cmathml.sty: A TEX/LATEX-based Syntax for Content MATHML*

Michael Kohlhase
Jacobs University, Bremen
http://kwarc.info/kohlhase

January 15, 2016

Abstract

The cmathml package is part of the STEX collection, a version of TEX/LATEX that allows to markup TEX/LATEX documents semantically without leaving the document format, essentially turning TEX/LATEX into a document format for mathematical knowledge management (MKM).

This package provides a collection of semantic macros for content MATHML and their LATEXML bindings. These macros form the basis of a naive translation from semantically preloaded LATEX formulae into the content MATHML formulae via the LATEXML system.

*Version ? (last revised ?)

Contents

1 Introduction **3**
- 1.1 Encoding Content MathML in TEX/LATEX . . . 3
- 1.2 Changing the TEX/LATEX Presentation . . . 3
- 1.3 The Future: Heuristic Parsing . . . 4

2 The User Interface **5**
- 2.1 Generalities of the Encoding . . . 5
- 2.2 The Token Elements . . . 5
- 2.3 The Basic Content Elements . . . 7
- 2.4 Elements for Arithmetic, Algebra, and Logic . . . 8
- 2.5 Relations . . . 9
- 2.6 Elements for Calculus and Vector Calculus . . . 9
- 2.7 Sets and their Operations . . . 10
- 2.8 Sequences and Series . . . 11
- 2.9 Elementary Classical Functions . . . 12
- 2.10 Statistics . . . 12
- 2.11 Linear Algebra . . . 13
- 2.12 Constant and Symbol Elements . . . 13
- 2.13 Extensions . . . 13

3 Limitations **14**

4 The Implementation **15**
- 4.1 Initialization and auxiliary functions . . . 15
- 4.2 The Basic Elements . . . 15
- 4.3 Elements for Arithmetic, Algebra, and Logic . . . 16
- 4.4 Relations . . . 17
- 4.5 Sets and their Operations . . . 19
- 4.6 Sequences and Series . . . 21
- 4.7 Elementary Classical Functions . . . 21
- 4.8 Statistics . . . 22
- 4.9 Linear Algebra . . . 22
- 4.10 Constant and Symbol Elements . . . 23
- 4.11 Extensions . . . 23

1 Introduction

This document describes the collection of semantic macros for content MathML and their LaTeXML bindings. These macros can be used to mark up mathematical formulae, exposing their functional/logical structure. This structure can be used by MKM systems for added-value services, either directly from the sTeX sources, or after translation. Even though it is part of the sTeX collection, it can be used independently. Note that this documentation of the package presupposes the discussion of the sTeX collection to be self-contained.

1.1 Encoding Content MathML in TeX/LaTeX

The `cmathml` packge presented here addresses part of transformation problem: representing mathematical formulae in the LaTeX workflow, so that content MathML representations can be derived from them. The underlying problem is that run-of-the-mill TeX/LaTeX only specifies the presentation (i.e. what formulae look like) and not their content (their functional structure). Unfortunately, there are no good methods (yet) to infer the latter from the former, but there are ways to get presentation from content.

The solution to this problem is to dump the extra work on the author (after all she knows what she is talking about) and give them the chance to specify the intended structure. The markup infrastructure supplied by the `cmathml` package lets the author do this without changing the visual appearance, so that the LaTeX workflow is not disrupted.

To use these `cmathml` macros in a LaTeX document, you will have to include the `cmathml` package using `\usepackage{cmathml}` somewhere in the document preamble. Then you can use the macros

```
$\Ceq{\Cexp{\Ctimes{\Cimaginaryi,\Cpi}},\Cuminus{\Ccn{1}}}$
```

which will result in $e^{i\pi} = -1$ when the document is formatted in LaTeX. If the document is converted to Xml using the LaTeXML conversion tool, then the result will be content MathML representation:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <eq/>
    <apply>
      <exp/>
      <apply><times><imaginaryi/><pi/></times></apply>
    </apply>
    <apply><minus/><cn>1</cn></apply>
  </apply>
</math>
```

Example 1: Content MathML Form of $e^{i\pi} = -1$

1.2 Changing the TeX/LaTeX Presentation

It is possible to change the default presentation (i.e. the result under LaTeX formatting): The semantic macros only function as interface control sequences, which call an internal macro that does the actual presentation. Thus we simply have to redefine the internal macro to change the presentation. This is possible locally or globally in the following way:

```
\makeatletter
\gdef\CMathML@exp#1{exp(#1)}
\def\CMathML@pi{\varpi}
\makeatother
```

The first line is needed to lift the LaTeX redefinition protection for internal macros (those that contain the character), and the last line restores it for the rest of the document. The second line has a *global* (i.e. the presentation will be changed from this point on to the end of the document.) redefinition of the presentation of the exponential function in the LaTeX output. The third line has a *local* redefinition of the presentation (i.e. in the local group induced by LaTeX's `begin/end` grouping or by TeX's grouping induced by curly braces). Note that the argument structure has to be respected by the presentation redefinitions. Given the redefinitions above, our equation above would come out as $exp(i\varpi) = -1$.

1.3 The Future: Heuristic Parsing

The current implementation of content MathML transformation from LaTeX to MathML lays a heavy burden on the content author: the LaTeX source must be semantically preloaded — the structure of the formulae must be fully annotated. In our example above, we had to write
EdN:1 `\Ceq{A,B}` instead of the more conventional (and more legible) `A=B`.[1]

The reason for this is that this keeps the transformation to content MathML very simple, predictable and robust at the expense of authoring convenience. The implementation described in this module should be considered as a first step and fallback solution only. Future versions of the LaTeXML tool will feature more intelligent solutions for determining the implicit structure of more conventional mathematical notations (and LaTeX representations), so that writing content MathML via LaTeX will become less tedious.

However, such more advanced techniques usually rely on linguistic, structural, and semantic information about the mathematical objects and their preferred representations. They tend to be
EdN:2 less predictable to casual users and may lead to semantically unexpected results.[2]

[1] EdNote: come up with a good mixed example
[2] EdNote: talk about sTeX and extensibility in MathML/OpenMath/OMDoc

2 The User Interface

We will now tabulate the semantic macros for the Content MathML elements. We have divided them into modules based on the sectional structure of the MathML2 recommendation (2^{nd} edition). Before we go into the specific elements one-by-one, we will discuss some general properties of the `cmatml` macros and their LaTeXML bindings.

2.1 Generalities of the Encoding

The semantic macros provided by the `cmatml` package differ mainly in the way they treat their arguments. The simplest case are those for constants 2.12 that do not take any. Others take one, two, three, or even four arguments, which have to be TeX tokens or have to be wrapped in curly braces. For operators that are associative like addition the argument sequence is provided as a single TeX argument (wrapped in curly braces) that contains a comma-separated sequence of arguments (wrapped in curly braces where necessary).

`\Capply` The current setup of the `cmathml` infrastructure minimizes the need of specifying the MathML `apply` element, since the macros are all in applied form: As we have seen in the example in the Introduction 1, a macro call like `\Cexp{A}` corresponds to the application of the exponential function to some object, so the necessary `apply` elements in the MathML representation are implicit in the LaTeX formulation and are thus added by the transformation. Of course this only works, if the function is a content MathML element. Often, in mathematics we will have situations, where the function is a variable (or "arbitrary but fixed") function. Then the formula $f(x)$ represented as `$f(x)$` in TeX could (and sometimes will) be misunderstood by the Math parser as $f \cdot x$, i.e. a product of the number f with the number x, where x has brackets for some reason. In this case, we can disambiguate by using `\Capply{f}x`, which will also format as $f(x)$.[3]

EdN:3

By the same token, we do not need to represent the qualifier elements `condition` and `domainofapplication`[1], for binding operators. They are are folded into the special forms of the semantic macros for the binding operators below (the ones with the `Cond` and `DA` endings):

For operators that are associative, commutative, and idempotent (ACI i.e. bracketing, order, and multiplicity of arguments does not matter) MathML supplies the a special form of application as a binding operator (often called the corresponding "big operator)", which ranges over a whole set of arguments. For instance for the ACI operator $\cup$ for set union has the "big" operator for unions over collections of sets e.g. used in the power set $\bigcup_{S \subseteq T} S$ of a set T. In some cases, the "big" operators are provided independently by MathML, e.g. the ACI addition operator has the sum operator as a corresponding "big operator": $\sum_{x \in \mathbb{N}} x^i$ is the sum of the powers of x for all natural numbers. Where they are not, we will supply extra macros in the `cmathml` package, e.g. the `\CUnion` macro as the big operator for `\Cunion`.

Finally, some of the binding operators have multiple content models flagged by the existence of various modifier elements. In these cases, we have provided different semantic macros for the different cases.

2.2 The Token Elements

The MathML token elements are very simple containers that wrap some presentation MathML text. The `csymbol` element is the extension element in MathML. It's content is the presentation of symbol, and it has a `definitionURL` attribute that allows to specify a URI that specifies the semantics of the symbol. This URL can be specified in an optional argument to the `\Ccsymbol` macro, in accordance with usual mathematical practice, the `definitionURL` is not presented.

`\Ccn`
`\Cci`
`\Ccsymbol`

[3] EdNote: what about n-ary functions?

[1] We do not support the `fn` element as it is deprecated in MathML2 and the `declare` and `sep` elements, since their semantic status is unclear (to the author, if you feel it is needed, please gripe to me).

macro	args	Example	Result
`\Ccn`	token	`\Ccn{t}`	t
`\Cci`	token	`\Cci{t}`	t
`\Ccsymbol`	token, URI	`\Ccsymbol[http://w3.org]{t}`	t

Like the `\Ccsymbol` macro, all other macros in the `camthml` package take an optional argument[2] for the `definitionURL` attribute in the corresponding MathML element.

[2]This may change into a KeyVaL argument in future versions of the `cmathml` package.

2.3 The Basic Content Elements

The basic elements comprise various pieces of the MathML infrastructure. Most of the semantic macros in this section are relatively uneventful.

\Cinverse
\Ccompose
\Cident
\Cdomain
\Ccodomain
\Cimage

macro	args	Example	Result
\Cinverse	1	\Cinverse{f}	f^{-1}
\Ccompose	1	\Ccompose{f,g,h}	$f \circ g \circ h$
\Cident	0	\Cident	id
\Cdomain	1	\Cdomain{f}	$\mathrm{dom}(f)$
\Ccodomain	1	\Ccodomain{f}	$\mathrm{codom}(f)$
\Cimage	1	\Cimage{f}	$\mathbf{Im}(f)$

\Clambda
\ClambdaDA
\Crestrict

For the lambda element, we only have the domainofapