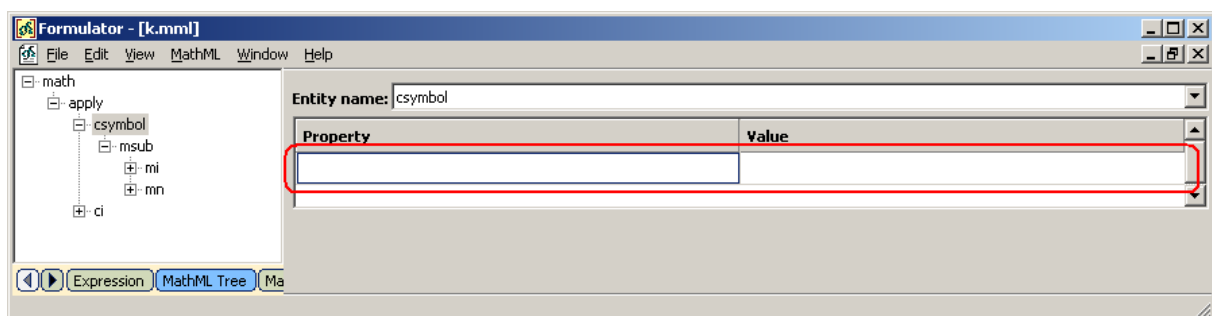




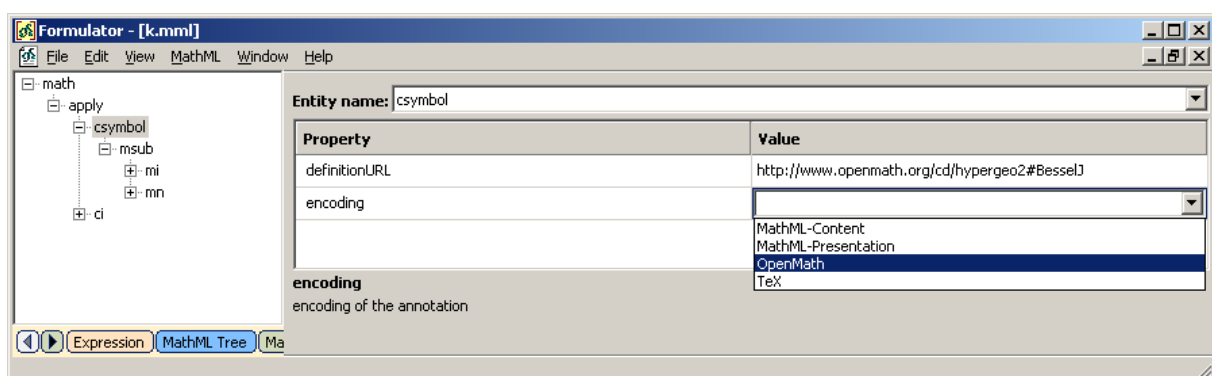
Navigate to the second input slot of the “apply” element and press ‘y’. This will fill the argument of the “apply” element with `<ci>y</ci>` content.



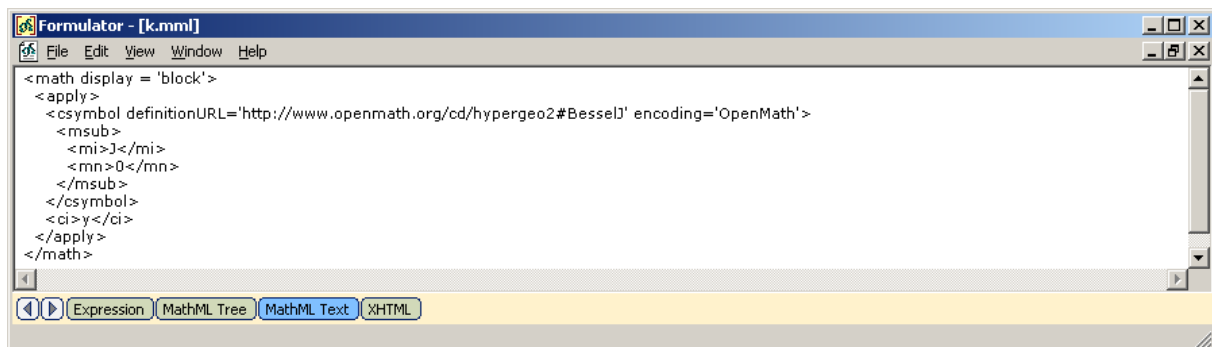
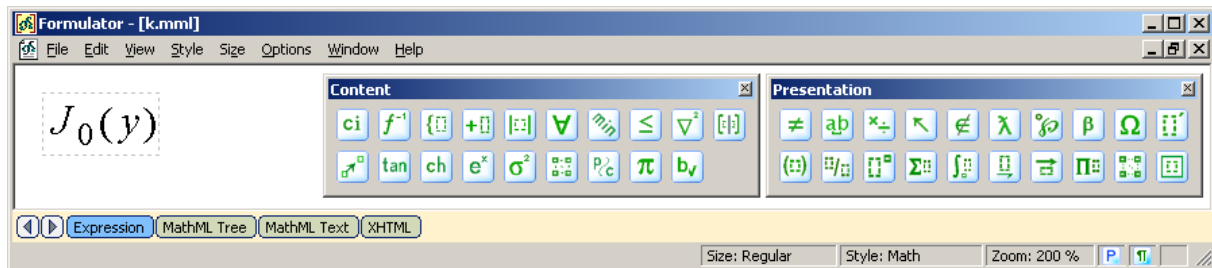
Now switch to the “MathML Tree” to add two attributes to the “csymbol” element. In order to do this click on the right side of the document (“Property-Value” dialog) and press the Insert button. We now see that an empty attribute record is added.



Using drop-down lists create “encoding” and “definitionURL” attributes.



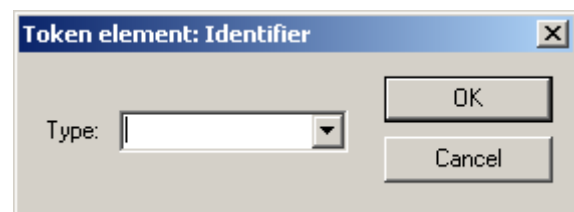
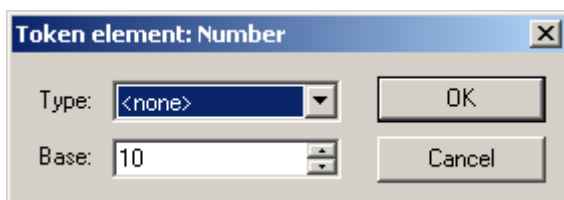
Check the results on the “Expression” and “MathML Text” pages.




Token Elements: numbers (cn) and identifiers (ci)

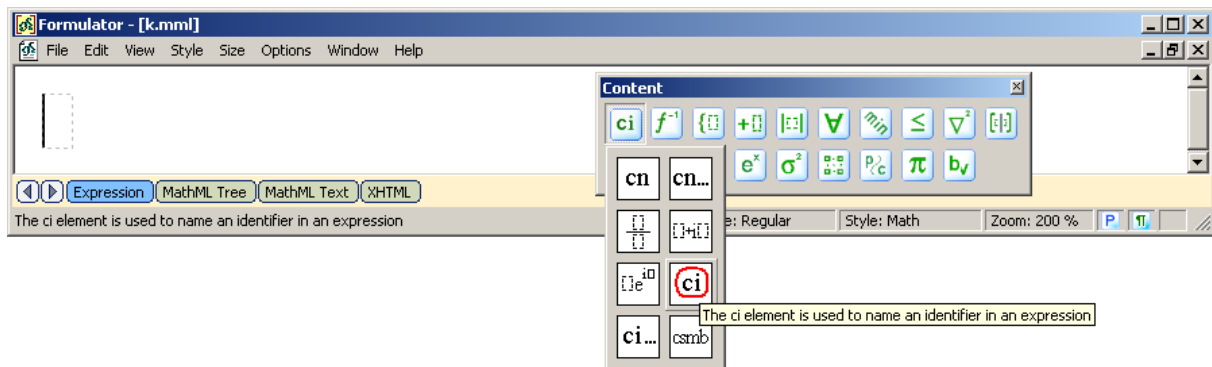
There are several attributes modifying Content Markup semantics for token elements. Thus the “base” attribute indicates numerical base of the number; the “type” attribute indicates type of the number or identifier. Moreover, different types of numbers will be rendered in a different way and even appearance of identifiers can be altered by changing the “type” attribute.

In order to visually support these requirements, Formulator 3.9 MathML Weaver proposes several ways to input Content Markup token elements: namely, a user can choose a button representing the needed type of a number from the toolbar or make some alterations in the attribute values via the “MathML Tree” page or by using corresponding property dialogs for buttons **cn...** and **ci...**:

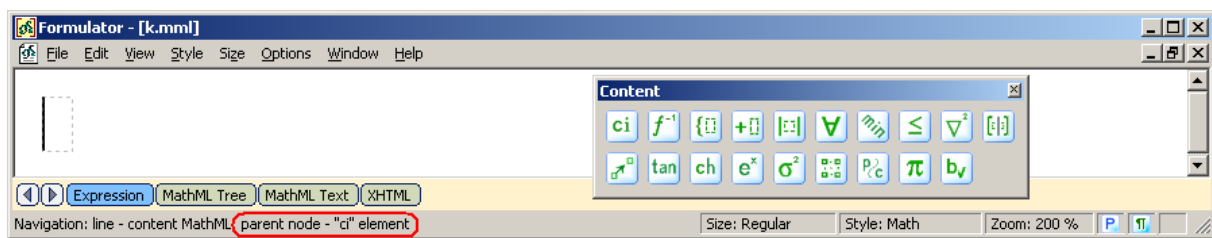


1. The next example shows how to create a simple identifier token element and how to change its appearance by editing the “type” attribute.

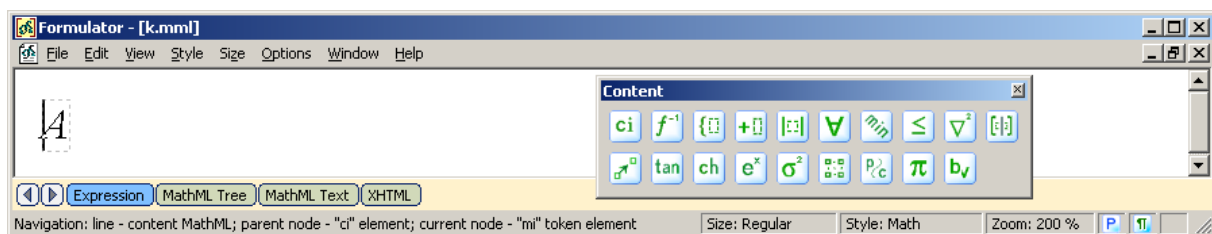
Insert the “ci” element by pressing the  button.



As it is shown on the next figure, the “ci” element appears as an empty slot and the navigation bar information makes sure that it is inserted.

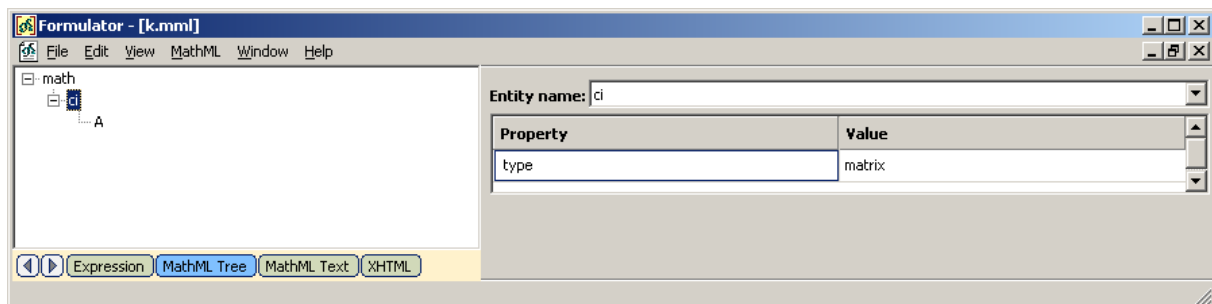


Now type an identifier in the current input slot. Note that the context of the “ci” element is one of three cases where presentation markup may appear in content markup properly. So, not only plain text can be typed into the “ci” input slot, but also some kind of presentation markup tree.

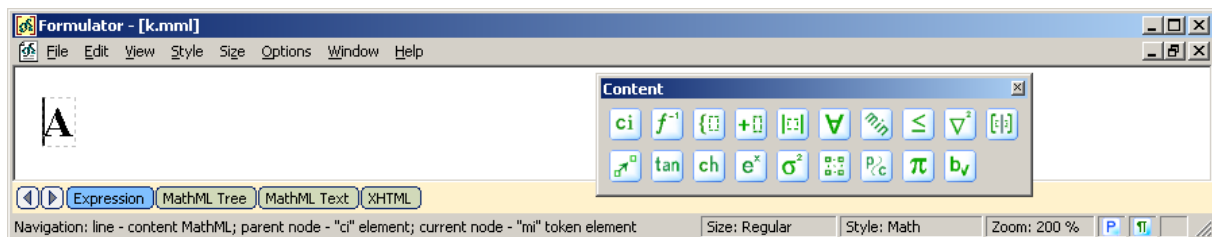


According to the default value of the “type” attribute of the “ci” element (unspecified type), contents of the “ci” element are represented as if it were contents of a “mi” element (italic style).

Now change the current page of MathML Weaver and add the “type” attribute into the “ci” node. Then choose the “matrix” value from the list of known identifier’s types.



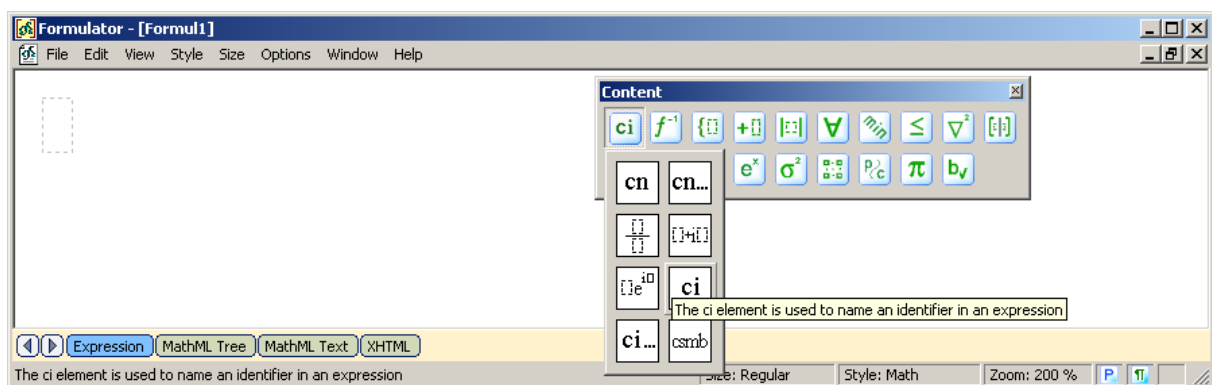
Switch back to the “Expression” page and see how rendering of the “ci” element is changed. Now A is bold according to the typographical rules of representing the “matrix” type.



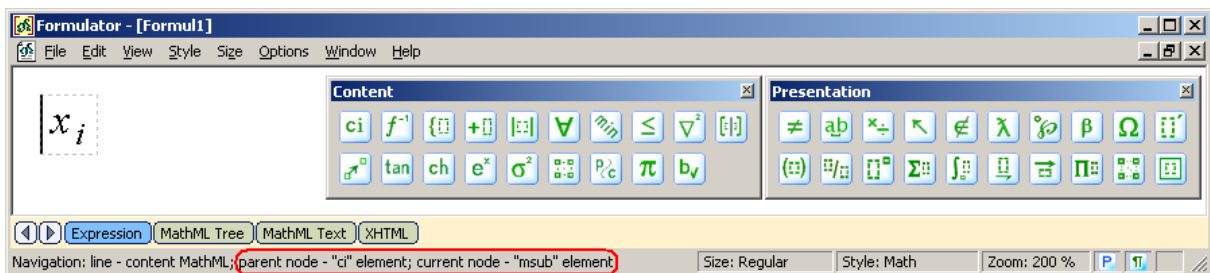
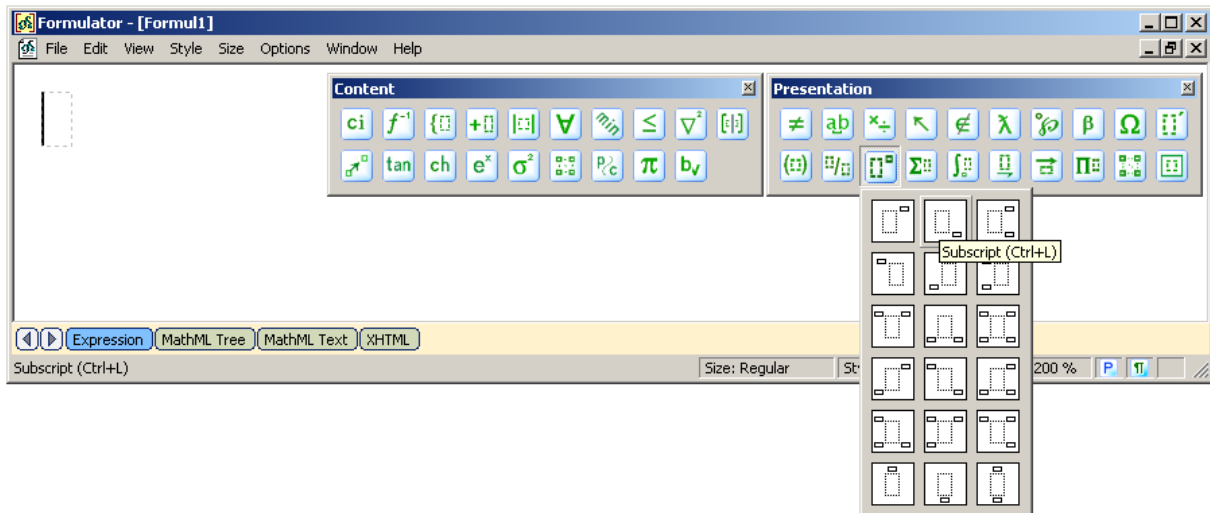
As in the case of the “csymbol” element, Content Markup identifiers can carry Presentation Markup. For example, if we use a set of Presentation toolbars we can create an example from the section 4.4.1.2.2 of the W3C MathML Recommendation:

```
<ci>
  <msub>
    <mi>x</mi>
    <mi>i</mi>
  </msub>
</ci>
```

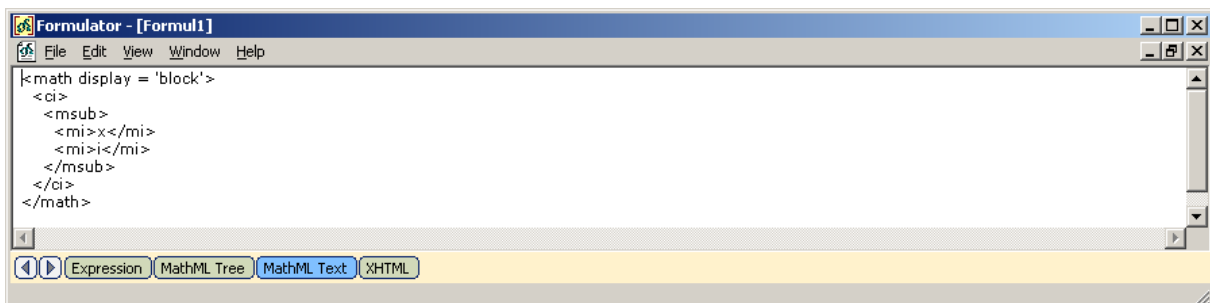
Insert the “ci” element.




Use the Presentation toolbar to insert Subscript node and fill it with x_i expression.

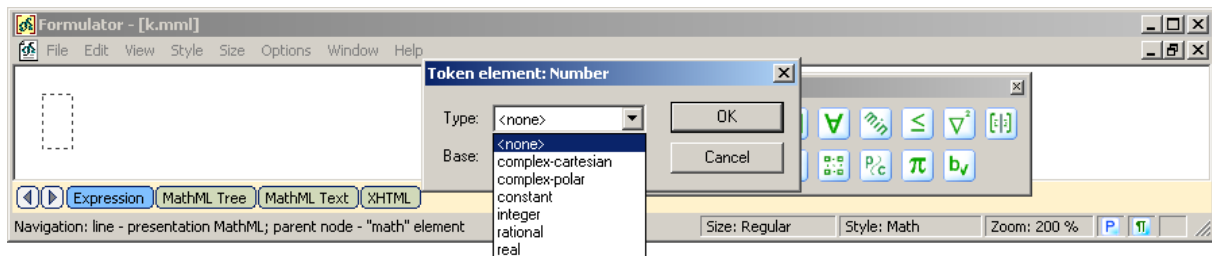


See results on the “MathML Text” page.

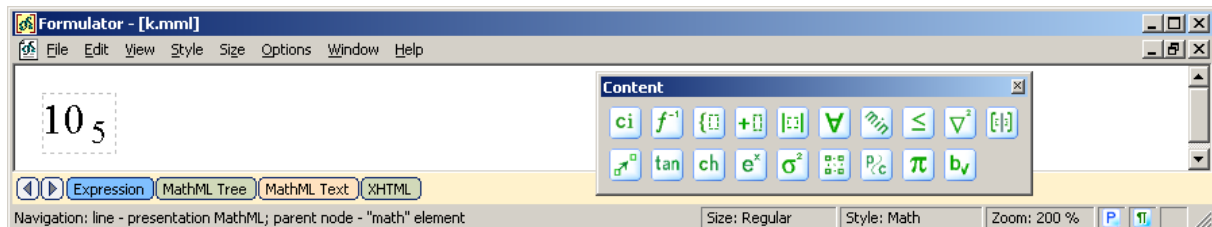


2. The next example shows how to create a simple number token element and how to control its appearance and semantics by altering “type” and “base” attributes.

Insert the “cn” element by pressing the  button. Then the initial property dialog for the “cn” element will be shown.



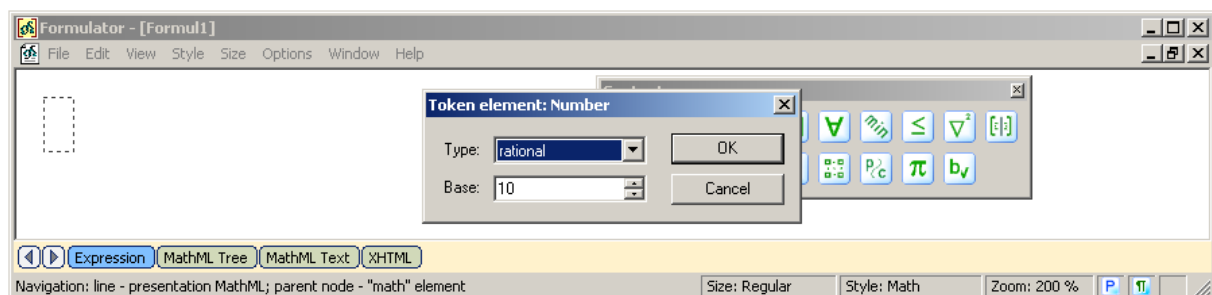
Choose the “integer” value in the “type” drop-down list and 5 as a base of a number. This feature lets creating numbers of different numerical bases.



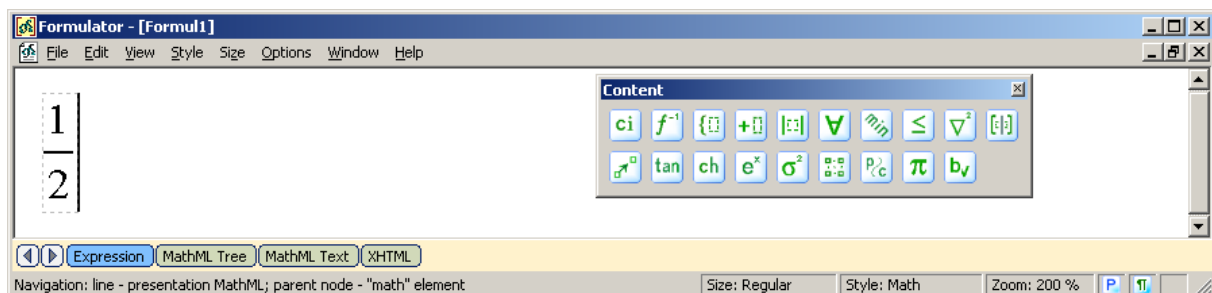
A user can control number’s type not only via different values proposed by the initial property dialog (“rational”, “complex-cartesian”, “complex-polar”), but also with the help other available buttons of the **ci** toolbar.


For example, rational numbers can be created by pressing the **cn...** button if the “rational” type is selected afterwards from the initial property dialog or at once with the **ci** button.

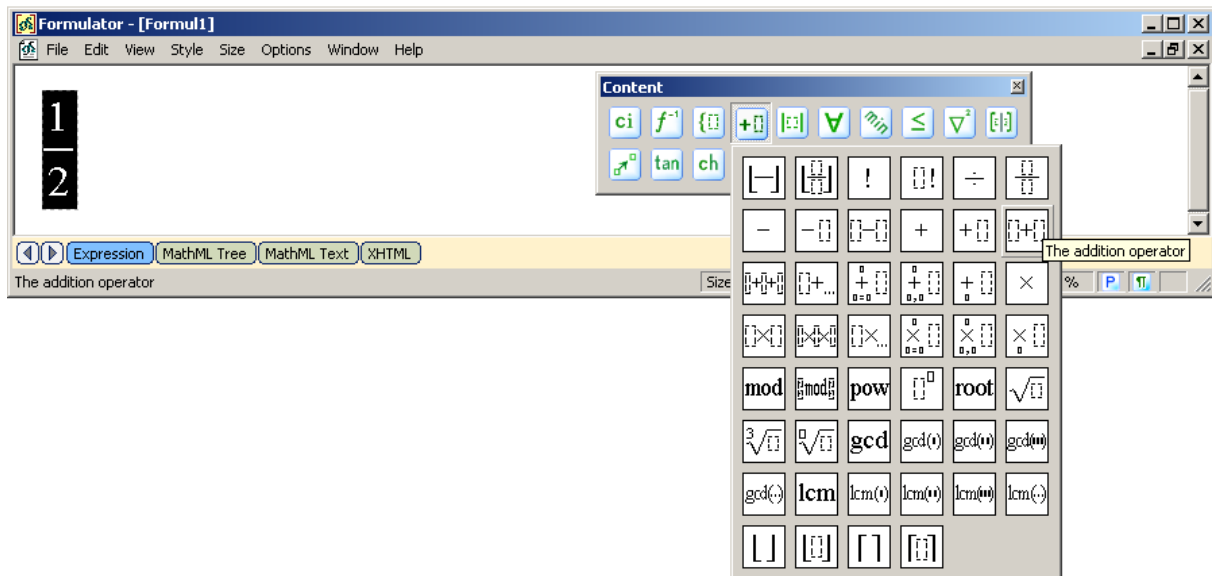
Press the **cn...** button and select the “rational” type.




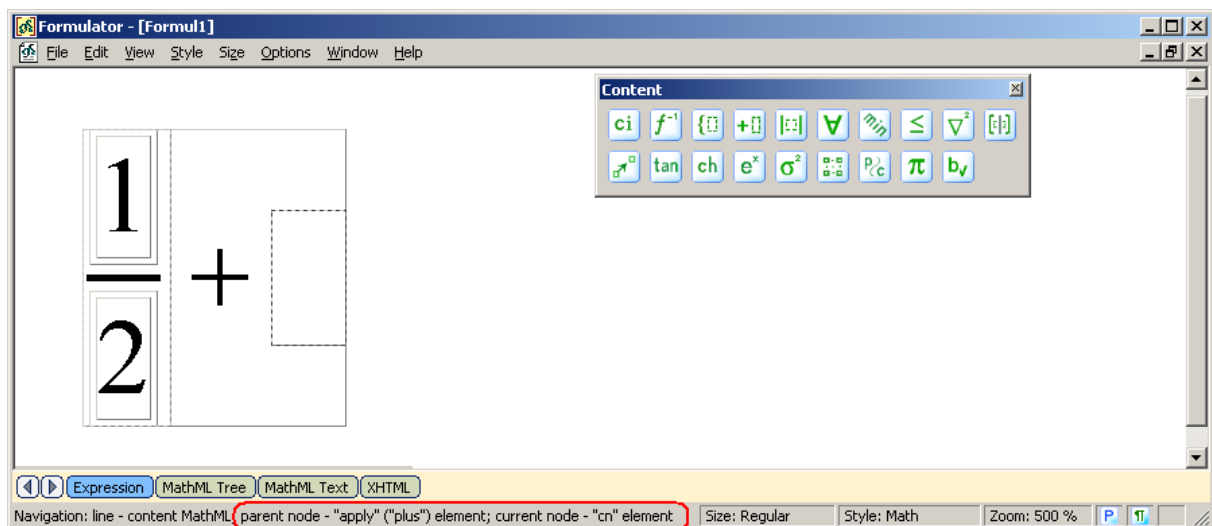
Fill numerator and denominator with the numerical values.




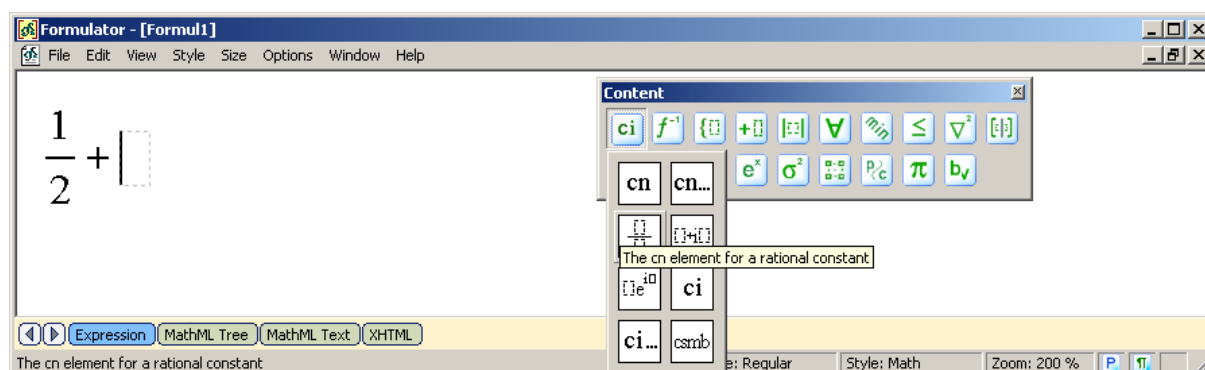
Select the created rational number (with Shift + arrow keys or with mouse) and click on the  button. The pane of Arithmetic Operators of the Content Markup is now available. After pressing on one of the buttons representing the “apply” element with one and more arguments the current selection will be considered as the first argument of this “apply” element.



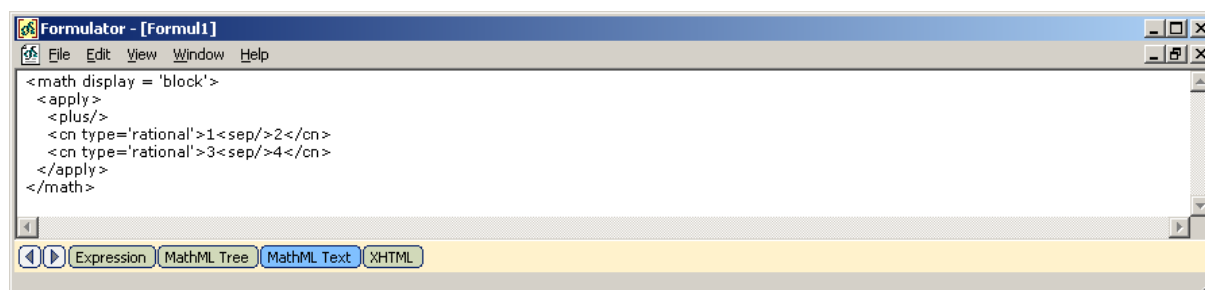
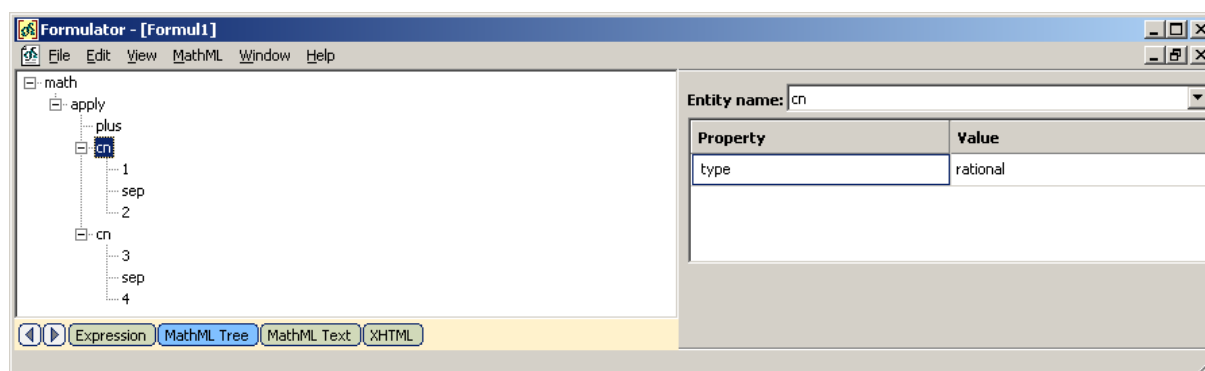
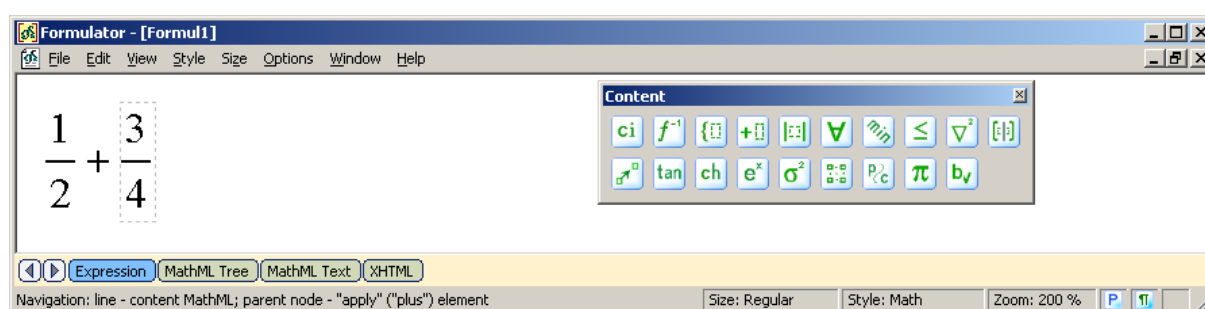
Let's press on the  button and see how the structure of the built MathML tree is changed. The figure suggests that the `<apply><plus/>...</apply>` pattern is inserted and the rational “cn” element is the first argument of this operation.



To fill the second input slot, navigate to it with the Right arrow and press the  button of the **ci** toolbar. This will insert another rational number without additional dialogs.




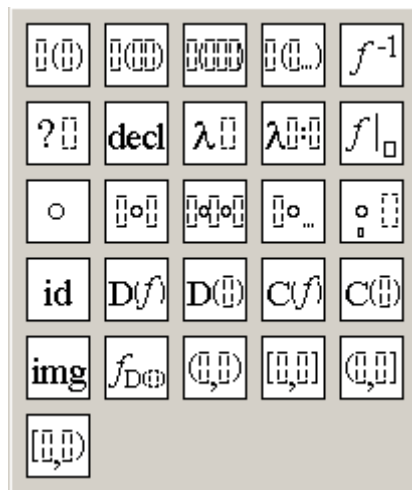
Compare results on different pages of MathML Weaver (“Expression”, “MathML Tree”, “MathML Test”).



Basic Content Elements: apply

The “apply” element of the Content Markup allows a function or operator to be applied to its arguments. It is the basic element in a sense that the overwhelming majority of expressions in MathML content markup is carried out by applying operators or functions to arguments.

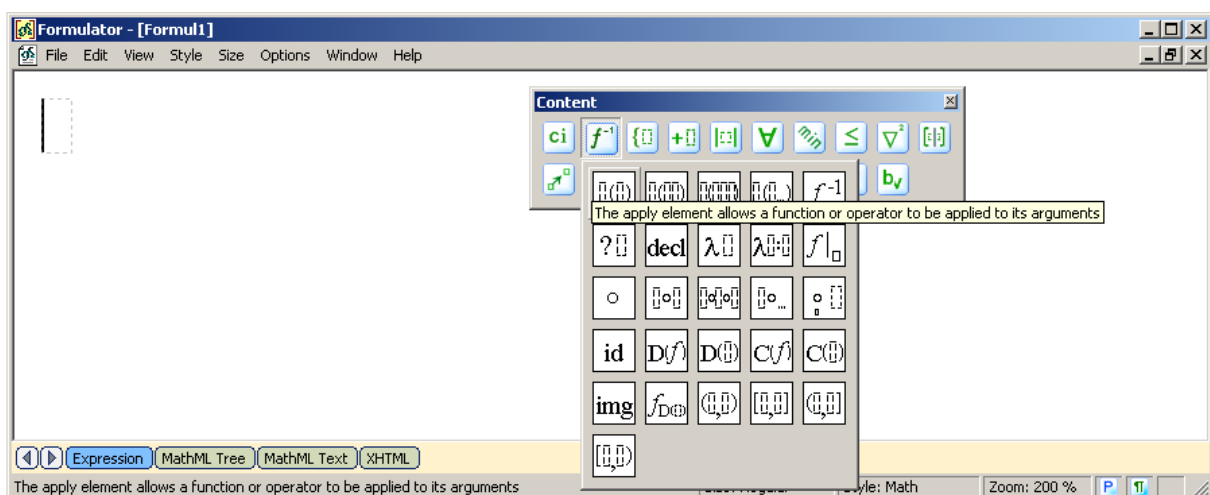
The toolbar of basic content elements  in MathML Weaver contains several groups of buttons for creating different forms of the “apply” element and some related to this element concepts:

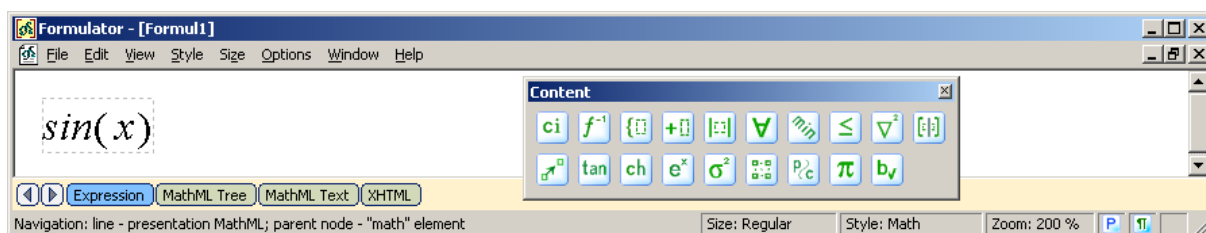
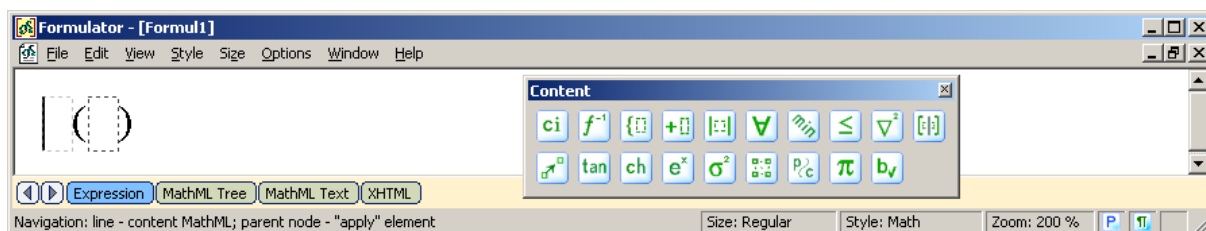


The first 4 buttons in the toolbar are responsible for the “apply” element itself: 1, 2, 3, and many-arguments forms. Please note, that these buttons represents the most general, functional form of the “apply” element appearance. Theoretically we can use these buttons for some other purpose, e.g., for arithmetic operators applying, but since there is a traditional infix form of rendering for arithmetic operators, such way of editing MathML content markup is not recommended. The next example shows this and others features of work with the “apply” element.

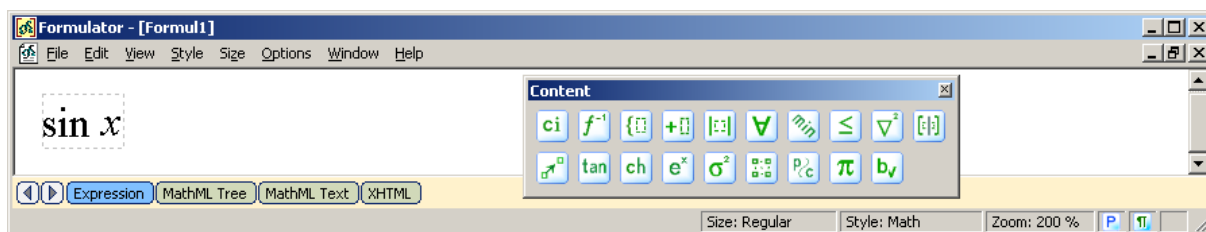
1. Insert $\sin(x)$ expression.

Press the  button; type “sin” in the first and “x” in the second input slot.



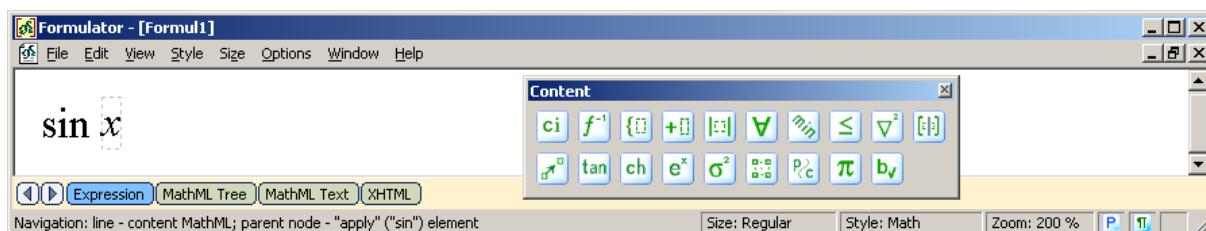
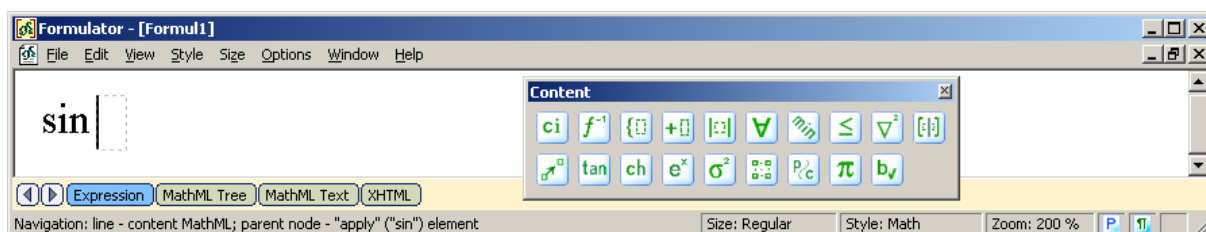


The rendering of this MathML fragment is not finished; to make it correct use the “Refresh All Through MathML” command from the “View” menu. This helps MathML Weaver to find the meaning of the entered text and to render it according to the mathematics representation rules.

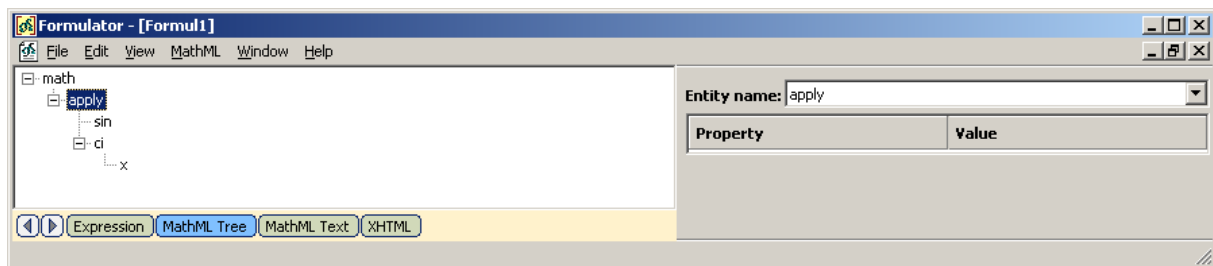


We did need such an additional action, because the most general form of the “apply” element was used. Normally, it is quite enough for users to have just special toolbars with common functions and operators. The previous example in that case would be more simple, as the next figures suggest.

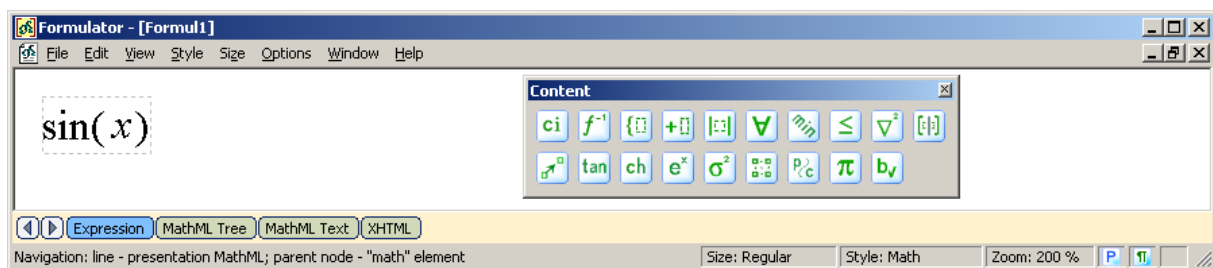
Press the **sin** button of the common trigonometric functions toolbar (**tan**) and type “x” in the input slot.



We have the same MathML tree for $\sin(x)$ as earlier, but now it was much faster.

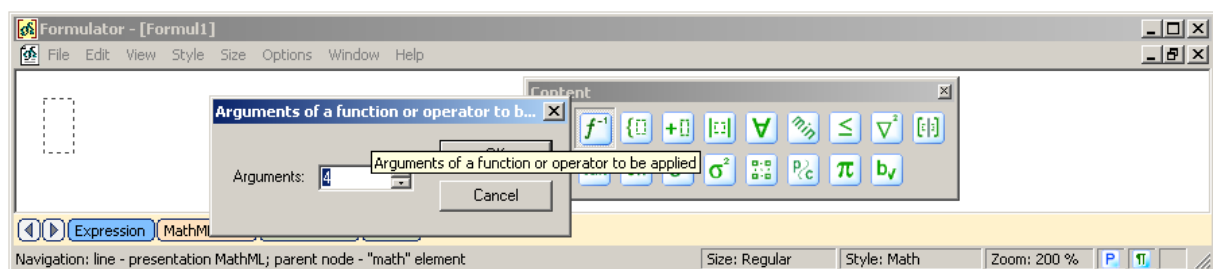


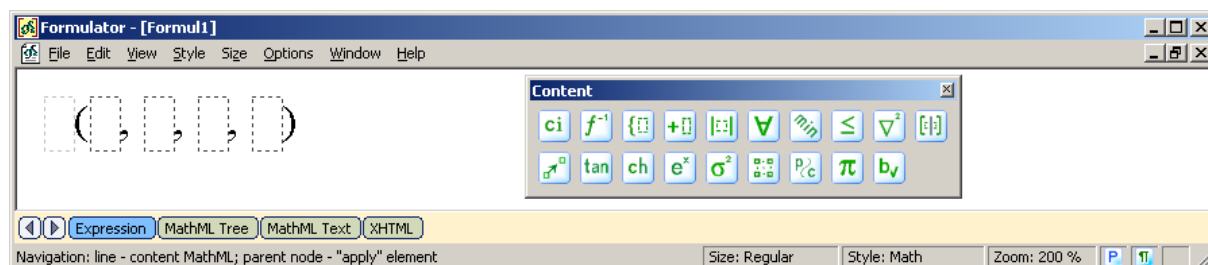
There is one more way to deal with this example, but it is similar to the first case and is shown here only for the sake of completeness. The only distinction is that we can not just type the “sin” value into the input slot, but press the corresponding **sin** button from the common trigonometric functions toolbar (**tan**). Then MathML Weaver will know about the special (plain, not italic) rendering of the first argument of the “apply” element, but still we should use the “Refresh All Through MathML” command from the “View” menu to help MathML Weaver understand that enclosing the “x” text in brackets is not needed:



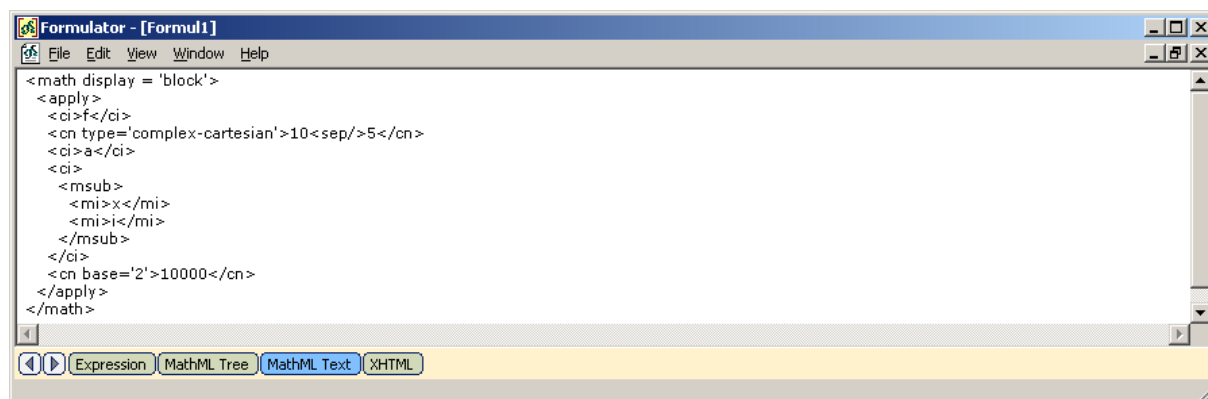
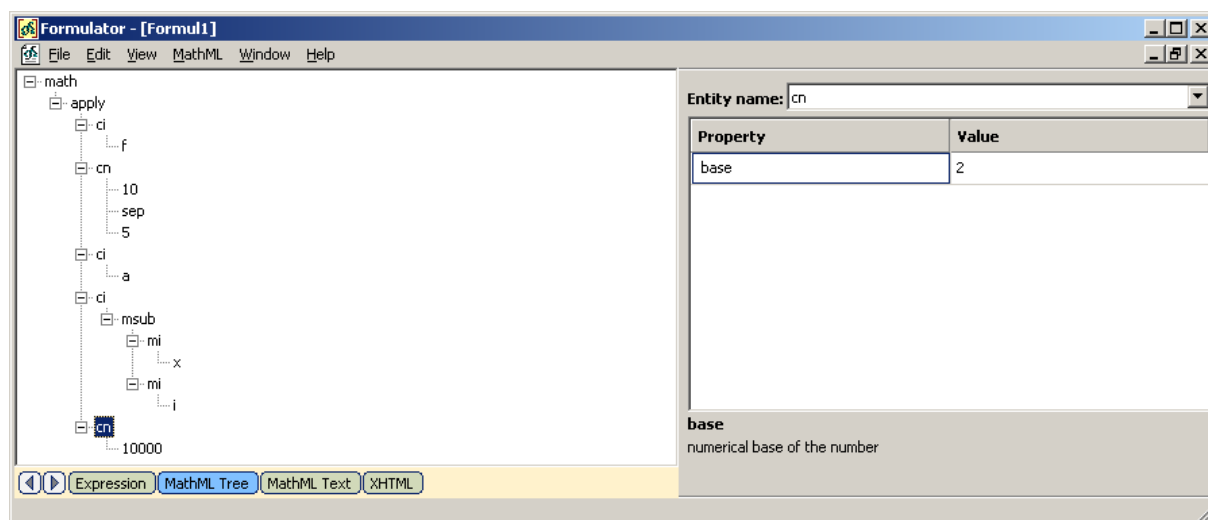
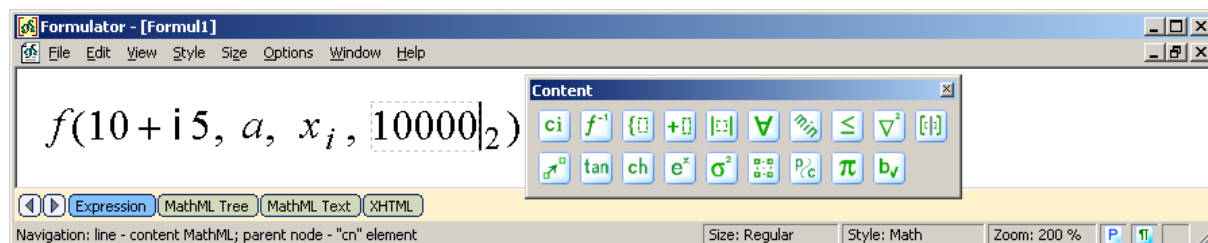
2. Insert a function of four arguments.

Press the **f(x₁, x₂, x₃, x₄)** button; fill the needed number of arguments in the shown dialog and see the newly created visual representation of the “apply” element.



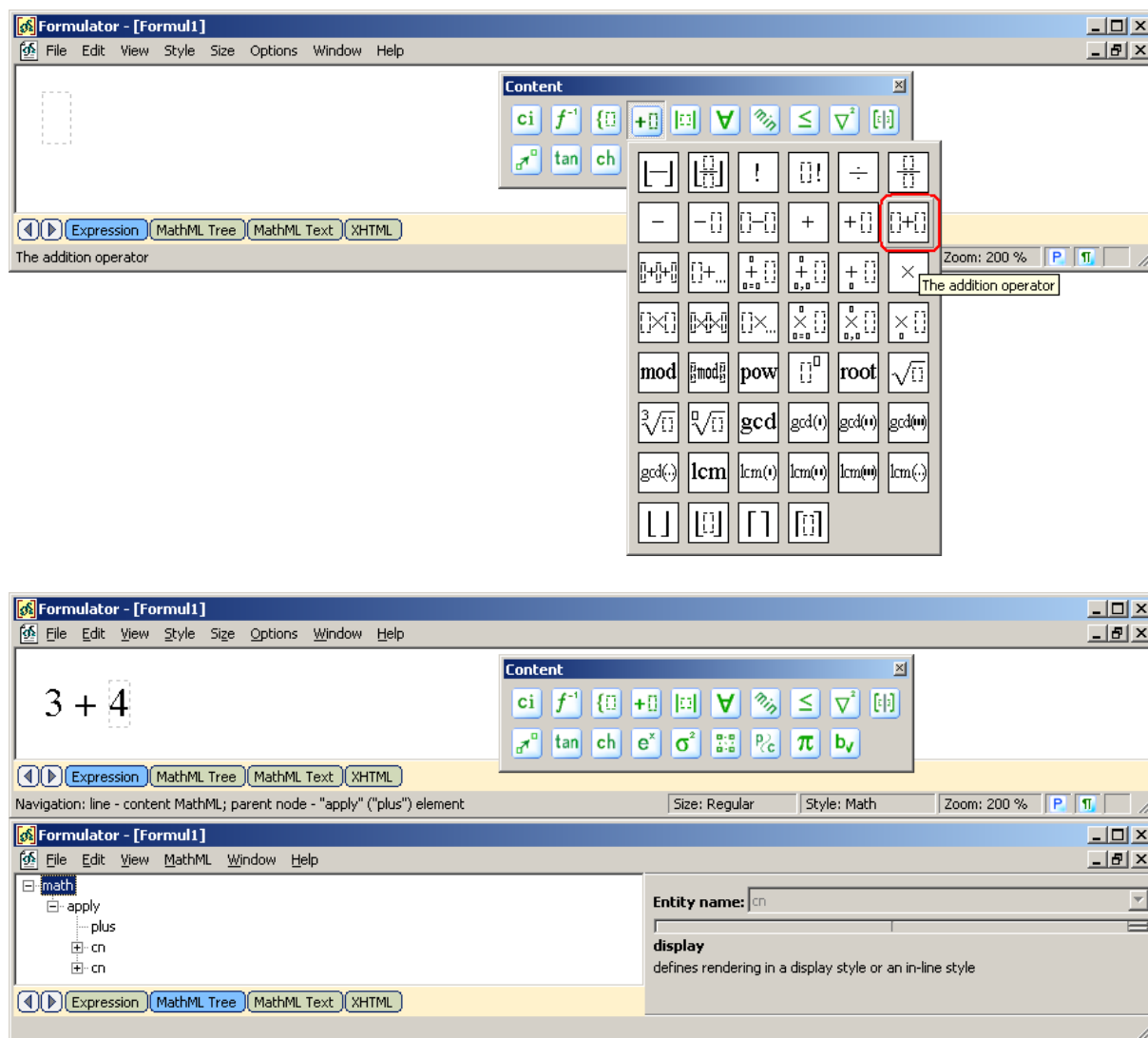


Now this form can be filled with other elements of the MathML content markup.

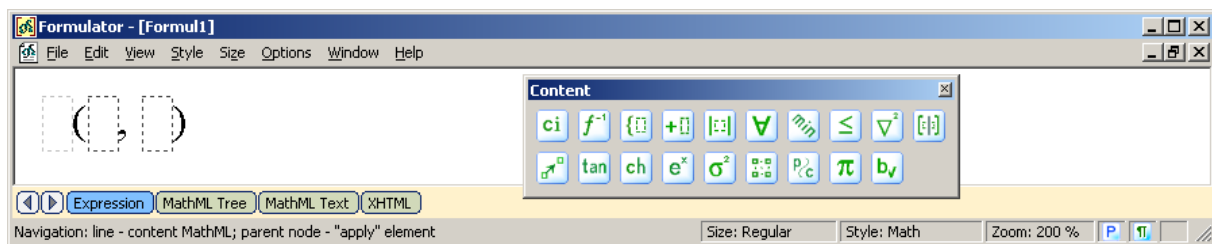
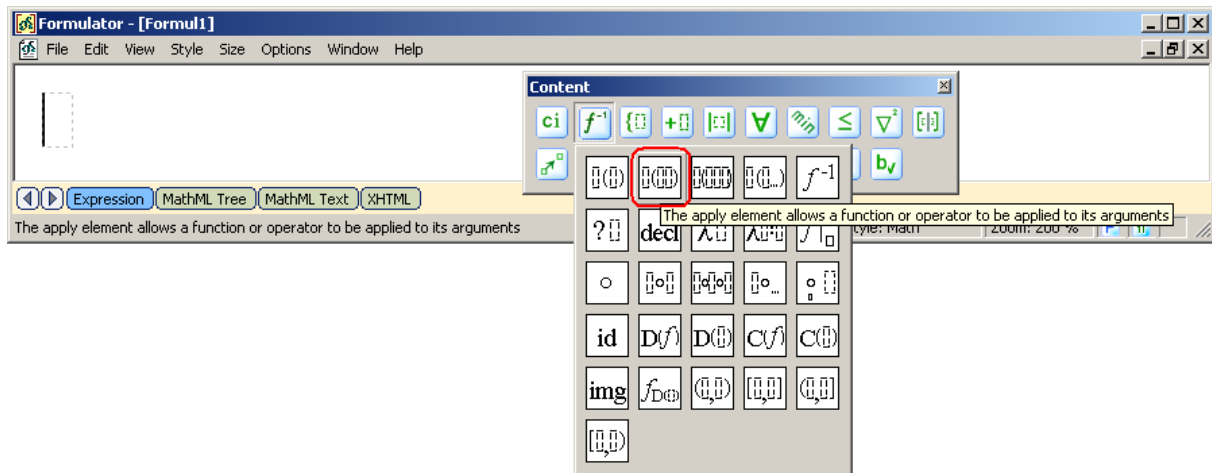


3. Create an arithmetic expression.

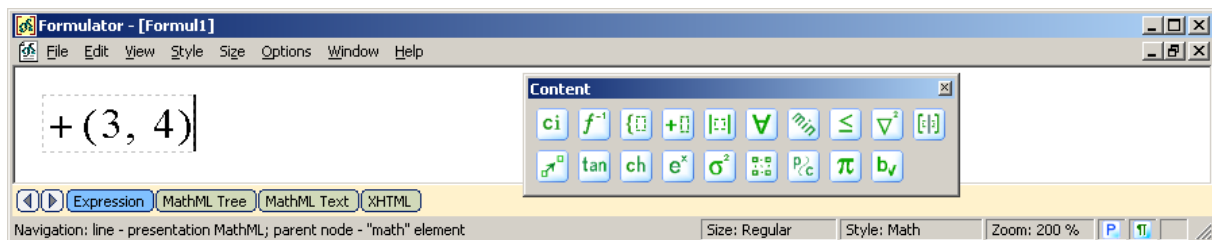
The easy way is to use the corresponding toolbar and special buttons:



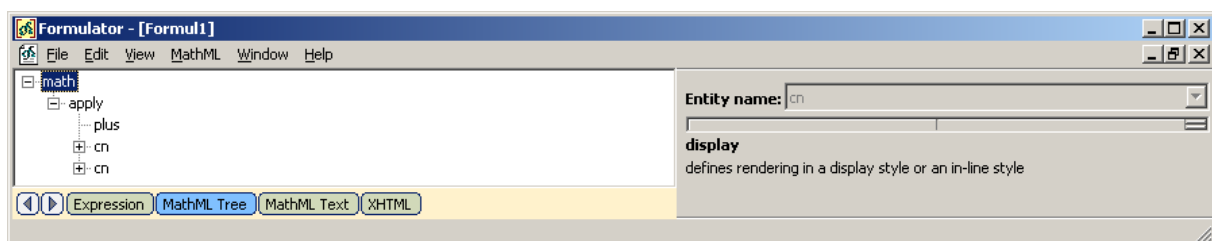
There is also a more long way to create this form by means of the “Basic Content Elements” toolbar.

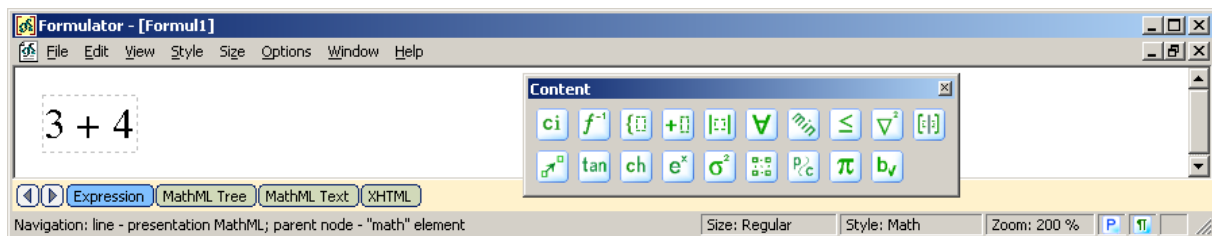
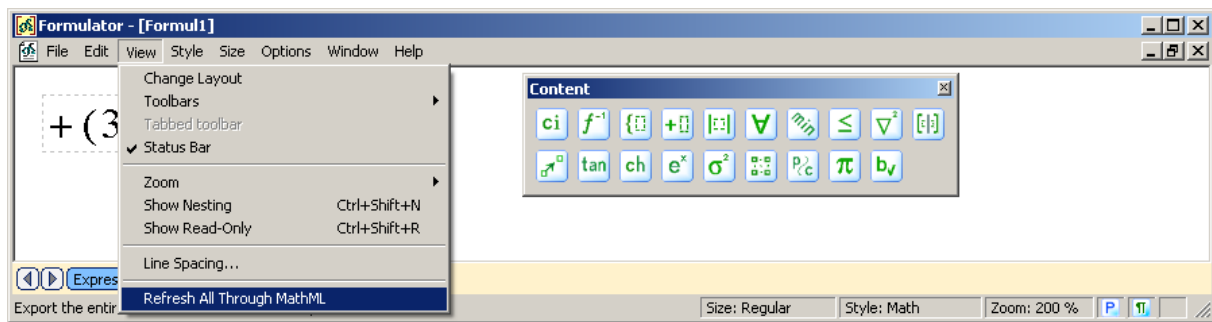


Type the operator '+' and values into input slots as if they represent the functional (in contrast to infix) notation.



The structure of the MathML tree is already built up, but the rendering will become correct after refreshing the text by means of the "Refresh All Through MathML" command.

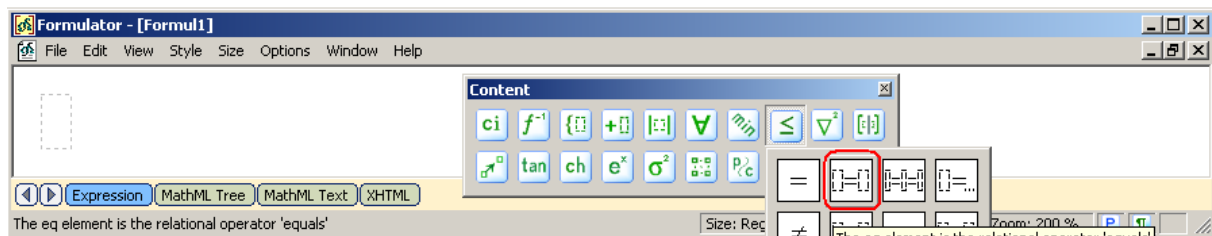


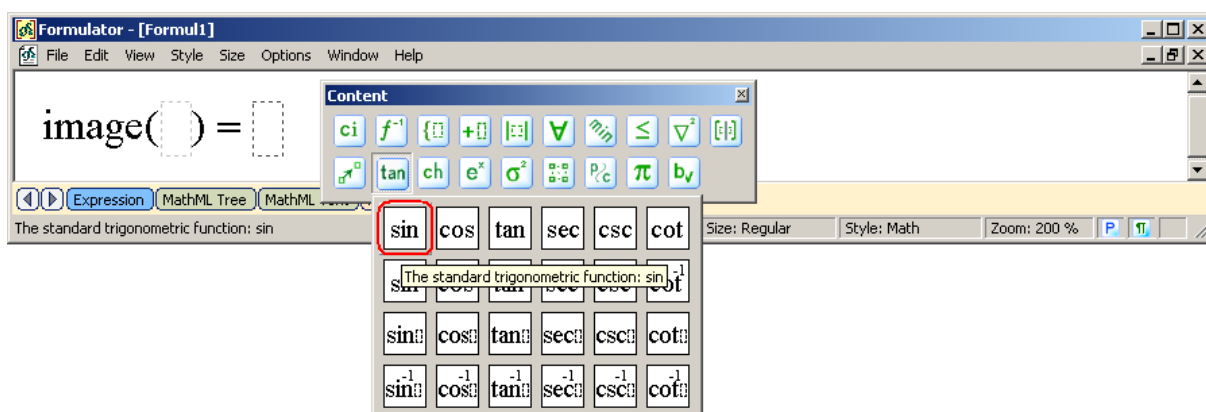
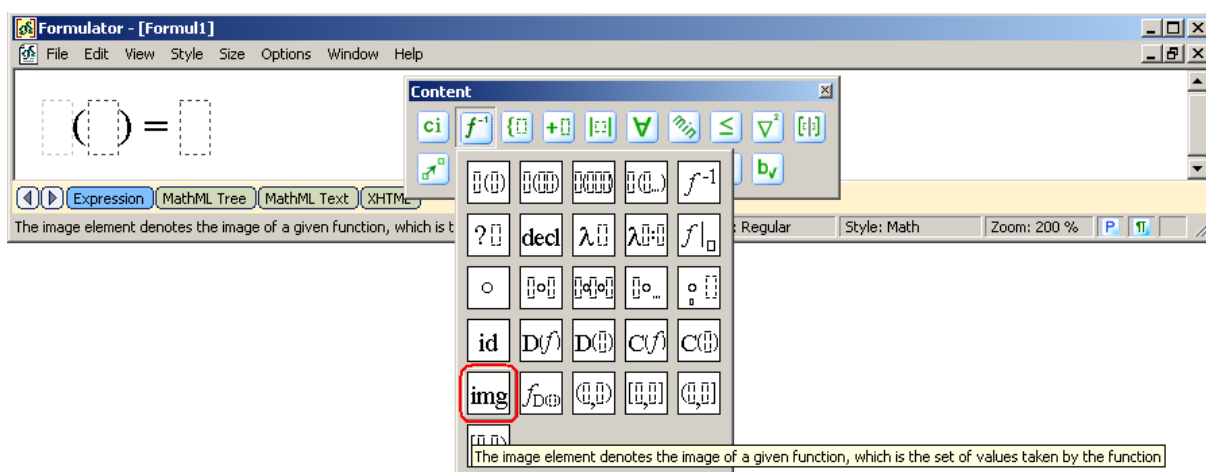
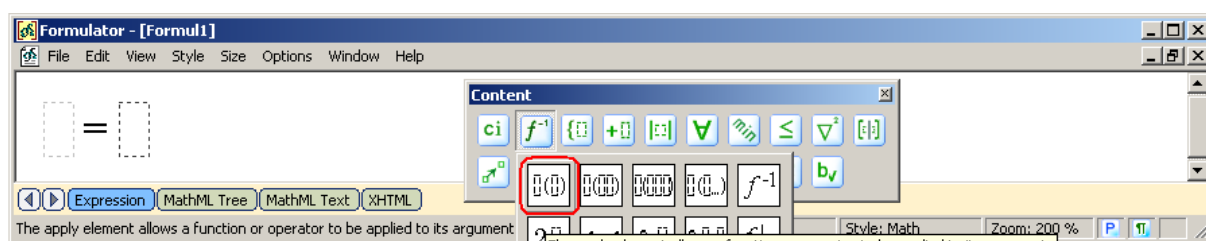


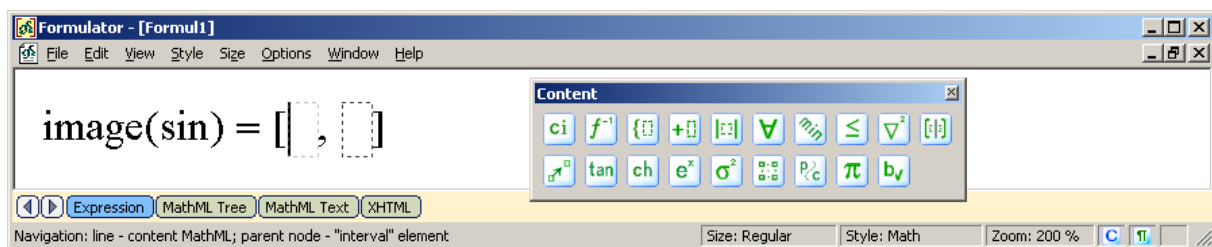
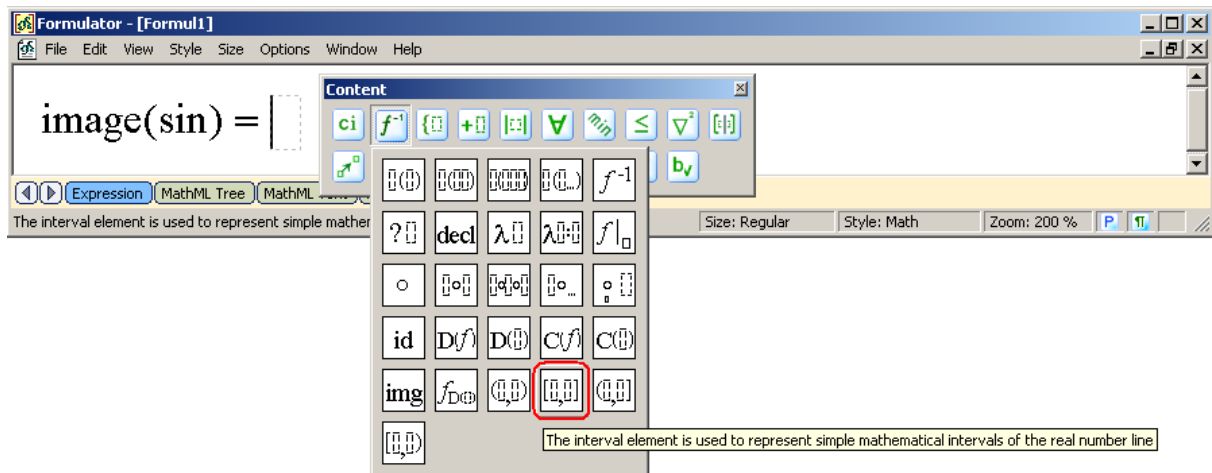
Basic Content Elements: functions and operators

1. Create the image of a given function, which is the set of values taken by the function (4.4.2.14.2 example from the W3C MathML Recommendation):

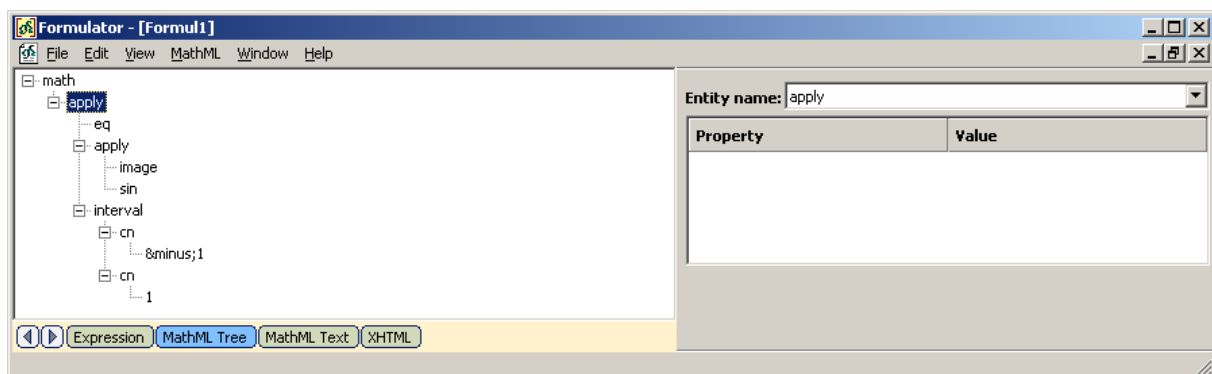
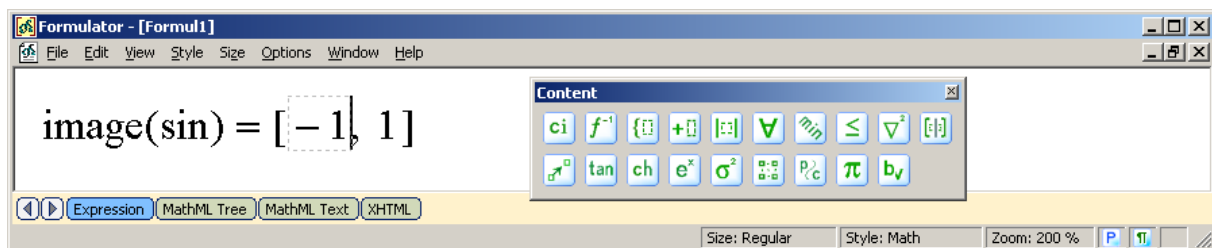
```
<apply>
  <eq/>
  <apply><image/>
    <sin/>
  </apply>
  <interval>
    <cn>-1</cn>
    <cn> 1</cn>
  </interval>
</apply>
```

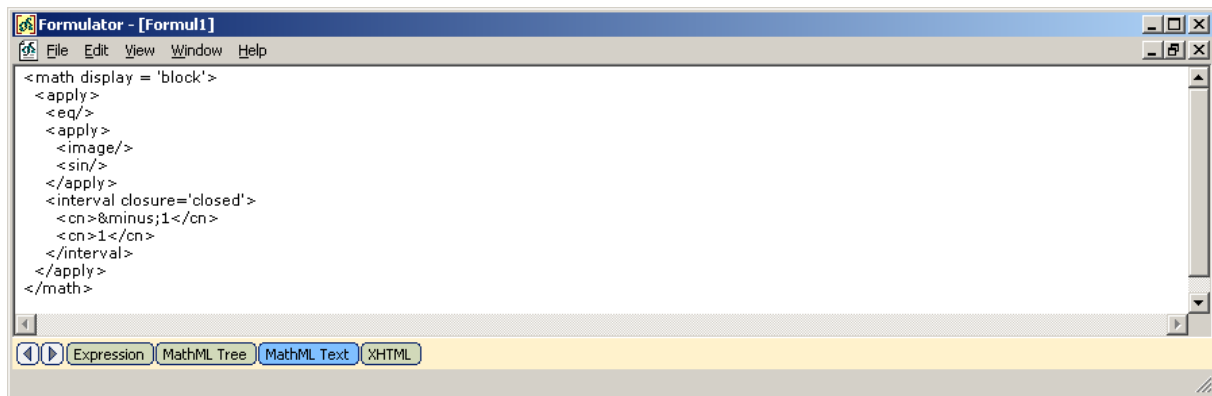






There comes a time when we should recall the difference between Presentation and Content markups. If we just type any text into the first and the second slot, MathML Weaver consider this as if we want to create some presentation for arguments of the “interval” element. But it is not true, since we are going to create content markup number elements there. So, now be careful, MathML Weaver will correctly detect the element of the content markup only if we type into these slots valid numeric constants.



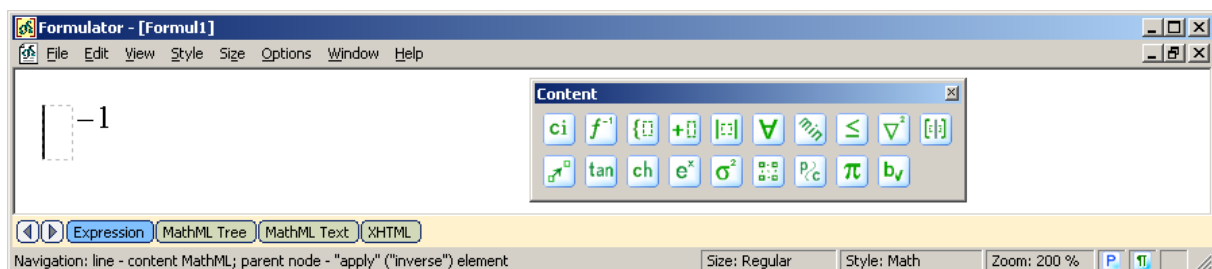
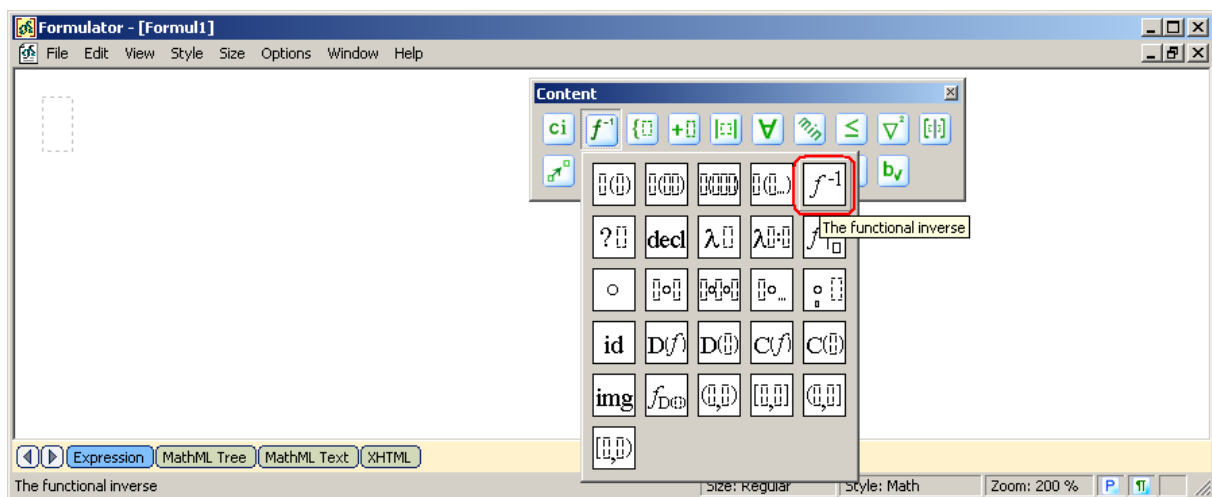


2. Create the inverse element in order to construct a generic expression for the functional inverse of that function (4.4.2.5.2 example from the W3C MathML Recommendation):

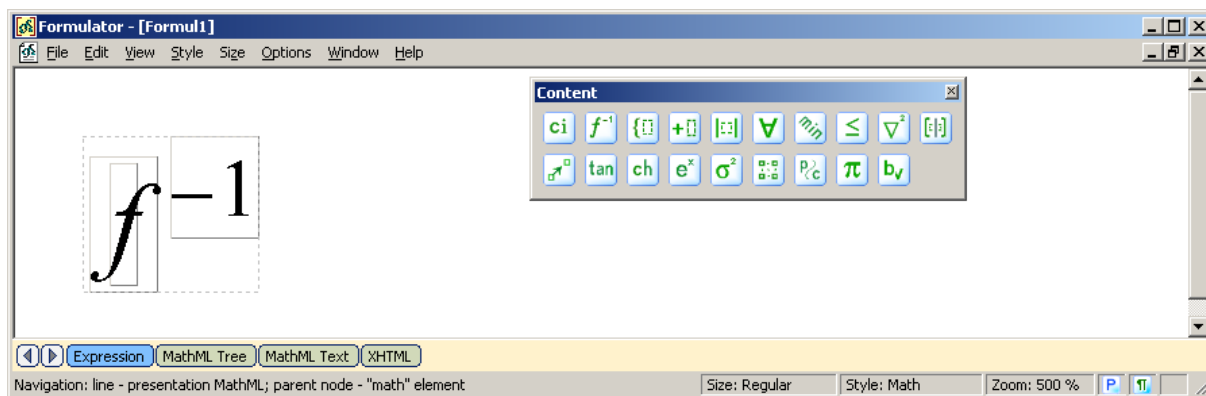
```

<apply>
  <inverse/>
  <ci> f </ci>
</apply>

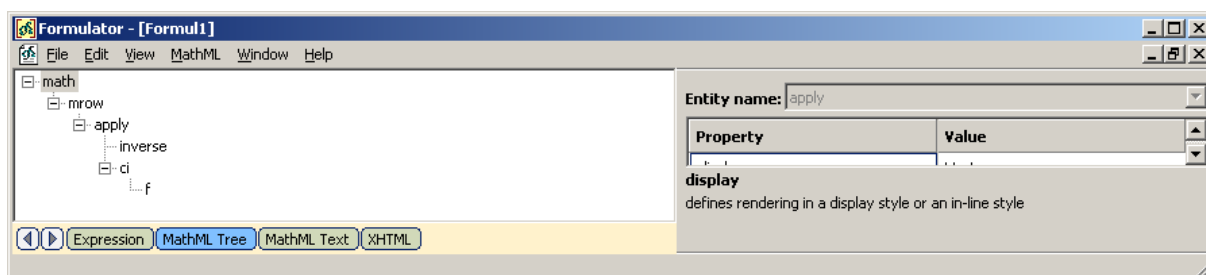
```



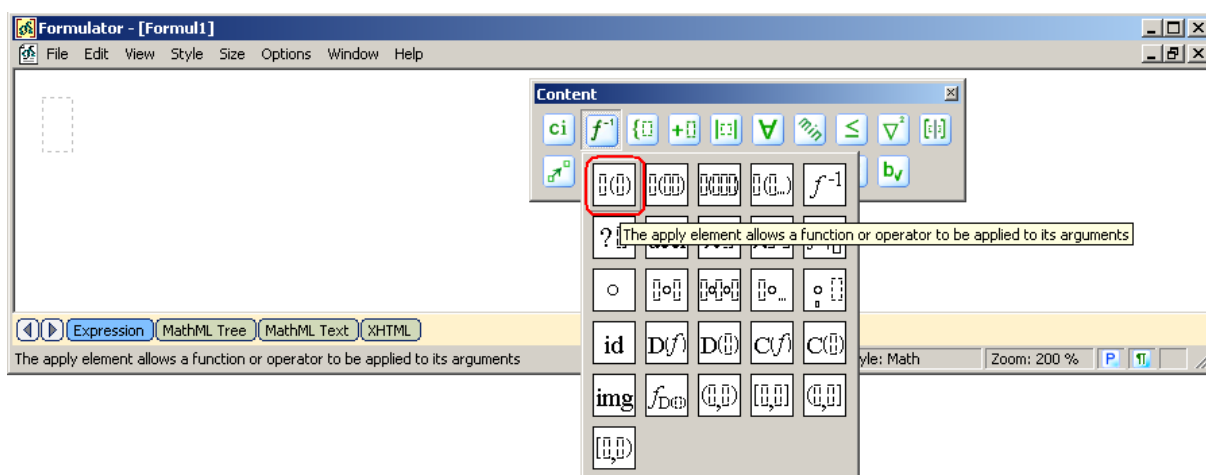
Type “f” into the input slot and see the structure of the MathML tree using nesting feature (Ctrl+Shift+N) and larger zoom (Ctrl+5).

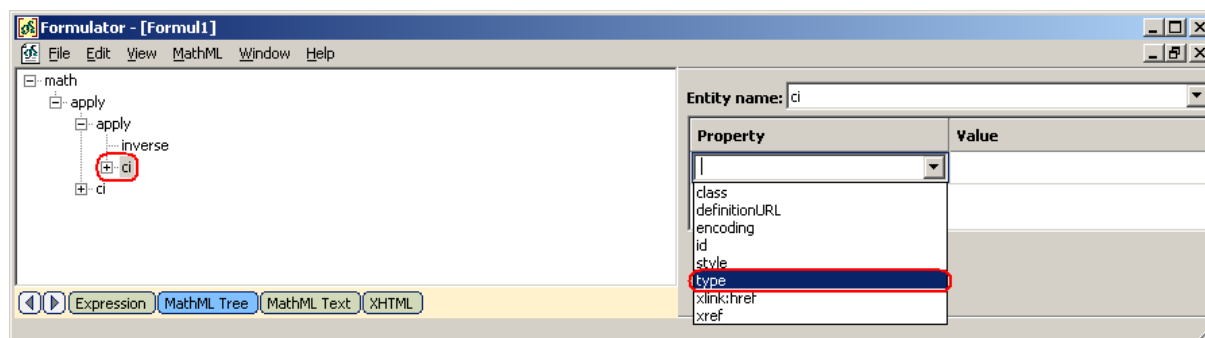
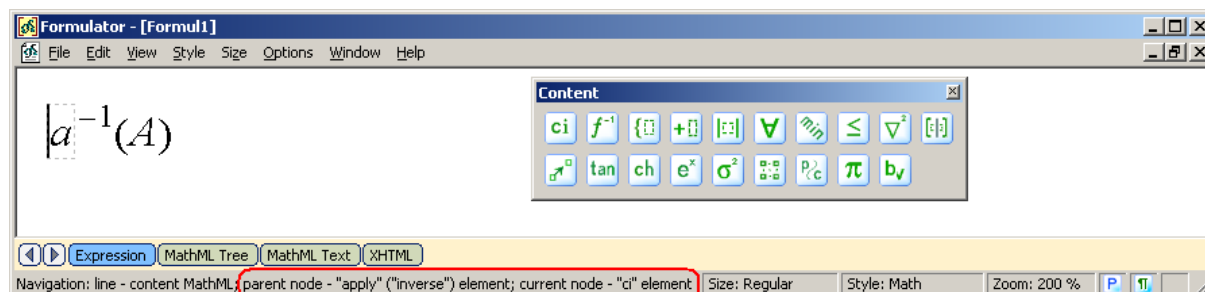
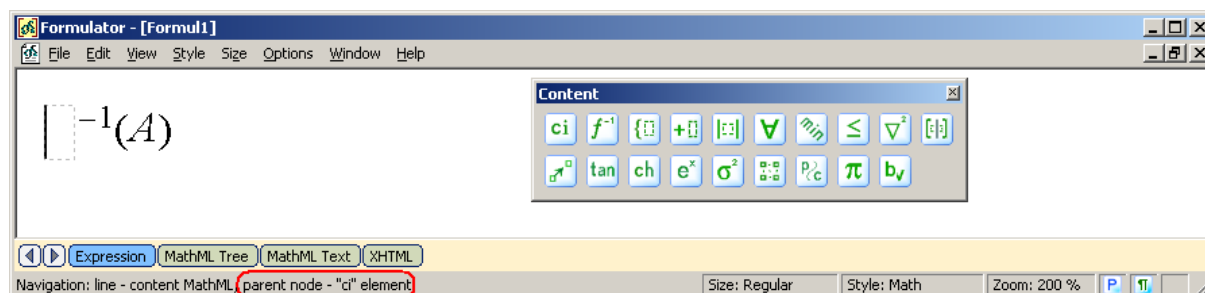
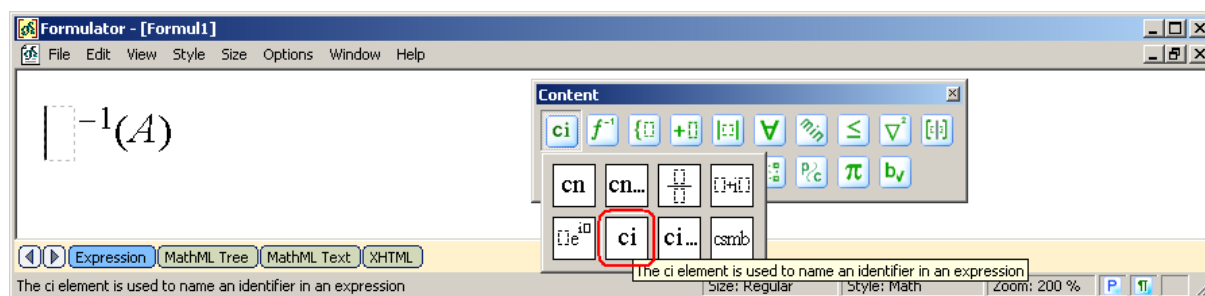
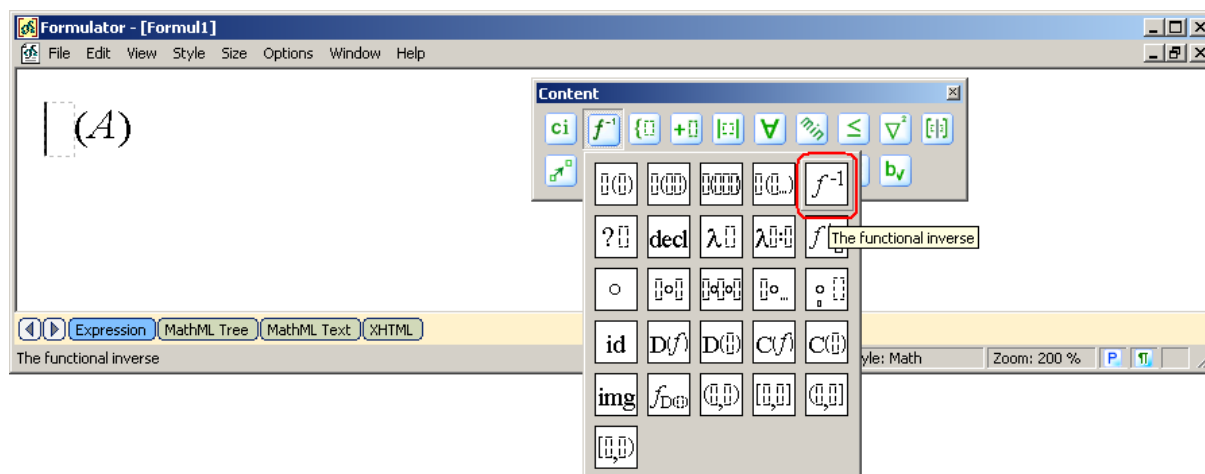


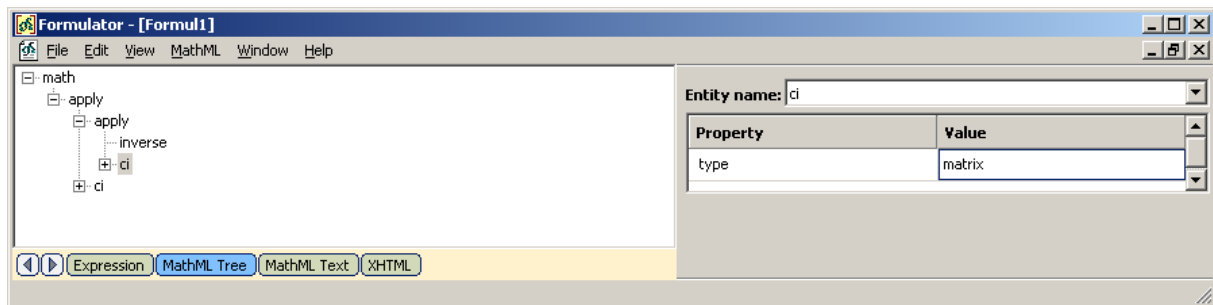
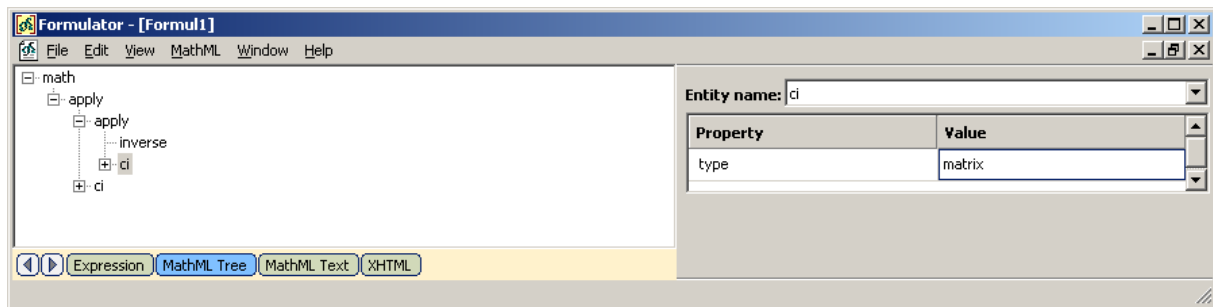
See results on the “MathML Tree” page.



```
<apply>
  <apply><inverse/>
    <ci type="matrix"> a </ci>
  </apply>
  <ci> A </ci>
</apply>
```

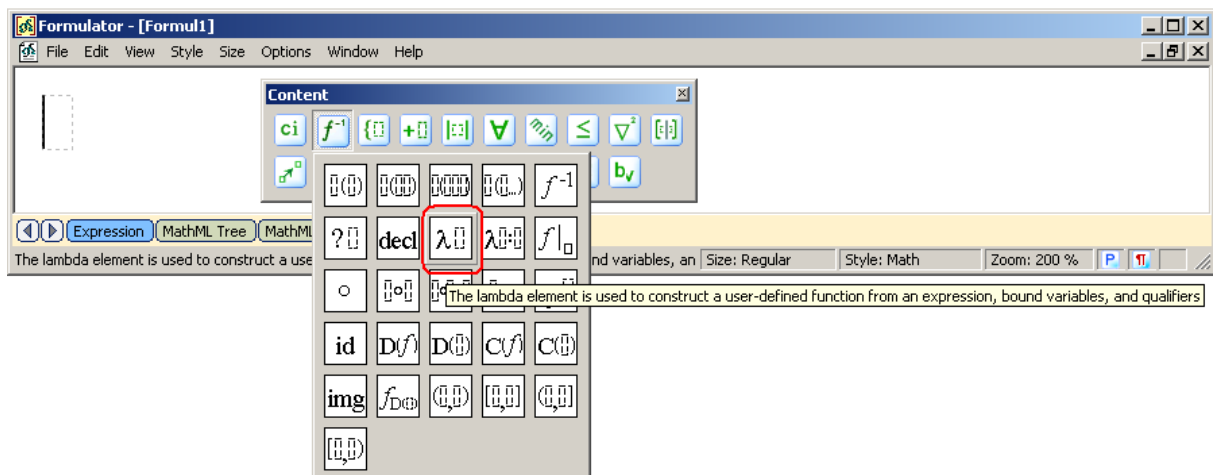




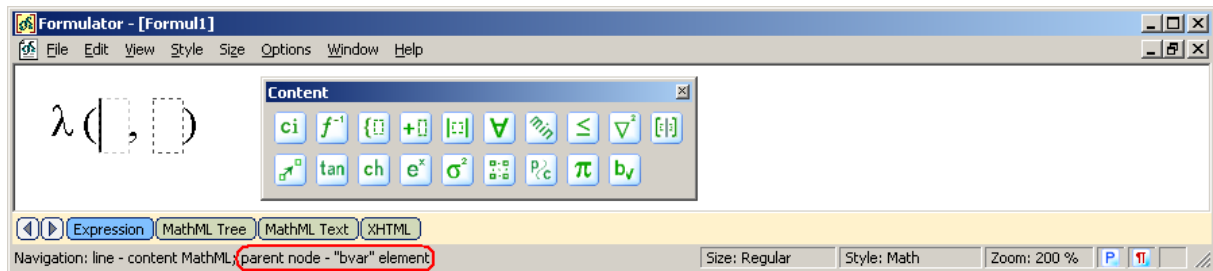


3. Create the lambda element that is used to construct a user-defined function from an expression, bound variables, and qualifiers (4.4.2.9.2 examples from the W3C MathML Recommendation):

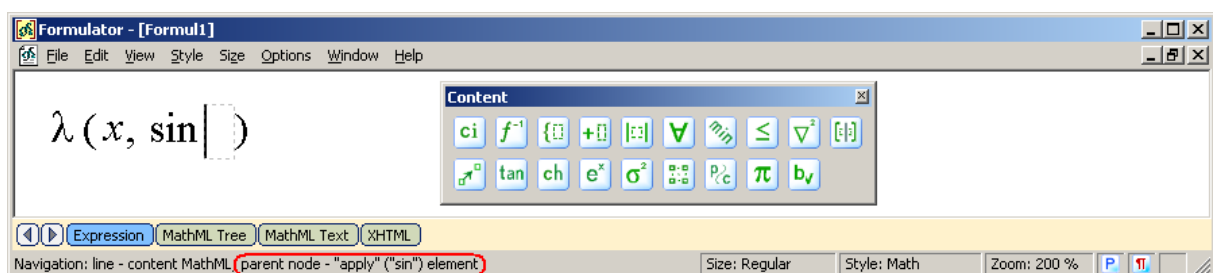
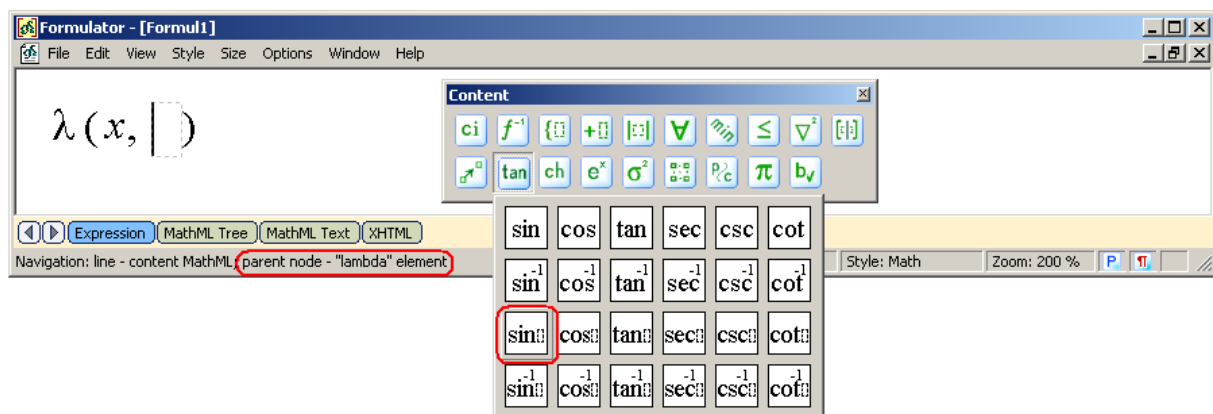
```
<lambda>
  <bvar><ci> x </ci></bvar>
  <apply><sin/>
    <apply>
      <plus/>
        <ci> x </ci>
        <cn> 1 </cn>
      </apply>
    </apply>
  </lambda>
```




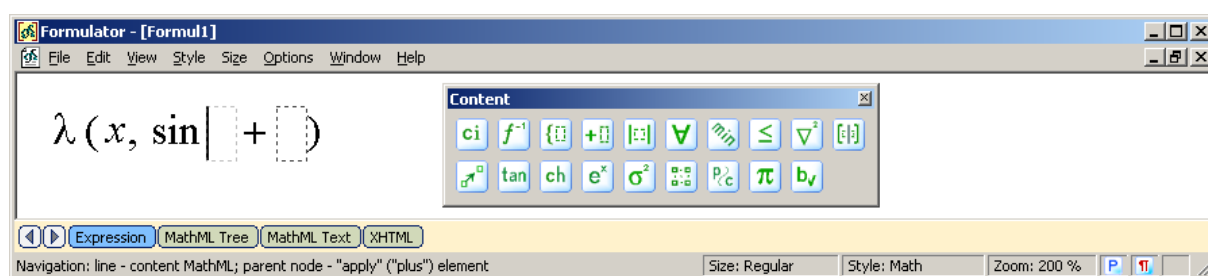
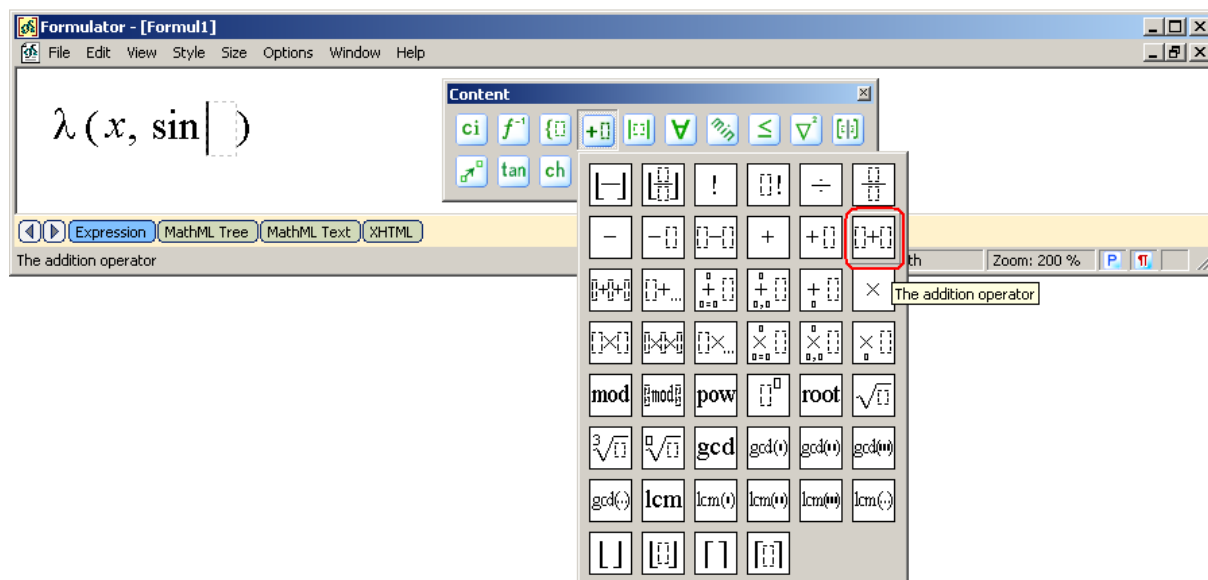
The first input slot is for the “bvar” element; type “x” text into it and it will be detected as a “ci” element when converting to MathML.



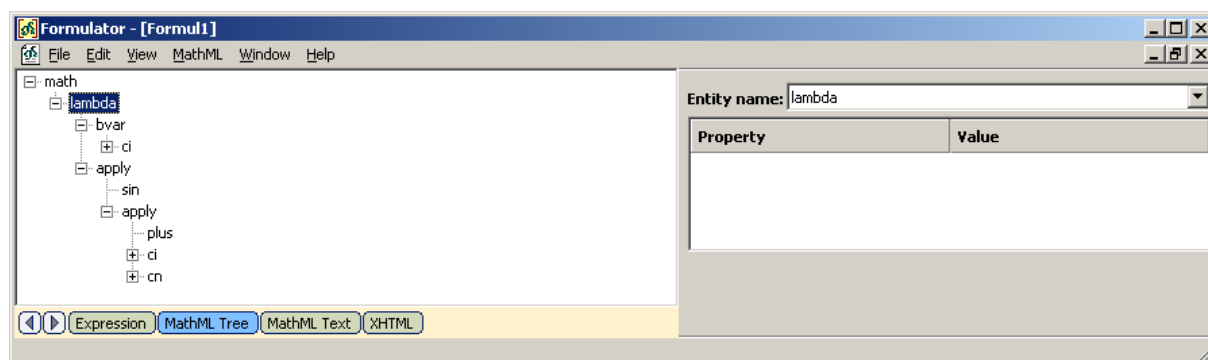
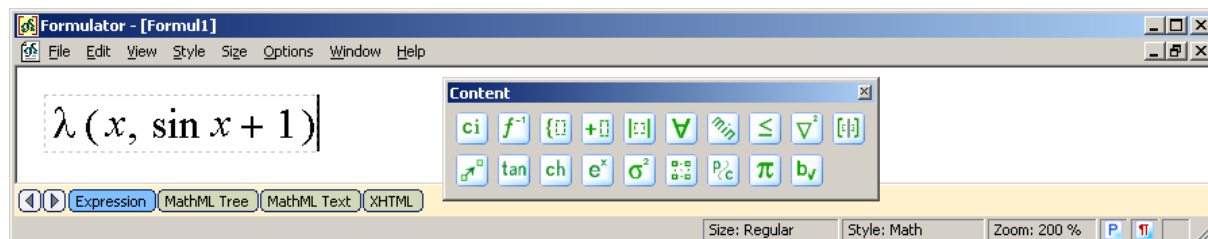
Now navigate to the second input slot (press the Right arrow) and insert a new visual form for editing sin(...) expression.

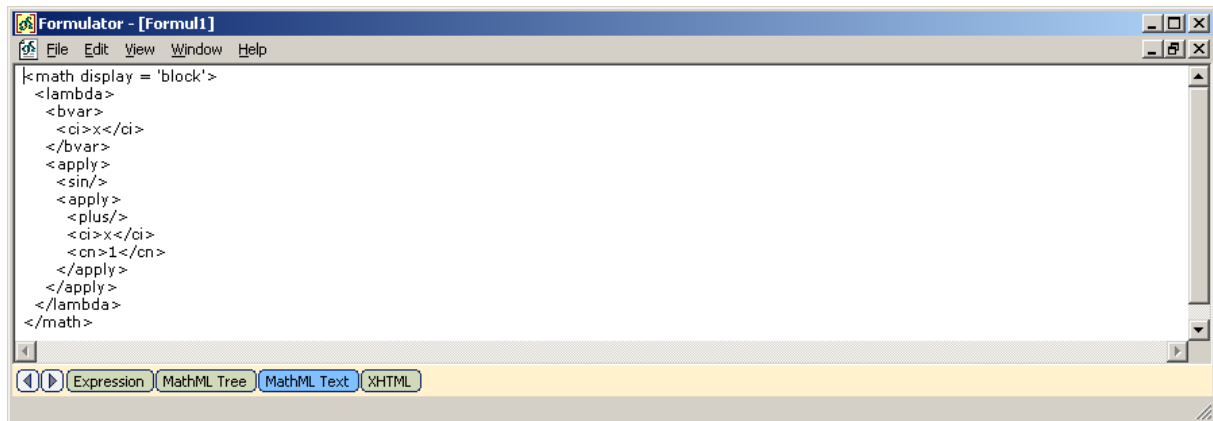


Now insert the “apply” element for the <plus/> operator. Please note that we cannot just type ‘+’ sign, because the current input mode is Presentation MathML (icon  in the right side of the Status Bar).



Type “x” and “1” into input slots. The results are shown on the next two figures.







Basic Content Elements: invisible and transparent elements

This section describes how to work with invisible and transparent MathML nodes: “declare” and “condition” elements of the content markup.

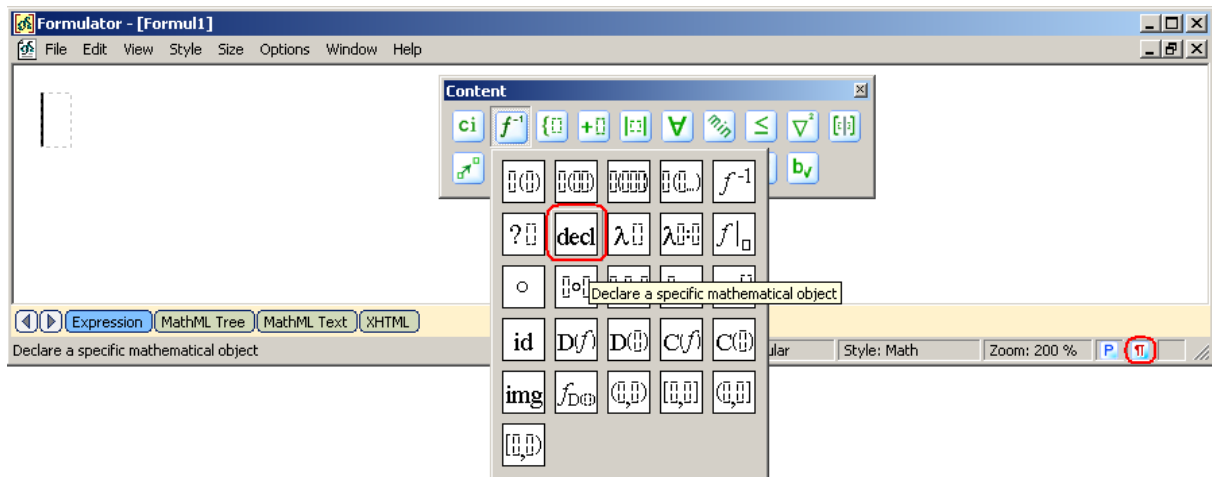
Working with the “declare” element asks for additional actions, because according to the W3C MathML Recommendation it should not be directly rendered. Thus the “declare” element is considered as *invisible* and is rendered in two cases only:


- a) just after insertion of this element from the mathematical toolbar of MathML Weaver (
- b) if the option “show invisible elements” is turned on (the green icon  in the right corner of the Status Bar).

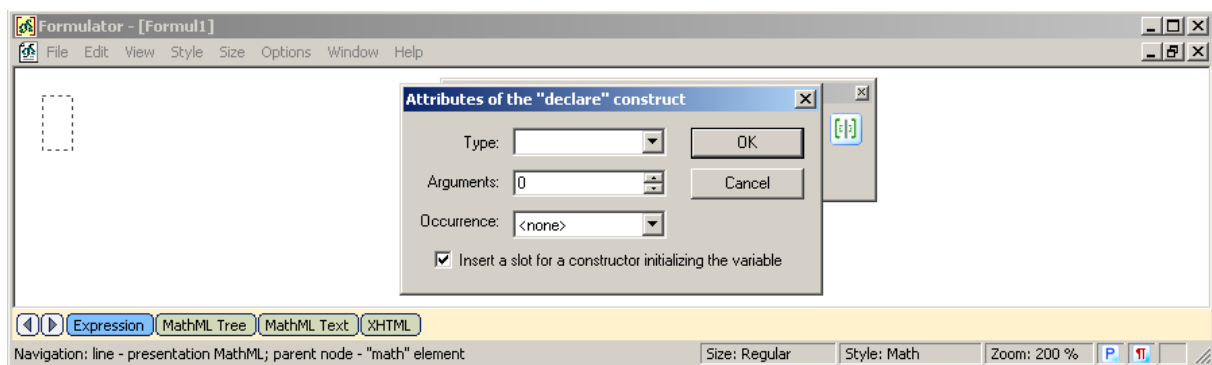
Additionally there is a limitation while using the “declare” element: MathML Weaver cannot hold a document that has no other elements except of the “declare”. This means that when a user tries to save the following example without any additional MathML elements, then an empty document will be created. This known limitation of MathML Weaver relates to the specific rendering requirements of the “declare” element and the point that the presence of this element makes a sense only when other elements of the document refers to this declaration.

1. In order to demonstrate how to work with the “declare” element let’s create a MathML tree for the following example from the W3C MathML Recommendation (4.4.2.8.1)

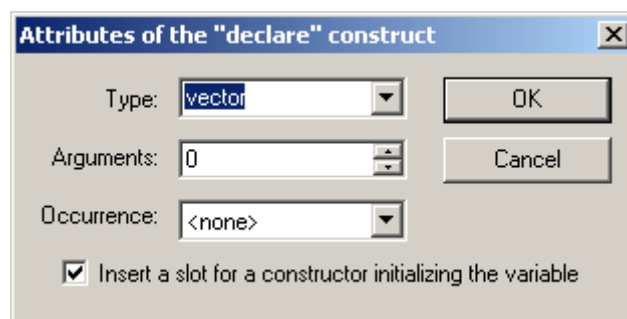
```
<declare type="vector">
  <ci> V </ci>
  <vector>
    <cn> 1 </cn><cn> 2 </cn><cn> 3 </cn>
  </vector>
</declare>
```

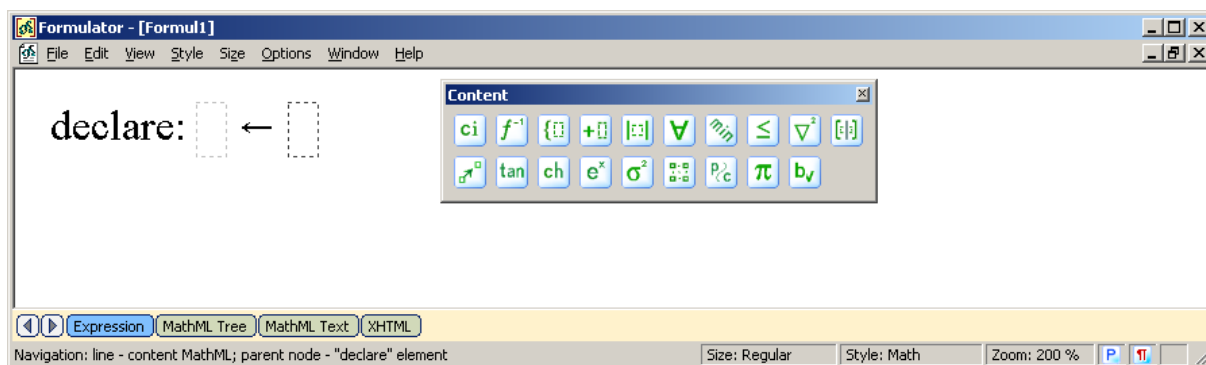


Please note that currently rendering of invisible elements is turned off ().

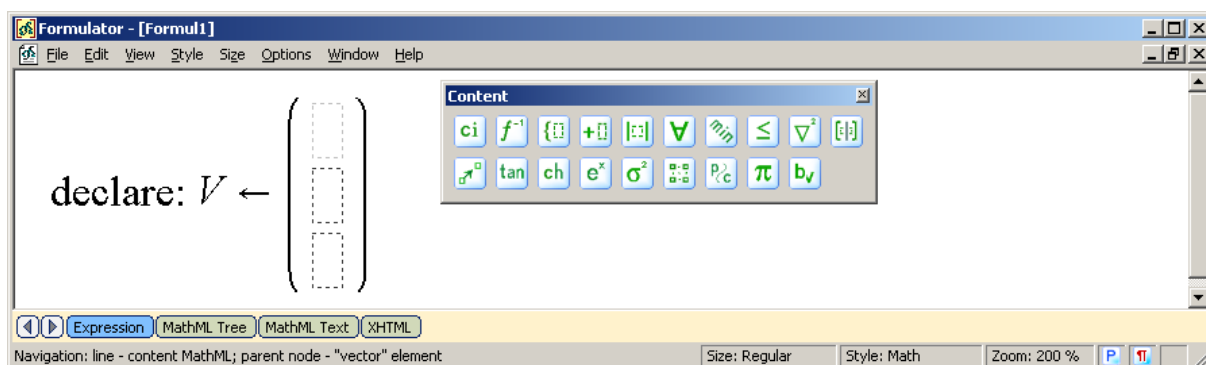
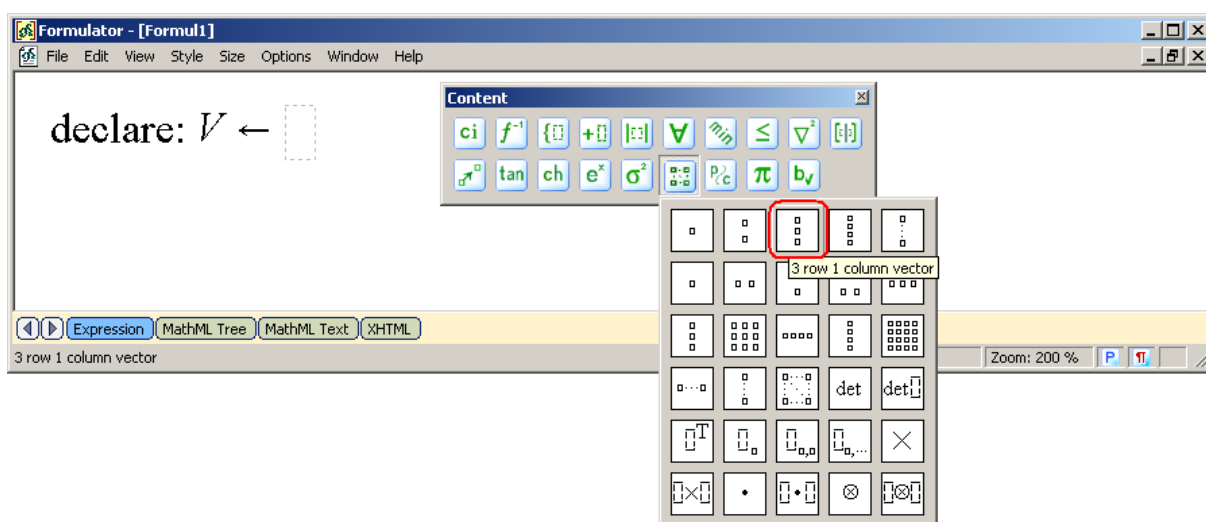


A dialog is shown to initially tune attributes of the “declare” element. Select the “vector” in the “Type” drop-down list. The next two attributes (“nargs”, “occurrence”) can be leave as they are, without editing. The check-box to the bottom of the dialog asks whether the “declare” element contains a constructor initializing the variable.

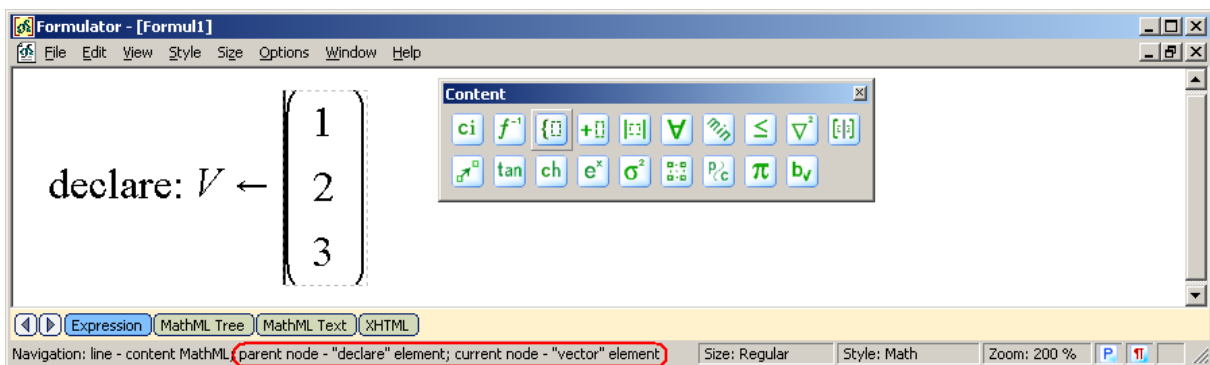




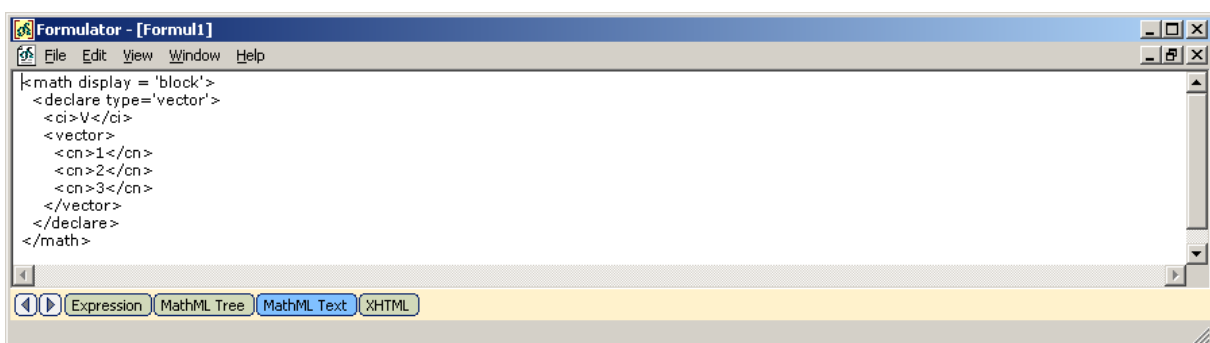
Type “V” into the first input slot; place the cursor into the second input slot and press the button for the “vector” element insertion.



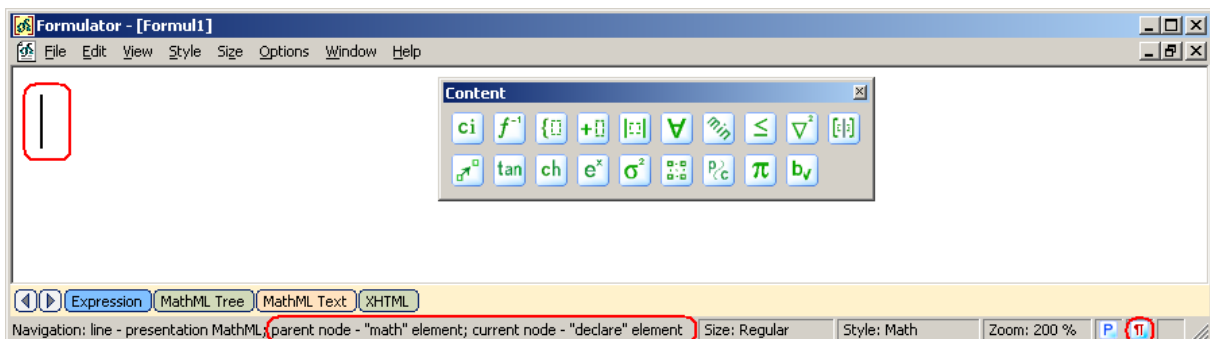
Type values of the vector into three input slots.





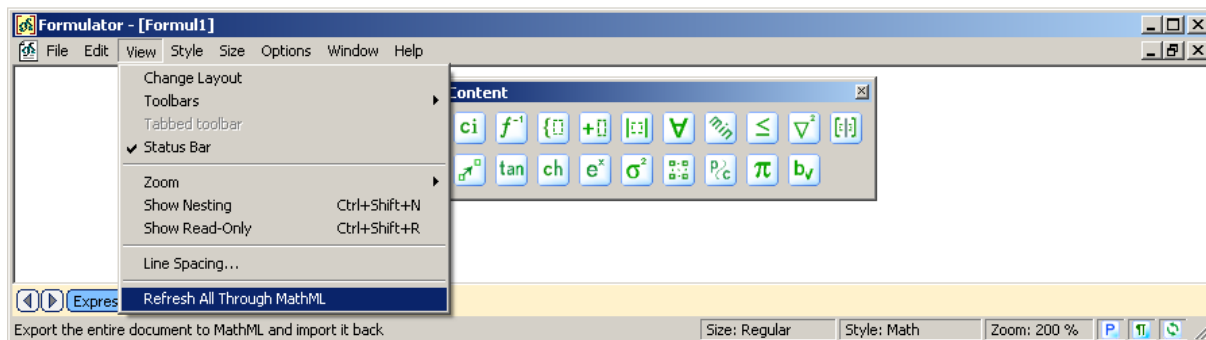
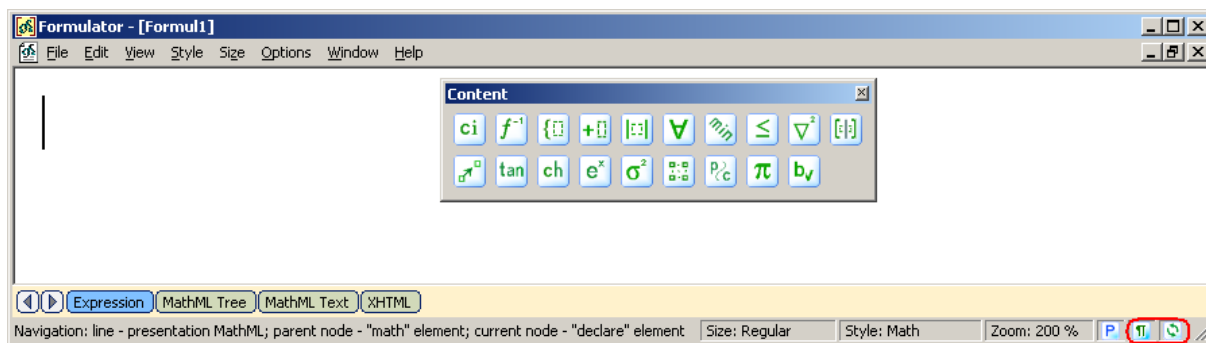
See the results.



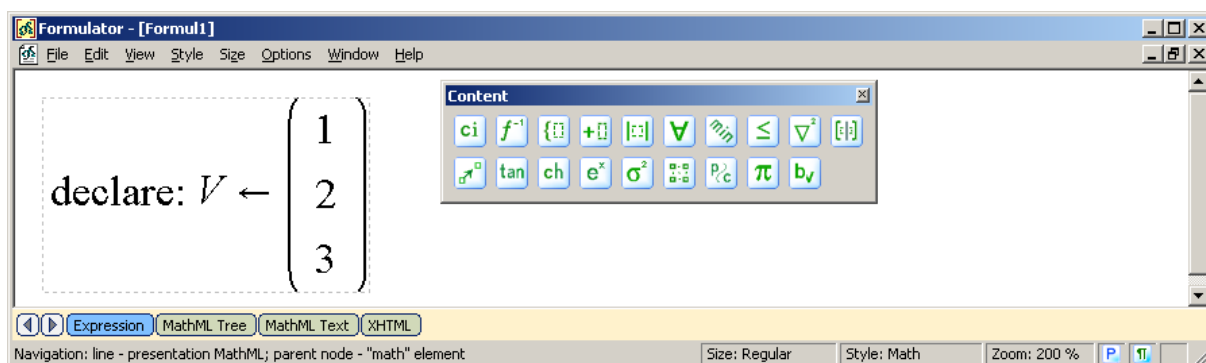
Now do a inessential editing of the MathML text (for example, just press a space somewhere between tags) and get back to the “Expression” page. We see an empty body of the document, because the “declare” element is not rendered normally and the option “show invisible elements” is turned off currently. About presence of the just created “declare” element indicates only the navigation information bar and absence of the dashed line around an empty input slot (it is shown only when the line of a document contains no elements).



Check the option “show invisible elements” from the “Display Content MathML Elements” submenu of the “Option” menu. We still can’t see the “declare” element, but the icon  suggests that the action was successful and the icon  prompts to the need of refreshing the document through MathML (the “Refresh All Through MathML” command from the “View” menu).

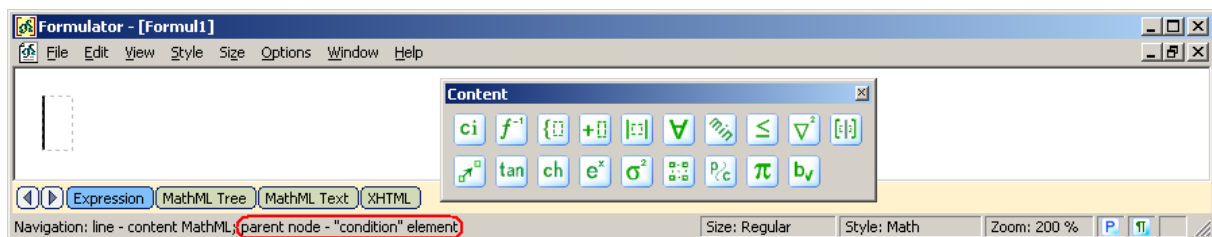
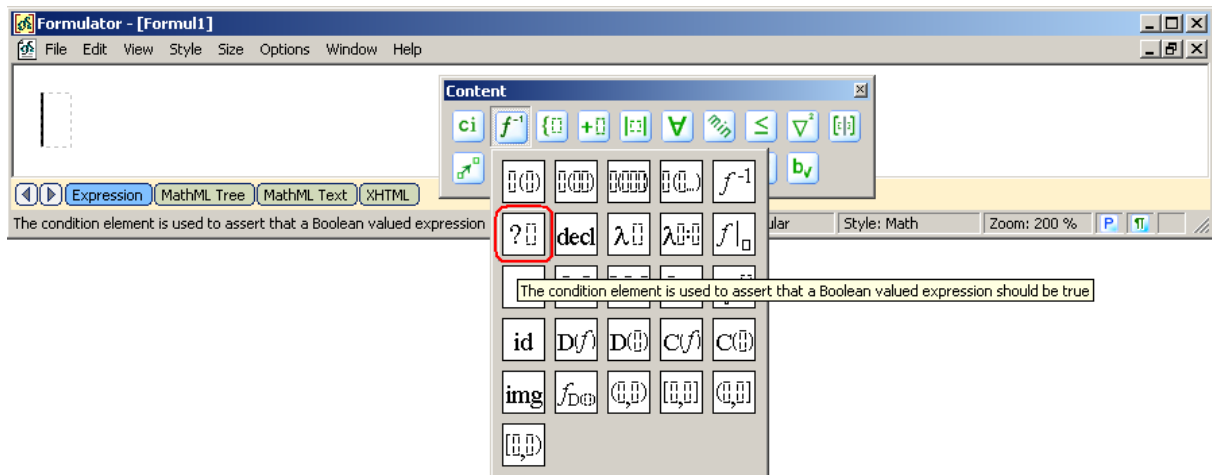


After refreshing the document rendering we see the “declare” element again, because the mode of invisible elements rendering is now turned on.



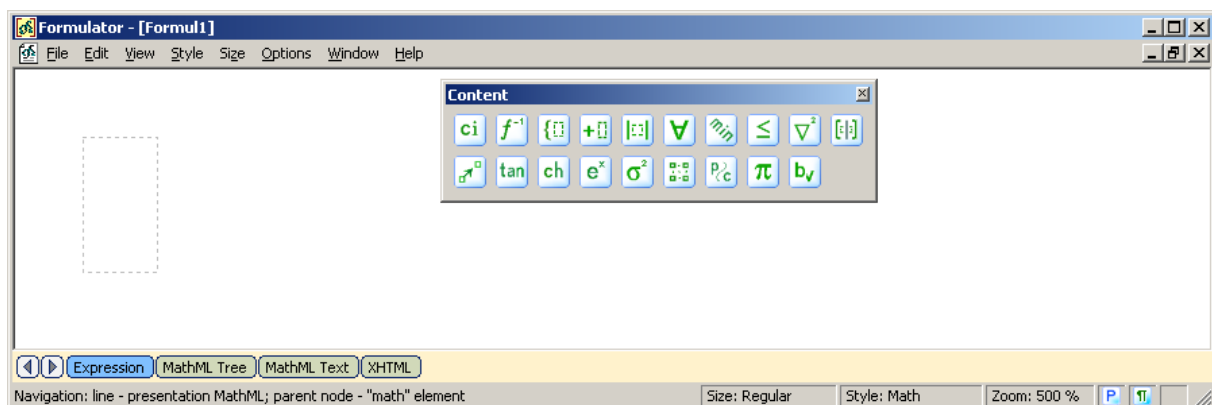
2. Create the “condition” element (4.4.2.7.2 examples from the W3C MathML Recommendation) to see how initially invisible (transparent) elements behave.

```
<condition>
  <apply><in/><ci> x </ci><ci type="set"> A </ci></apply>
</condition>
```

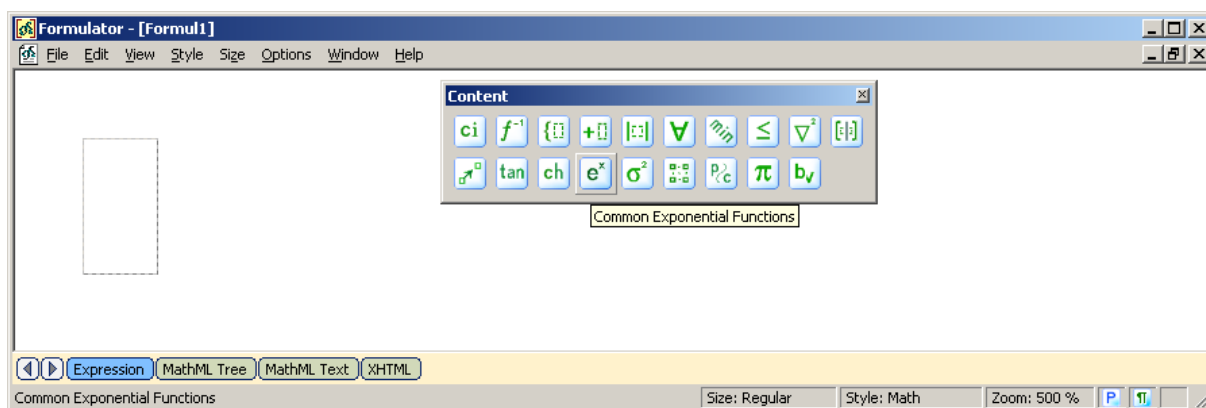


At the first glance, the rendering of the document is not changed, but the navigation information suggests that the "condition" element is successfully inserted and it is not rendered just because there is no presentation for it; the presentation of the "condition" element coincides with its contents rendering.

The changed structure of the document can be viewed using the "nesting view" feature. Compare view of the document before insertion of the "condition" element (dashed line around the empty input slot):

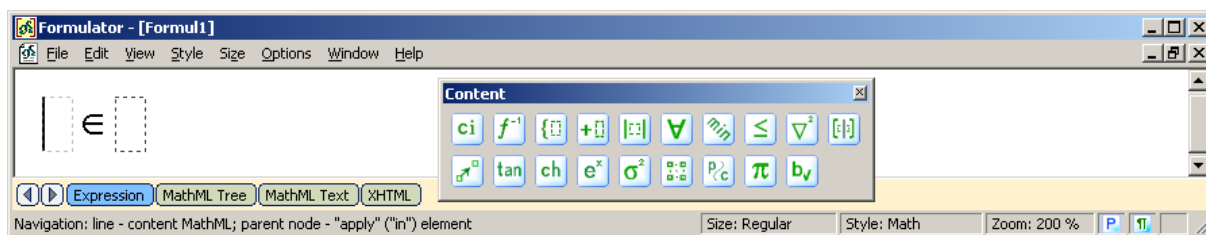
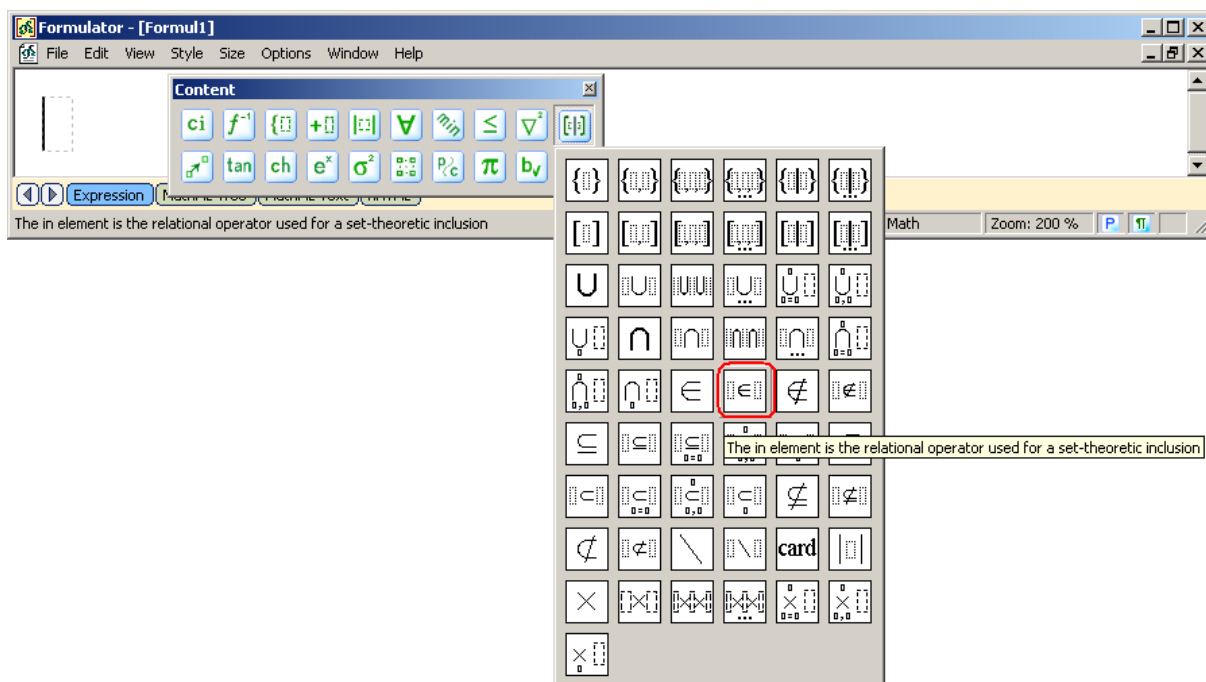


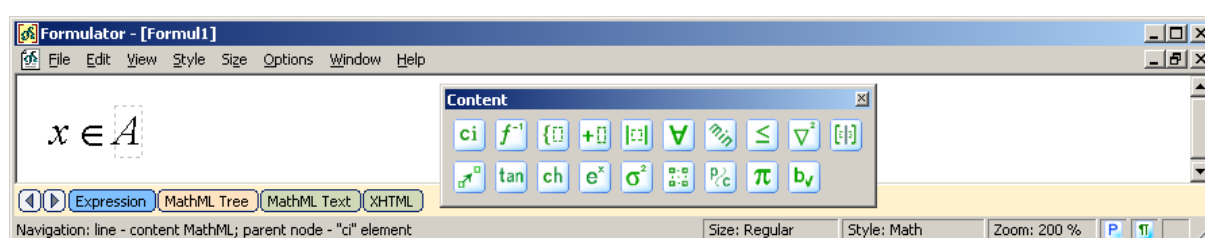
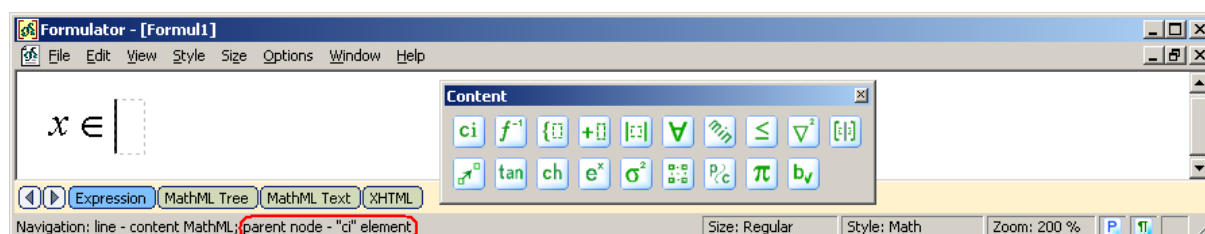
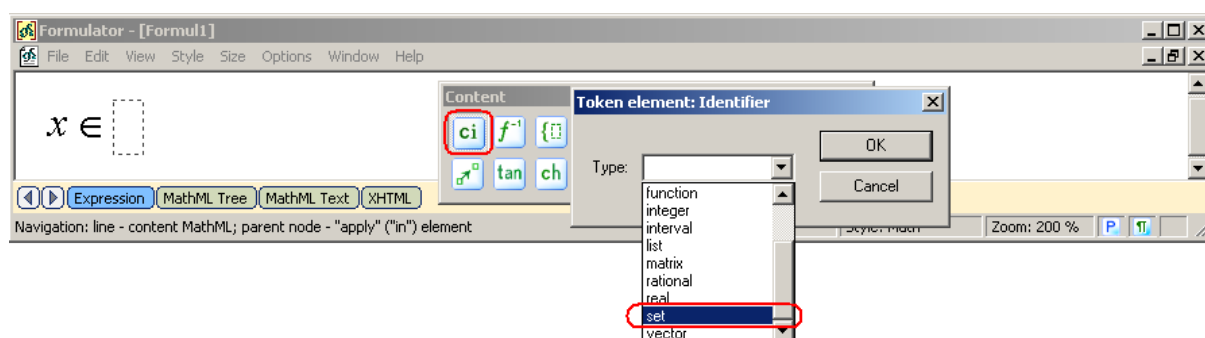
and after it (solid line over the dashed line around the empty input slot):



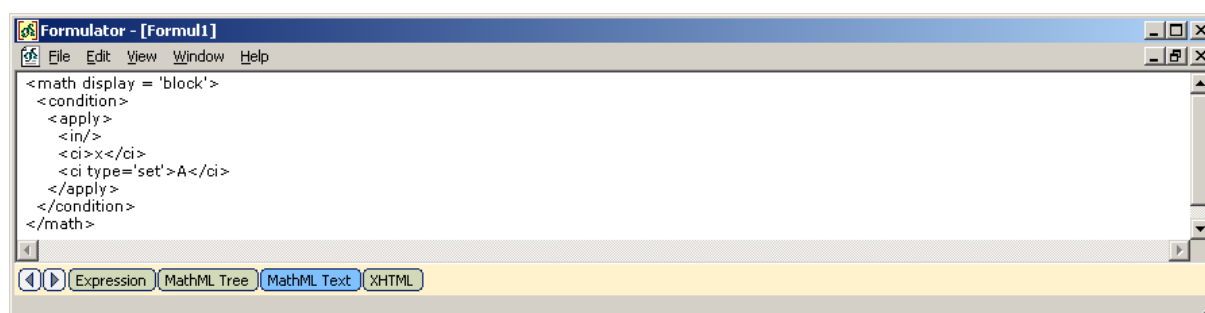
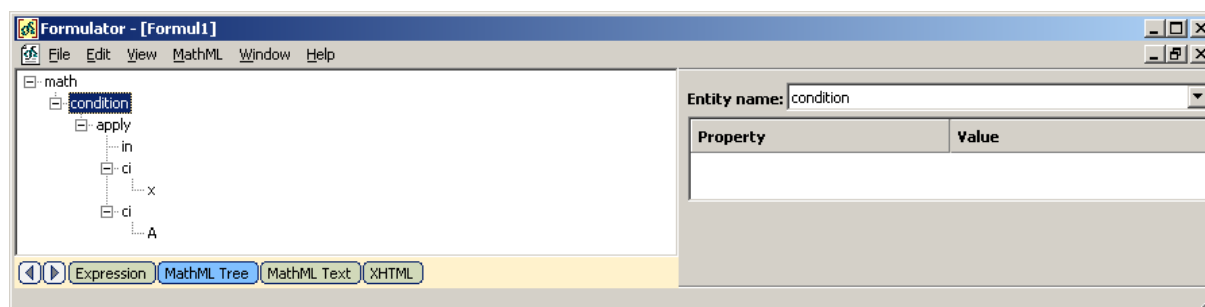
This solid line suggests to the thought that there is an internal frame element to hold a newly created element of the content markup.

Now let's back to our example and create the "apply" element inside of the "condition" element.





See the results

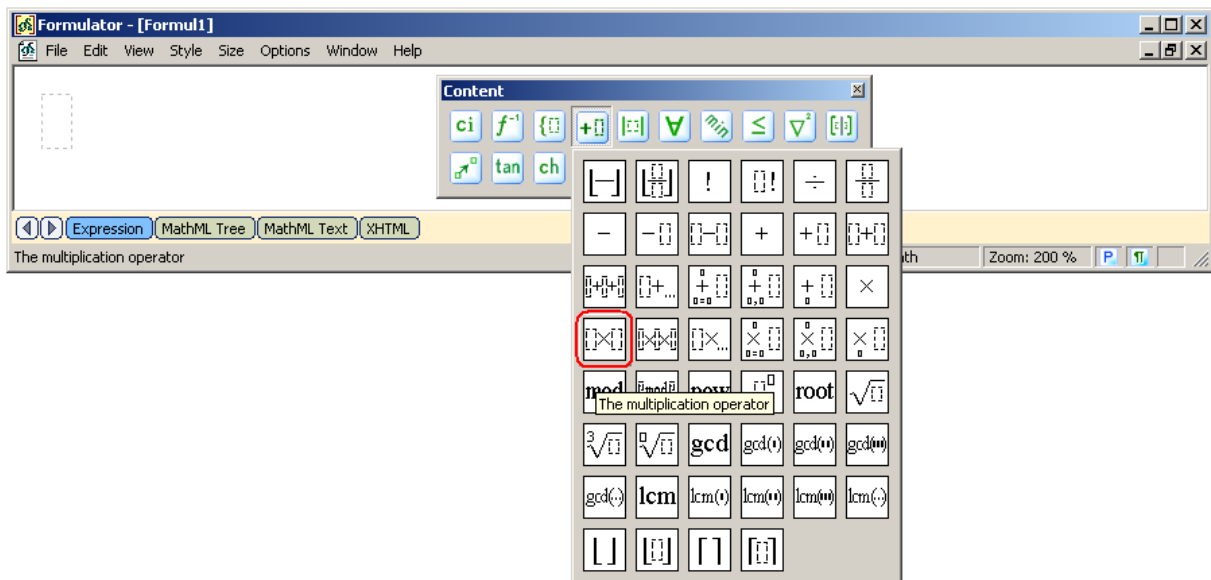


Arithmetic, Algebra, Logic and Relations

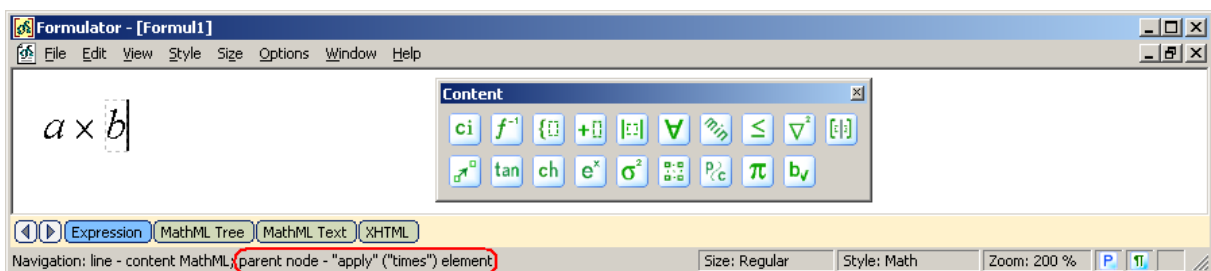
1. Example 4.4.3.9.2 from the W3C MathML Recommendation).

```
<apply>
  <times/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

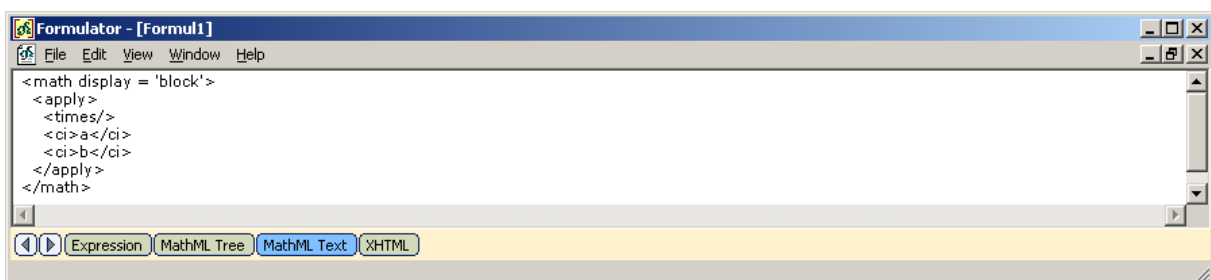
Press the button on the mathematical toolbar for arithmetic operators or switch the input mode to the Content Markup and just press the '*' character.



Type 'a', press the Right arrow, type 'b'.

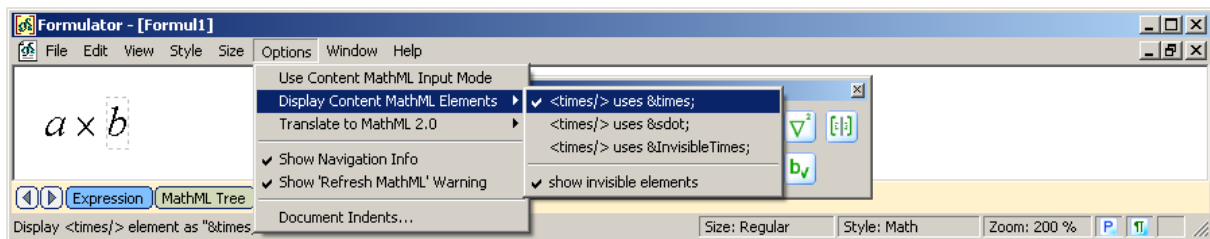



See the results

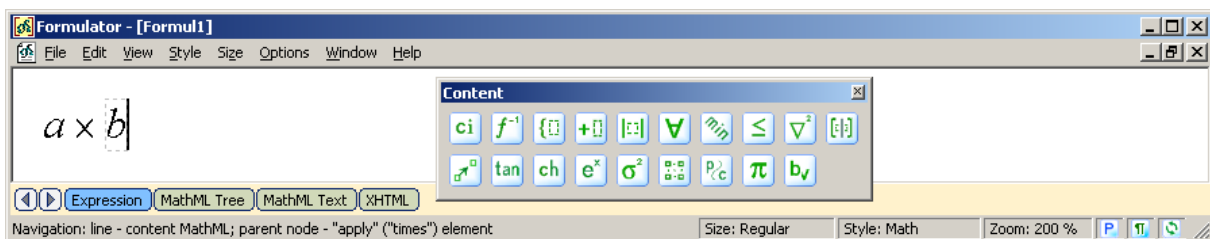


Now get back to the “Expression” page and see how to change the appearance of the `<times/>` element.

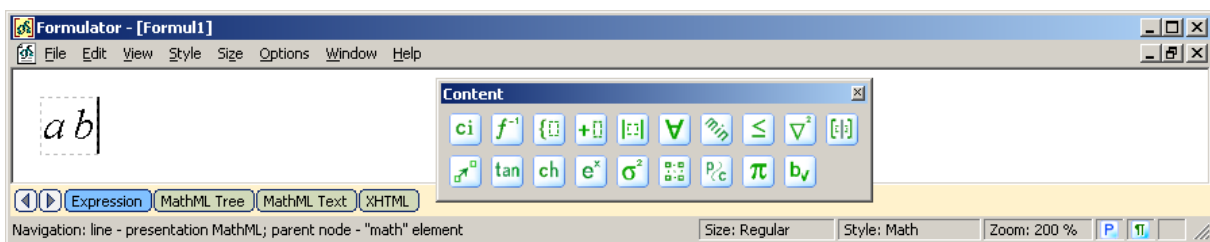
The next figures shows which item of menu controls the rendering option of the `<times/>` element.



If we select another option (“⁢”) the appearance of the formula is not changed, but the icon  prompts to the need of refreshing the document through MathML (the “Refresh All Through MathML” command from the “View” menu).

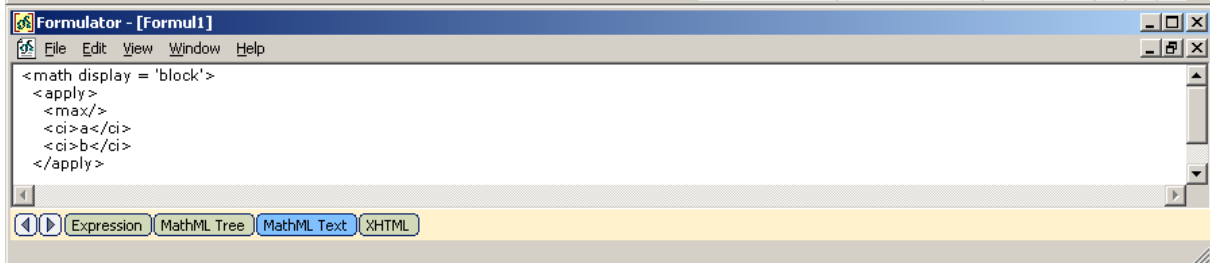
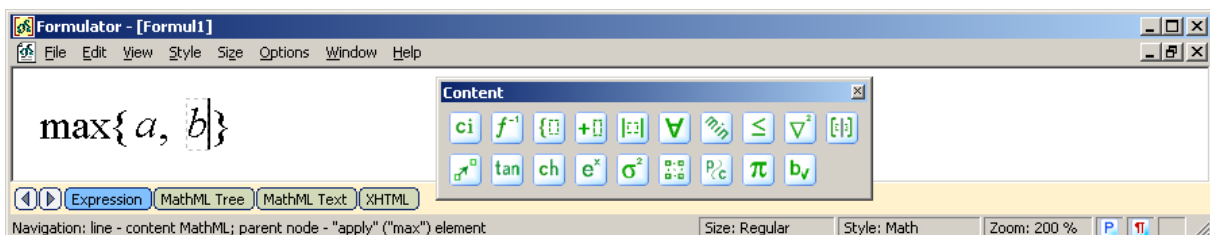
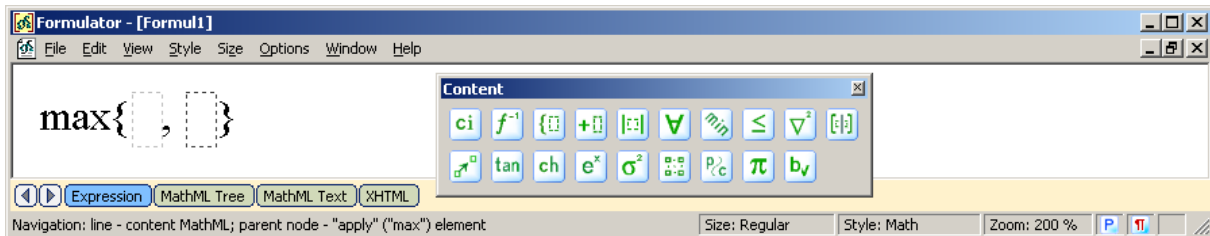
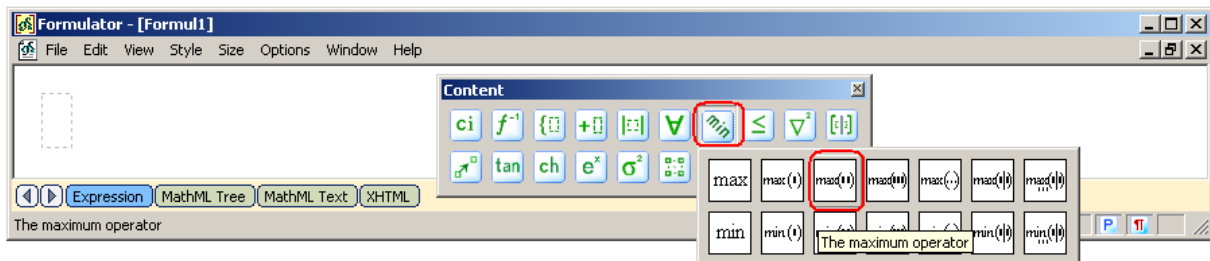


After refreshing the formula is rendered as if instead of the `<times/>` element is used the presentation element `<mo>&InvisibleTimes</mo>`.



2. Example 4.4.3.4.2 from the W3C MathML Recommendation): Maximum and minimum.

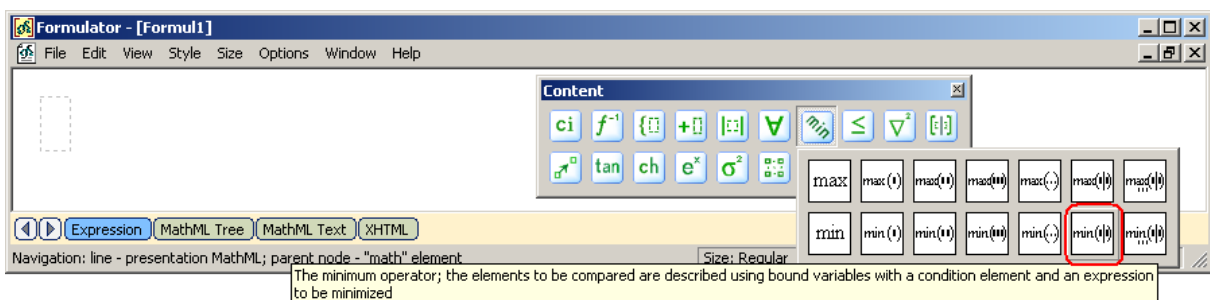
```
<apply>
  <max/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

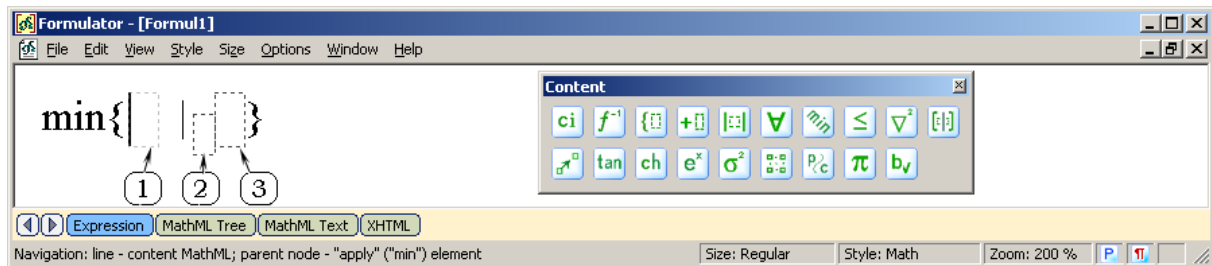


```

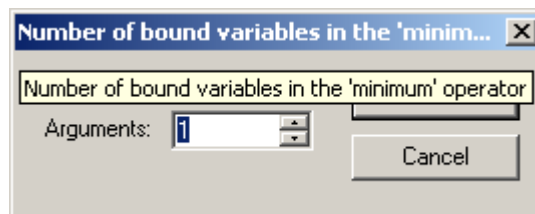
<apply>
  <min/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><notin/><ci> x </ci><ci type="set"> B </ci></apply>
  </condition>
  <apply>
    <power/>
    <ci> x </ci>
    <cn> 2 </cn>
  </apply>
</apply>

```



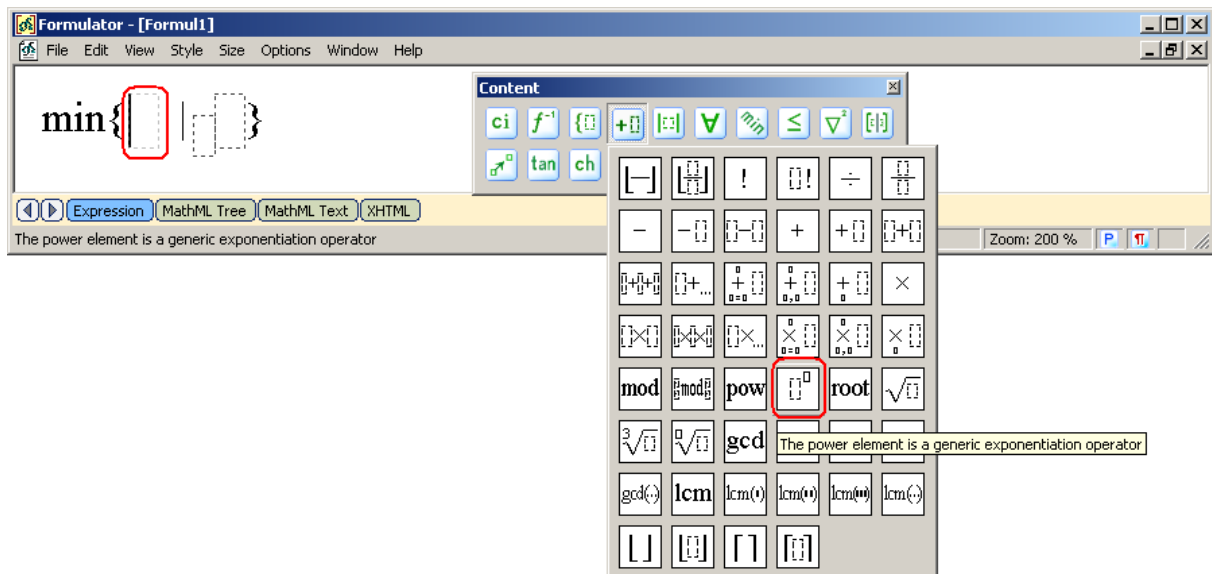


Here (1) is an expression to be minimized; (2) is an input slot for one bound variable (the “bvar” element); (3) a condition element. There is also a button for many bounded variables ($\min_{i=1}^n$) that shows the following dialog:

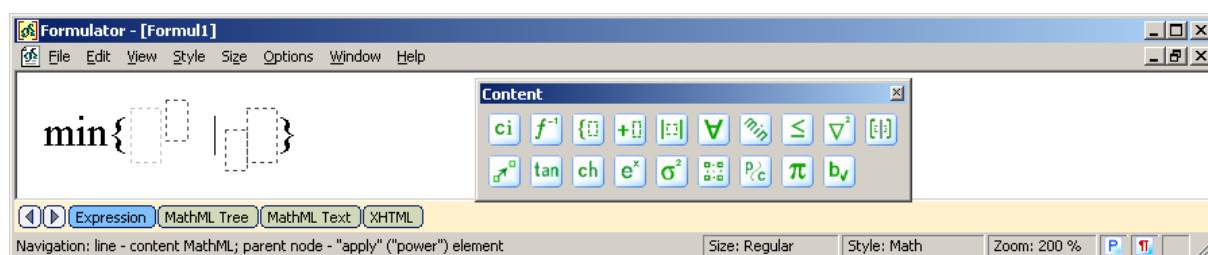


This approach is common for all the cases when qualifier elements of the MathML content markup are needed.

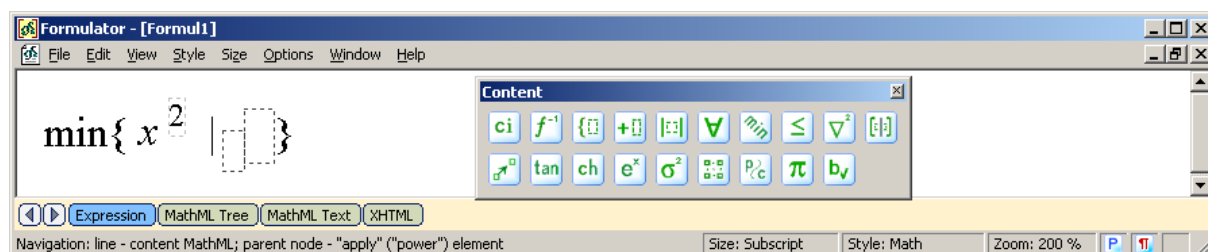
Please note that according to the W3C MathML Recommendation the “bvar” element should not be rendered in such a content markup element, so we consider it as an invisible (see the section “Basic Content Elements: invisible and transparent elements” for detailed discussion). This means that the corresponding input slot is available only after insertion of the “min” element, because currently rendering of invisible elements is turned off (⚙️).



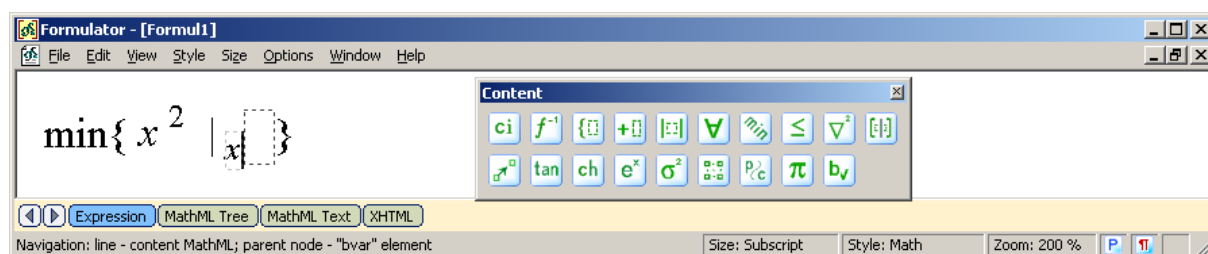
Insert the “apply” element with the <power/> operation.



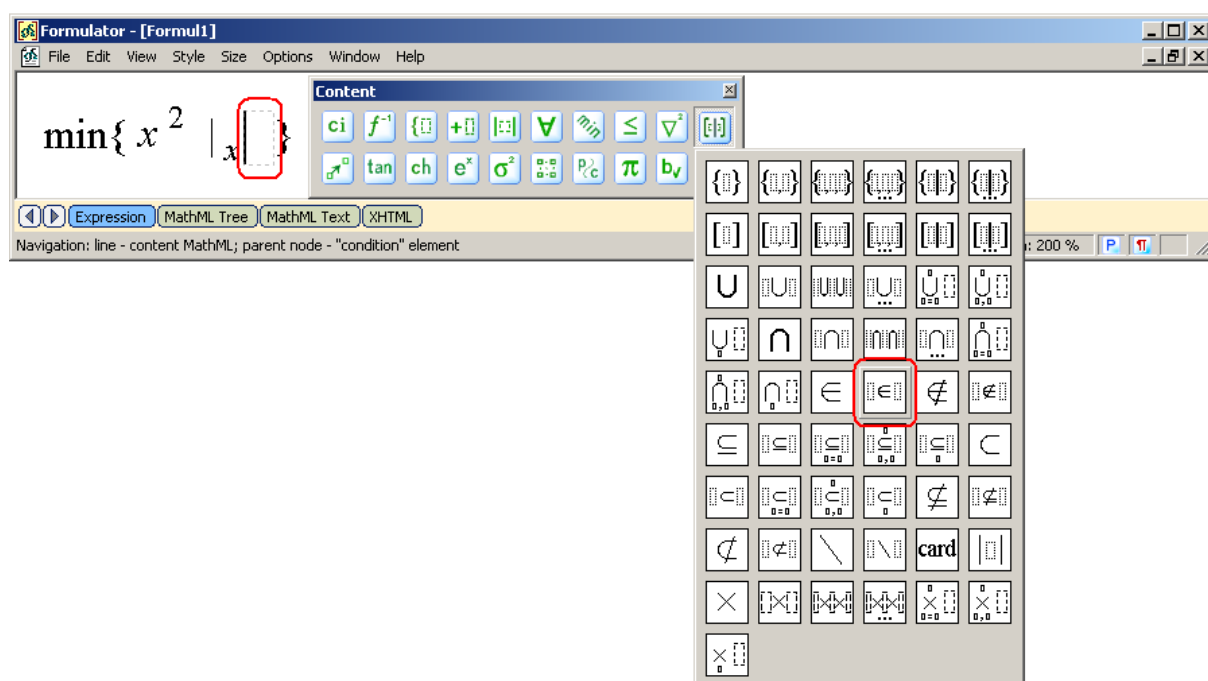
Type “x”, “2”.

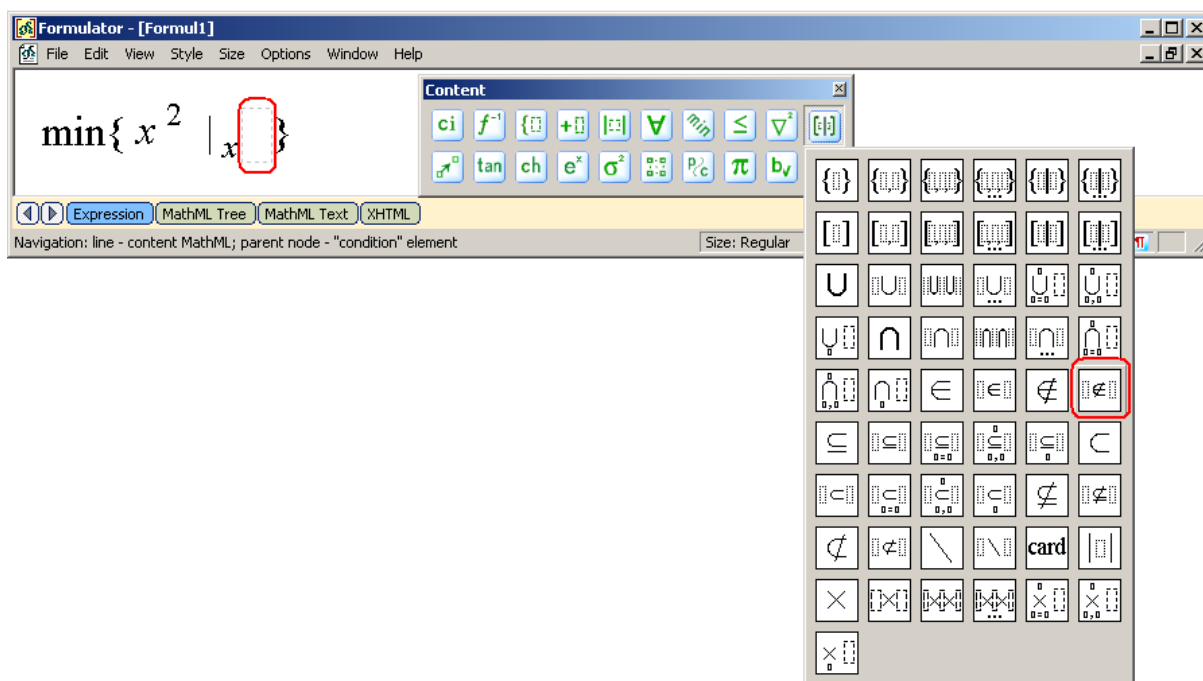


Navigate to the “bar” input slot and type a name of the bounded variable (“x”).

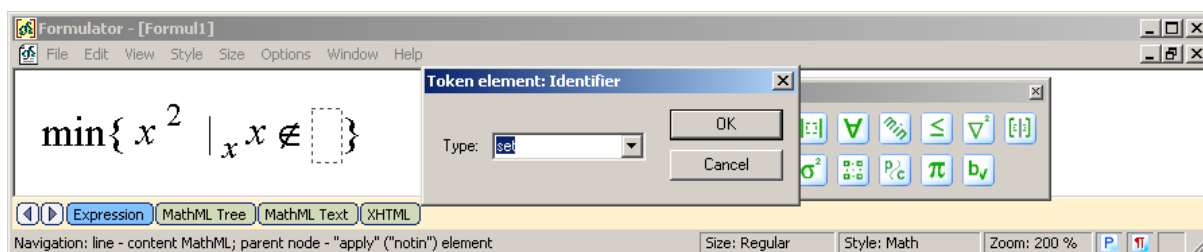
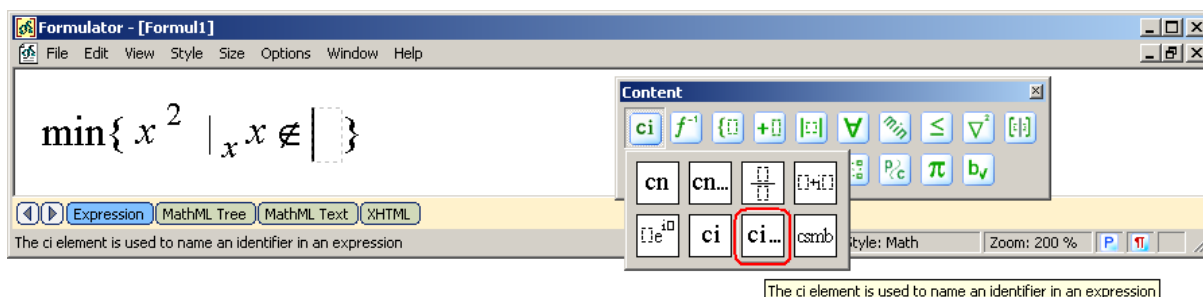


Create an expression for the “condition” element.

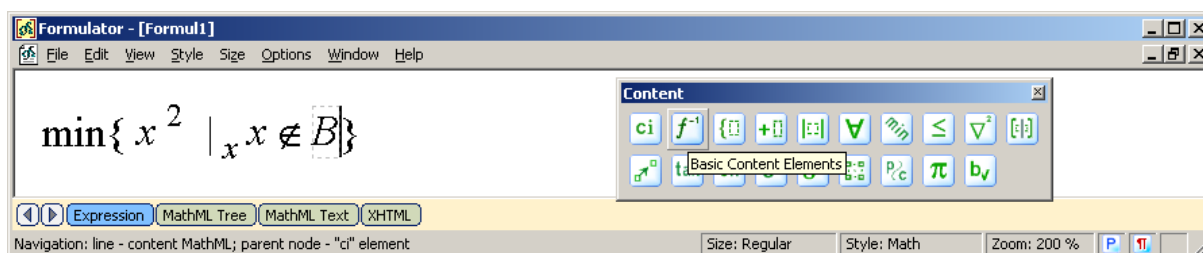


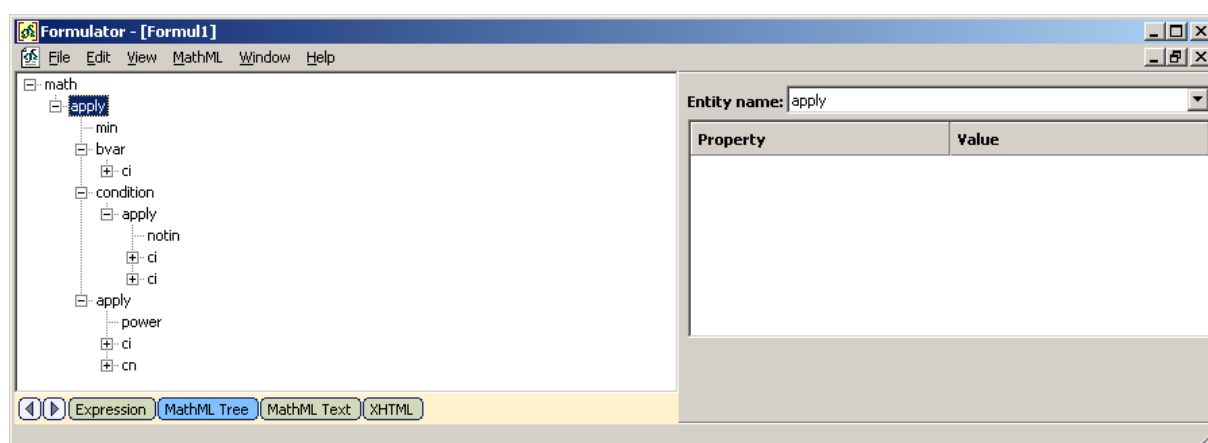
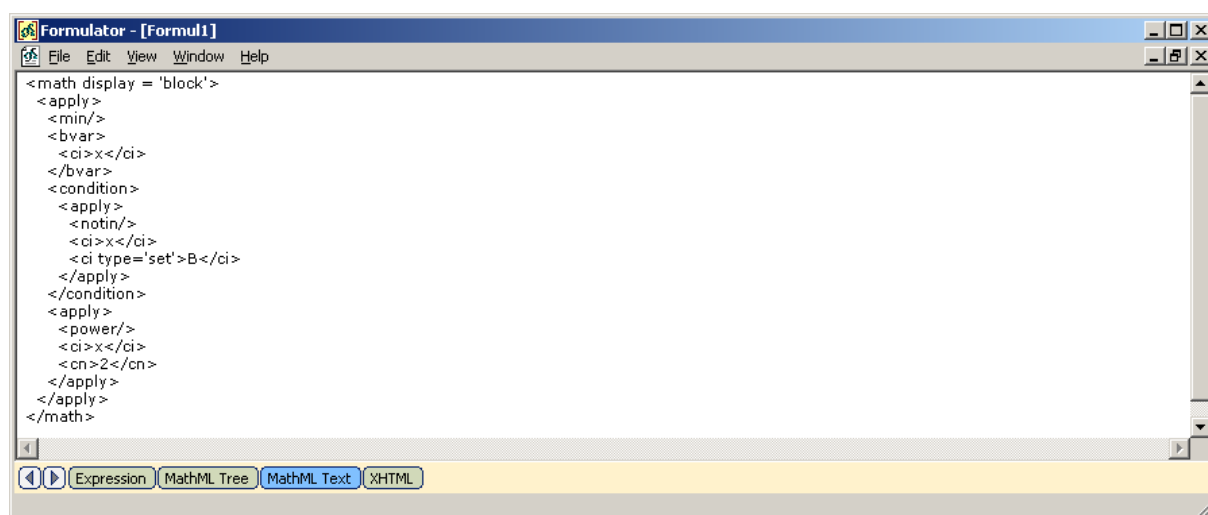


Type “x” as a value of the first argument of the “apply” element in the condition; create the “ci” element of the “set” type as the second argument of the “apply”.

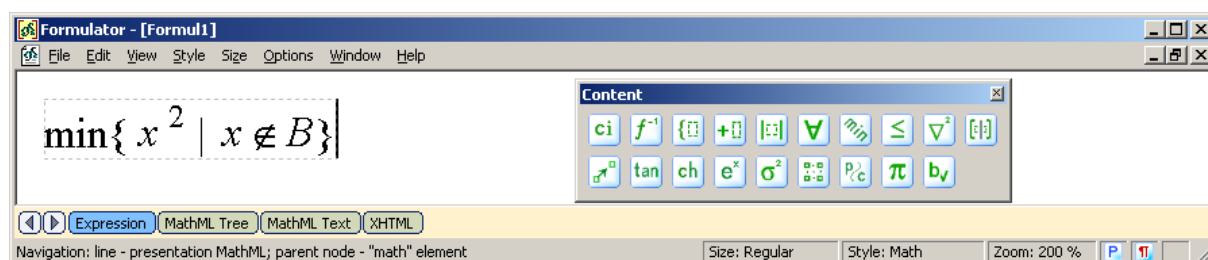
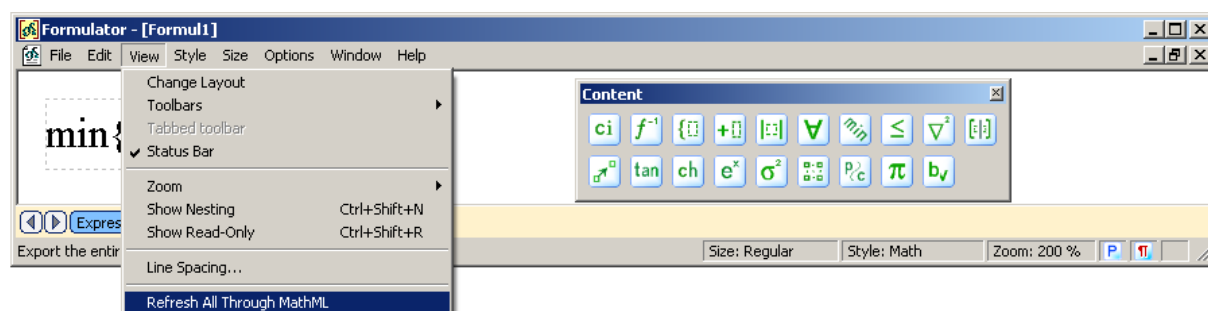


See results on different pages of Formulator 3.9 MathML Weaver.

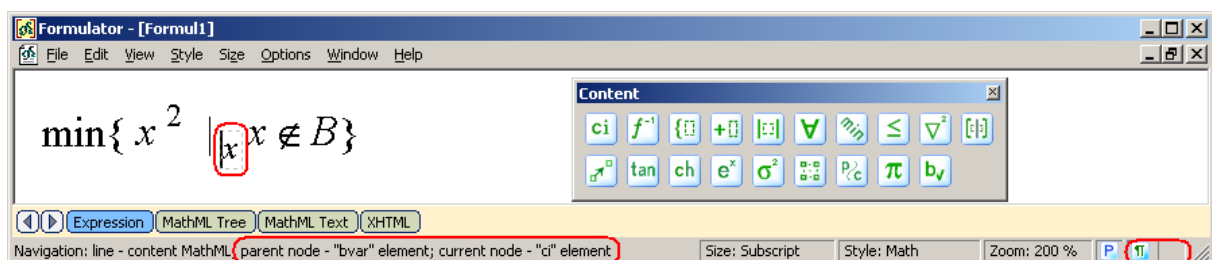
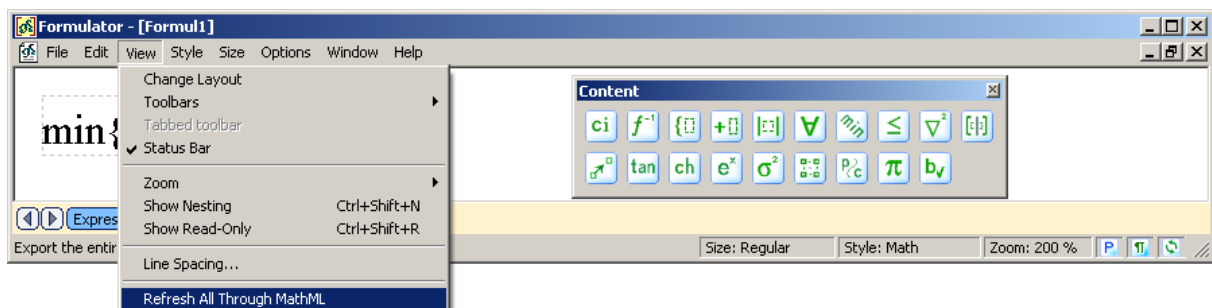
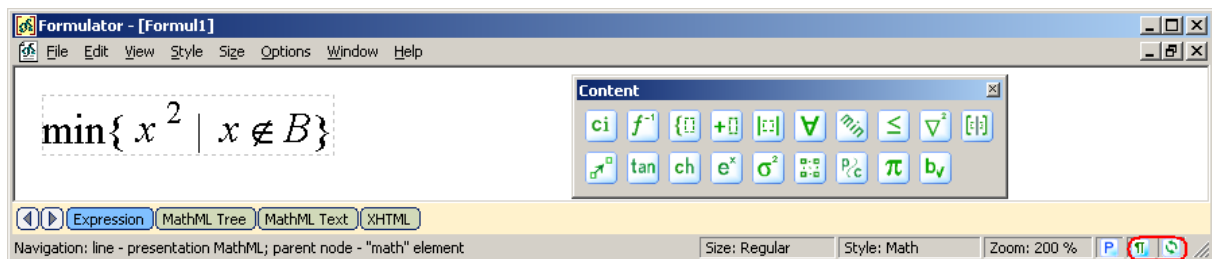
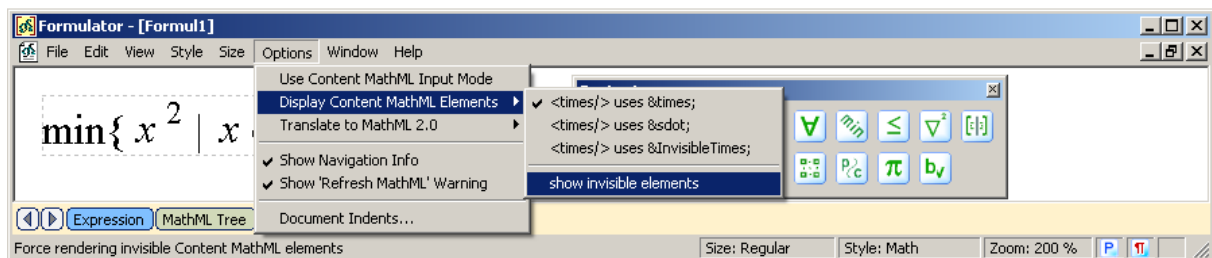




Now get back to the “Expression” page and refresh the document contents through MathML.



Note that the bounded variable is hidid according to the proper rendering of the “min” element with condition qualifier. We can edit the bounded variable at any time if we turn on rendering of invisible elements and refresh the document through MathML once more.



2. Example 4.4.3.17.2 from the W3C MathML Recommendation): universal quantifier (forall):

$$\forall p, q, p \in \mathbb{Q} \wedge q \in \mathbb{Q} \wedge p < q : p < q^2$$

```
<apply>
  <forall/>
  <bvar><ci> p </ci></bvar>
  <bvar><ci> q </ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci> p </ci><rational/></apply>
```

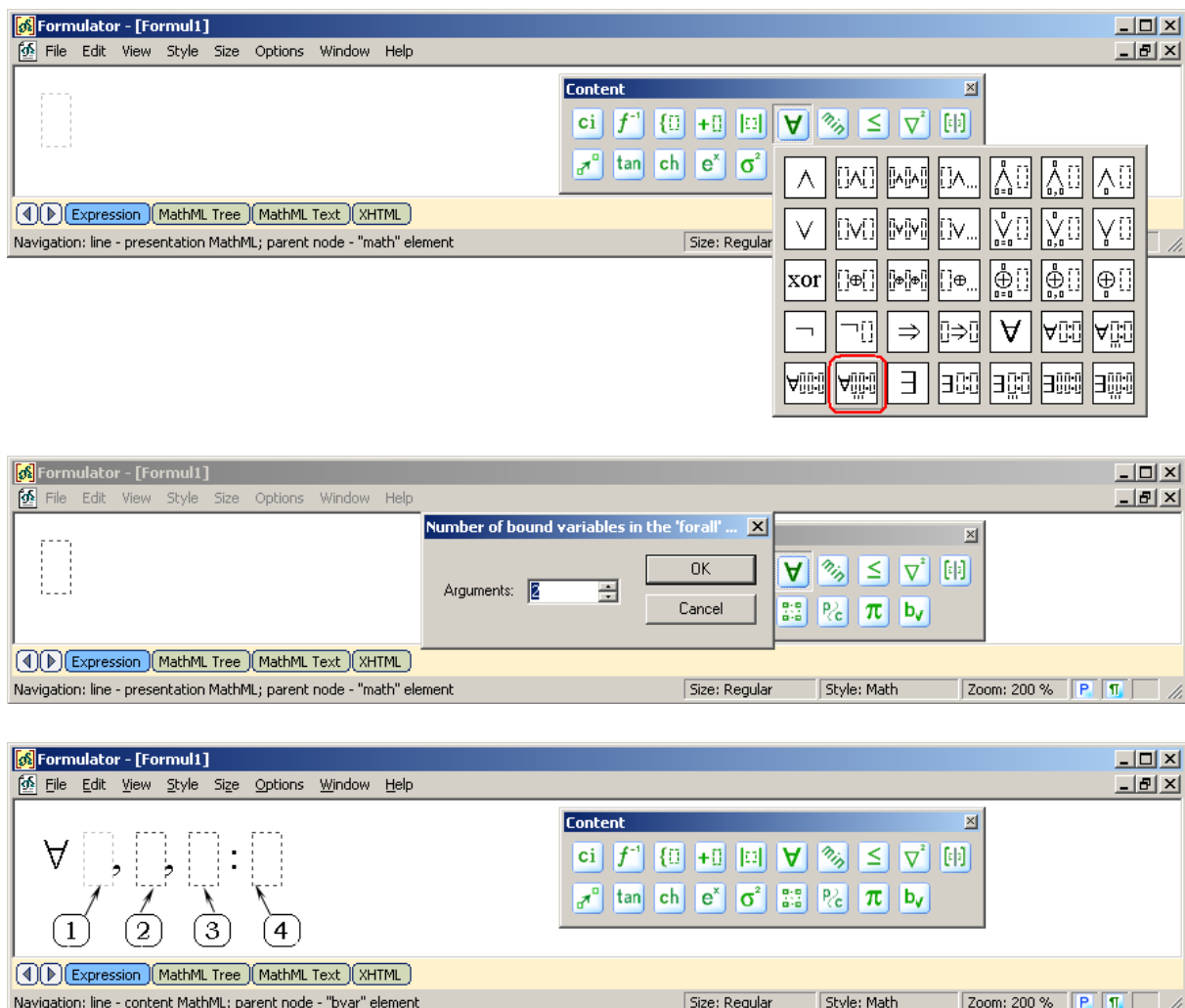
```

    <apply><in/><ci> q </ci><rationals/></apply>
    <apply><lt/><ci> p </ci><ci> q </ci></apply>
  </apply>
</condition>
<apply><lt/>
  <ci> p </ci>
  <apply>
    <power/>
    <ci> q </ci>
    <cn> 2 </cn>
  </apply>
</apply>
</apply>

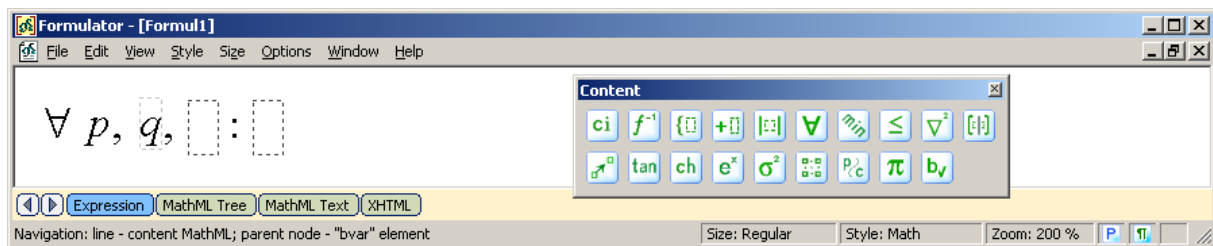
```

The forall element is usually used in conjunction with one or more bound variables, an optional condition element, and an assertion. There is lot of available combination of different constructs of this element (this is true also for other similar statements, like the “exists” element, sums and products, integrals and so on). So MathML Weaver propose several buttons on mathematical toolbars, trying to make it faster creating common MathML trees.

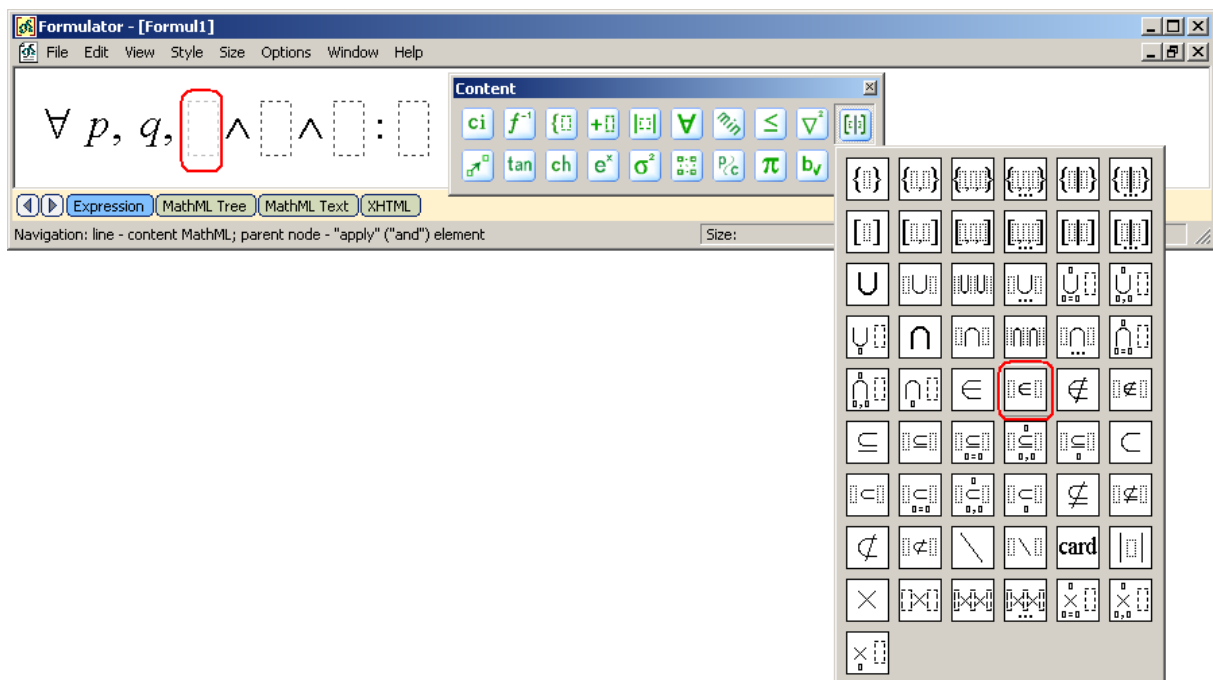
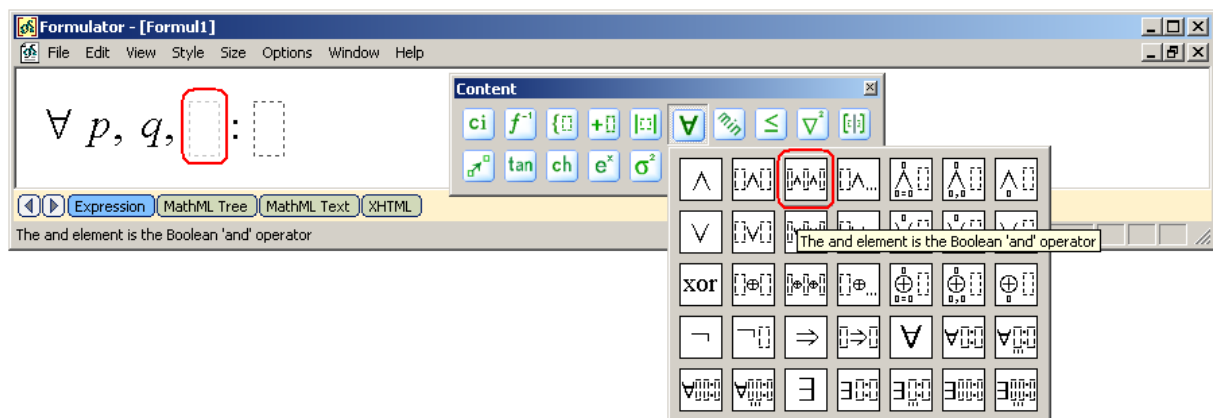
In the case of our example there is the button for the “forall” element with several bounded variables and a conditional element:



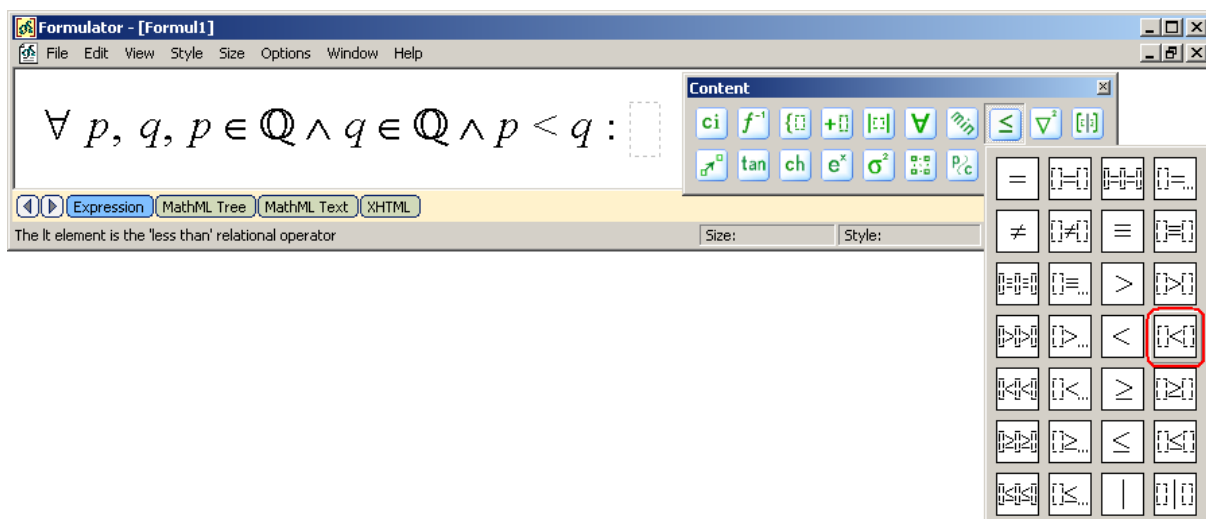
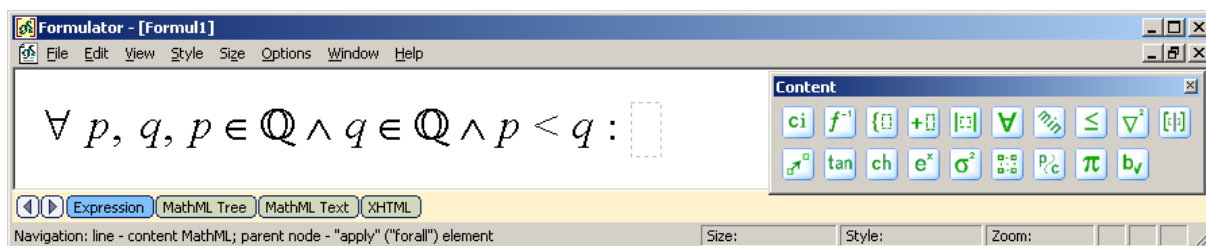
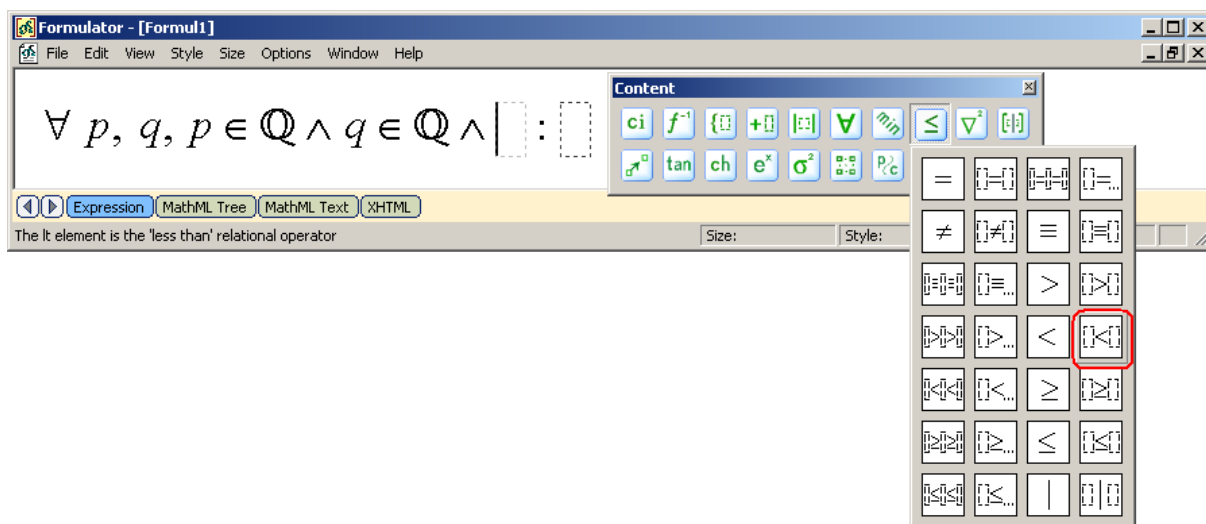
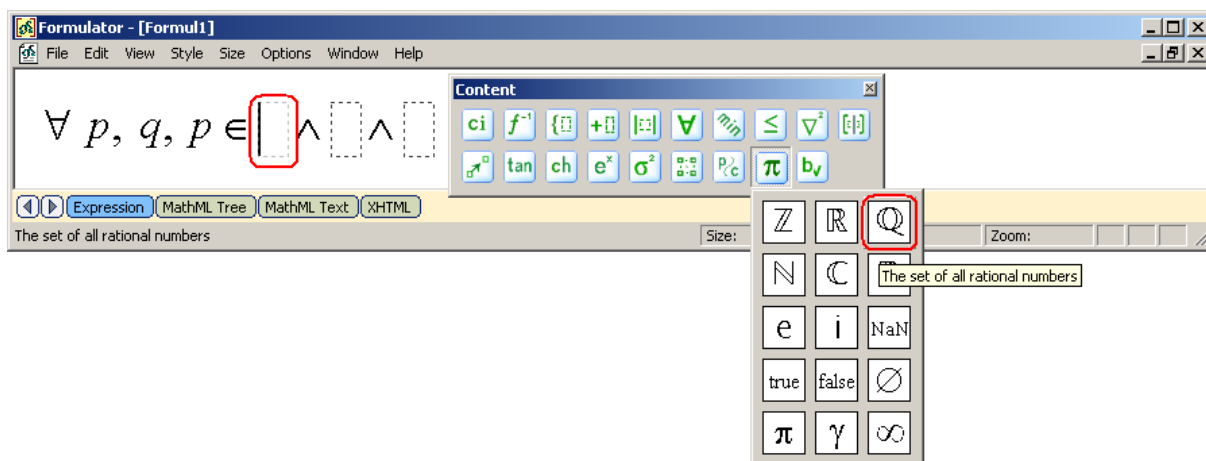
Here (1) and (2) are input slots for two bounded variables; (3) is a condition element; (4) is an assertion.

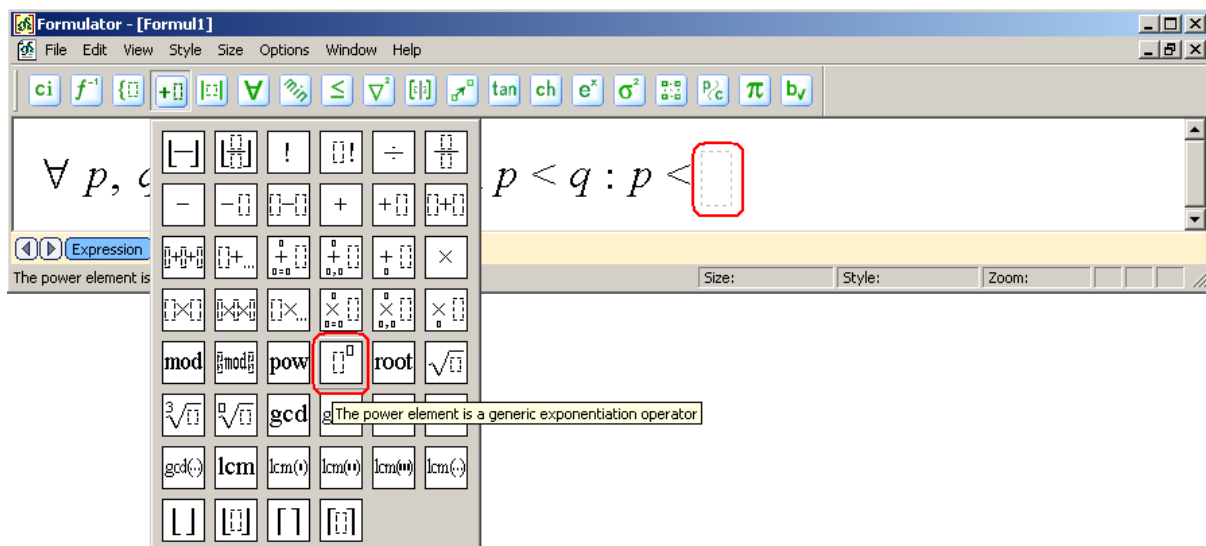
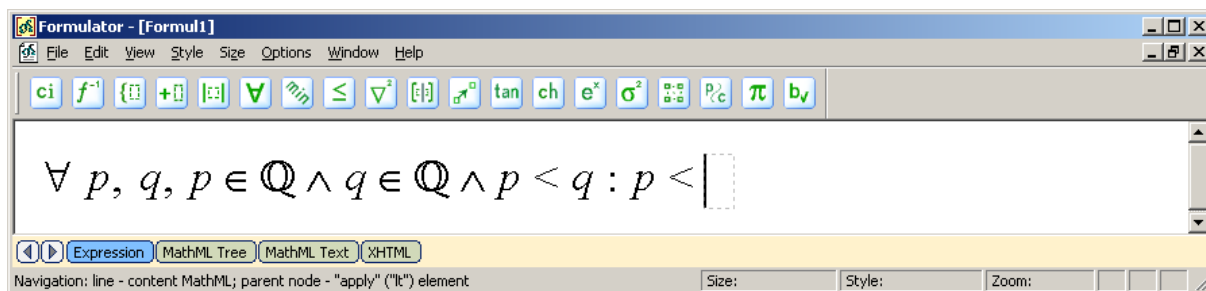


Now we should insert the “apply” element with the `<and/>` operation and three arguments.

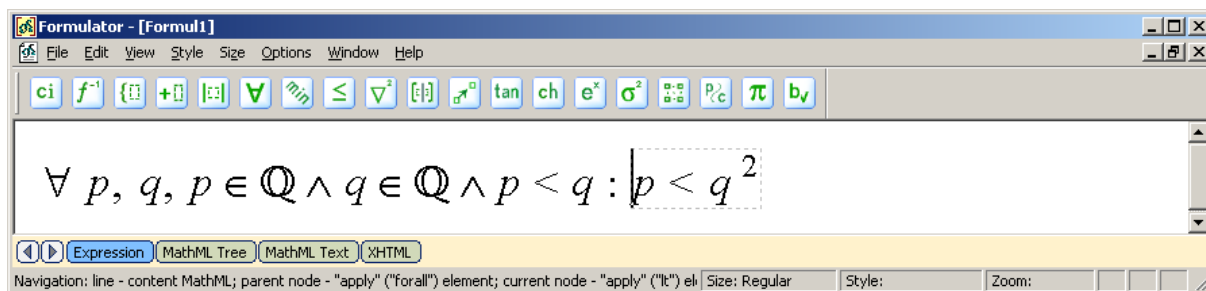


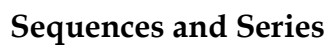
In order to insert the `<rationals/>` element use the π button:



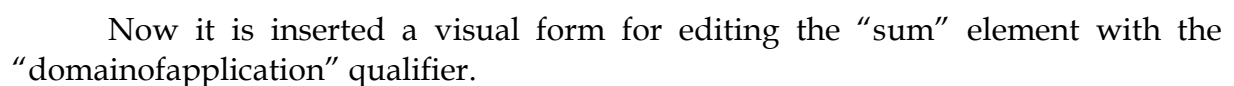


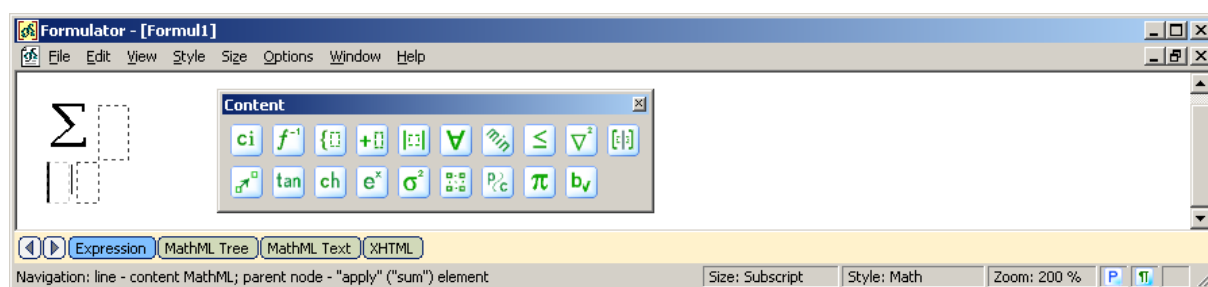
See results on different pages of MathML Weaver.





```
<apply>
  <sum/>
  <domainofapplication>
    <ci type="set"> B </ci>
  </domainofapplication>
  <ci type="function"> f </ci>
</apply>
```

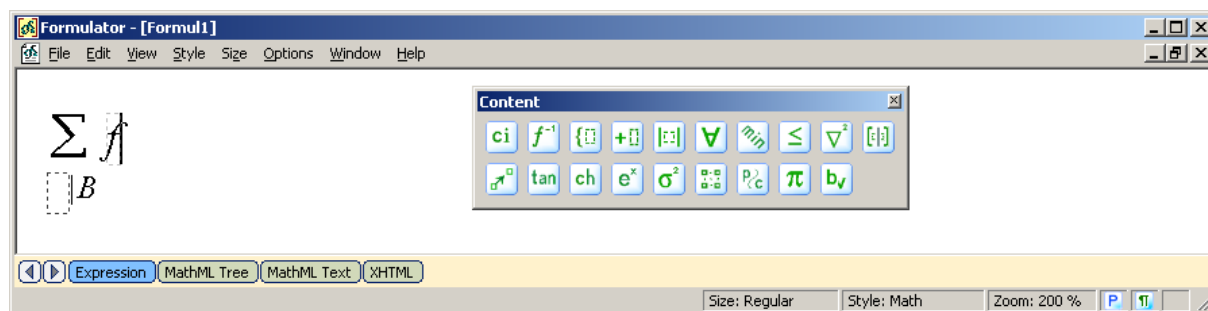
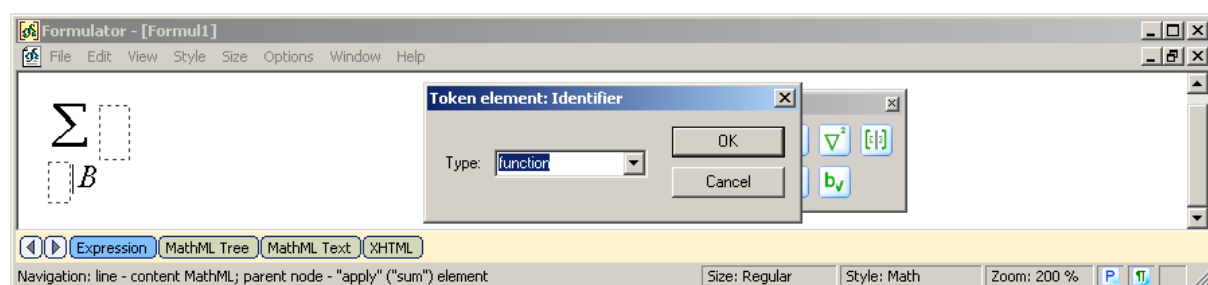




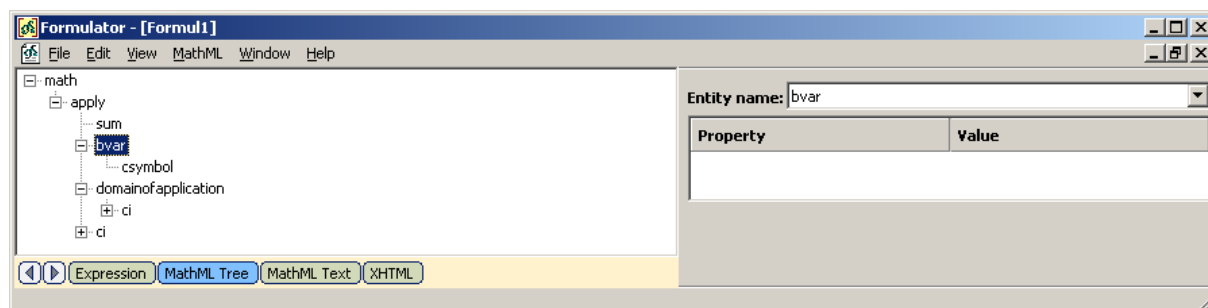
Input slots below the sum sign are rendered as

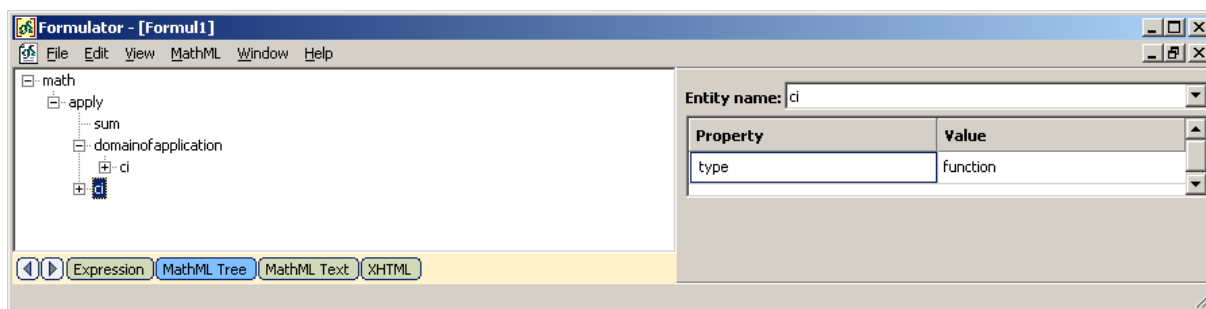


in order to allow to a user editing both the bounded variable (the first input slot) and the “domainofapplication” element. Leave the first input slot empty and create a “ci” element of type “set” for the second input slot. Then create another “ci” element for the argument of the “sum” element.

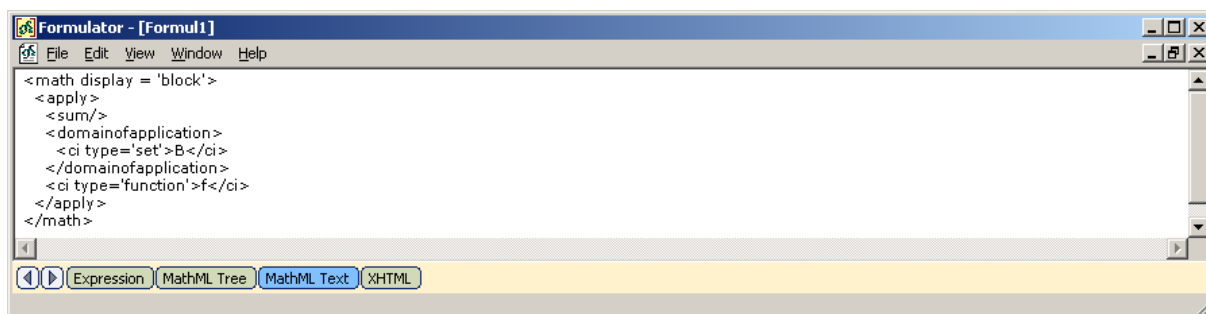
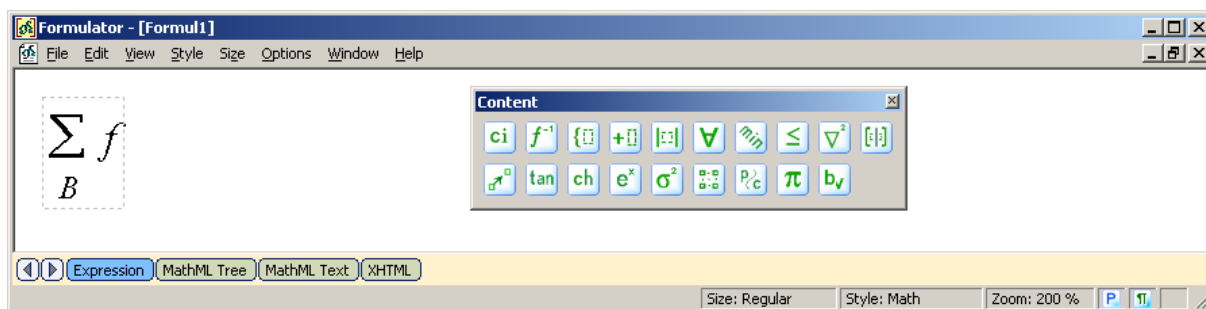


Switch to the “MathML Tree” page and delete the node for the bounded variable, since the example don’t need the “bvar” presence. In order to do this, select the “bvar” node and press Delete.



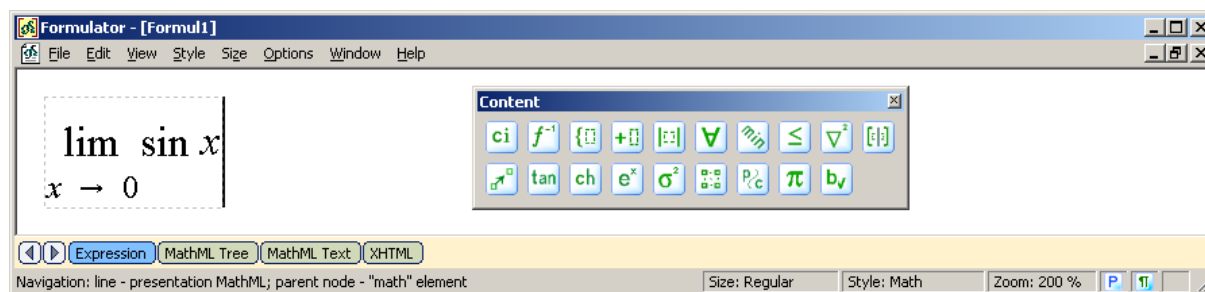
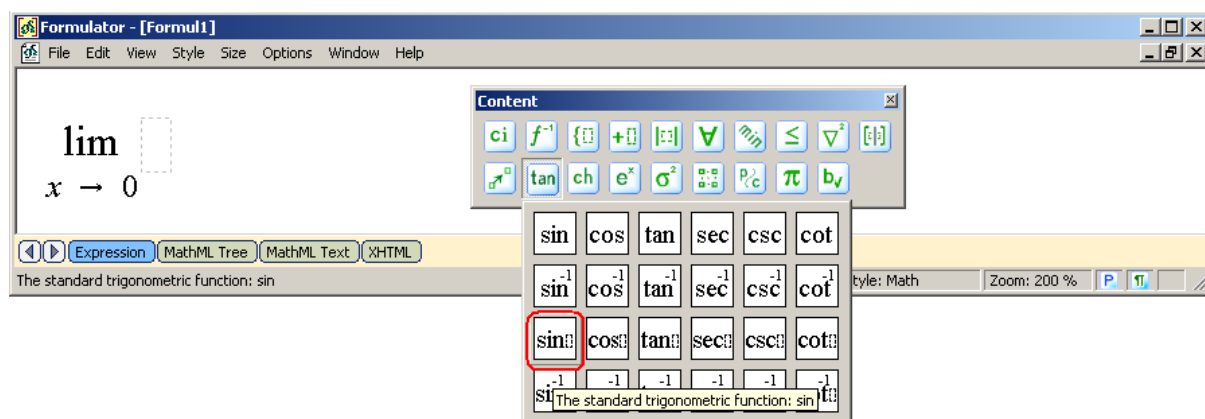
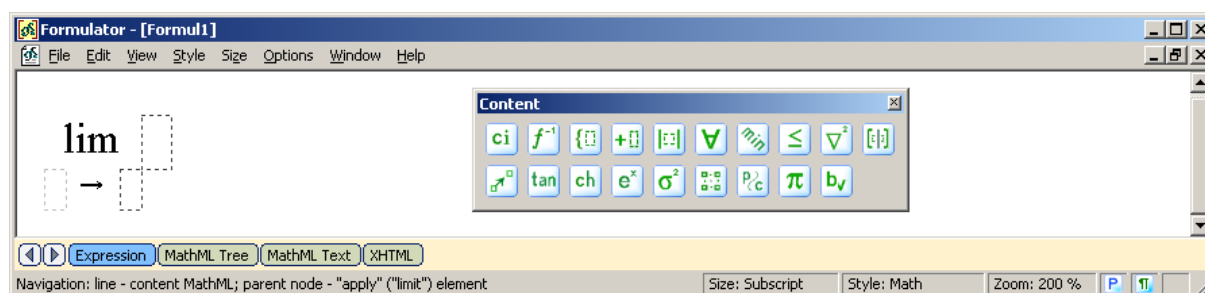
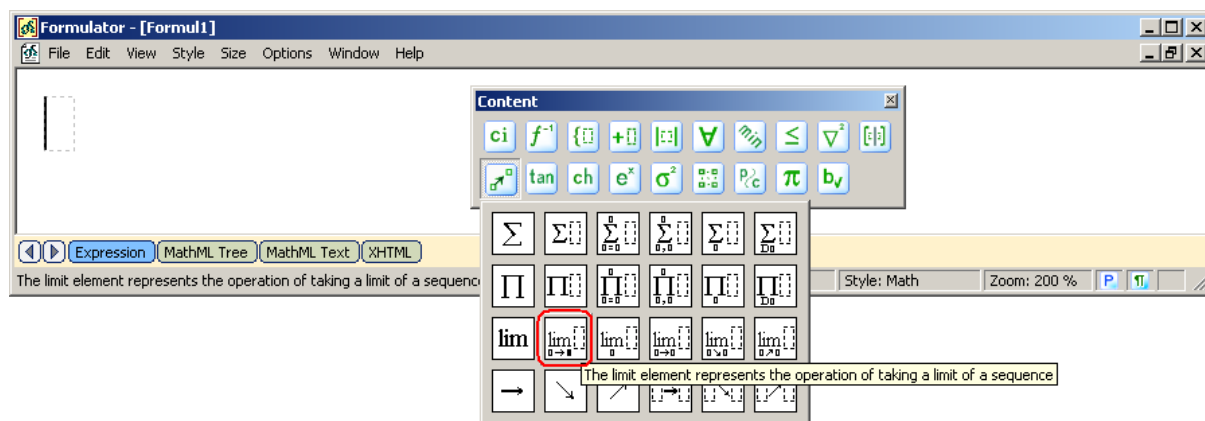


Then get to the “Expression” and “MathML Text” pages to see results.



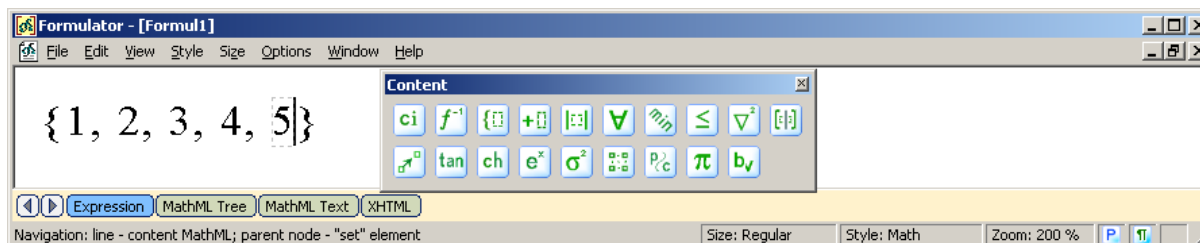
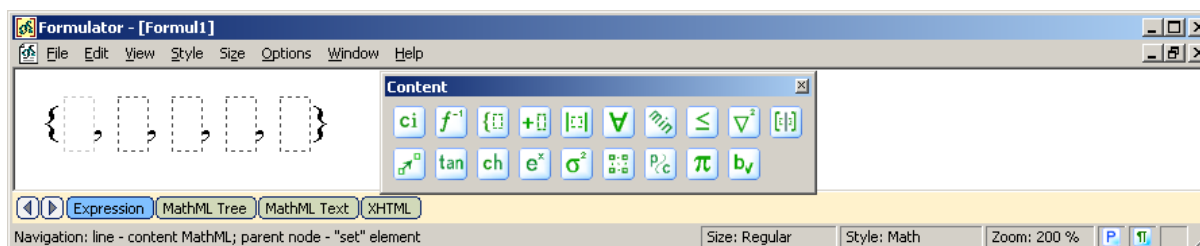
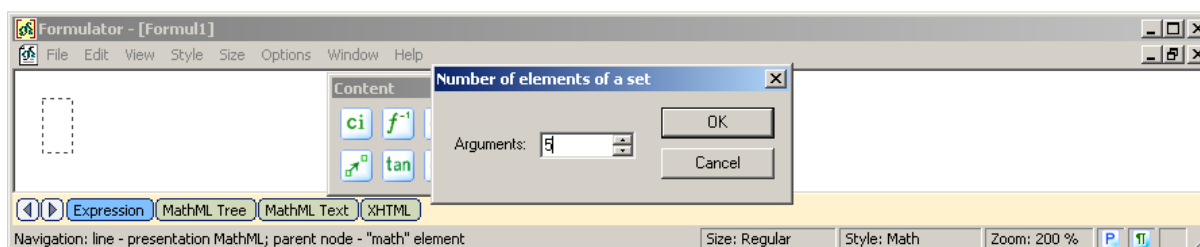
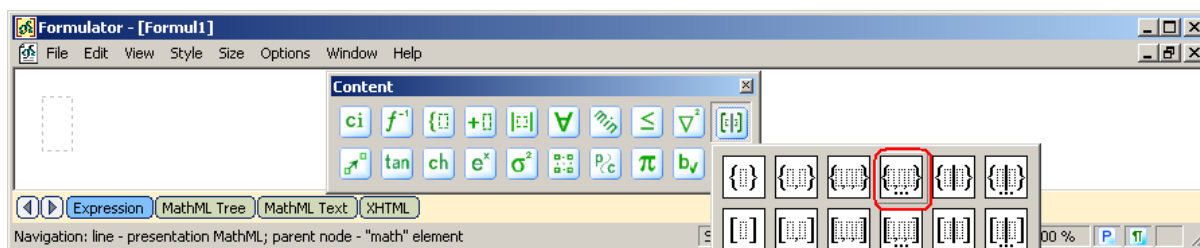
2. Examples 4.4.7.3.2 from the W3C MathML Recommendation): the limit element:

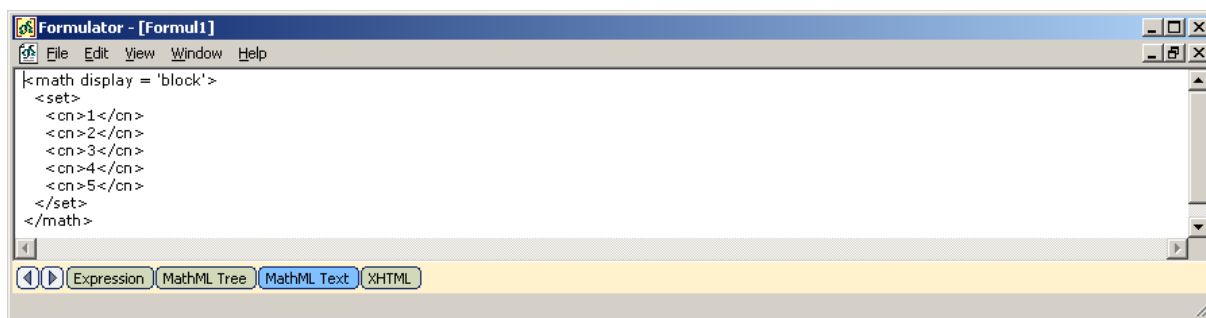
```
<apply>
  <limit/>
  <bvar><ci> x </ci></bvar>
  <lowlimit><cn> 0 </cn></lowlimit>
  <apply><sin/><ci> x </ci></apply>
</apply>
```



Constructor elements: theory of sets and linear algebra

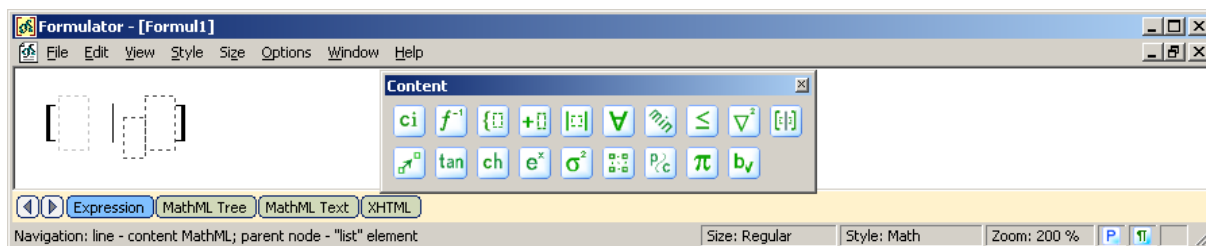
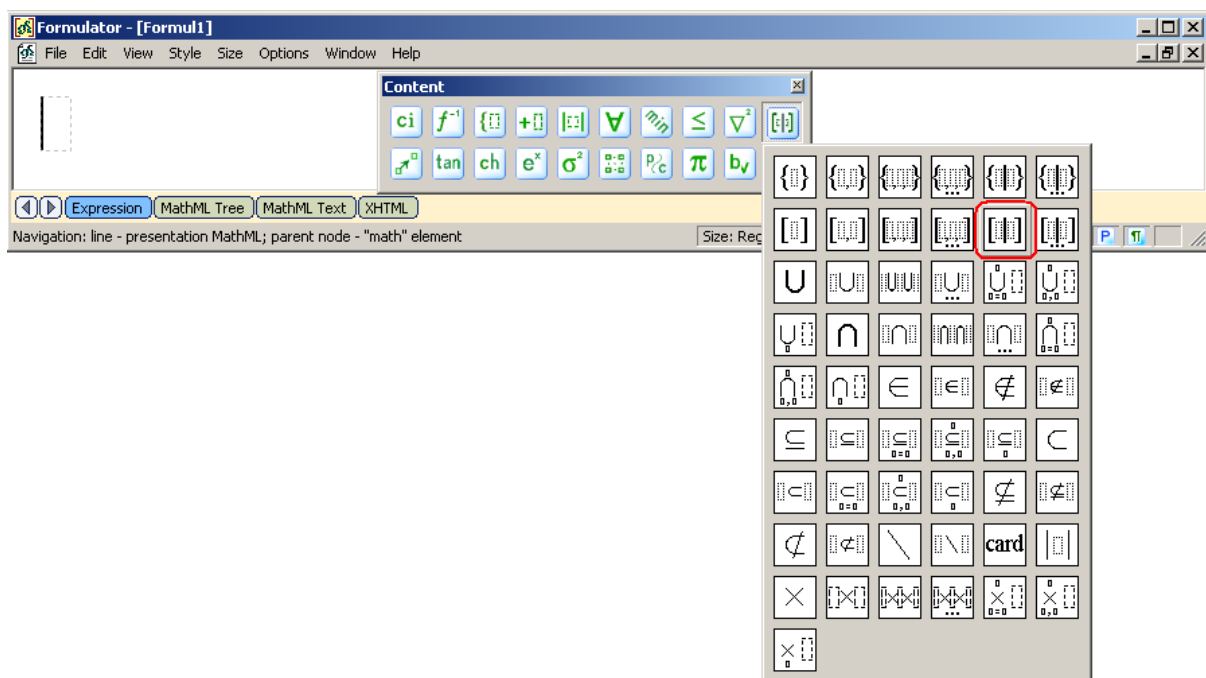
1. The next example shows how to create elements of the theory of sets.



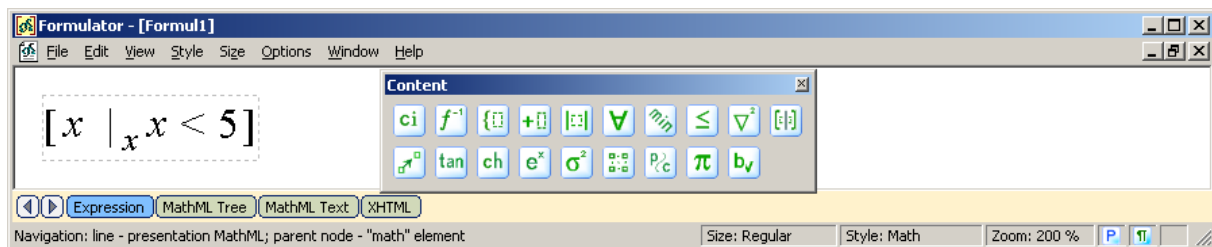
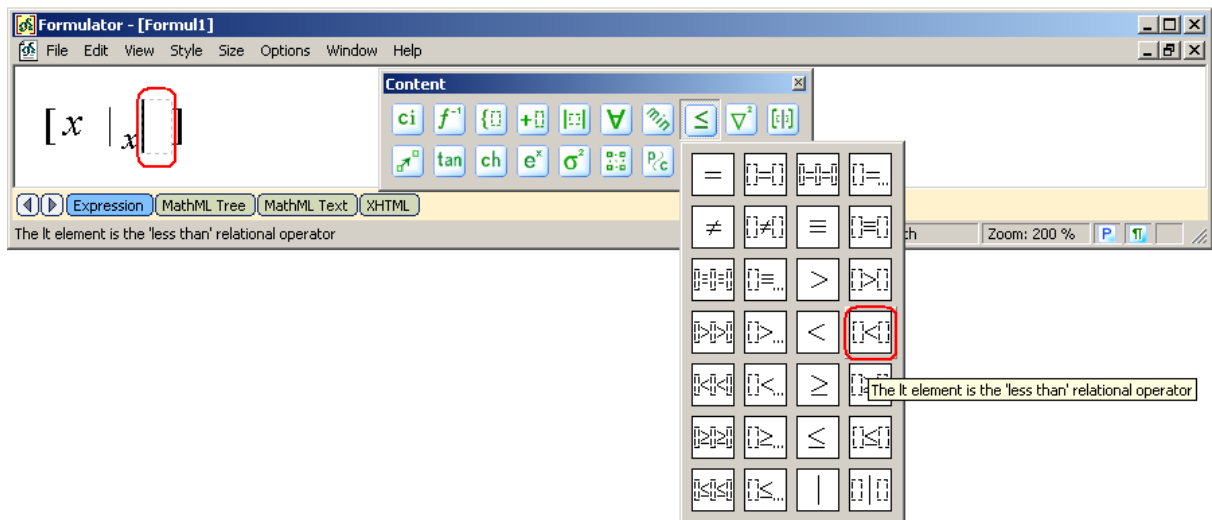


Now create a list using a bounded variable and a condition element (example 4.4.6.2.2 from the W3C MathML Recommendation):

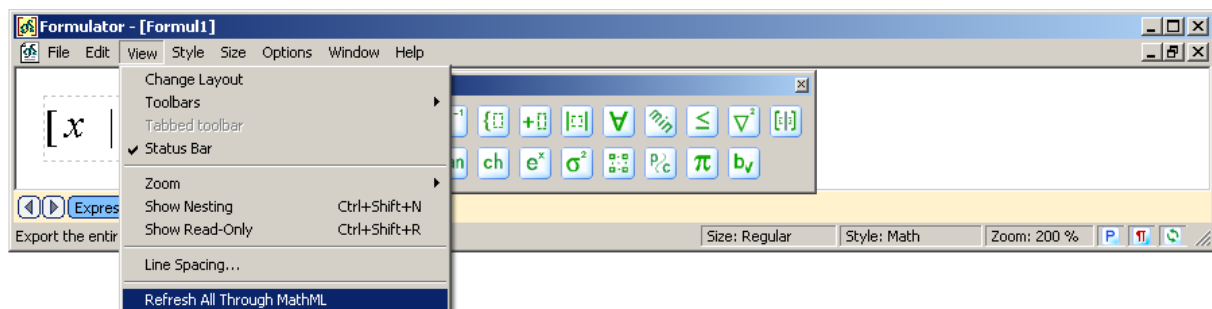
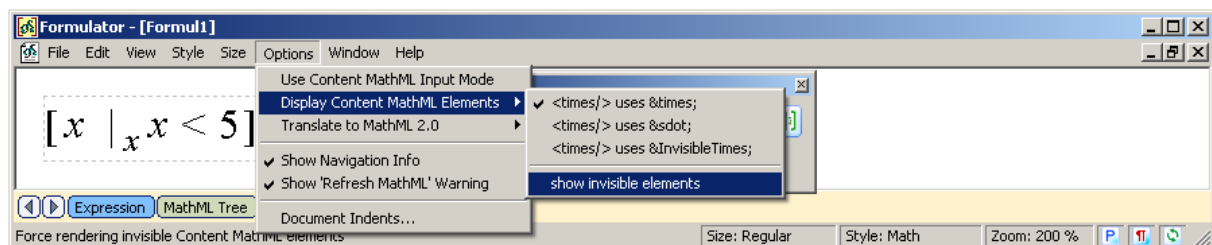
```
<list order="numeric">
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/>
      <ci> x </ci>
      <cn> 5 </cn>
    </apply>
  </condition>
  <ci> x </ci>
</list>
```

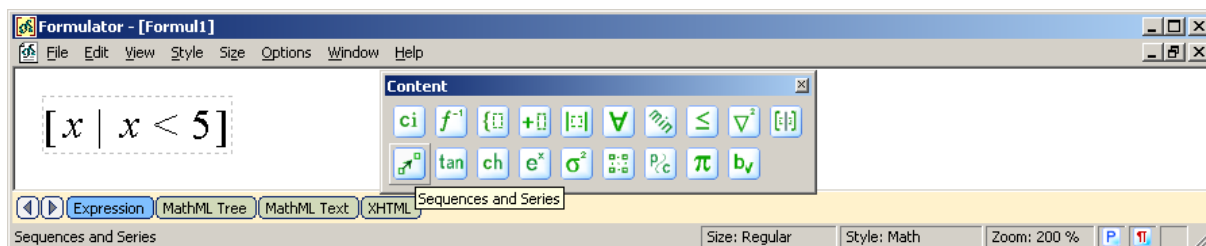


If we recall the case of the minimum operator from the “Arithmetic, Algebra, Logic and Relations” section, it will be obvious that the structure of the just created “list” element is quite similar to that case. Namely, the starting input slot is for a rule of constructing list’s items; after the vertical line there is a smaller input slot for a bounded variable; the last input slot is for a condition.

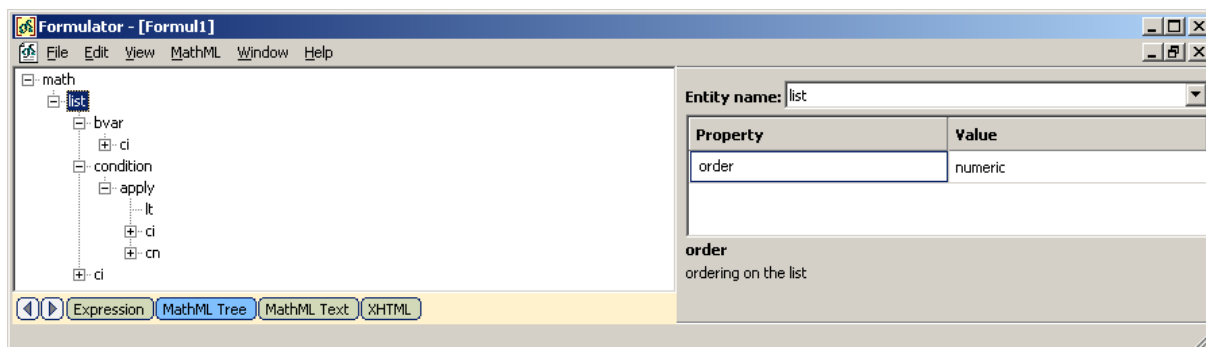


In order to get proper rendering of the list we should turn off the invisible elements rendering and refresh the document through MathML, as it is shown on the next figures.



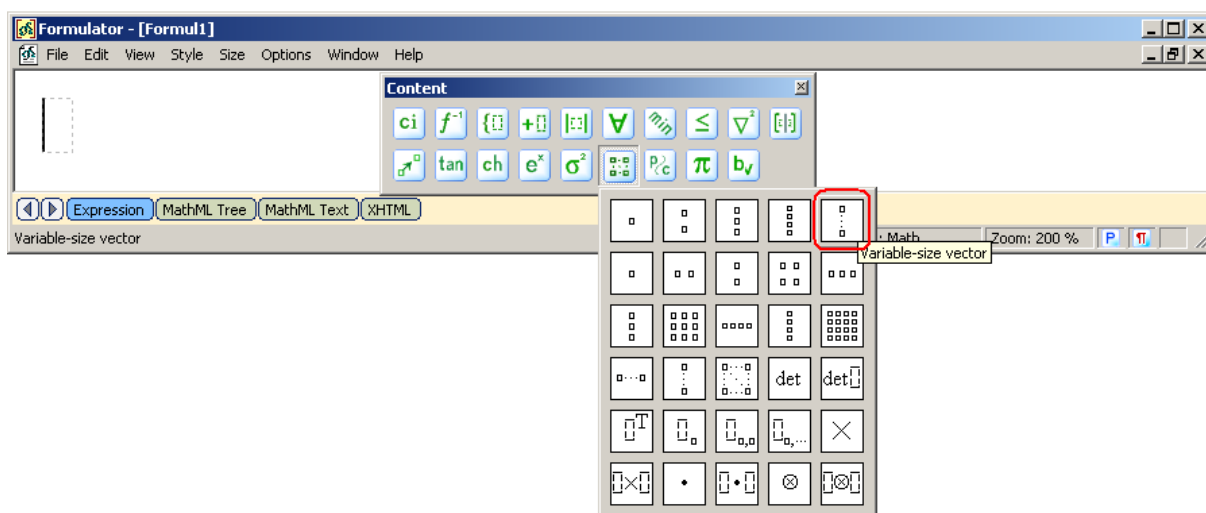


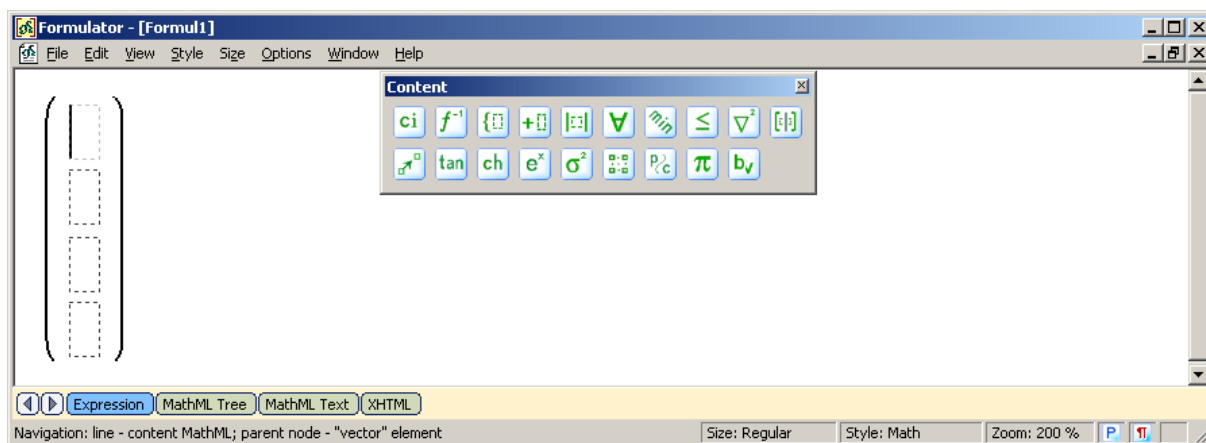
Switch to the “MathML Tree” page to see results.



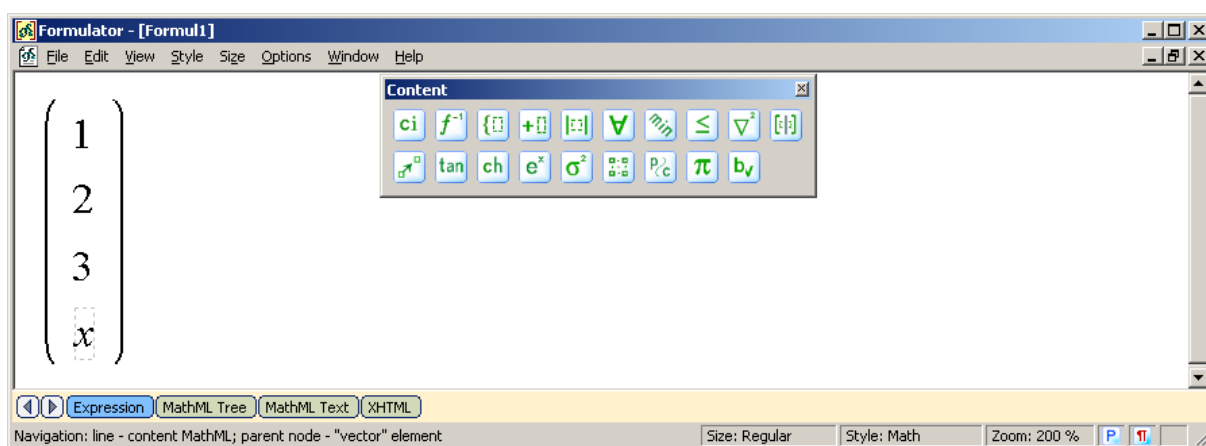
2. The next examples shows how to create elements of the linear algebra.

```
<vector>
  <cn> 1 </cn>
  <cn> 2 </cn>
  <cn> 3 </cn>
  <ci> x </ci>
</vector>
```

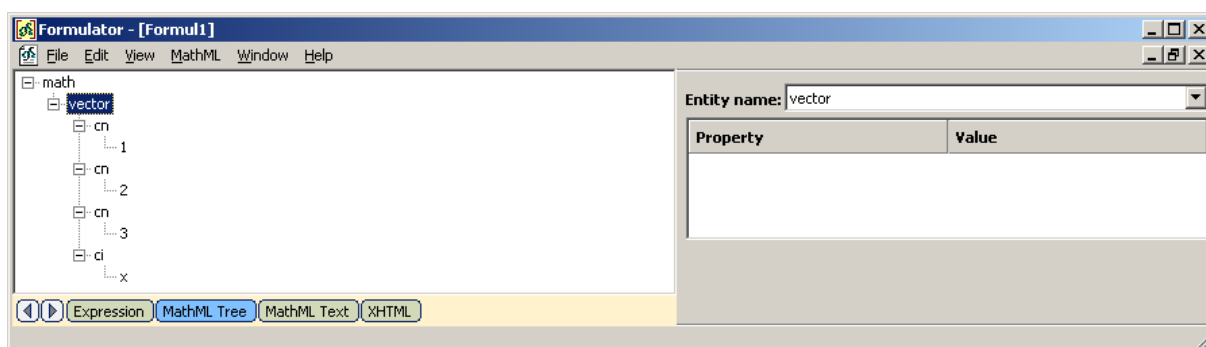




Press '1', the Down arrow, '2', the Down arrow, '3', the Down arrow, 'x'.



See results:



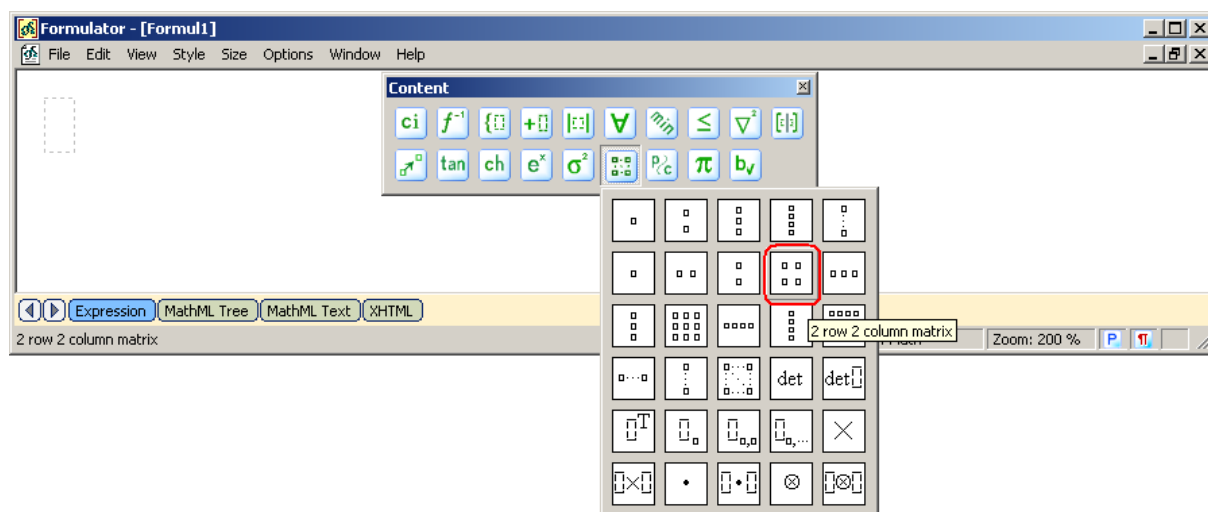
The next example show how to construct a matrix and use the “selector” element.

```
<apply>
  <selector/>
  <matrix>
    <matrixrow>
      <cn> 1 </cn> <cn> 2 </cn>
    </matrixrow>
    <matrixrow>
      <cn> 3 </cn> <cn> 4 </cn>
    </matrixrow>
  </matrix>
</apply>
```

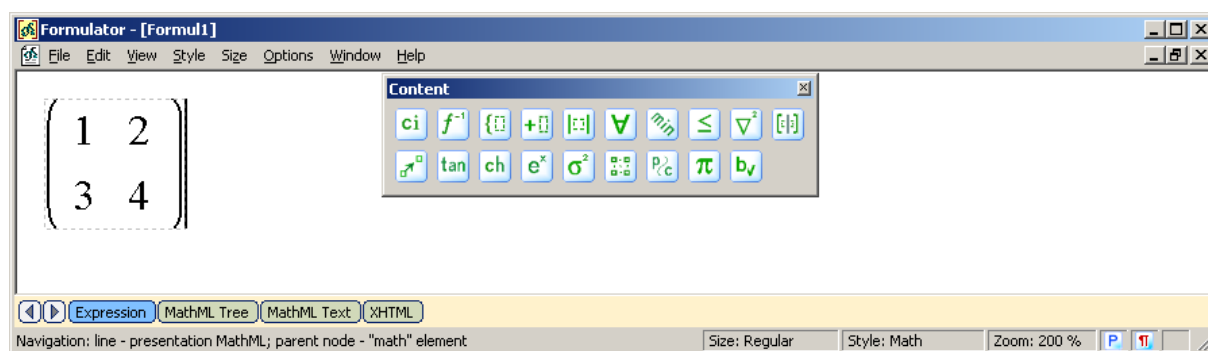
```

    </matrixrow>
  </matrix>
  <cn> 1 </cn>
</apply>

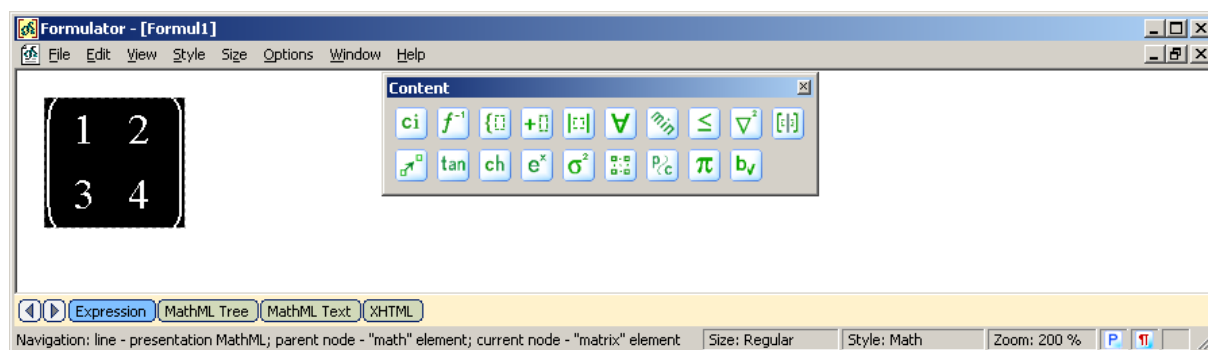
```



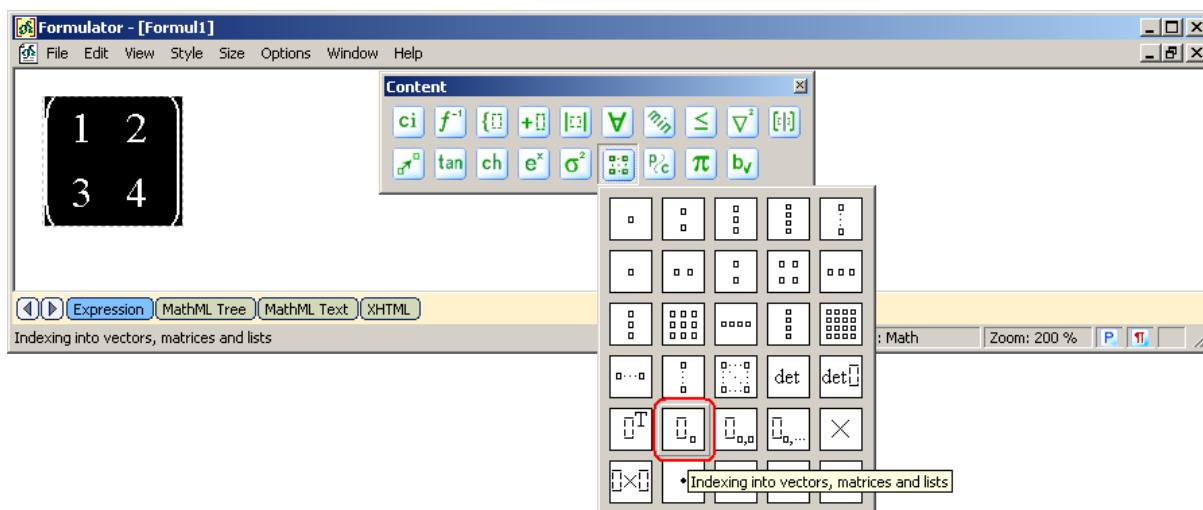
Press '1', the Right arrow, '2', the Right arrow, '3', the Right arrow, '4', the Right arrow.



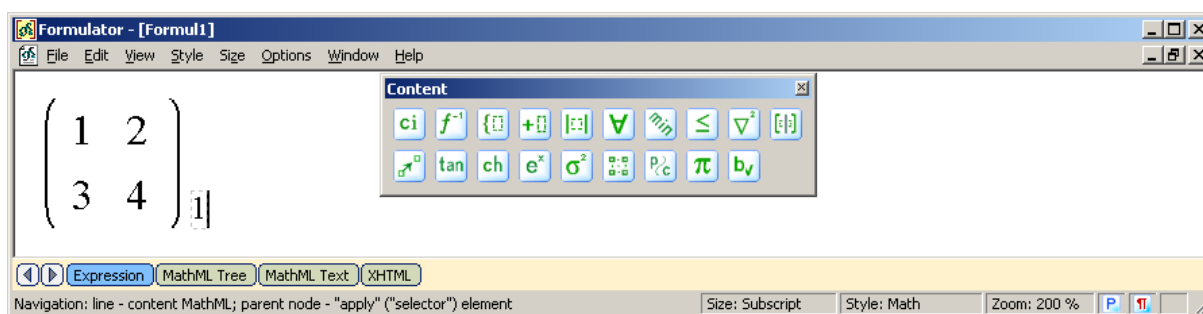
Select the whole matrix.



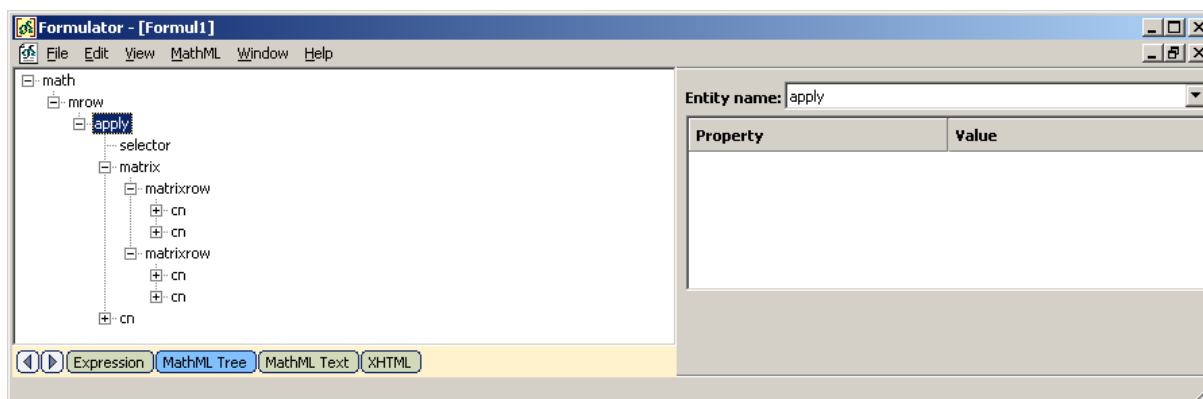
Insert the "separator" element.



Press the Right arrow, '1'.



See the resulting MathML tree:



Calculus and Vector Calculus: Integral and Differentiation


1. Example 4.4.5.1.2 from the W3C MathML Recommendation): the int element:

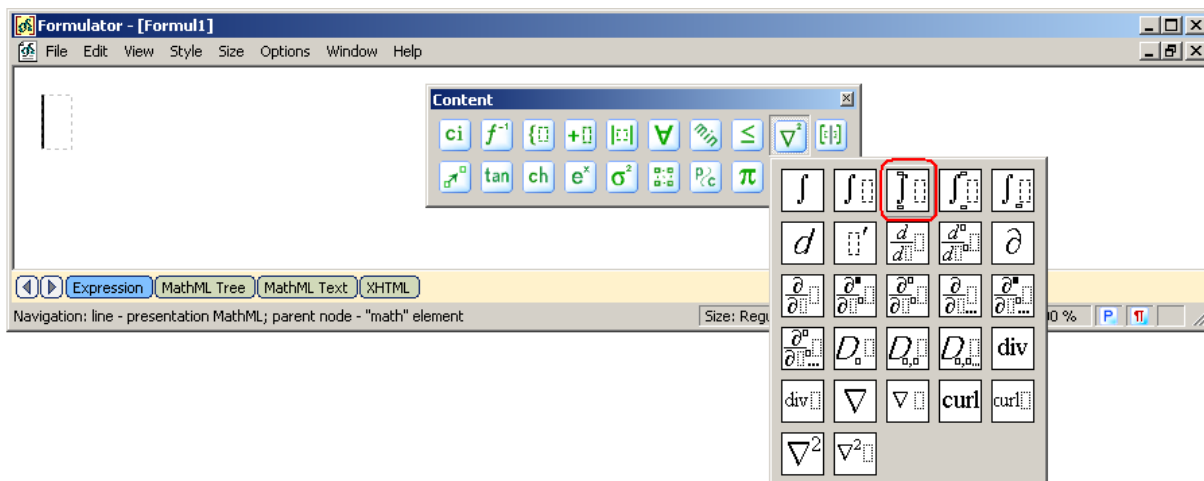
```
<apply>
  <int/>
  <bvar><ci> x </ci></bvar>
  <lowlimit><cn> 0 </cn></lowlimit>
```

```

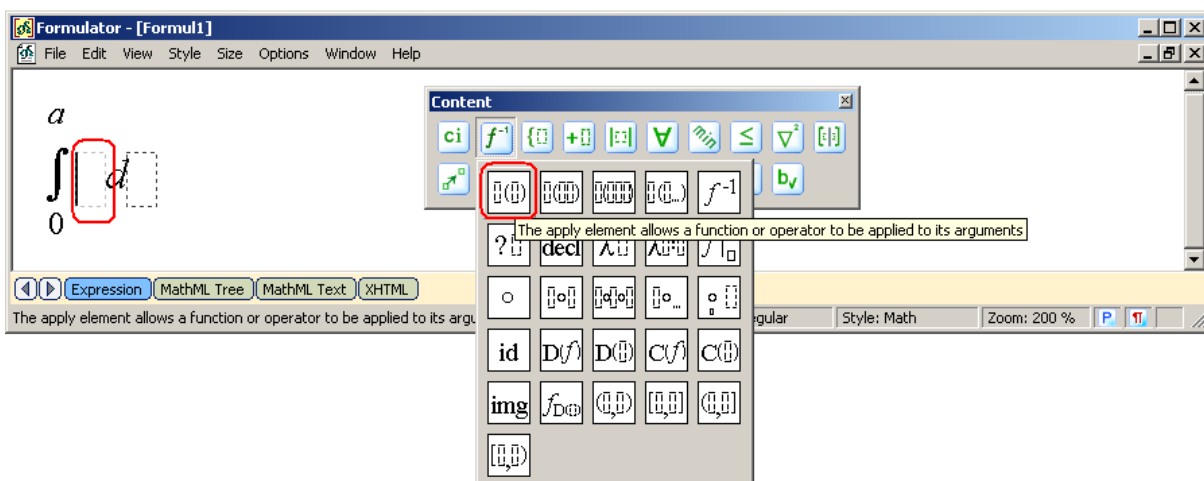
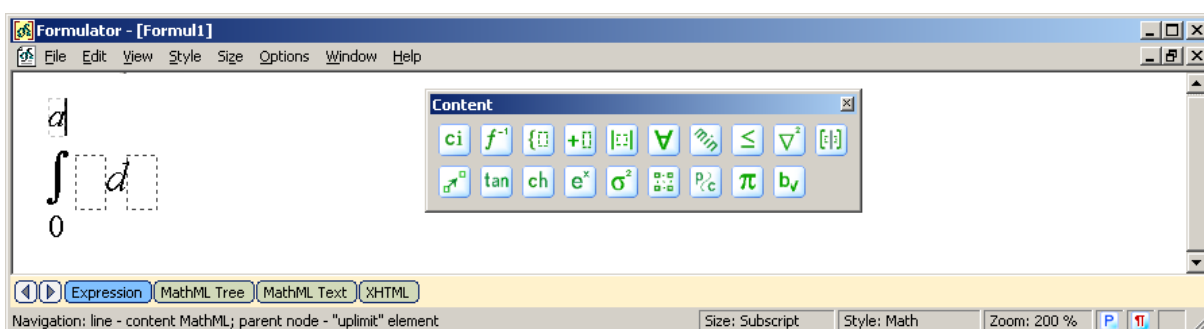
<uplimit><ci> a </ci></uplimit>
<apply>
  <ci> f </ci>
  <ci> x </ci>
</apply>
</apply>

```

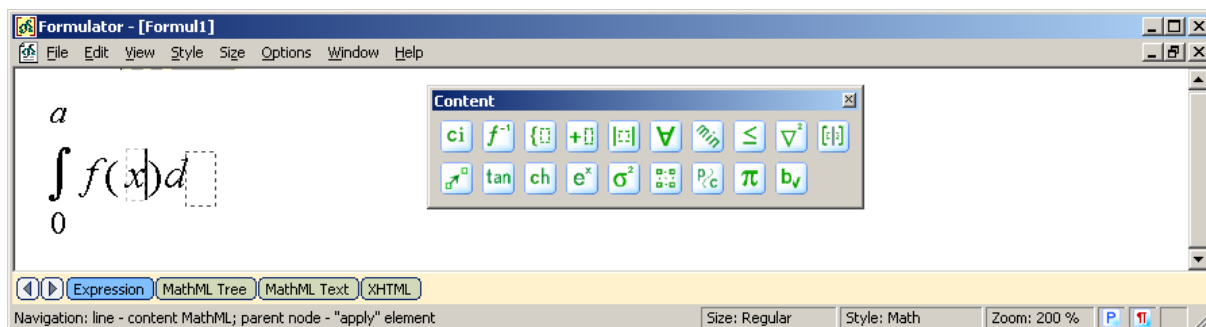
Use the mathematical toolbar .



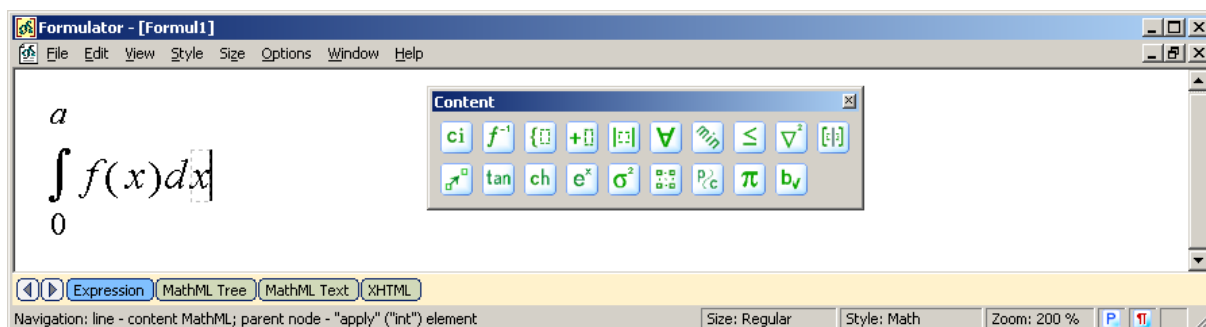
Press '0', the Up arrow, 'a' (a low and upper limits of the integral).



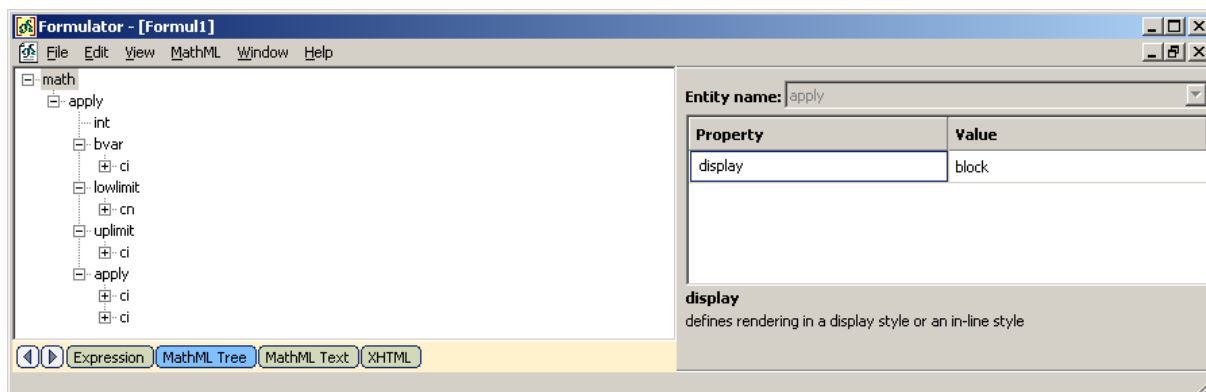
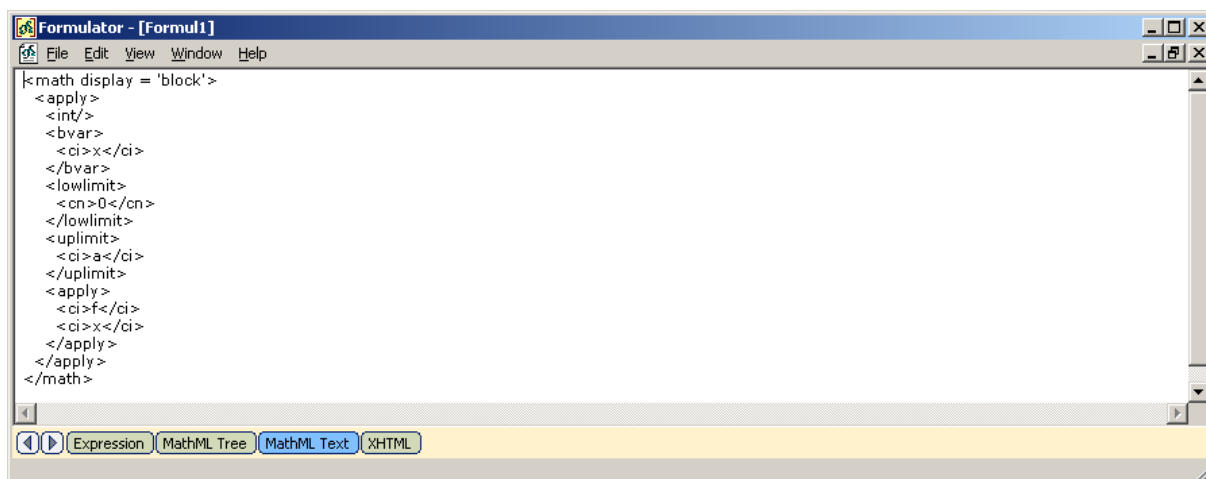
Type 'f', the Right arrow, 'x'.



Press the Right arrow two times, type 'x' (a bounded variable).

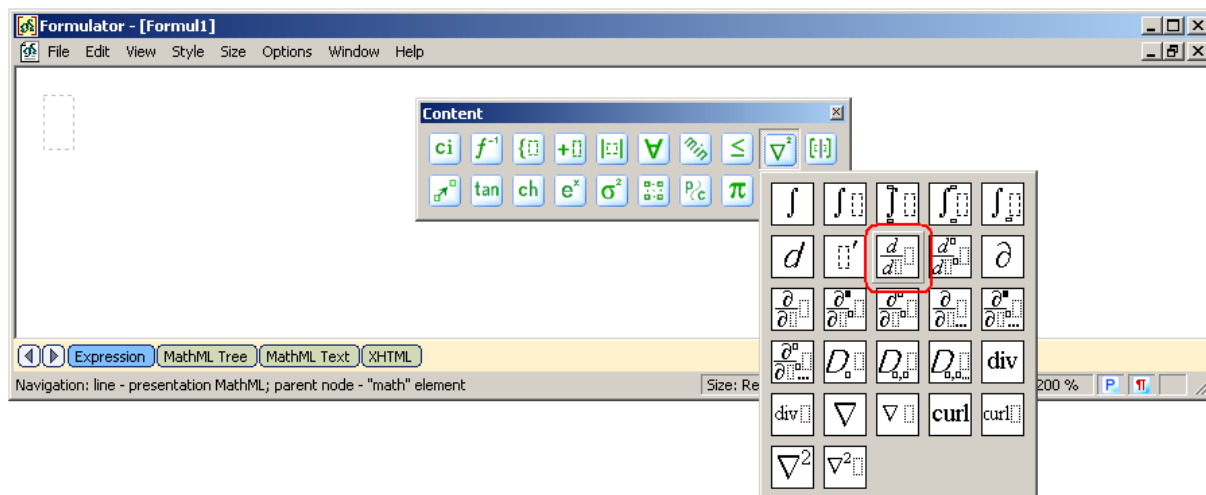


See results:

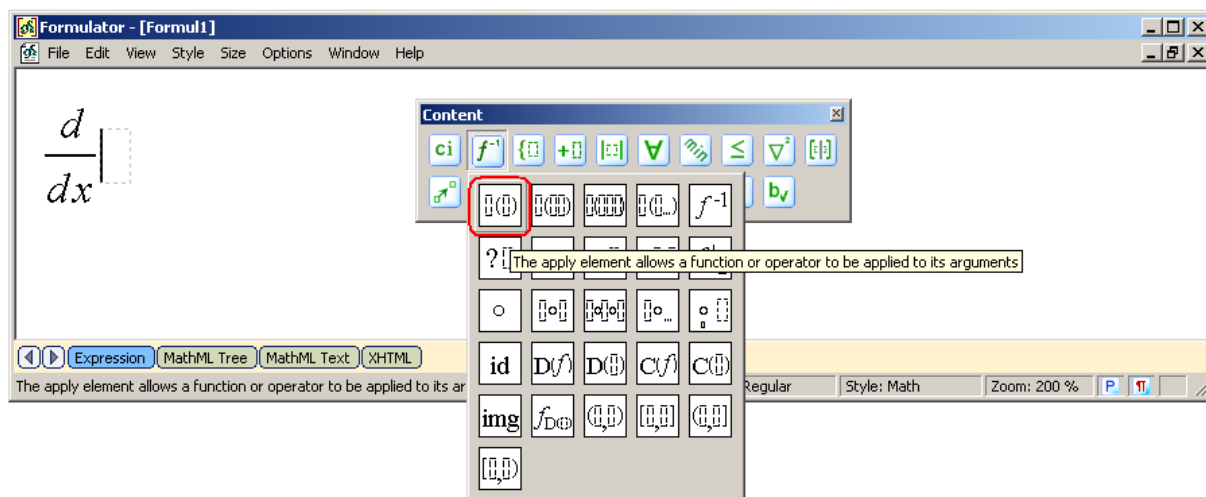


2. Example 4.4.5.2.2 from the W3C MathML Recommendation): the diff element:

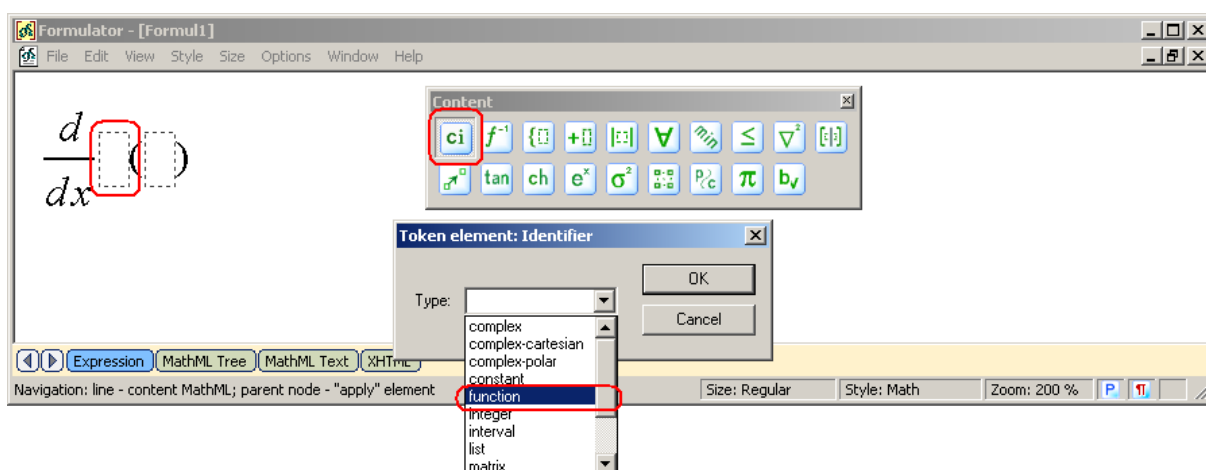
```
<apply>
  <diff/>
  <bvar><ci> x </ci></bvar>
  <apply><ci type="function"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
```



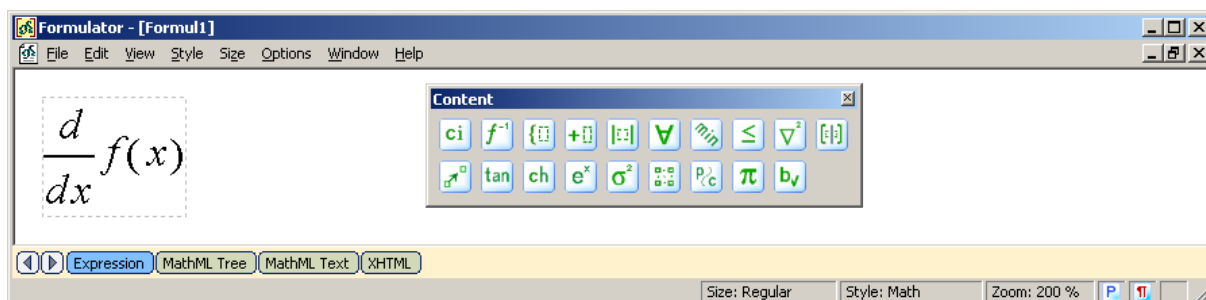
Type 'x' (a name of the bounded variable), press the Right arrow.



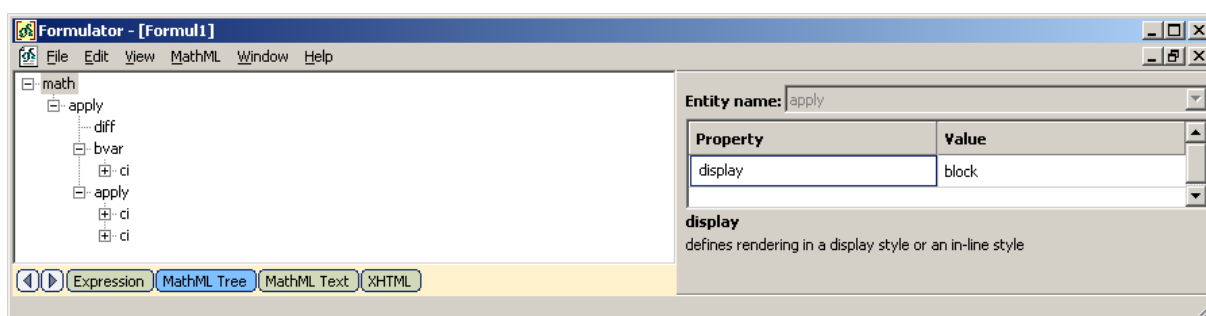
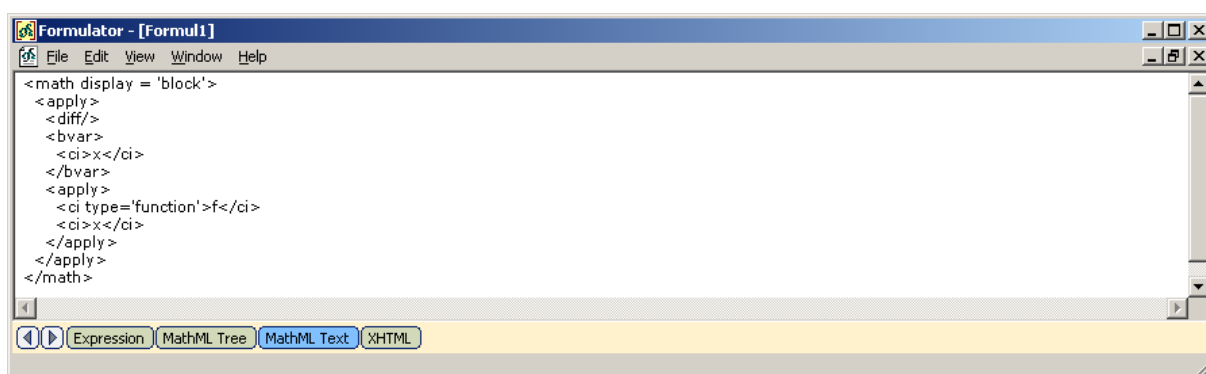
Insert the “ci” element of type “function”; name it “f”.



Type ‘f’, press the Right arrow two times (the first in order to get out of the “ci” node, the second to get to the second input slot of the “apply” element). Type ‘x’.

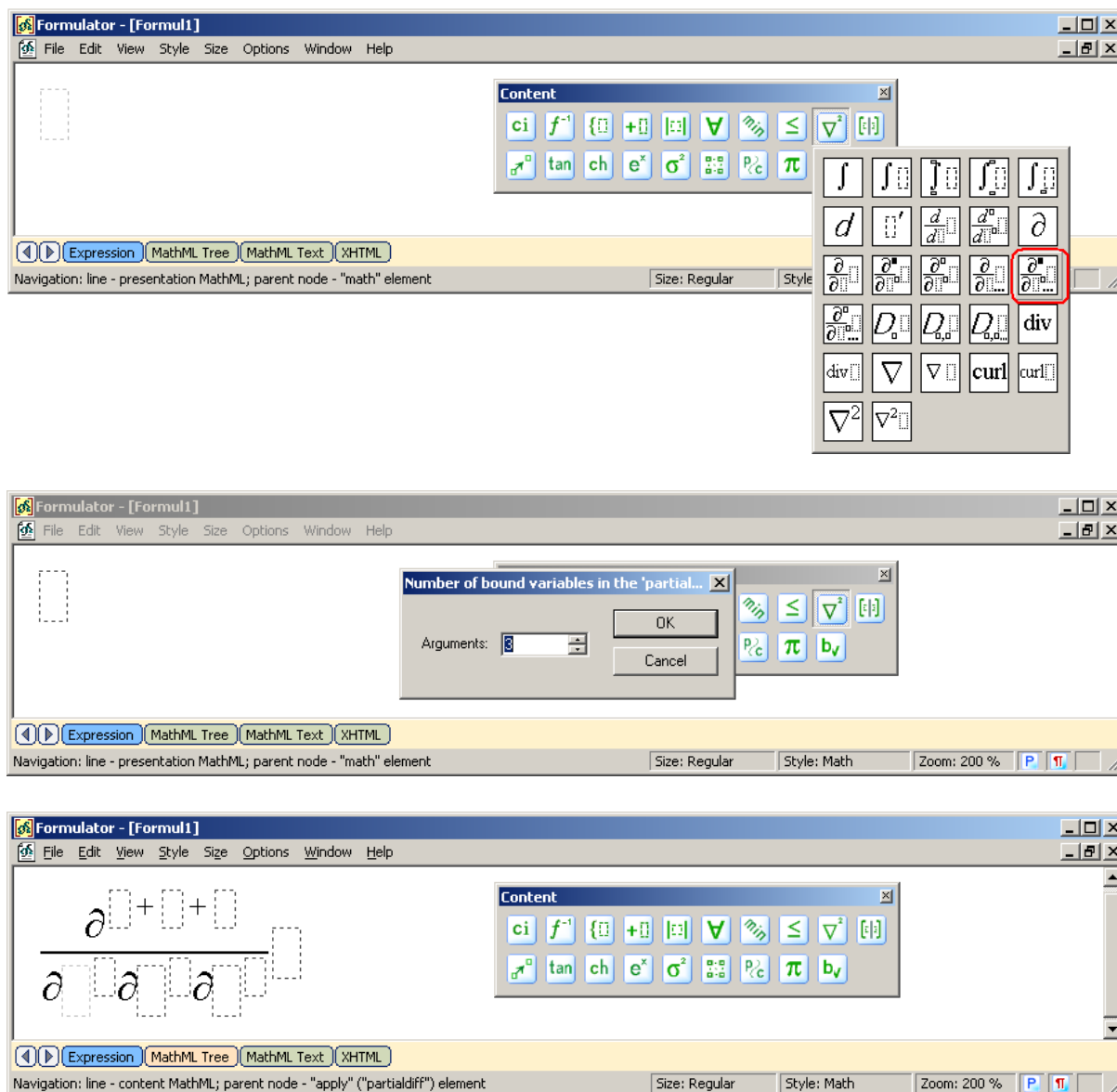


See results:

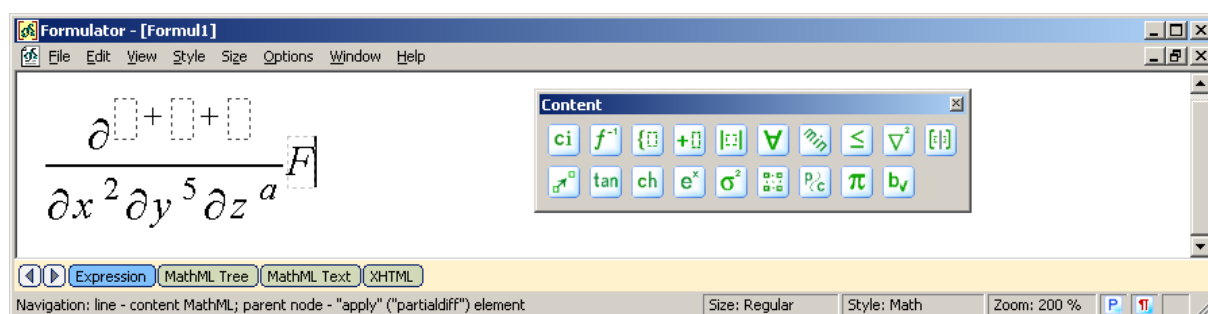


Partial Differentiation

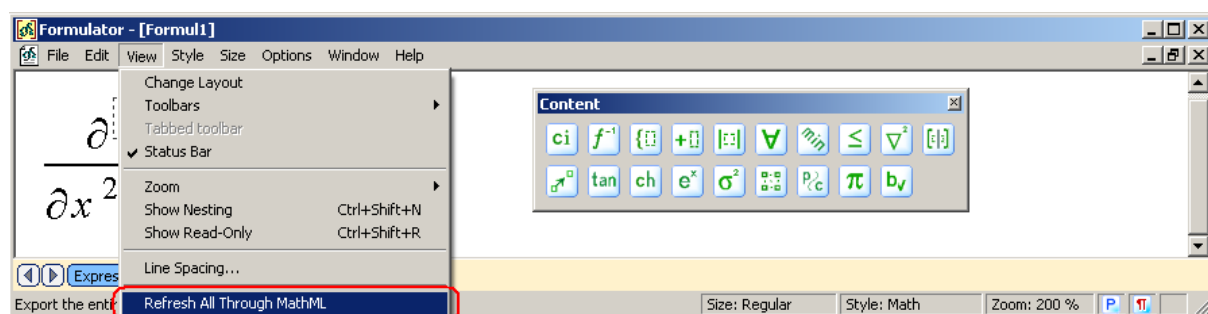
1. The “partialdiff” element can automatically calculate a total degree of differentiation by use of child `degree` elements. The next example demonstrates this feature.



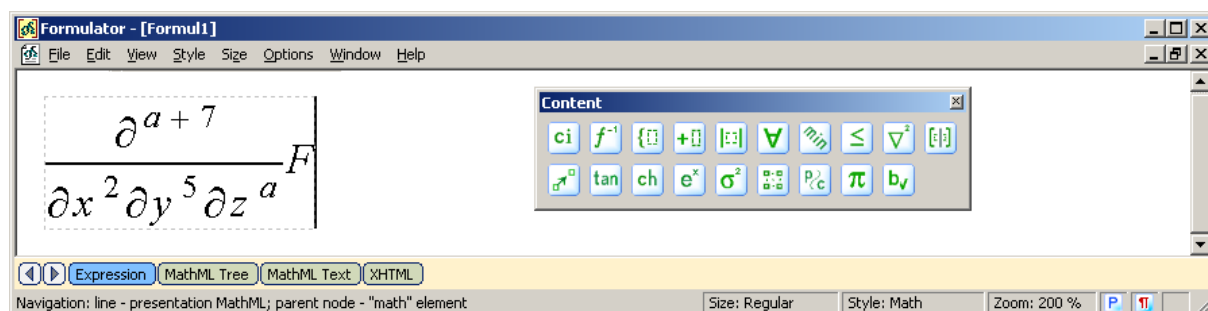
Input names of bounded variables and choose a numeric and symbolic values of their degree.



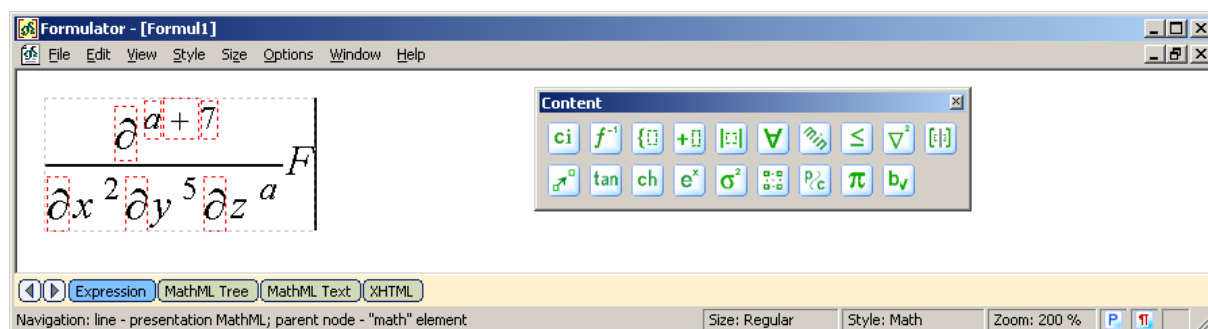
Now a total degree of differentiation can be calculated if we refresh the text through MathML.



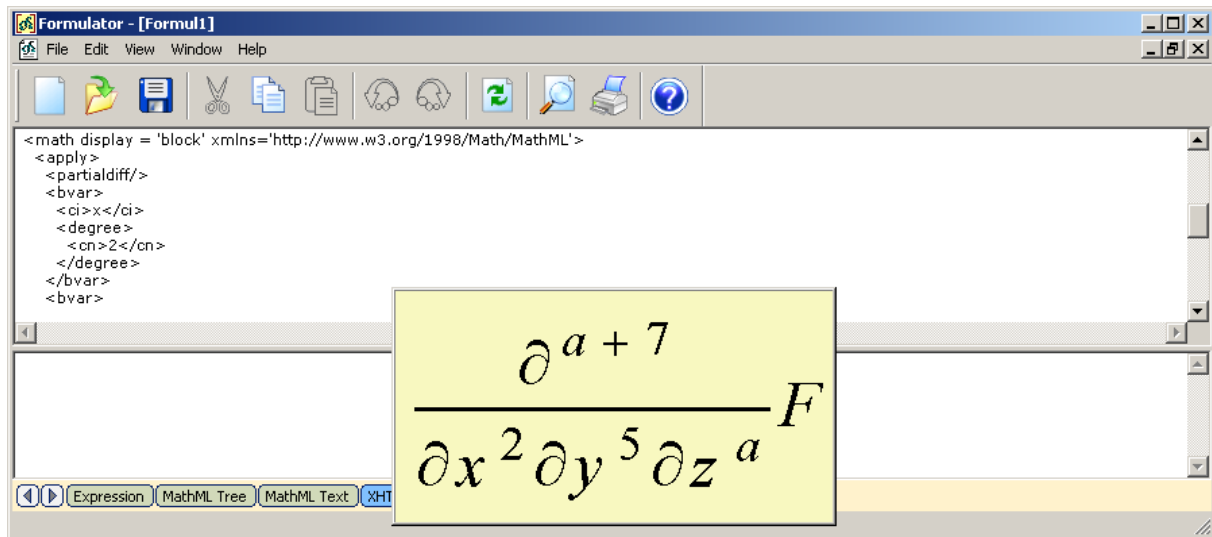
See results:



Turn on the “Show Read-Only” option from the “View” menu to see which areas can’t be edited in the created formula. The next figure suggests that automatically detected total degree of the “partialdiff” element can’t be edited (dashed red line around elements).

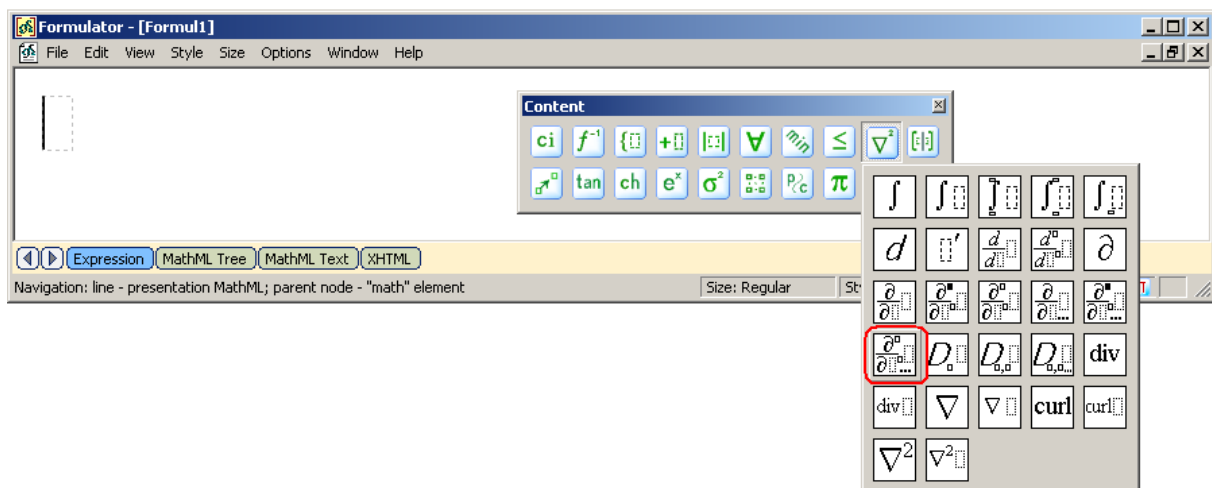


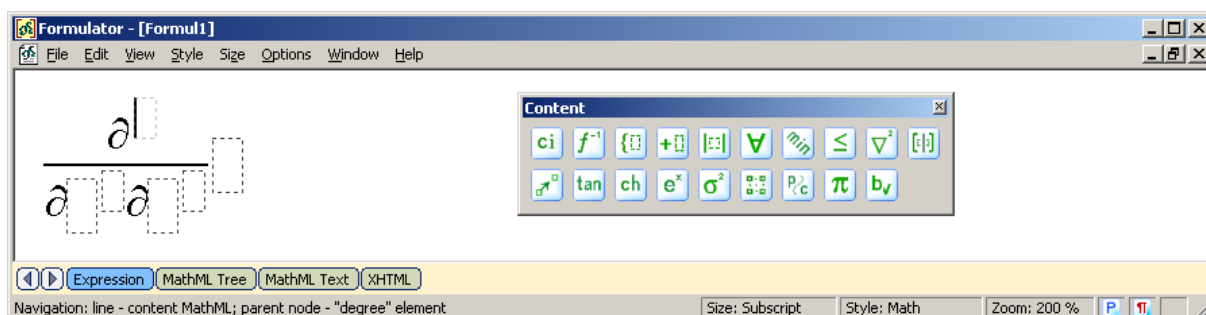
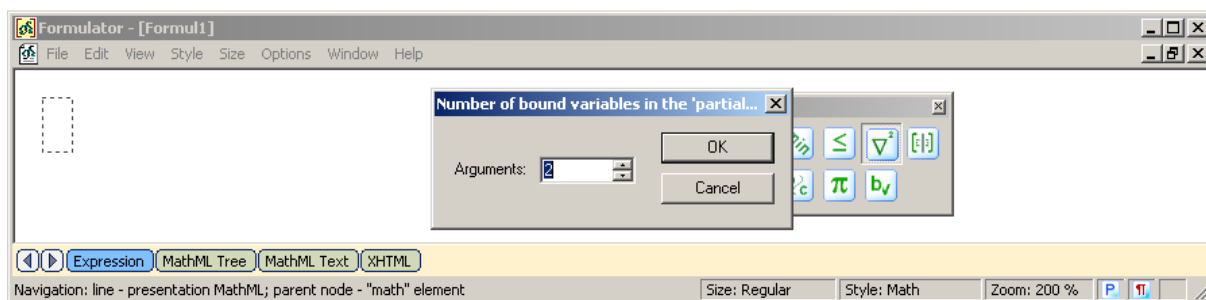
See the same results on the “XHTML” page of MathML Weaver, using the ‘Zoom’ feature from the context menu.



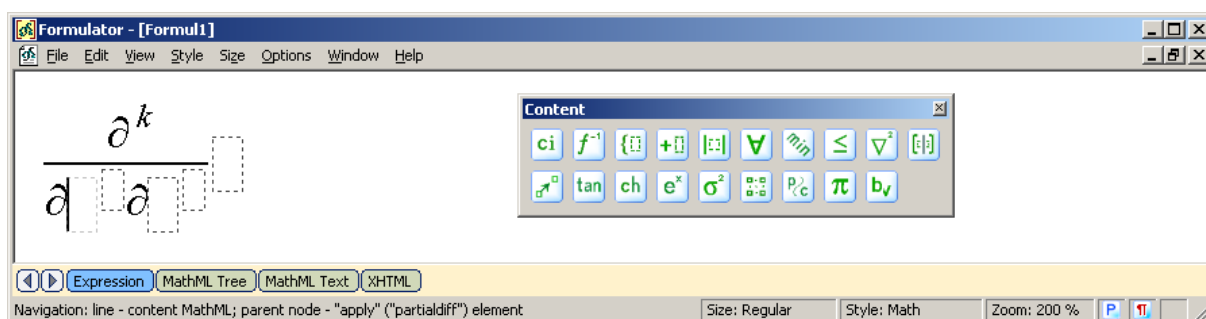
2. Example 4.4.5.3.2 from the W3C MathML Recommendation): the partialdiff element:

```
<apply><partialdiff/>
<bvar><ci> x </ci><degree><ci> m </ci></degree></bvar>
<bvar><ci> y </ci><degree><ci> n </ci></degree></bvar>
<degree><ci> k </ci></degree>
<apply><ci type="function"> f </ci>
<ci> x </ci>
<ci> y </ci>
</apply>
</apply>
```

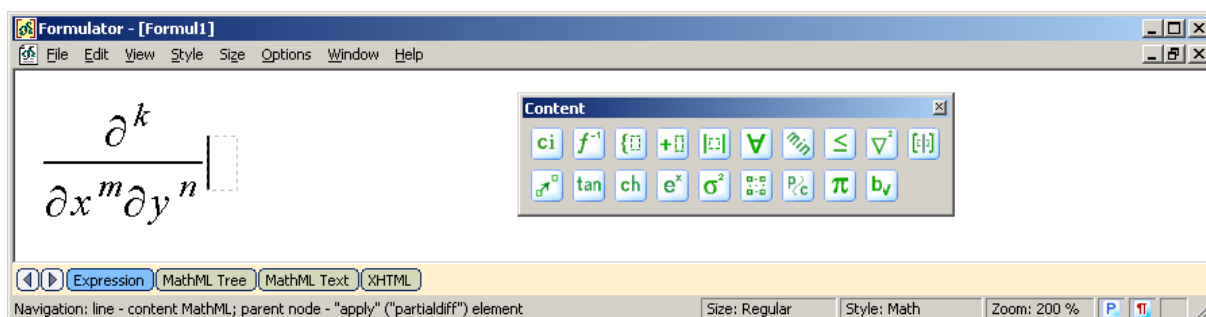


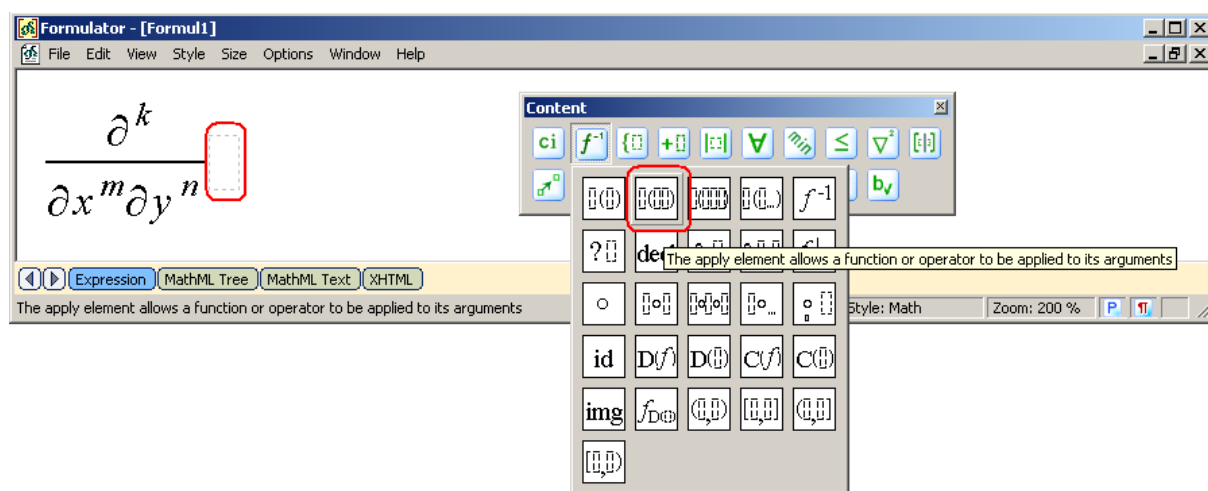


Input a total degree of differentiation by typing 'k'. Press the Right arrow to start inputting bounded variables.

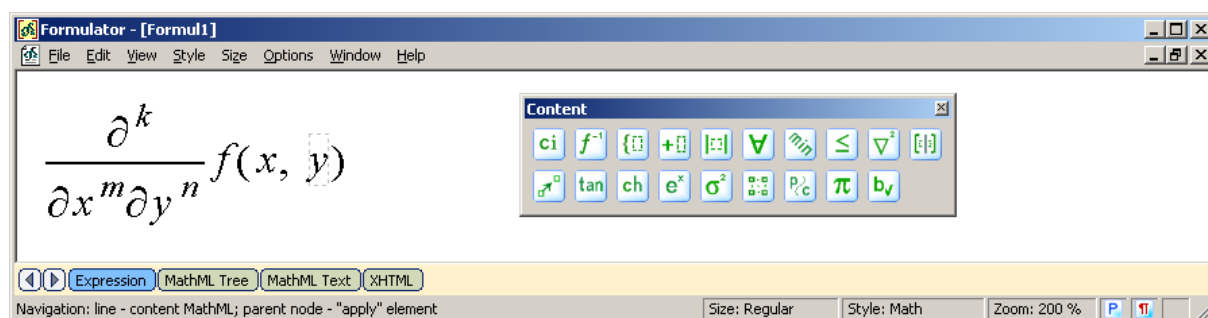


Press 'x', the Right arrow, 'm', the Right arrow, 'y', the Right arrow, 'n', the Right arrow.

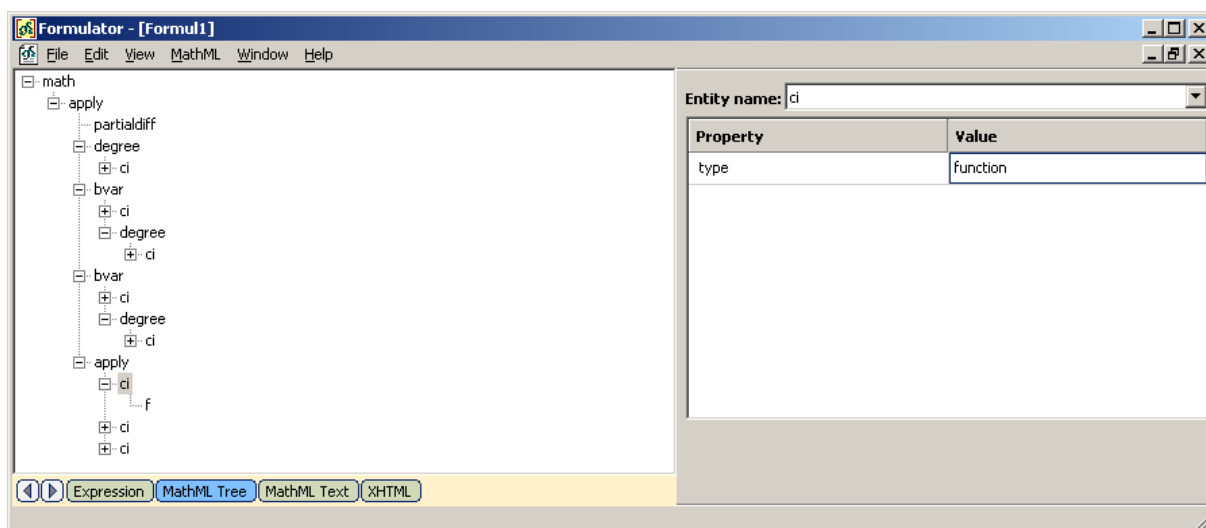
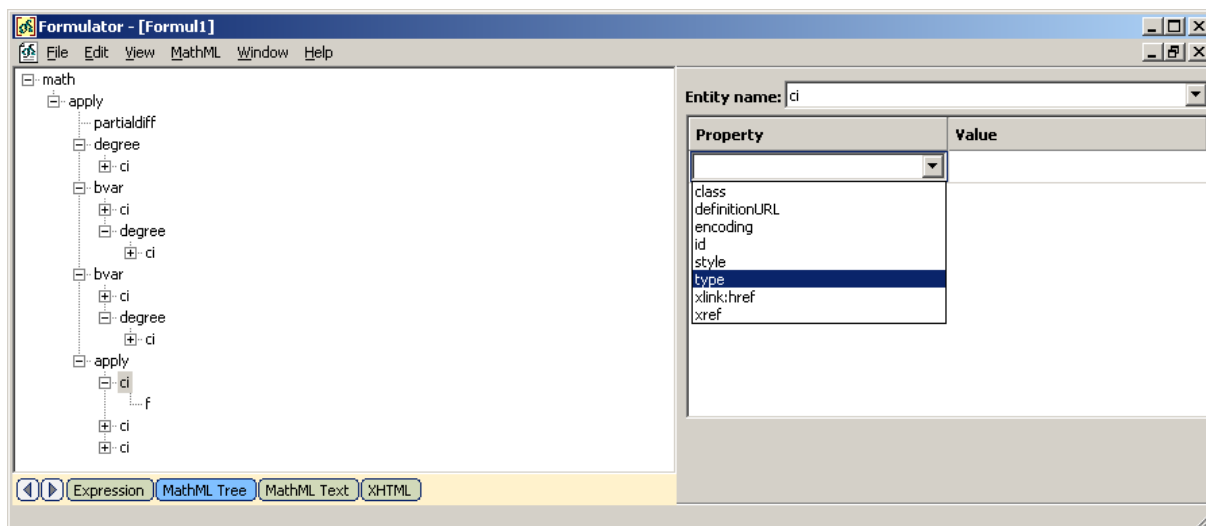




Press 'f', the Right arrow, 'x', the Right arrow, 'y'.



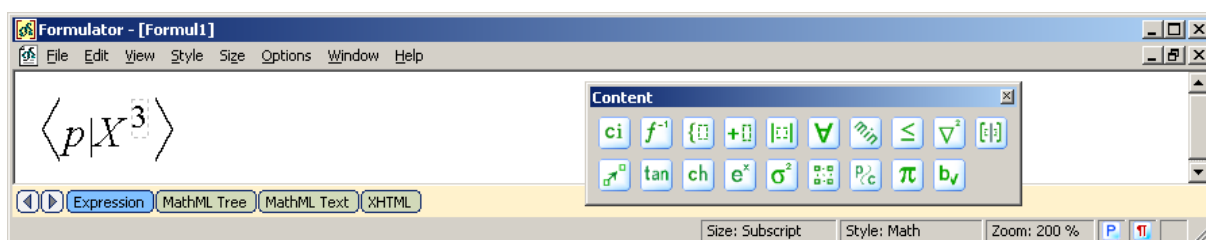
Switch to the "MathML Tree" and add the "type" attribute to the function 'f'.



Statistics

Lets' consider an example 4.4.9.6.2 from the W3C MathML Recommendation):
the partialdiff element:

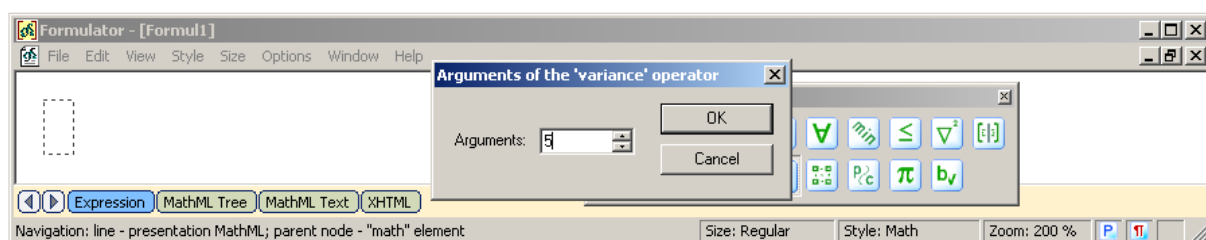
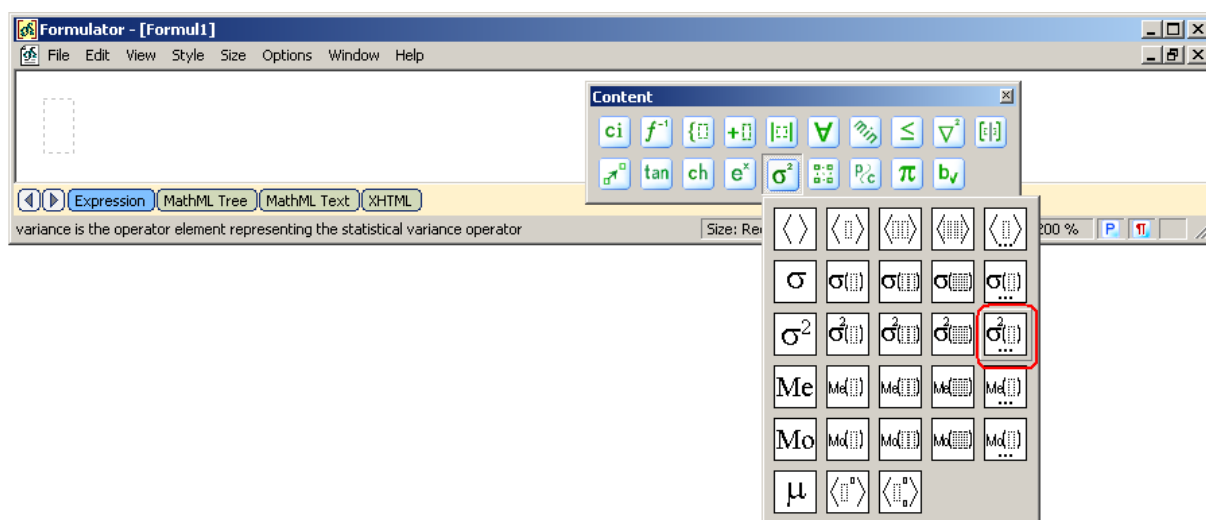
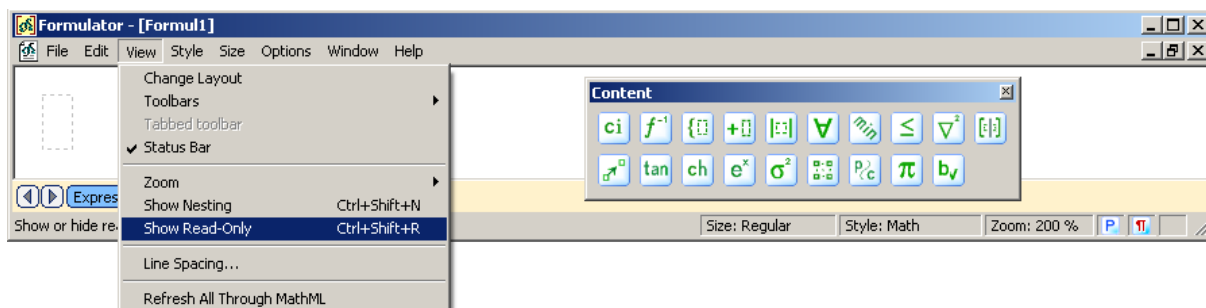
```
<apply>
  <moment/>
  <degree><cn> 3 </cn></degree>
  <momentabout>
    <ci> p </ci>
  </momentabout>
  <ci> X </ci>
</apply>
```



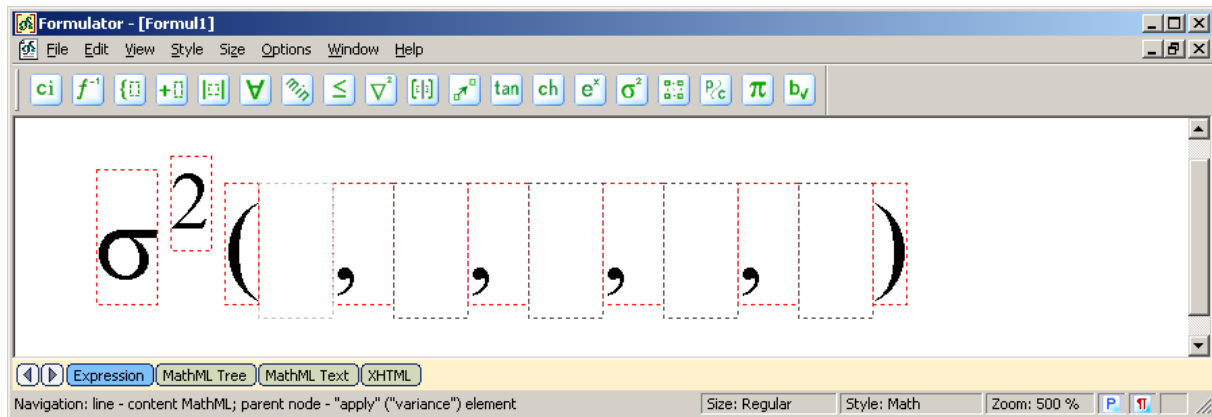
Read-only elements

Create the “variance” element with 5 arguments and see which elements are read-only in the visual editing form.

Turn on the “Show Read-Only” option from the “View” menu.

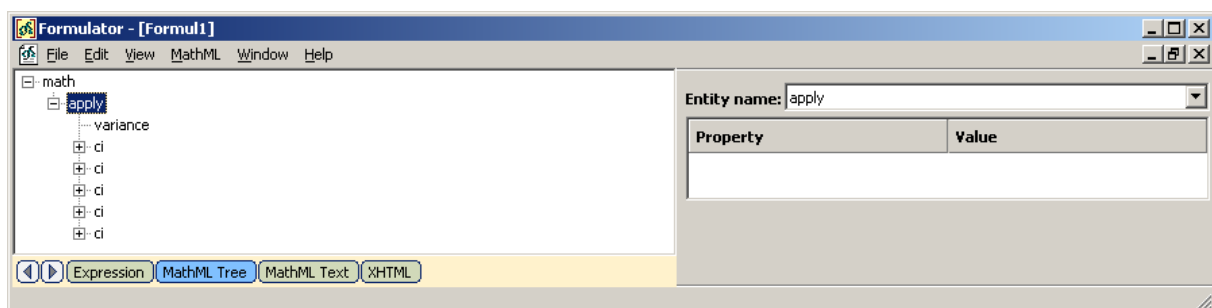
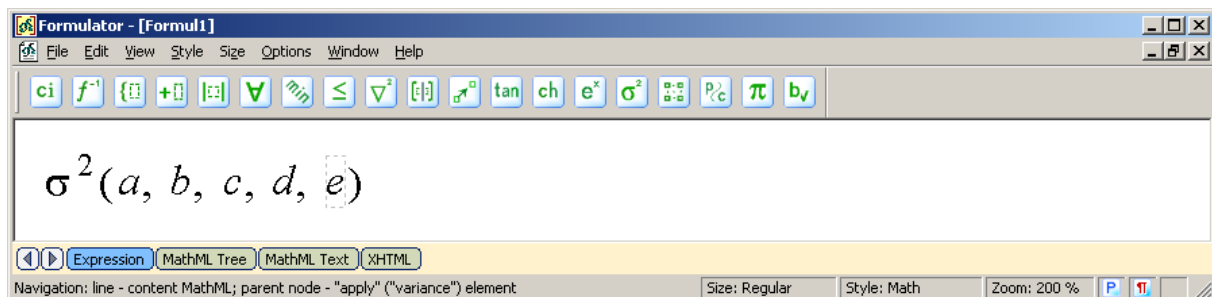


See how MathML Weaver make it easy to navigate through the formula by marking some MathML nodes as *read-only*.



Red dashed line around an element means that this element can't be edited, because it is an essential part of a visual editing form for some element of the content markup. When a user presses navigation buttons (e.g., arrows), MathML Weaver omits visiting every element of the form, but rather chooses only not read-only elements. Due to this smart behavior a user can navigate through the MathML tree much faster, without positioning on every tree hierarchy level and every existing node.

Finish editing the "variance" element and see results.



Combining Presentation and Content Markup

MathML Weaver is able to combine Presentation and Content Markup of MathML in three ways:

1. Presentation elements can be inserted into the input slots of editing form of the content markup. When the presence of such combining seems inappropriate by W3C Recommendation, presentation elements are wrapped into the "csymbol" an

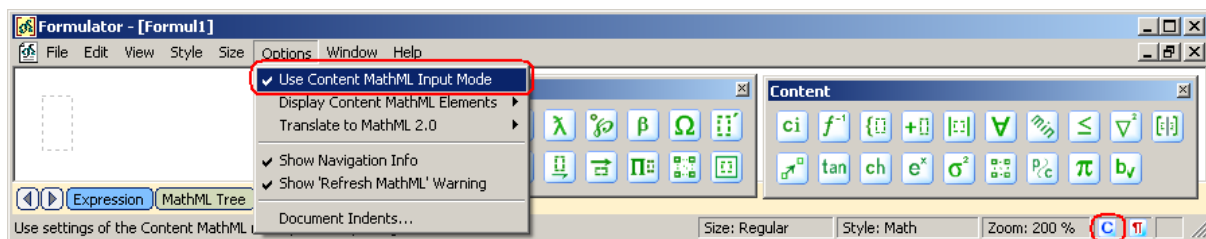
element, because MathML Weaver consider such cases as definition on-the-fly of an element in MathML whose semantics are externally defined.

2. The simplest way how content markup can be contained in presentation markup is just simultaneously using the Presentation and Content mathematical toolbars, thus embedding content markup nodes into the existing presentation hierarchy of elements. Note that the resulting expression should still have an unambiguous rendering.

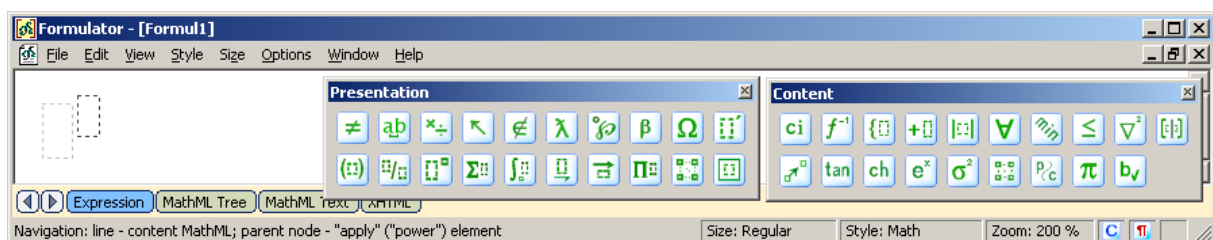
An example from the 5.2.1 section of the W3C Recommendation.

```
<mrow>
  <apply>
    <power/>
    <ci>x</ci>
    <cn>2</cn>
  </apply>
  <mo>+</mo>
  <msup>
    <mi>v</mi>
    <mn>2</mn>
  </msup>
</mrow>
```

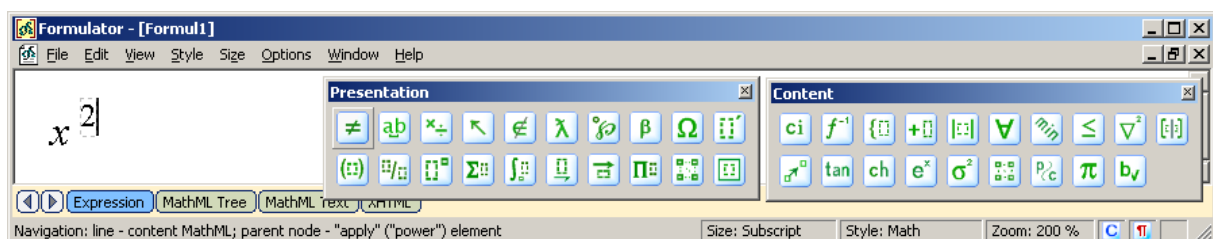
Switch to the Content MathML input mode.



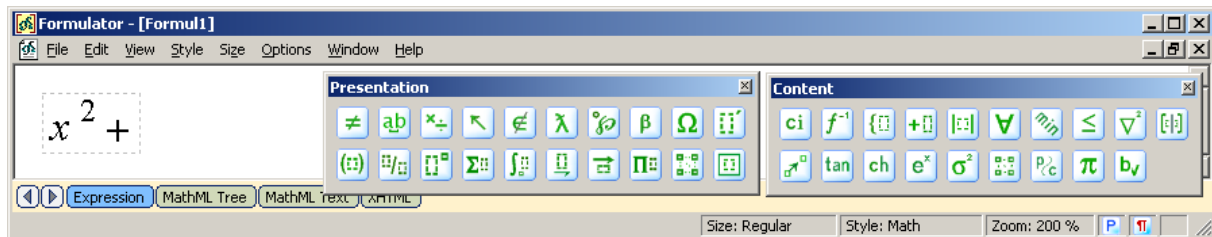
Type '^' to insert the "apply" element with the <power/> operation.



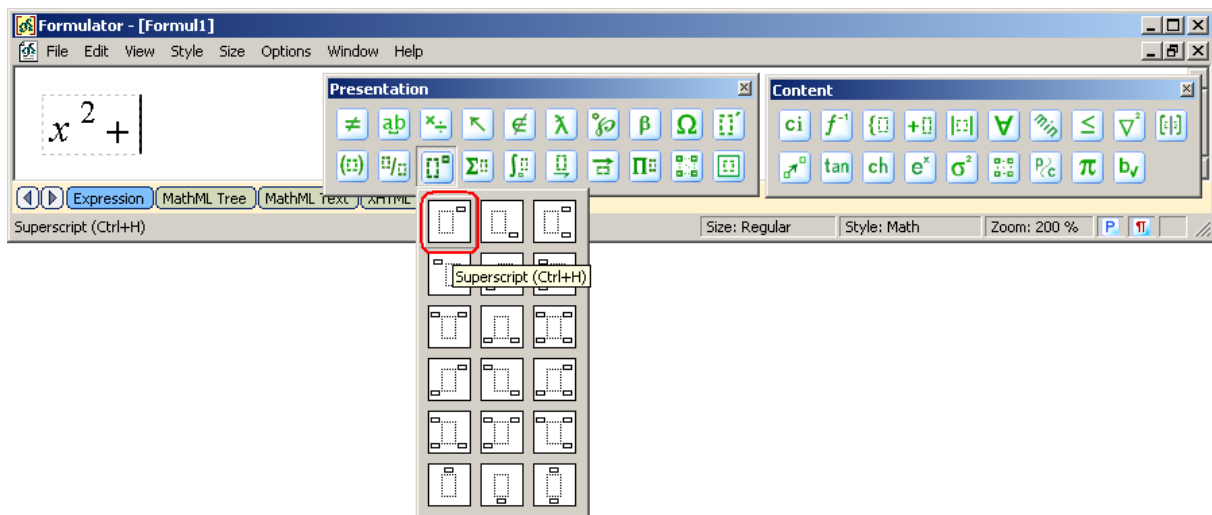
Press 'x', the Right arrow, '2'.



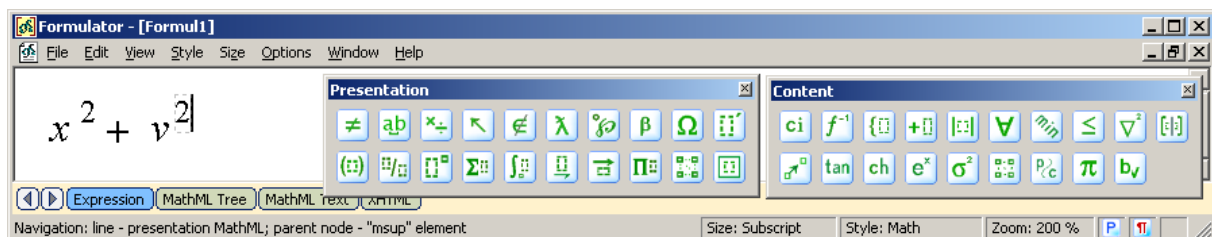
Switch to the Presentation MathML input mode; type ‘+’.



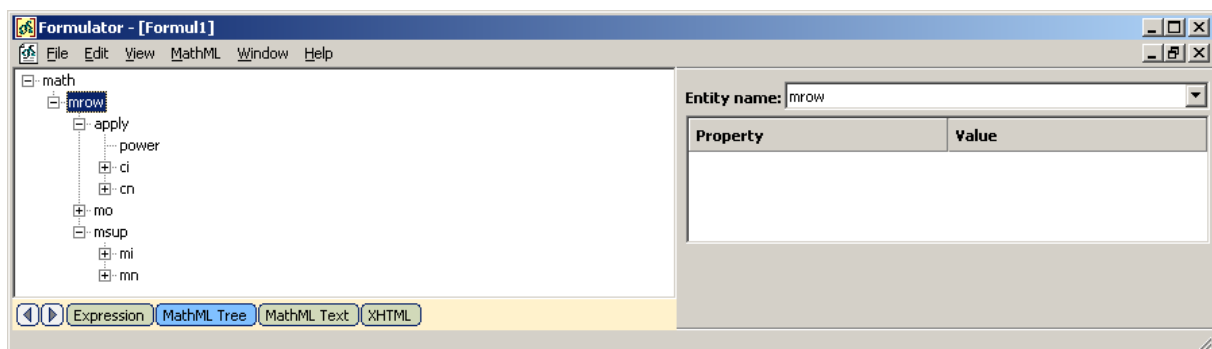
Press the Right arrow to get out of the “apply” element; using presentation toolbar insert the superscript MathML node.

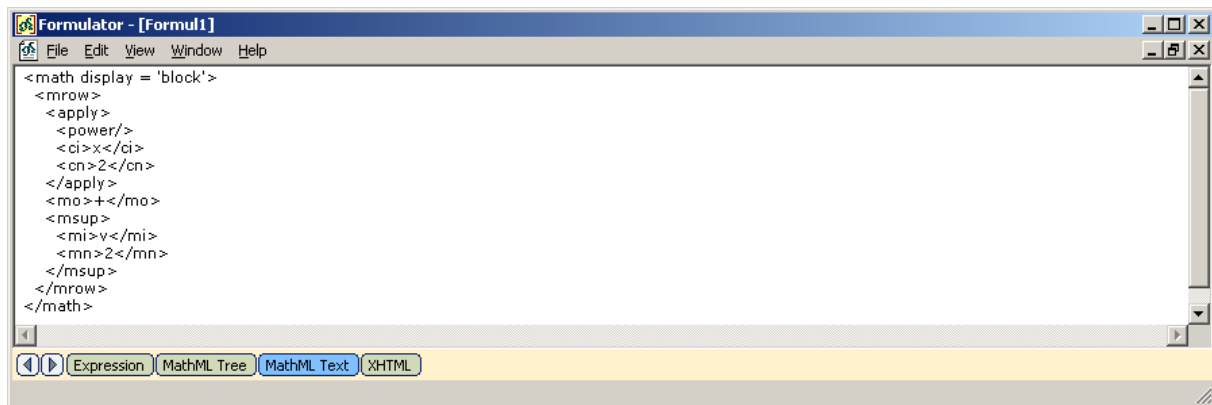



Press ‘v’, the Right arrow, ‘2’.



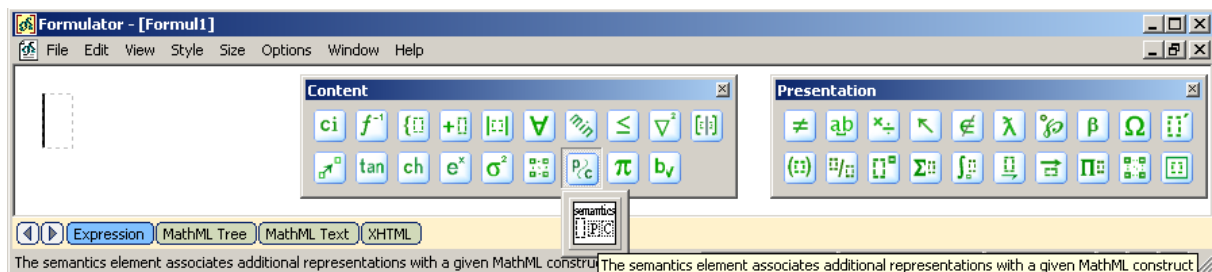
See the resulting MathML tree and text.



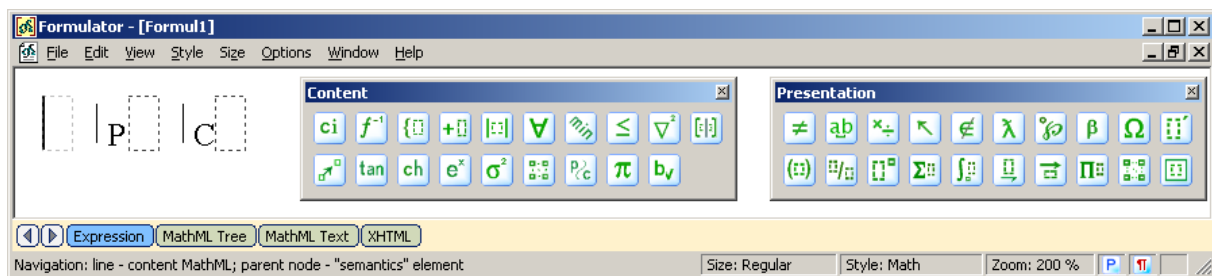


3. By using Semantic Mapping Elements from the  toolbar content markup can be contained in presentation markup as a *Parallel Markup*, thus making use of both presentation and content information for the same element.

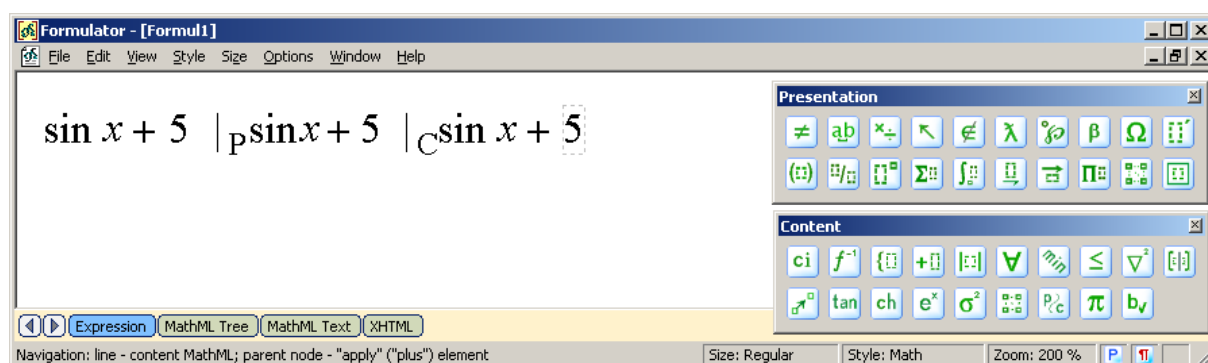
Insert the “semantics” element.



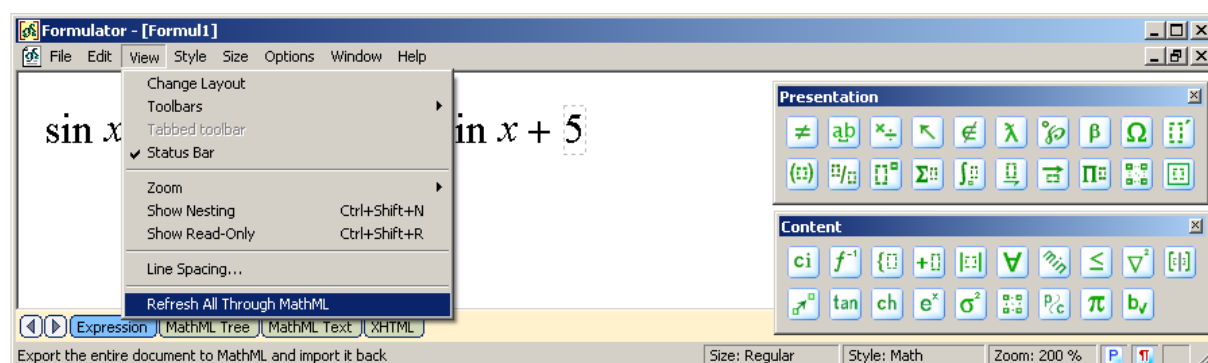
Its rendering can't be correct right away, since a user should input needed values of child “annotation” elements.



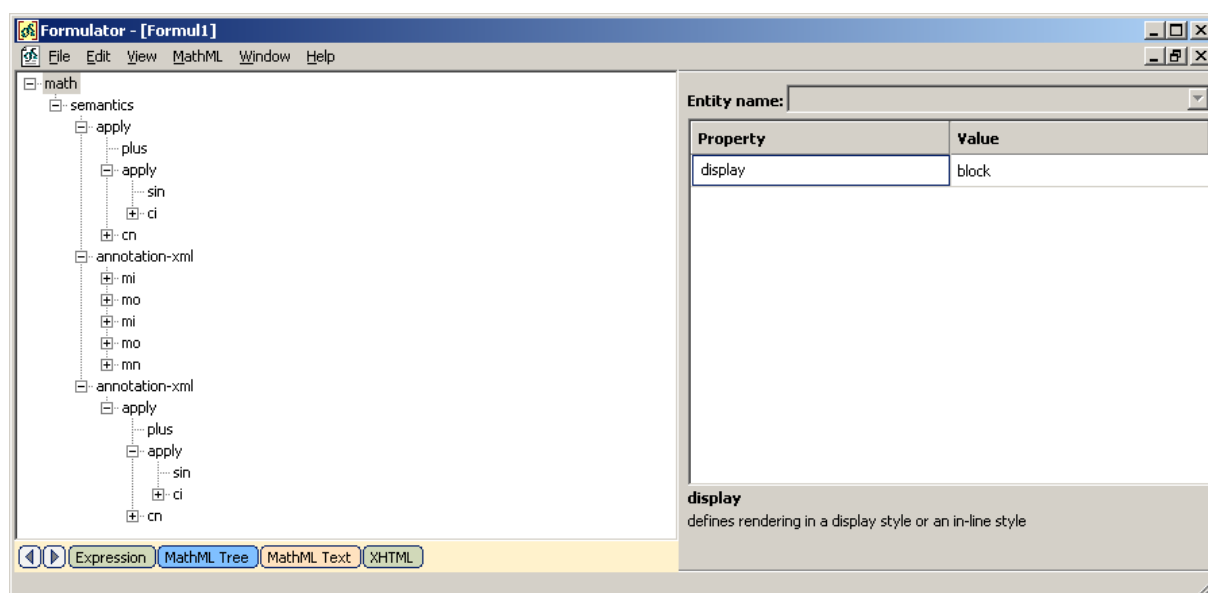
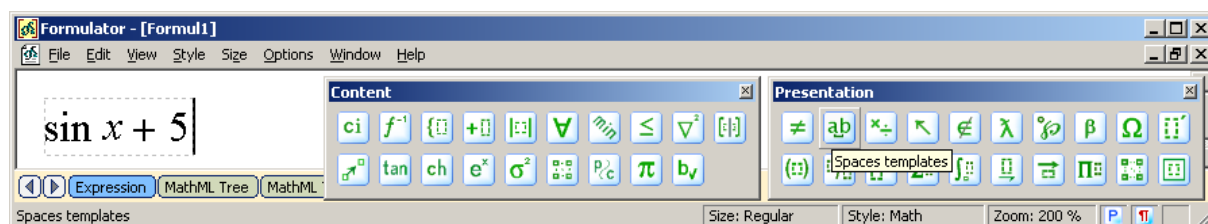
Create “ $\sin(x) + 5$ ” expression using different markups.

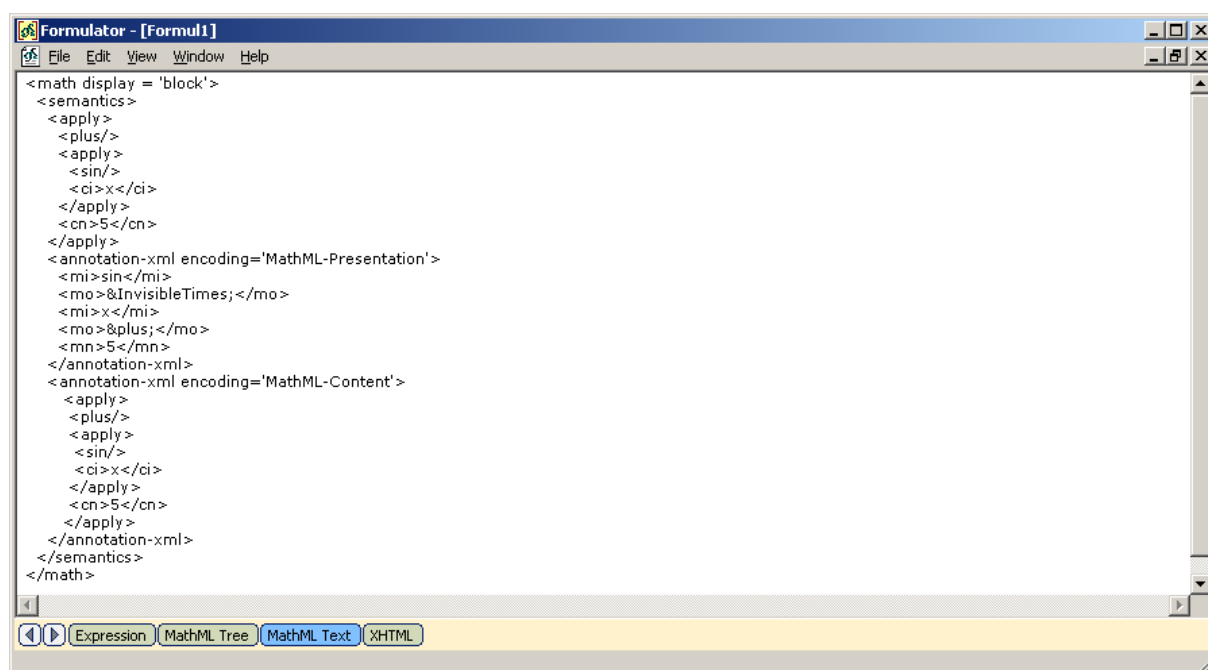


Refresh appearance of the document.



See results on different pages of MathML Weaver.





Tutorial: Formulator ActiveX Control

There is a special edition of our software for people seeking to develop new software that is acquainted with presentation or content side of mathematics. This component edition, Formulator ActiveX Control, is the perfect way for software developers to insert mathematics rendering/editing/processing functionality in their applications.

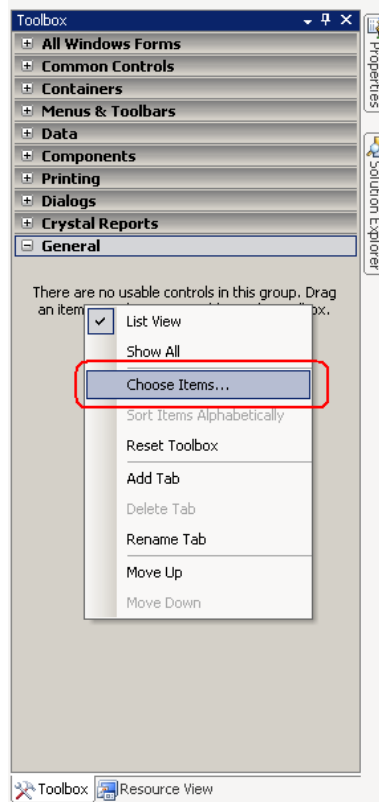
As in the case of Formulator MathML Weaver and Formulator Express products, Hermitech Laboratory provides both commercial and free versions of the component edition: Formulator ActiveX Control and Formulator Express ActiveX Control. The last version has all the restrictions of the Formulator Express (since it is built on the top of this limited free version). Additionally, Formulator Express ActiveX Control is limited in its API, providing only means of exporting mathematical expressions into MathML 2.0. In spite of the reductive level of functionality, Formulator Express ActiveX Control remains extremely useful software component. It allows developers to add a new value to their products by giving a software the competence in math and ready easy-to-use graphical interface that is keenly aware of the mathematical typesetting and semantics rules.

The following tutorial briefly describe how to make use of Formulator ActiveX Control by developing a very simple Visual C# Windows Application, capable of:

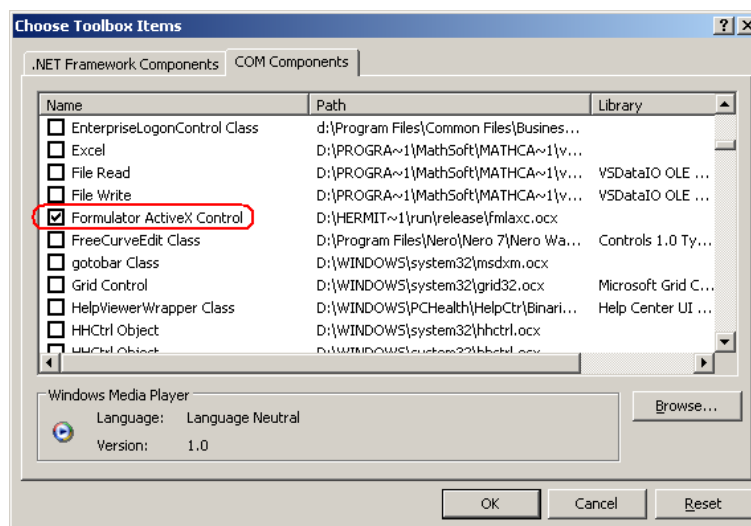
- editing mathematical formulas,
- importing them from MathML 2.0, and
- exporting formulas into MathML 2.0, raster and vector graphics.

This example is part of the standard samples pack, distributing along with Formulator ActiveX Control.

1. On the first step we add a custom Toolbox item for Formulator ActiveX Control.

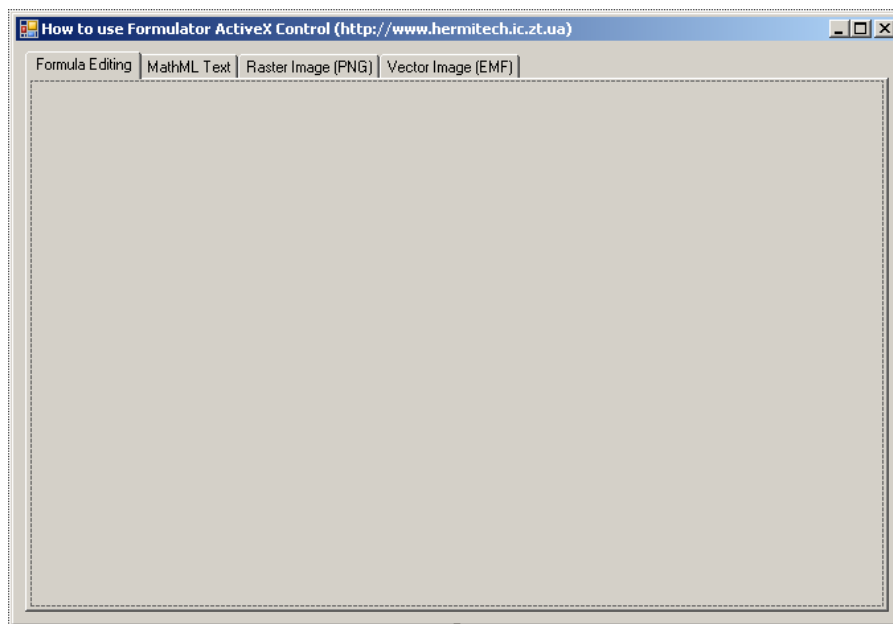


Microsoft Visual Studio environment shows the dialog “Choose Toolbox Items”. Switch to the tab page “COM Components” and in the list of all software components registered on your computer choose the entry for Formulator ActiveX Control.

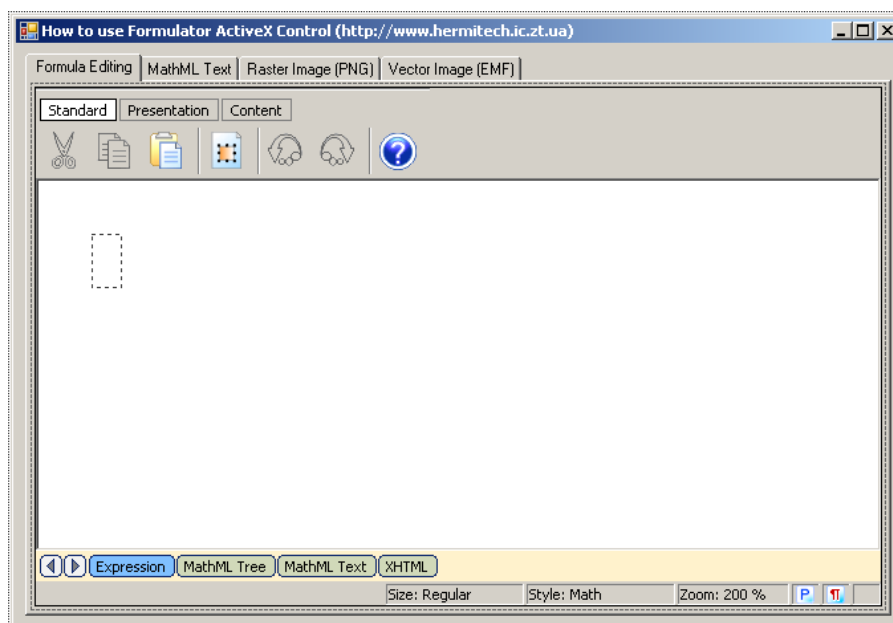


Now there is a new item on the Toolbox. You can use it anytime when you want to enrich your application with mathematical editing and MathML processing functionality.

2. In this example we plan to have a dialog with a tab control and four tab pages. We use a separate tab page for editing mathematics, viewing and editing plain MathML 2.0 text, viewing a raster image of the just typed formula, viewing the image in the vector format (EMF).



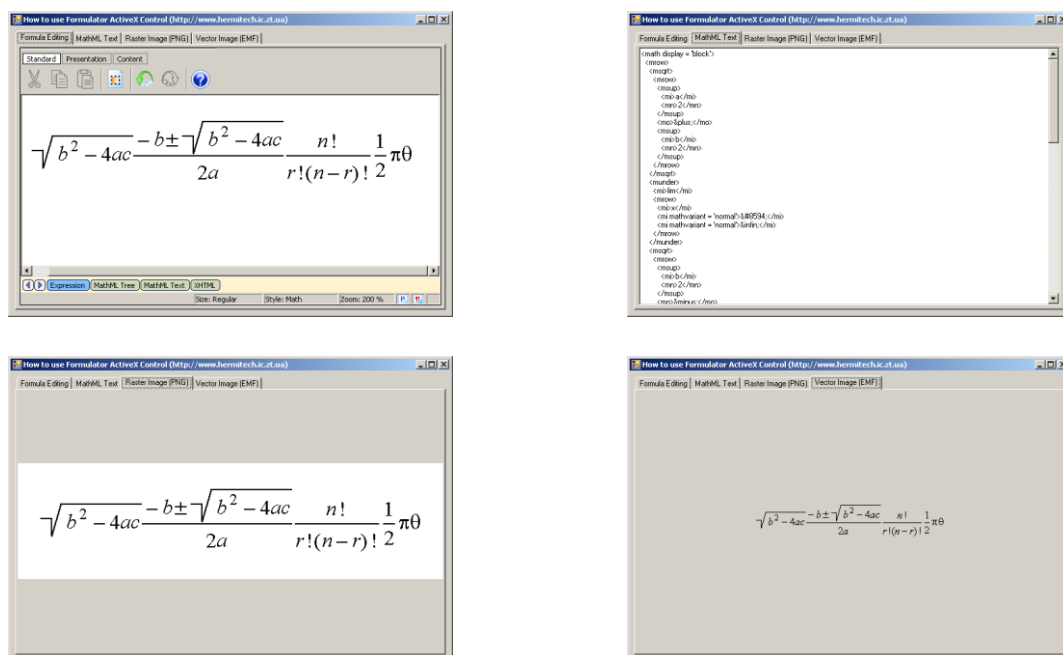
3. Now using the corresponding item of the Toolbox, place Formulator ActiveX Control on the first tab page of the form.



4. Our application is almost ready and the only thing we should do now is to select an event for import/export actions (e.g., switching tab pages of the form) and to call the needed methods of Formulator ActiveX Control (*SetMathML()*, *GetMathMLStr()*, *ExportImage()*, *ExportEMF()*).

```
protected int PreviousSelectedIndex = -1;    // the last selected tab page
protected int PreviousSelectedIndex = -1;    // the last selected tab page
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
    switch ( tabControl1.SelectedIndex )
    {
    case 0:
        // check whether our ActiveX is licensed as the Express Edition
        // (free version with limited functionality)
        if ( PreviousSelectedIndex == 1 && axFmlax1.GetFormulatorLicenseInfo() != 1 )
        {
            // update formula according to the contents of the MathML editing page
            axFmlax1.SetMathML( richTextBox1.Text );
        }
        break;
    case 1:
        // check whether our ActiveX is licensed as the Express Edition
        // (free version with limited functionality)
        if ( axFmlax1.GetFormulatorLicenseInfo() == 1 )
        {
            richTextBox1.ReadOnly = true;
        }
        // update MathML 2.0 text according to the edited formula
        richTextBox1.Text = axFmlax1.GetMathMLStr();
        break;
    case 2:
        // check whether our ActiveX is licensed as the Express Edition
        // (free version with limited functionality)
        if ( axFmlax1.GetFormulatorLicenseInfo() == 1 )
        {
            pictureBox1.Enabled = false;
        }
        else
        {
            // create a raster image for the mathematical formula
            // type of the image file is detected by the given file extension
            axFmlax1.ExportImage("out.png");
            pictureBox1.Load("out.png");
        }
        break;
    case 3:
        // check whether our ActiveX is licensed as the Express Edition
        // (free version with limited functionality)
        if ( axFmlax1.GetFormulatorLicenseInfo() == 1 )
        {
            pictureBox1.Enabled = false;
        }
        else
        {
            // create a vector image for the mathematical formula
            axFmlax1.ExportEMF("out.emf");
            pictureBox2.Load("out.emf");
        }
        break;
    }
    PreviousSelectedIndex = tabControl1.SelectedIndex;
}
```


5. Start our application and see how it works.



6. In several very simple steps we get ready to use application that has features of a professional mathematical editor. Our example shows how Formulator ActiveX Control helps to rapidly develop a software that is aware of the mathematical typesetting and semantics rules and is a good start for projects in such a different domains as computer-aided education, computer algebra systems, authoring tools, and many other applications for mathematics, science, business, economics, etc.

Working with Mathematical Web Pages in Formulator

Exporting expressions to XHTML web pages with the MathML markup

Ability to publish mathematics on web is provided in Formulator 3.9 MathML Weaver by following to W3C recommendation to have web pages written using XHTML with the MathML markup inlined. This feature is available from the menu command or by means of an additional mode of editing mathematical expressions. The resulting web page can be easily viewed with modern popular internet browsers, either using additional plug-ins (e.g., Hermitech's own plug-in for Internet Explorer, known also as "Formulator MathML IE Performer", which is based on Formulator), or by browser's native methods.

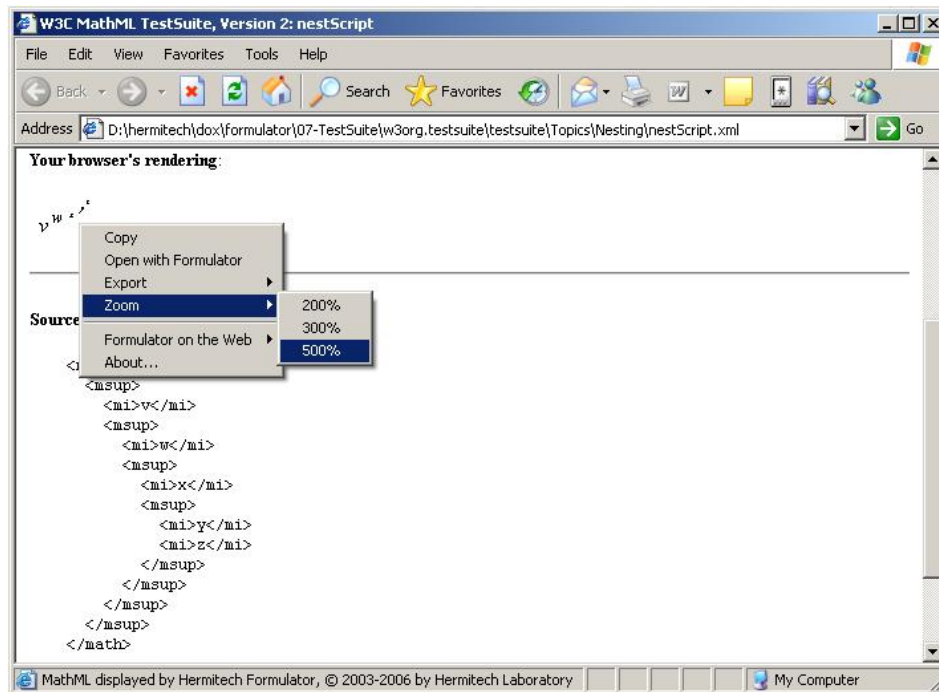
It's worthy of notice that when using the full version of Formulator a user benefits from several modes of editing mathematical expressions. The list of such modes is given at the bottom of a document and includes "Expression", "MathML Tree", "MathML Text" and "XHTML" options. The last option shows an easy way to deal with XHTML web pages with the MathML markup inlined. A user can switch between the "Expression" option and the "XHTML" option, editing mathematical expressions and viewing the results on-the-fly.

Viewing XHTML web pages with the MathML markup

Formulator enhances Internet Explorer (versions 5.5 and later) to display mathematical notation within a web page. If the math in a web page is written in the MathML language, Internet Explorer gives it to Hermitech's plug-in for Internet Explorer, known also as "Formulator MathML IE Performer", to display the page using the standard math notation. Formulator MathML IE Performer is a part of Formulator 3.9 MathML Weaver, and so it is installed by default along with Formulator.

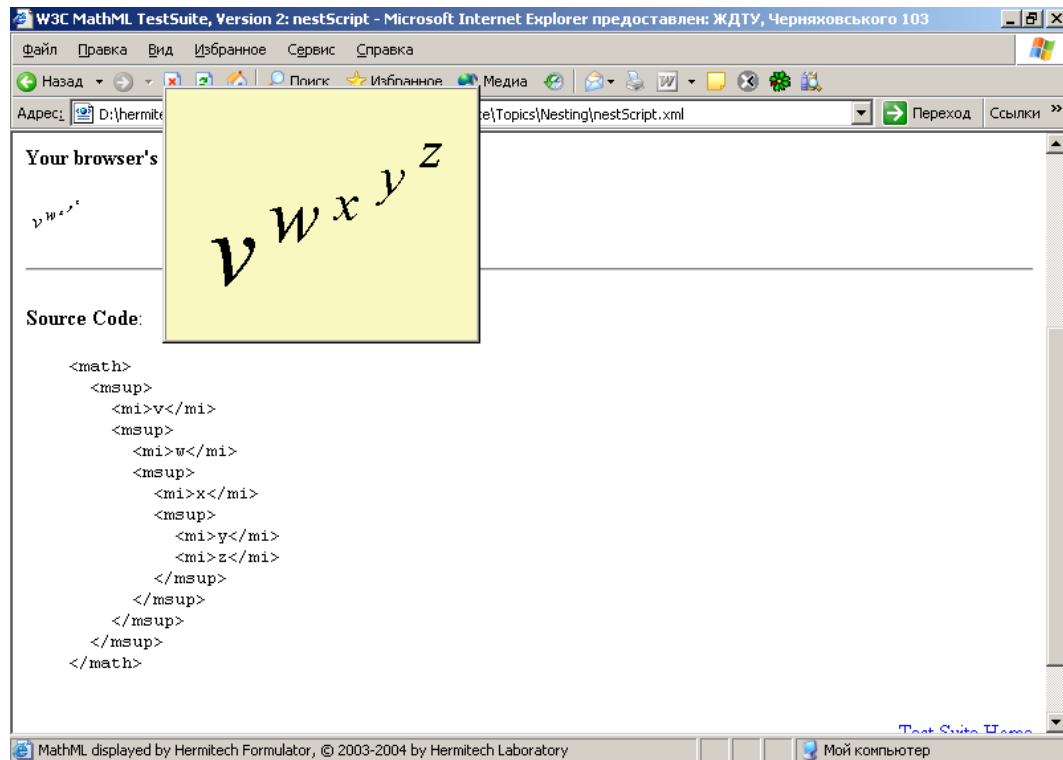
Another way to view XHTML web pages with MathML inlined is to freely download Formulator MathML IE Performer and to install it separately of Formulator 3.9 MathML Weaver. Formulator MathML IE Performer is a free software.

When displayed the equation within a web page can be accessed by placing the mouse pointer over an equation and clicking the right mouse button. This brings up Formulator's shortcut menu:



In addition to the commands operating with the clicked-on equation, there are commands to view Formulator's information, version number and copyright and to visit the Formulator web site.

- The Copy MathML command copies the MathML equivalent of the equation to the Clipboard. After this action the MathML text can be pasted into a text editor or computer algebra system.
- The Open with Formulator command opens the equation in a new Formulator window.
- The Exports command translates the current version of the equation that you're working on into a graphic file or a web-page.
- Choose the Zoom command to change the viewing scale and get a closer look at the equation. This feature can be extremely useful when you examine view small scripts and accents. To bring the equation back down to normal size just use a single mouse click or press ESCAPE:



Following to W3C recommendation [<http://www.w3.org/Math/XSL/>], in order to render math on the web page with the help of Formulator, the web page should be written using XHTML with the MathML markup inlined, as in the following example:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="pmathml.xsl"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    ....
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      ....
    </math>
  </body>
</html>
```

Please note that you can use just `pmathml.xsl` stylesheet since Formulator doesn't support Content Markup yet. Make sure that 'href' attribute in the 'xml-stylesheet' declaration points to actual location of `pmathml.xsl` stylesheet.

Since your browser can have more than one way to render MathML markup, there is a way to specify that your preferred method is using a Formulator. This is achieved through the use of an attribute 'renderer' belonging to the 'pref' namespace:

```
<?xml-stylesheet type="text/xsl" href="pmathml.xsl"?>

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:pref="http://www.w3.org/2002/Math/preference"
      pref:renderer="hermitech">
  <head>...</head>
```

```
<body>...</body>  
</html>
```

Values for the renderer attributes here is "hermitech" and that means that you prefer to use Formulator for viewing web pages containing math.

Please note that an adapted version of pmathml.xsl stylesheet is included in the installation package of Formulator. You can find this changed stylesheet at the following path:

```
...\\Hermitech Laboratory\\Formulator\\Template\\bin\\pmathml.xsl
```

where ... denotes folder to which Formulator is installed (e.g., "Program Files").

Formulator Conformance with MathML 2.0

Formulator conforms to the Mathematical Markup Language (MathML) version 2.0 (W3C recommendation, second edition, 21 October 2003). The following tables show the implemented MathML elements and attributes:

- $[+]$ in the cell means that the element or attribute is implemented;
- $[\pm]$ means that the element or attribute is partially implemented;
- $[-]$ means that the element is not implemented or attribute is saved, but not used.

Implemented Elements and Attributes

General

Element	Support	Comments / Attributes
$\langle\text{math}\rangle$	+	The 'display' attribute is effective when rendering MathML in the Internet Explorer browser.

Presentation: Token Elements

Element	Support	Comments / Attributes
Mathematics style attributes (mathvariant, mathsize, mathcolor, mathbackground) are used both for rendering (when importing MathML) and for exporting formulas to MathML 2.0. Deprecated style attributes (fontsize, fontweight, fontstyle, fontfamily, color) are recognized and used for rendering by converting them to the four above mentioned style attributes. For the 'mathvariant' attributes there is no check whether needed for rendering font is present in a system.		
$\langle\text{mi}\rangle$	+	MathML 2.0 specification advices to render identifiers depending on their content: as 'italic' when the content is a single character and as 'normal' otherwise. Since the default editing model of Formulator is to keep user choice of a specific font and character style for an identifier regardless of its content, it is usual for multicharacter 'mi' elements to have 'mathvariant' attribute set to 'italic'.
$\langle\text{mn}\rangle$	+	
$\langle\text{mo}\rangle$	+	'maxsize' and 'minsize' attributes do not effect on rendering formulas.
$\langle\text{mtext}\rangle$	+	
$\langle\text{mspace}\rangle$	\pm	
$\langle\text{ms}\rangle$	+	
$\langle\text{mglyph}\rangle$	+	The 'alt' attribute is used for substitution of the element body in the case when needed font can't be applied. There are no checks whether a position given by the 'index' attributes is valid for the specified font, so it may be that user see blank rectangle instead of the expected symbol.

Presentation: General Layout

Element	Support	Comments / Attributes
<mrow>	+	
<mfrac>	+	
<msqrt>	+	
<mroot>	+	
<mstyle>	±	For further editing purposes in the internal document model 'mstyle' element is implemented as insertion of an additional frame (slot) which contents are set according to children of the 'mstyle' element. Among own special attributes of 'mstyle' element only 'background' takes effect. In order to preserve formatting even if a child of the 'mstyle' element is removed from this extra frame node, attributes of every child element in the 'mstyle' are set as a combination of explicitly set parameters and inherited parameters (own parameters have higher priority).
<merror>	+	
<mpadded>	±	If 'height' and 'depth' attributes lead to lessening of the vertical formula size, they are ignored.
<mphantom>	+	
<mfenced>	+	
<menclose>	+	Supports all notations which are stated in the current MathML 2.0 specification (longdiv, actuarial, radical, box, roundedbox, circle, left, right, top, bottom, updiagonalstrike, downdiagonalstrike, verticalstrike, horizontalstrike). Additionally several new values are implemented according to the content of Formulator's mathematical toolbars (joint-status, strike-through, top-left, top-right, bottom-left, bottom-right). Their effect on 'menclose' element contents is the same as of Formulator's buttons with corresponding names (toolbars 'Underbar and overbar templates' and 'Box templates').

Presentation: Scripts and Limits

Element	Support	Comments / Attributes
<msub>	+	Except for the 'subscriptshift' attribute.
<msup>	+	Except for the 'superscriptshift' attribute.
<msubsup>	+	Except for the 'subscriptshift' attribute and the 'superscriptshift' attribute.
<munder>	+	Except for the 'accentunder' attribute.

<code><mover></code>	+	Except for the 'accent' attribute.
<code><munderover></code>	+	Except for the 'accentunder' attribute and the 'accent' attribute.
<code><mmultiscripts></code>	+	

Presentation: Tables and Matrices

Element	Support	Comments / Attributes
<code><mtable></code>	±	Attributes which relates to cell size and spacing are not processed.
<code><mtr></code>	±	The 'groupalign' attribute is not processed.
<code><mlabeldtr></code>	–	This element is treated simply as 'mtr' element.
<code><mtd></code>	±	The 'groupalign' attribute, the 'rowspan' attribute and the 'colspan' attribute are not processed.
<code><malignngrop></code>	–	
<code><malignngmark></code>	–	

Presentation: Enlivening Expressions

Element	Support	Comments / Attributes
<code><maction></code>	–	

Content: Token Elements

Element	Support	Comments / Attributes
<code><cn></code>	+	
<code><ci></code>	+	
<code><csymbol></code>	+	

Content: Relations

Element	Support	Comments / Attributes
<code><eq></code>	+	
<code><neq></code>	+	
<code><gt></code>	+	
<code><lt></code>	+	
<code><geq></code>	+	
<code><leq></code>	+	
<code><equivalent></code>	+	
<code><approx></code>	+	
<code><factorof></code>	+	

Content: Sequences and Series

Element	Support	Comments / Attributes
<sum>	+	
<product>	+	
<limit>	+	
<tendsto>	+	

Content: Basic Content Elements

Element	Support	Comments / Attributes
<apply>	+	
<reln>	+	
<fn>	+	
<interval>	+	
<inverse>	+	
<sep>	+	
<condition>	+	
<declare>	+	This element usually is not rendered (by W3C MathML 2.0 recommendation), but there are two cases when a user can access it. The first is connected with a moment of insertion of the 'declare' element using Formulator's mathematical toolbars. The second case is when a user explicitly commands Formulator to show invisible elements ('declare', 'momentabout', 'annotation-xml' and in some expressions 'bvar').
<lambda>	+	
<compose>	+	
<ident>	+	
<domain>	+	
<codomain>	+	
<image>	+	
<domainofapplication>	+	
<piecewise>	+	
<piece>	+	
<otherwise>	+	

Content: Theory of Sets

Element	Support	Comments / Attributes
<set>	+	
<list>	+	
<union>	+	

<intersect>	+	
<in>	+	
<notin>	+	
<subset>	+	
<prsubset>	+	
<notsubset>	+	
<notprsubset>	+	
<setdiff>	+	
<card>	+	
<cartesianproduct>	+	

Content: Arithmetic, Algebra and Logic

Element	Support	Comments / Attributes
<quotient>	+	
<factorial>	+	
<divide>	+	
<max>	+	
<min>	+	
<minus>	+	
<plus>	+	
<power>	+	
<rem>	+	
<times>	+	
<root>	+	
<gcd>	+	
<and>	+	
<or>	+	
<xor>	+	
<not>	+	
<implies>	+	
<forall>	+	
<exists>	+	
<abs>	+	
<conjugate>	+	
<arg>	+	
<real>	+	
<imaginary>	+	
<lcm>	+	
<floor>	+	
<ceiling>	+	

Content: Linear Algebra

Element	Support	Comments / Attributes
<vector>	+	
<matrix>	+	
<matrixrow>	+	
<determinant>	+	
<transpose>	+	
<selector>	+	
<vectorproduct>	+	
<scalarproduct>	+	
<outerproduct>	+	

Content: Calculus and Vector Calculus

Element	Support	Comments / Attributes
<int>	+	
<diff>	+	
<partialdiff>	+	
<lowlimit>	+	
<uplimit>	+	
<bvar>	+	<p>Rendering of this element is strongly depends on its context. There are several MathML constructions with 'bvar' element which don't presuppose its visual representation. Then a user can access and edit 'bvar' elements either at the moment of insertion of a MathML element using Formulator's mathematical toolbars, or by explicitly selecting an option to show invisible MathML elements.</p> <p>Note that after insertion of a MathML element containing invisible 'bvar' its rendering can differ from normative one in order to enable further editing a 'bvar' element. This will become normal after exporting expressions to MathML and importing them back to Formulator (by using 'Refresh All Through MathML' menu command or by switching to "MathML text" page, editing and return back).</p>
<degree>	+	
<divergence>	+	
<grad>	+	
<curl>	+	
<laplacian>	+	

Content: Constants and Symbol Elements

Element	Support	Comments / Attributes
<integers>	+	
<reals>	+	
<rational>	+	
<naturalnumbers>	+	
<complexes>	+	
<primes>	+	
<exponentiale>	+	
<imaginaryi>	+	
<notanumber>	+	
<>true>	+	
<>false>	+	
<emptyset>	+	
<pi>	+	
<eulergamma>	+	
<infinity>	+	

Content: Elementary Functions

Element	Support	Comments / Attributes
<exp>	+	
<ln>	+	
<log>	+	
<sin>	+	
<cos>	+	
<tan>	+	
<sec>	+	
<csc>	+	
<cot>	+	
<sinh>	+	
<cosh>	+	
<tanh>	+	
<sech>	+	
<csch>	+	
<coth>	+	
<arcsin>	+	
<arccos>	+	
<arctan>	+	
<arccosh>	+	
<arccot>	+	
<arccoth>	+	
<arccsc>	+	

<arccsch>	+	
<arcsec>	+	
<arcsech>	+	
<arcsinh>	+	
<arctanh>	+	

Content: Statistics

Element	Support	Comments / Attributes
<mean>	+	
<sdev>	+	
<variance>	+	
<median>	+	
<mode>	+	
<moment>	+	
<momentabout>	+	

Content: Semantic Mapping Elements

Element	Support	Comments / Attributes
<semantics>	+	<p>The default rendering of a 'semantics' element is the default rendering of its first child. Other children elements are contained, but usually are not rendered. There are two cases when a user can access and edit 'annotation-xml' child element. The first is connected with a moment of insertion of the 'semantics' element using Formulator's mathematical toolbars. The second case is when a user explicitly commands Formulator to show invisible elements ('declare', 'momentabout', 'annotation-xml' and in some expressions 'bvar').</p> <p>Note that after insertion of a 'semantics' element its rendering is non-normative in order to enable editing of 'annotation-xml' child elements for 'Presentation' and 'Content' encoding. This situation becomes normal after exporting the expression to MathML and importing its back to Formulator (by using 'Refresh All Through MathML' menu command or by switching to "MathML text" page, editing and return back).</p>
<annotation>	+	
<annotation-xml>	+	

Notes

- Author can accompany MathML tags with arbitrary attributes. In the case of unknown or currently unsupported attributes Formulator keeps them, but ignores their values during rendering. Among well-known examples of such attributes are 'id', 'xref', 'class', etc.
- There is a set of mathematical operators which should be rendered as horizontally stretchy, but currently are not always conform to this rule. E.g., arrows in some uncommon cases might not be able to be extended in their bounding box.

Formulator Conformance with MathML 2.0: Test Suite

Formulator conforms to the Mathematical Markup Language (MathML) version 2.0 (W3C recommendation, second edition, 21 October 2003). The following tables show how the current MathML implementation passes the W3C MathML 2.0 test suite that is to help insure uniformity and interoperability between MathML implementations.

- $[+]$ implementation passes the test;
- $[\pm]$ implementation passes part of the test;
- $[-]$ implementation does not pass the test.

General

Math

Test Name	Result	Comments
emptymath2	+	
math1	+	
math3	+	
mathAdisplay1	+	
mathAdisplay2	+	
mathAmacros1	+	
mathAmode1	+	

GenAttribs

Test Name	Result	Comments
attribQuote1	+	
class1	+	
class2	+	
id1	+	
id2	+	
style1	\pm	CSS-compatible attributes 'color' and 'background' currently are not used for rendering
style2	\pm	CSS-compatible attributes 'color' and 'background' currently are not used for rendering
xref1	+	

Presentation

TokenElements

Test Name	Result	Comments
mi/mi1	+	
mi/mi2	+	
mi/mi3	+	
mi/mi4	+	
mi/miAtoken5	+	
mi/miScolorname15	+	
mi/miScolorname6	+	
mi/miScolorscope7	+	
mi/miSfont8	+	Γ is italic according to the 'fontstyle' attribute of the 'mstyle' element.
mi/miSfontsize9	+	
mi/miSmathsize16	+	
mi/miSmathsize17	+	
mi/miStoken10	+	
mi/miequivalents11	±	Some glyphs (double-struck, fraktur) are displayed by font substitution.
mi/mifontstyle12	+	
mi/mimathvariant13	±	Some glyphs (double-struck, fraktur) are displayed by font substitution.
mi/mimathvariant14	±	Some glyphs (double-struck, fraktur) are displayed by font substitution.

Test Name	Result	Comments
mn/mn1	+	
mn/mn2	+	
mn/mn3	+	
mn/mn4	+	
mn/mnAcolorname5	+	
mn/mnAtoken6	+	
mn/mnAtoken7	+	
mn/mnScolor8	+	
mn/mnSdisplaystyle9	+	
mn/mnSfont10	+	
mn/mnSscriptlevel11	+	The 'scriptlevel' attribute is not used for rendering.

Test Name	Result	Comments
mo/mo1	+	
mo/mo2	+	
mo/mo3	+	
mo/mo4	+	
mo/mo5	±	The 'maxsize' attribute is not used for rendering.
mo/mo6	+	
mo/mo7	+	
mo/mo8	+	
mo/moAaccent10	+	
mo/moAaccent9	+	
mo/moAform11	+	
mo/moAlargeop12	±	Usage of the 'displaystyle' attribute don't result in changes of the operator size.
mo/moAlrspace13	+	
mo/moAminmax14	±	'maxsize' and 'minsize' attributes are not used for rendering.
mo/moAmoveable15	±	'movablelimits' and 'displaystyle' attributes are not used for rendering.
mo/moAprime16	+	
mo/moAsep17	+	
mo/moAstretchy18	+	
mo/moAsymmetric19	±	The 'symmetric' attribute is not used for rendering.
mo/moSminmax20	±	'maxsize' and 'minsize' attributes are not used for rendering.

Test Name	Result	Comments
mtext/mtext1	+	
mtext/mtext2	+	
mtext/mtextAtoken3	+	
mtext/mtextSbg4	+	
mtext/mtextScolor5	+	
mtext/mtextStoken6	+	
mtext/mtextspaces7	+	

Test Name	Result	Comments
mspace/mspace1	+	
mpace/mspace2	+	Choice of the value ('-10px') for the 'width' attribute is not advantageous, but using Zoom option we can see that behavior of Formulator is absolutely correct. By applying another value (e.g., '-9px') for the 'width' attribute the essential effect of this test is evident on other

		scales also, including 100%.
mspace/mspaceAbreak3	+	
mspace/mspacestruts2	+	

Test Name	Result	Comments
ms/ms1	+	
ms/msAdisplaystyle2	+	
ms/msAquotes3	+	
ms/msAtoken4	+	
ms/msAtoken5	+	
ms/msScolorscope6	+	
ms/msSinheritance7	+	
ms/msSinheritance8	+	Γ is italic according to the 'fontstyle' attribute of the 'mstyle' element.

Test Name	Result	Comments
mglyph/rec-mglyph1	±	The 'alt' attribute is used for substitution of the element body in the case when needed font can't be applied, but there are no checks whether a position given by the 'index' attributes is valid for the specified font, so it may be that user see blank rectangle instead of the expected symbol or instead of a value of the 'alt' attribute.
mglyph/rec-mglyph2	–	The embedded version of the 'mglyph' tag is not supported.

Test Name	Result	Comments
CommonAttributes/hexcolors2	+	
CommonAttributes/sizeunits3	+	
CommonAttributes/sizeunits4	±	The 'superscript' attribute is not used for rendering.

GeneralLayout

Test Name	Result	Comments
mrow/mrow1	+	
mrow/mrowAbg4	+	
mrow/mrowBinferred2	+	
mrow/mrowBnested3	+	

Test Name	Result	Comments
mfrac/mfrac1	+	
mfrac/mfrac2	+	
mfrac/mfrac3	+	
mfrac/mfrac4	+	
mfrac/mfrac5	+	
mfrac/mfrac6	+	
mfrac/mfrac7	+	
mfrac/mfracAbevelled16	+	
mfrac/mfracAcss8	+	
mfrac/mfracAkeyword9	±	'medium' and 'thin' values of the 'linethickness' attributes have similar effect on rendering.
mfrac/mfracAmultiplier10	+	
mfrac/mfracBalign16	+	
mfrac/mfracBheight17	+	
mfrac/mfracBhoriz11	+	
mfrac/mfracBvert12	+	
mfrac/mfracSfonts13	+	
mfrac/mfracSinheritance14	+	
mfrac/mfracSscriptlevel15	±	The 'displaystyle' attribute is not used for rendering.
mfrac/mfracZComp1	+	

Test Name	Result	Comments
msqrt-mroot/mrootB1	+	
msqrt-mroot/mrootSfonts3	+	
msqrt-mroot/mrootSscriptlevel4	±	'scriptlevel' and 'displaystyle' attributes are not used for rendering.
msqrt-mroot/msqrt5	+	
msqrt-mroot/msqrt6	+	
msqrt-mroot/msqrtB7	+	
msqrt-mroot/msqrtBimplied8	+	
msqrt-mroot/msqrtSinheritance9	+	

Test Name	Result	Comments
mstyle/mstyle1	±	The 'maxsize' attribute is not used for rendering.
mstyle/mstyleA2	+	
mstyle/mstyleAdep1	+	
mstyle/mstyleAlinethick1	+	
mstyle/mstyleAminscript1	±	The 'scriptminsize' attribute is not used for rendering.

mstyle/mstyleAscriptlevel1	±	The 'scriptlevel' attribute is not used for rendering.
----------------------------	---	--

Test Name	Result	Comments
merror/merror1	+	
merror/merrorB3	+	

Test Name	Result	Comments
mpadded/mpadded1	+	
mpadded/mpadded10	+	
mpadded/mpadded11	+	
mpadded/mpadded12	+	
mpadded/mpadded13	+	
mpadded/mpadded14	±	If 'height' and 'depth' attributes lead to lessening of the vertical formula size, they are ignored.
mpadded/mpadded15	±	If 'height' and 'depth' attributes lead to lessening of the vertical formula size, they are ignored.
mpadded/mpadded16	±	If 'height' and 'depth' attributes lead to lessening of the vertical formula size, they are ignored.
mpadded/mpadded17	+	
mpadded/mpadded18	±	If 'height' and 'depth' attributes lead to lessening of the vertical formula size, they are ignored.
mpadded/mpadded2	+	
mpadded/mpadded3	+	
mpadded/mpadded4	+	
mpadded/mpadded5	+	
mpadded/mpadded6	+	
mpadded/mpadded7	+	
mpadded/mpadded8	+	
mpadded/mpadded9	+	
mpadded/mpaddedAdepth2	+	
mpadded/mpaddedAheight3	+	
mpadded/mpaddedAleft4	+	
mpadded/mpaddedAmixed5	+	
mpadded/mpaddedAwidth6	+	
mpadded/mpaddedScolor7	+	

Test Name	Result	Comments
mphantom/mphantomB1	+	
mphantom/mphantomBinferred2	+	
mphantom/mphantomBoperators3	+	
mphantom/mphantomScolor4	+	
mphantom/mphantomSinheritance5	±	The 'scriptlevel' attribute is not used for rendering.

Test Name	Result	Comments
mfenced/mfenced1	+	
mfenced/mfenced2	+	
mfenced/mfenced3	+	
mfenced/mfenced4	+	
mfenced/mfenced5	+	
mfenced/mfencedAdelims6	+	
mfenced/mfencedAempty	+	
mfenced/mfencedBfences7	+	
mfenced/mfencedBseparators8	+	
mfenced/mfencedSfonts9	+	

Test Name	Result	Comments
menclose/rec-enclose1	±	Attributes which relates to cell size and spacing are not processed.
menclose/rec-enclose2	+	
menclose/rec-enclose3	+	

ScriptsAndLimits

Test Name	Result	Comments
msub/msub1	+	
msup/msup1	+	
msup/msup2	+	
msup/msupBnest1	+	
msup/msupBsize1	+	
msup/msupBsize2	+	
msup/msupBsize3	+	
msup/msupSadvance1	+	
msubsup/msubsup1	+	
msubsup/msubsupBsize1	+	
msubsup/msubsupBsize2	+	

Test Name	Result	Comments
munder/munder1	+	
munder/munder2	+	
mover/mover1	+	
mover/mover2	+	
mover/mover3	+	

Test Name	Result	Comments
mmultiscript/mmultiscripts1	+	
mmultiscript/mmultiscripts2	+	

TablesAndMatrices

Test Name	Result	Comments
Note: for all tests with 'mtable' element it is common that attributes which relates to cell size, spacing and the 'groupalign' attribute are not processed. In those tests where it leads to noticeable visual effects, the implementation is considered as partially conforming the test.		
mtable/deprecated-mtd1	+	
mtable/deprecated-mtd2	+	
mtable/deprecated-test3	+	
mtable/maligngroup1	±	The 'maligngroup' element and the 'groupalign' attribute are not processed.
mtable/mtable1	+	
mtable/mtable2	+	
mtable/mtableAalign1	+	
mtable/mtableAalign2	+	
mtable/mtableAalign3	+	
mtable/mtableAframe1	+	
mtable/mtableAframe2	+	
mtable/mtableAgroupalign	+	
mtable/mtableAlines1	+	
mtable/mtableAlines2	+	
mtable/mtableAspacing1	+	
mtable/mtableAspacing2	+	
mtable/mtableAspacing3	±	
mtable/mtableAspacing4	±	
mtable/mtableAwidth1	±	
mtable/mtableAwidth2	+	
mtable/mtableAwidth3	±	
mtable/mtableAwidth4	±	
mtable/mtableBgap1	-	The 'mlabeledtr' element is treated as 'mtr'.
mtable/mtableBsize1	±	Arrow is not extended in its bounding

		box.
mtable/mtableBsize2	\pm	Arrow is not extended in its bounding box.
mtable/mtableBspan3	-	The 'rowspan' attribute and the 'colspan' attribute are not processed.
mtable/rec-mtable1	+	
mlabeledtr/mlabeledtr1	-	The 'mlabeledtr' element is treated as 'mtr'.
mlabeledtr/mlabeledtrAside1	-	The 'mlabeledtr' element is treated as 'mtr'.
mlabeledtr/mlabeledtrAside2	-	The 'mlabeledtr' element is treated as 'mtr'.
mlabeledtr/rec-mlabeledtr	-	The 'mlabeledtr' element is treated as 'mtr'.
nested/mtableAwidth1	\pm	
nested/nestedAwidth1	\pm	

DynamicExpressions

Test Name	Result	Comments
maction/mactionBhigh1	-	
maction/mactionBstatus1	-	
maction/mactionBtoggle1	-	
maction/mactionBtooltip1	-	

Content

TokenElements

Test Name	Result	Comments
cn/cn2	+	
cn/rec-cn1	+	
ci/ci4	+	
ci/rec-ci1	+	
ci/rec-ci2	+	Formulator's behavior is correct, because 'ci' elements of type 'vector' should be rendered as bold.
ci/rec-ci3	+	
csymbol/rec-csymbol1	+	
csymbol/rec-csymbol2	+	
csymbol/rec-csymbol3	+	Formulator's behavior is correct, because there is no presentation markup so that the 'csymbol' element should be rendered as if it were wrapped in an 'mo' presentation element

BasicContentElements

Test Name	Result	Comments
apply/rec-apply1	+	
apply/rec-apply2	+	
apply/rec-apply3	+	
apply/rec-apply4	+	
reln/rec-reln1	+	
reln/rec-reln2	+	
reln/rec-reln3	+	
fn/fn1	+	
fn/fn4	+	
fn/rec-fn2	+	
fn/rec-fn3	+	
interval/rec-interval1	+	
interval/rec-interval2	+	
inverse/rec-inverse1	+	
inverse/rec-inverse2	+	
inverse/rec-inverse3	+	Formulator's behavior is correct: a 'ci' element of type 'matrix' should be rendered as bold.
inverse/rec-inverse4	+	
condition/rec-condition1	+	
declare/rec-declare1	+	
declare/rec-declare2	+	
declare/rec-declare3	+	
declare/rec-declare4	+	
declare/rec-declare5	+	
lambda/rec-lambda1	+	
lambda/rec-lambda2	+	
lambda/rec-lambda3	+	
compose/rec-compose1	+	
compose/rec-compose2	+	
compose/rec-compose3	+	
compose/rec-compose4	+	
ident/ident1	+	
ident/rec-ident2	+	
domain/rec-domain1	+	
codomain/rec-codomain1	+	
image/rec-image1	+	
domainofapplication/rec-domainofapplication1	+	
piecewise/rec-piecewise1	+	
piecewise/rec-piecewise2	+	

ArithmeticAlgebraLogic

Test Name	Result	Comments
arithmetic_1	+	Rendering of the 'times' element is regulated with Formulator's option to use one of the following values: '⋅', '⁢', '×'. The last value is default that differs rendering from the test suite example.
arithmetic_2	+	
arithmetic_3	+	
arithmetic_4	+	
logic5	+	There is an empty slot after condition expression of the 'exists' element, since it is correct to expect user to fill it further.
logic6	+	
logic7	+	
quotient/rec-quotient1	+	
factorial/factorial1	+	Formulator's rendering is correct, because the nested 'apply' element changes precedence.
factorial/factorial3	+	
factorial/factorial4	+	
factorial/rec-factorial2	+	
divide/divide1	+	
divide/divide2	+	
divide/divide3	+	
divide/divide5	+	
divide/rec-divide4	+	
max/max3	+	
max/rec-max1	+	
max/rec-max2	+	
min/min1	+	
min/rec-min2	+	
minus/minus1	+	
minus/minus2	+	
minus/minus3	+	
minus/minus4	+	
minus/minus5	+	
minus/minus6	+	
minus/minus7	+	
minus/minus9	+	
minus/rec-minus8	+	
plus/plus1	+	Formulator's rendering is correct, because the nested 'apply' element changes precedence.
plus/plus2	+	

plus/plus3	+	
plus/plus4	+	
plus/plus6	+	
plus/plus7	+	
plus/rec-plus5	+	
power/power1	+	
power/power2	+	
power/power3	+	
power/power5	+	
power/power6	+	
power/power7	+	
power/power8	+	
power/rec-power4	+	
rem/rec-rem1	+	
times/rec-times1	+	
times/times2	+	
times/times3	+	
times/times4	+	
times/times5	+	
times/times6	+	
times/times7	+	
root/rec-root1	+	
root/root2	+	
gcd/rec-gcd1	+	
and/and1	+	
and/and2	+	
and/rec-and2	+	
or/rec-or1	+	
xor/rec-xor1	+	
not/not1	+	
not/rec-not2	+	
not/rec-not3	+	There is an empty slot after condition expression of the 'forall' element, since it is correct to expect user to fill it further.
implies/implies2	+	
implies/rec-implies1	+	
forall/forall1	+	
forall/forall2	+	There is an empty slot after condition expression of the 'forall' element, since it is correct to expect user to fill it further.
forall/rec-forall2	+	
forall/rec-forall3	+	There are different ways to render this and the next test and Formulator uses one that conforms to the example in the W3C MathML

		2.0 specification (see section 4.4.3.17 of the specification).
forall/rec-forall4	+	
forall/rec-forall5	+	
forall/rec-forall6	+	There is an empty slot after condition expression of the 'forall' element, since it is correct to expect user to fill it further.
forall/rec-forall7	+	
exists/rec-exists1	+	
abs/abs1	+	
abs/abs2	+	
abs/rec-abs3	+	
conjugate/rec-conjugate1	+	
arg/rec-arg1	+	
real/rec-real1	+	
imaginary/rec-imaginary1	+	
lcm/rec-lcm1	+	
floor/rec-floor1	+	
floor/rec-floor2	+	
ceiling/rec-ceiling1	+	
ceiling/rec-ceiling2	+	

Relations

Test Name	Result	Comments
eq/eq2	+	
eq/rec-eq1	+	
neq/neq2	+	
neq/rec-neq1	+	
gt/gt2	+	
gt/rec-gt1	+	
lt/lt2	+	
lt/rec-lt1	+	
geq/geq2	+	
geq/rec-geq1	+	
leq/rec-leq1	+	
equivalent/rec-equivalent1	+	
approx/rec-approx1	+	
factorof/rec-factorof1	+	

Calculus

Test Name	Result	Comments
int/int1	+	
int/int2	+	
int/rec-int3	+	
int/rec-int4	+	
int/rec-int5	+	
int/rec-int6	+	
diff/rec-diff1	+	
diff/rec-diff2	+	
partialdiff/partialdiff1	+	
partialdiff/partialdiff2	+	There are different ways to render this and the next test and Formulator uses one that conforms to the example in the W3C MathML 2.0 specification (see sections 4.4.5.3 and 4.4.5.7 of the specification).
partialdiff/rec-partialdiff3	+	
partialdiff/rec-partialdiff4	+	
partialdiff/rec-partialdiff5	+	
lowlimit/rec-lowlimit1	+	
uplimit/rec-uplimit1	+	
bver/rec-bvar1	+	
bver/rec-bvar2	+	
degree/degree2	+	
degree/rec-degree1	+	
divergence/rec-divergence1	+	
divergence/rec-divergence2	+	
grad/rec-grad1	+	
curl/rec-curl1	+	
laplacian/rec-laplacian1	+	

TheoryOfSets

Test Name	Result	Comments
equation1	+	
set/rec-set1	+	
set/rec-set2	+	
set/set-empty	+	
set/set3	+	
set/set4	+	
set/set5	+	
set/set6	+	
list/list-empty	+	

list/list3	+	
list/rec-list1	+	
list/rec-list2	+	
union/rec-union1	+	
union/union2	+	
union/union3	+	
intersect/intersect1	+	
intersect/rec-intersect2	+	
in/in2	+	
in/in3	+	
in/rec-in1	+	
notin/notin2	+	
notin/rec-notin1	+	
subset/rec-subset1	+	
subset/subset2	+	
prsubset/prsubset2	+	
prsubset/rec-prsubset1	+	
notsubset/notsubset2	+	
notsubset/rec-notsubset1	+	
notprsubset/notprsubset2	+	
notprsubset/rec-notprsubset1	+	
setdiff/rec-setdiff1	+	
card/rec-card1	+	
cartesianproduct/rec-cartesianproduct1	+	
cartesianproduct/rec-cartesianproduct2	+	

SequencesAndSeries

Test Name	Result	Comments
sum/rec-sum1	+	
sum/sum2	+	
sum/sum3	+	
product/product1	+	
product/product2	+	
product/rec-product3	+	
limit/limit1	+	Formulator's rendering is correct in this test and in next similar tests, because the 'tendsto' element has an attribute 'type' with a value of 'above' (or 'below' in later examples).
limit/limit2	+	
limit/limit3	+	
limit/limit4	+	
limit/limit5	+	
limit/limit6	+	

limit/limit7	+	
limit/rec-limit8	+	
limit/rec-limit9	+	
tendsto/rec-tendsto1	+	
tendsto/rec-tendsto2	+	
tendsto/tendsto3	+	
tendsto/tendsto4	+	
tendsto/tendsto5	+	
tendsto/tendsto6	+	
tendsto/tendsto7	+	
tendsto/tendsto8	+	
tendsto/tendsto9	+	

ElementaryFunctions

Test Name	Result	Comments
rec-trig1	+	
rec-trig2	+	
trigonometry_3	+	
trigonometry_4	+	
trigonometry_5	+	
trigonometry_6	+	
trigonometry_7	+	
trigonometry_8	+	
exp/exp1	+	
exp/exp2	+	
exp/rec-exp3	+	
ln/rec-ln1	+	
log/log1	+	
log/rec-log2	+	
sin/factorial3	+	
sin/sin1	+	
sin/sin2	+	
sin/sin3	+	
cos/cos1	+	
cos/cos2	+	
cos/cos3	+	
cos/cos4	+	
tan/tan1	+	
tan/tan2	+	
tan/tan3	+	
sec/sec1	+	
sec/sec2	+	
sec/sec3	+	

csc/csc1	+	
csc/csc2	+	
csc/csc3	+	
cot/cot1	+	
cot/cot2	+	
cot/cot3	+	
sinh/sinh1	+	
sinh/sinh2	+	
sinh/sinh3	+	
cosh/cosh1	+	
cosh/cosh2	+	
cosh/cosh3	+	
cosh/cosh4	+	
tanh/tanh1	+	
tanh/tanh2	+	
tanh/tanh3	+	
sech/sech1	+	
sech/sech2	+	
sech/sech3	+	
csch/csch1	+	
csch/csch2	+	
csch/csch3	+	
coth/coth1	+	
coth/coth2	+	
coth/coth3	+	
arcsin/arcsin1	+	
arcsin/arcsin2	+	
arcsin/arcsin3	+	
arcsin/factorial3	+	
arccos/arccos1	+	
arccos/arccos2	+	
arccos/arccos3	+	
arccos/arccos4	+	
arctan/arctan1	+	
arctan/arctan2	+	
arctan/arctan3	+	
arcsec/arcsec1	+	
arcsec/arcsec2	+	
arcsec/arcsec3	+	
arccsc/arccsc1	+	
arccsc/arccsc2	+	
arccsc/arccsc3	+	
arccot/arccot1	+	
arccot/arccot2	+	

arccot/arccot3	+	
arcsinh/arcsinh1	+	
arcsinh/arcsinh2	+	
arcsinh/arcsinh3	+	
arccosh/arccosh1	+	
arccosh/arccosh2	+	
arccosh/arccosh3	+	
arccosh/arccosh4	+	
arctanh/arctanh1	+	
arctanh/arctanh2	+	
arctanh/arctanh3	+	
arcsech/arcsech1	+	
arcsech/arcsech2	+	
arcsech/arcsech3	+	
arccsch/arccsch1	+	
arccsch/arccsch2	+	
arccsch/arccsch3	+	
arccoth/arccoth1	+	
arccoth/arccoth2	+	
arccoth/arccoth3	+	

Statistics

Test Name	Result	Comments
mean/rec-mean1	+	
sdev/rec-sdev1	+	
variance/rec-variance1	+	
median/rec-median1	+	
mode/rec-mode1	+	
moment/rec-moment1	+	There are different ways to render this and the next two tests and Formulator uses one that conforms to the example in the W3C MathML 2.0 specification (see section 4.4.9.7 of the specification).
moment/rec-moment2	+	
momentabout/rec-momentabout1	+	

LinearAlgebra

Test Name	Result	Comments
vector/rec-vector1	+	
vector/rec-vector2	+	
vector/vector3	+	

matrix/inverse1	+	
matrix/matrix3	+	
matrix/rec-matrix1	+	
matrix/rec-matrix2	+	
matrix/rec-matrix3	+	
determinant/rec-determinant1	+	
transpose/rec-transpose1	+	Formulator's behavior is correct, because a 'ci' element of type 'matrix' should be rendered as bold.
selector/rec-selector1	+	
selector/rec-selector2	+	
vectorproduct/rec-vectorproduct1	+	
scalarproduct/rec-scalarproduct1	+	
outerproduct/rec-outerproduct1	+	

SemanticMappingElements

Test Name	Result	Comments
annotation/rec-annotation1	+	

ConstantsAndSymbols

Test Name	Result	Comments
integers/rec-integers1	+	
reals/rec-reals1	+	
rationals/rec-rationals1	+	
naturalnumbers/rec-naturalnumbers1	+	
complexes/rec-complexes1	+	
primes/rec-primes1	+	
exponentiale/rec-exponentiale1	+	
imaginaryi/rec-imaginaryi1	+	
notanumber/rec-notanumber1	+	
true/rec-true1	+	
false/rec-false1	+	
emptyset/rec-emptyset1	+	
pi/rec-pi1	+	
eulergamma/rec-eulergamma1	+	There are different ways to render this test and Formulator uses one that conforms to the example in the W3C MathML 2.0 specification (see section 4.4.12.14 of the specification).
infinity/rec-infinity1	+	

CharactersEntityNames

Test Name	Result	Comments
a	±	Formulator uses system's preinstalled fonts which are currently can have poor support of mathematics needs, so sometimes it may be that user see blank rectangle instead of the expected symbol.
b	±	
c	±	
d	±	
e	±	
f	±	
g	±	
h	±	
i	±	
j	±	
k	±	
l	±	
m	±	
n	±	
o	±	
p	±	
q	±	
r	±	
s	±	
t	±	
u	±	
v	±	
w	±	
x	±	
y	±	
z	±	

NumericRefs

Test Name	Result	Comments
a	±	Formulator uses system's preinstalled fonts which are currently can have poor support of mathematics needs, so sometimes it may be that user see blank rectangle instead of the expected symbol.
b	±	
c	±	
d	±	
e	±	
f	±	
g	±	
h	±	
i	±	
j	±	
k	±	

l	±	
m	±	
n	±	
o	±	
p	±	
q	±	
r	±	
s	±	
t	±	
u	±	
v	±	
w	±	
x	±	
y	±	
z	±	

UTF8

Test Name	Result	Comments
a	±	Formulator uses system's preinstalled fonts which are currently can have poor support of mathematics needs, so sometimes it may be that user see blank rectangle instead of the expected symbol.
b	±	
c	±	
d	±	
e	±	
f	±	
g	±	
h	±	
i	±	
j	±	
k	±	
l	±	
m	±	
n	±	
o	±	
p	±	
q	±	
r	±	
s	±	
t	±	
u	±	
v	±	
w	±	
x	±	
y	±	
z	±	

Error Handling

BadAttribs

Test Name	Result	Comments
badAttribs2	+	Author can accompany MathML tags with arbitrary attributes. In the case of unknown or currently unsupported attributes Formulator keeps them, but ignores their values during rendering.
badAttribsAction	+	
badAttribsGlyph4	+	Nested 'mglyph' elements are not supported.
badAttribsVal3	+	

BadChildren

Test Name	Result	Comments
Formulator is not just viewer, but an editor of MathML 2.0, so in all “bad” examples Formulator tries to repair or to reconstruct an expression in the hope of further user help in this matter.		
badBvar1	+	
badCondContent1	+	
badMatrix1	+	
badMoment1	+	
badPiecewise1	+	
badReln1	+	
emptyContent1	+	

BadEntities

Test Name	Result	Comments
badEntity1	+	Insert contents of the 'mn' element “as is”.

BadTags

Test Name	Result	Comments
badTag1	+	The 'merror' element is used to inform a user about problems with the MathML fragment.
badTagPhantom2	+	The 'merror' element is used to inform a user about problems with the MathML fragment. Unfortunately, the diagnostic message is hidden by the parent 'mphantom' element.
nestedMath3	+	

NumChildren

Test Name	Result	Comments
emptyContent	+	
mrootE2	+	
noChildContent	+	
noChildPresentation	+	
singleBinary	+	
tooFewContentContainer	+	

Torture TestsSize

Test Name	Result	Comments
10	+	
100	+	
1000	+	

Complexity

Test Name	Result	Comments
complex1	+	
complex2	+	
complex3	+	
complex4	+	
simplePres	+	

TopicsEmbellishedOp

Test Name	Result	Comments
embStretch1	+	

LargeOp

Test Name	Result	Comments
In all the following tests usage of the 'displaystyle' attribute don't result in changes of the operator size.		
chain1	+	
chain2	+	
coprod1	+	
coprod2	+	

doubleint1	+	
doubleint2	+	
int1	+	
int10	+	
int2	+	
int3	+	
int4	+	
int5	+	
int6	+	
int7	+	
int8	+	
int9	+	
largeop1	\pm	
largeop2	\pm	
largeopPos3	+	
oint1	+	
oint2	+	
prod1	+	
prod2	\pm	
sum1	+	
sum2	\pm	
tripleint1	+	
tripleint2	\pm	

LineBreak

Test Name	Result	Comments
linebreak1	+	Formulator is an editor of MathML 2.0, so there is no need to break lines without user request.
linebreakFrac	+	
linebreakNum1	+	
linebreakRow1	+	
linebreakString1	+	
goodbreak/goodbreak1	+	
badbreak/badbreak1	+	
nobreak/nobreak1	+	
nobreak/nobreak2	+	
newline/indent1	+	
newline/indent2	+	
newline/mixed4	+	
newline/multinewline3	+	
newline/newline1	+	
newline/newline2	+	

Nesting

Test Name	Result	Comments
nestAction1	-	The 'maction' element is rendered as its contents in the bounding slot.
nestFrac1	+	
nestScript	+	
nestTable1	±	Attributes which relates to cell size and spacing are not processed.

Primes

Test Name	Result	Comments
primes1	+	

Accents

Test Name	Result	Comments
accents1	±	
accents2	±	
accents3	±	
accents4	±	

StretchyChars

Test Name	Result	Comments
vertical/abs1	+	
vertical/abs2	+	
vertical/mid1	+	
vertical/mid2	+	
vertical/stretchVert1	±	
vertical/stretchVert2	+	
vertical/stretchVertNest2	+	
vertical/verbar1	+	
vertical/verbar2	+	
horizontal/genBvert1	+	
horizontal/stretchAccents1	±	
horizontal/stretchAccents2	+	
horizontal/stretchBrack1	+	
horizontal/stretchHoriz1	±	There is a set of mathematical operators which should be rendered as horizontally stretchy, but currently are not always conform to this rule. E.g., arrows in some cases might not be able to be extended in

		their bounding box.
horizontal/stretchHoriz2	+	
horizontal/stretchHoriz3	+	
horizontal/stretchTilde1	+	Formulator's rendering is correct, because the '∼' operator is not stretchy by default (see W3C MathML 2.0 specification, Appendix F, Operator Dictionary).
integral/int1	+	
integral/int10	+	
integral/int2	+	
integral/int3	+	
integral/int4	+	
integral/int5	+	
integral/int6	+	
integral/int7	+	
integral/int8	+	
integral/int9	+	
integral/intDisplayStyle	±	Usage of the 'displaystyle' attribute don't result in changes of the operator size.
integral/intNested3	+	
integral/intSize1	+	
integral/intSize2	+	
tables/stretchTable1	±	Arrows are not extended in their bounding box.
tables/stretchTable2	±	Arrow is not extended in its bounding box.

WhiteSpace

Test Name	Result	Comments
invChars	+	
whBcomments1	+	
whBgeneral1	+	
white1	+	
white2	+	
white3	+	
white4	+	

Notes

- Content Markup tests fairly often can be rendered in different manners, so that all of them will be correct. There are several cases when Formulator's ways to render Content Markup differs from the images supplied with the MathML 2.0 Test Suite only because of utilization of different rendering standard (e.g., max, min, sets, lambda, etc.). These cases are considered as a correct behavior.
- Sometimes differing approaches to representation of Content Markup can be soften by using Formulator's option. E.g., for rendering of the 'times' element there is a way to use several values ('⋅', '⁢', '×').
- In the case of rendering of the 'partialdiff' element Formulator enhances the potential of the 'degree' element by applying simple calculations during parsing of MathML text. E.g., if a user indicates degrees not in identifiers, but in integer constants, then the overall degree will be calculated automatically.

Appendix A. Keyboard Shortcuts

You can execute some Formulator operations directly from the keyboard via shortcuts. Note that some shortcuts require you to type two keystroke combinations consecutively.

Formatting

Commands for formatting equation elements:

- Insert Tab (Tab);
- New Line (Enter).

Menu Commands

Commands for items appearing on the File menu:

- New (Ctrl+N);
- Open (Ctrl+O);
- Close (Ctrl+F4);
- Save (Ctrl+S);
- Print (Ctrl+P);
- Exit (Alt+F4).

Commands for items appearing on the Edit menu:

- Undo (Ctrl+Z);
- Redo (Ctrl+Y);
- Cut (Ctrl+X);
- Copy (Ctrl+C);
- Paste (Ctrl+V);
- Select All (Ctrl+A).

Commands for items appearing on the View menu:

- Zoom/100% (Ctrl+1);
- Zoom/200% (Ctrl+2);
- Zoom/300% (Ctrl+3);
- Zoom/500% (Ctrl+5);
- Zoom/1000% (Ctrl+0);
- Show Nesting (Ctrl+Shift+N);
- Show Read-Only (Ctrl+Shift+R);
- Toolbar;
- Status Bar.

Commands for items appearing on the Style menu:

- Math (Ctrl+Shift+M);
- Text (Ctrl+Shift+E);
- Function (Ctrl+Shift+F);
- Variable (Ctrl+Shift+V);
- Greek (Ctrl+Shift+G);
- Vector-Matrix (Ctrl+Shift+B);

- Fixed (Ctrl+Shift+X);
- Operator (Ctrl+Shift+O);
- User 1 (Ctrl+Shift+U);
- User 2 (Ctrl+Alt+Shift+U).

Commands for items appearing on the Size menu:

- Smaller (Alt+<);
- Larger (Alt+>).

Commands for items appearing on the Help menu:

- Contents and Index (F1).

Navigation and Selection

Commands for moving around and/or selecting items in the current equation:

- Beginning of Slot (Home);
- Delete Left (Backspace);
- Delete Right (Delete);
- End of Slot (End);
- Move Down (Down);
- Move Down Extend Selection (Shift+Down);
- Move Left (Left);
- Move Left Extend Selection (Shift+Left);
- Move Right (Right);
- Move Right Extend Selection (Shift+Right);
- Move Up (Up);
- Move Up Extend Selection (Shift+Up);
- Scroll Up (Page Up);
- Scroll Down (Page Down).

Toolbar Commands

Commands for items of the "Relational and logical symbols" toolbar:

- Less-than or equal to (Ctrl+K,,);
- Greater-than or equal to (Ctrl+K,.);
- Not equal to (Ctrl+K,+);
- Identical to (Ctrl+K,=);
- Tilde operator (Ctrl+K,Alt+~);
- Almost equal to (Ctrl+K,~);
- Proportional to (Ctrl+K,P);
- Therefore (Ctrl+Shift+K,T);
- There exists (Ctrl+Shift+K,E);
- For all (Ctrl+Shift+K,A);
- Not sign (Ctrl+Shift+K,N);
- Logical and (Ctrl+Shift+K,7);
- Logical or (Ctrl+Shift+K,\).

Commands for items of the "Spaces templates" toolbar:

- 1-point space (Ctrl+Alt+Space);
- Thin space (1/6 EM) (Ctrl+Space);
- Thick space (1/3 EM) (Ctrl+Shift+Space);
- EM space (Ctrl+K,4).

Commands for items of the "Operator symbols" toolbar:

- Plus-minus sign (Ctrl+Shift+K,=);
- Multiplication sign (Ctrl+K,T);
- Asterisk operator (Ctrl+Shift+K,*);
- Division sign (Ctrl+Shift+K,/);
- Dot operator (Ctrl+Shift+K,.);
- Bullet (Ctrl+Shift+K,8);
- Left-pointing angle bracket (Ctrl+Shift+K,<);
- Right-pointing angle bracket (Ctrl+Shift+K,>).

Commands for items of the "Arrow symbols" toolbar:

- Left right arrow (Ctrl+K,Alt+Left);
- Rightwards arrow (Ctrl+K,Right);
- Leftwards arrow (Ctrl+K,Left);
- Up down arrow (Ctrl+K,Alt+Up);
- Upwards arrow (Ctrl+K,Up);
- Downwards arrow (Ctrl+K,Down);
- Left right double arrow (Ctrl+K,Alt+Shift+Left);
- Rightwards double arrow (Ctrl+K,Shift+Right);
- Leftwards double arrow (Ctrl+K,Shift+Left);
- Up down double arrow (Ctrl+K,Alt+Shift+Up);
- Upwards double arrow (Ctrl+K,Shift+Up);
- Downwards double arrow (Ctrl+K,Shift+Down);
- Rightwards arrow from bar (Ctrl+K,Tab);
- Downwards arrow with corner leftwards (Ctrl+K,Enter).

Commands for items of the "Set theory symbols" toolbar:

- Element of (Ctrl+K,E);
- Not an element of (Ctrl+K,Shift+E);
- Union (Ctrl+K,U);
- Intersection (Ctrl+K,X);
- Subset (Ctrl+K,C);
- Superset (Ctrl+K,S);
- Not a subset of (Ctrl+K,Shift+C);
- Empty set (Ctrl+K,O).

Commands for items of the "Special constants" toolbar:

- Partial Differential (Ctrl+K,D);
- Greek small letter pi (Ctrl+G,P);
- Planck constant over two pi (Ctrl+K,H);

- Infinity (Ctrl+K,I);
- Latin small letter lambda with stroke (Ctrl+K,L);
- Script small l (Ctrl+Shift+K,L).

Commands for items of the "Miscellaneous symbols" toolbar:

- Fraktur capital I, imaginary part (Ctrl+K,Shift-I);
- Fraktur capital R, real part (Ctrl+K,Shift-R);
- Alef symbol (Ctrl+K,A);
- Blackboard-bold capital R, the set of all real numbers (Ctrl+D,Shift+R);
- Blackboard-bold capital Z, the set of all integer numbers (Ctrl+D,Shift+Z);
- Blackboard-bold capital C, the set of all rational numbers (Ctrl+D,Shift+C);
- Blackboard-bold capital Q, the set of all rational numbers (Ctrl+D,Shift+Q);
- Blackboard-bold capital N, the set of all natural numbers (Ctrl+D,Shift+N);
- Greek capital letter delta (Ctrl+G,Shift+D);
- Greek capital letter omega (Ctrl+G,Shift+W);
- Inverted ohm sign (Ctrl+Shift+K,Shift+O);
- Degree sign (Ctrl+Shift+K,D);
- Angle (Ctrl+Shift+K,Shift+A);
- Measured angle (Ctrl+Shift+K,Alt+A);
- Spherical angle (Ctrl+Shift+K,Alt+Shift+A);
- Perpendicular (Ctrl+Shift+K,P);
- Parallel (Ctrl+Shift+K,I).

Commands for items of the "Greek characters (lowercase)" toolbar:

- Greek small letter alpha (Ctrl+G,A);
- Greek small letter beta (Ctrl+G,B);
- Greek small letter gamma (Ctrl+G,G);
- Greek small letter delta (Ctrl+G,D);
- Greek small letter epsilon (Ctrl+G,E);
- Greek small letter zeta (Ctrl+G,Z);
- Greek small letter eta (Ctrl+G,H);
- Greek small letter theta (Ctrl+G,Q);
- Greek small letter iota (Ctrl+G,I);
- Greek small letter kappa (Ctrl+G,K);
- Greek small letter lambda (Ctrl+G,L);
- Greek small letter mu (Ctrl+G,M);
- Greek small letter nu (Ctrl+G,N);
- Greek small letter xi (Ctrl+G,X);
- Greek small letter omicron (Ctrl+G,O);
- Greek small letter pi (Ctrl+G,P);
- Greek small letter rho (Ctrl+G,R);
- Greek sigma symbol (Ctrl+G,Shift+V);
- Greek small letter sigma (Ctrl+G,S);
- Greek small letter tau (Ctrl+G,T);
- Greek small letter upsilon (Ctrl+G,U);
- Greek small letter phi (Ctrl+G,F);

- Greek phi symbol (Ctrl+G,J);
- Greek small letter chi (Ctrl+G,C);
- Greek small letter psi (Ctrl+G,Y);
- Greek small letter omega (Ctrl+G,W).

Commands for items of the "Greek characters (uppercase)" toolbar:

- Greek capital letter alpha (Ctrl+G,Shift+A);
- Greek capital letter beta (Ctrl+G,Shift+B);
- Greek capital letter gamma (Ctrl+G,Shift+G);
- Greek capital letter delta (Ctrl+G,Shift+D);
- Greek capital letter epsilon (Ctrl+G,Shift+E);
- Greek capital letter zeta (Ctrl+G,Shift+Z);
- Greek capital letter eta (Ctrl+G,Shift+H);
- Greek capital letter theta (Ctrl+G,Shift+Q);
- Greek capital letter iota (Ctrl+G,Shift+I);
- Greek capital letter kappa (Ctrl+G,Shift+K);
- Greek capital letter lambda (Ctrl+G,Shift+L);
- Greek capital letter mu (Ctrl+G,Shift+M);
- Greek capital letter nu (Ctrl+G,Shift+N);
- Greek capital letter xi (Ctrl+G,Shift+X);
- Greek capital letter omicron (Ctrl+G,Shift+O);
- Greek capital letter pi (Ctrl+G,Shift+P);
- Greek capital letter rho (Ctrl+G,Shift+R);
- Greek capital letter sigma (Ctrl+G,Shift+S);
- Greek capital letter tau (Ctrl+G,Shift+T);
- Greek capital letter phi (Ctrl+G,Shift+F);
- Greek capital letter chi (Ctrl+G,Shift+C);
- Greek capital letter psi (Ctrl+G,Shift+Y);
- Greek capital letter omega (Ctrl+G,Shift+W).

Commands for items of the "Differentiation templates" toolbar:

- Prime (Ctrl+Alt+');)
- Double prime (Ctrl+").

Commands for items of the "Fence templates" toolbar:

- Parenthesis or round-brackets (Ctrl+9);
- Brackets or square-brackets (Ctrl+[);
- Braces or curly-brackets (Ctrl+{);
- Angle brackets (Ctrl+,);
- Single vertical bars (Ctrl+T,|);
- Left brackets (Ctrl+T,[);
- Right brackets (Ctrl+T,]);
- Left braces (Ctrl+T,{);
- Right braces (Ctrl+T,});
- Left angle brackets (Ctrl+T,<);
- Right angle brackets (Ctrl+T,>).

Commands for items of the "Fraction and radical templates" toolbar:

- Full-size fraction (Ctrl+F);
- Full-size diagonal fraction (Ctrl+ /);
- Slash fraction (Ctrl+T, Alt+ /);
- Square root (Ctrl+R);
- nth root (Ctrl+T,N).

Commands for items of the "Subscript and superscript templates" toolbar:

- Superscript (Ctrl+H);
- Subscript (Ctrl+L);
- Superscript and subscript (Ctrl+J);
- Over-script (Ctrl+T, Shift+L);
- Under-script (limit) (Ctrl+T, Alt-L);
- Over-script and under-script (Ctrl+T, L).

Commands for items of the "Summation templates" toolbar:

- Summation with no limits (Ctrl+T,Shift+S);
- Summation with underscript limit (Ctrl+T,Alt+S);
- Summation with underscript and overscript (Ctrl+T,S).

Commands for items of the "Integral templates" toolbar:

- Indefinite integral (no limits) (Ctrl+Shift+I,!);
- Definite integral with underscript and overscript limits (Ctrl+Shift+I,S);
- Definite integral with subscript and superscript limit (Ctrl+I);
- Definite integral with underscript limit (Ctrl+Shift+I,Alt+S);
- Definite integral with subscript limit (Ctrl+Shift+I,Alt+1);
- Double integral with no limits (Ctrl+Shift+I,@);
- Double integral with underscript limit (Ctrl+Shift+I,2);
- Double integral with subscript limit (Ctrl+Shift+I,Alt+2);
- Triple integral with no limits (Ctrl+Shift+I,#);
- Triple integral with underscript limit (Ctrl+Shift+I,3);
- Triple integral with subscript limit (Ctrl+Shift+I,Alt+3);
- Contour integral with no limits (Ctrl+Shift+I,Shift+C);
- Contour integral with underscript limit (Ctrl+Shift+I,C);
- Contour integral with subscript limit (Ctrl+Shift+I,Alt+C);
- Area integral with no limits (Ctrl+Shift+I,Shift+A);
- Area integral with underscript limit (Ctrl+Shift+I,A);
- Area integral with subscript limit (Ctrl+Shift+I,Alt+A);
- Volume integral with no limits (Ctrl+Shift+I,Shift+V);
- Volume integral with underscript limit (Ctrl+Shift+I,V);
- Volume integral with subscript limit (Ctrl+Shift+I,Alt+V);
- Integral with counter-clockwise loop with no limits (Ctrl+Shift+I,Shift+Left);
- Integral with counter-clockwise loop with underscript limit (Ctrl+Shift+I,Left);

- Integral with counter-clockwise loop with subscript limit (Ctrl+Shift+I,Alt+Left);
- Integral with clockwise loop with no limits (Ctrl+Shift+I,Shift+Right);
- Integral with clockwise loop with underscript limit (Ctrl+Shift+I,Right);
- Integral with clockwise loop with subscript limit (Ctrl+Shift+I,Alt+Right).

Commands for items of the "Underbar and overbar templates" toolbar:

- Tilde (Ctrl+^,~);
- Hat (Ctrl+^,6);
- Arc (Ctrl+^,9);
- Joint status (Ctrl+^,J);
- Over-bar (Ctrl+^,-);
- Double over-bar (Ctrl+^,D);
- Under-bar (Ctrl+^,_);
- Double Under-bar (Ctrl+^,Shift+D);
- Right arrow over-bar (Ctrl+^,Right);
- Left arrow over-bar (Ctrl+^,Left);
- Right harpoon over-bar (Ctrl+^,Alt+Right);
- Double-headed arrow over-bar (Ctrl+^,Up);
- Right arrow under-bar (Ctrl+^,Shift+Right);
- Left arrow under-bar (Ctrl+^,Shift+Left);
- Right harpoon under-bar (Ctrl+^,Alt+Shift+Right);
- Double-headed arrow under-bar (Ctrl+^,Shift+Up);
- Mid-line strike through (Ctrl+^,Alt+-);
- Strike through (Ctrl+^,X);
- Strike-through (bottom-left to upper-right) (Ctrl+^,/);
- Strike-through (top-left to bottom-right) (Ctrl+^,\).

Commands for items of the "Labeled arrow templates" toolbar:

- Right arrow with upper text slot (Ctrl+T,Shift+Right);
- Right arrow with lower text slot (Ctrl+T,Alt+Right);
- Right arrow with upper and lower text slot (Ctrl+T,Right);
- Left arrow with upper text slot (Ctrl+T,Shift+Left);
- Left arrow with lower text slot (Ctrl+T,Alt+Left);
- Left arrow with upper and lower text slot (Ctrl+T,Left);
- Double-headed arrow with upper text slot (Ctrl+T,Shift+Up);
- Double-headed arrow with lower text slot (Ctrl+T,Alt+Up);
- Double-headed arrow with upper and lower text slot (Ctrl+T,Up).

Commands for items of the "Products and set theory templates" toolbar:

- Product with no limits (Ctrl+T,Shift+P);
- Product with underscript limit (Ctrl+T,Alt+P);
- Product with underscript and overscript limits (Ctrl+T,P);
- Coproduct with no limits (Ctrl+T,Shift+C);
- Coproduct with underscript limit (Ctrl+T,Alt+C);
- Coproduct with underscript and overscript limits (Ctrl+T,C);

- Intersection with no limits (Ctrl+T,Shift+I);
- Intersection with underscript limit (Ctrl+T,Alt+I);
- Intersection with underscript and overscript limits (Ctrl+T,I);
- Union with no limits (Ctrl+T,Shift+U);
- Union with underscript limit (Ctrl+T,Alt+U);
- Union with underscript and overscript limits (Ctrl+T,U).

Commands for items of the "Box templates" toolbar:

- Box (Ctrl+Shift+T,X).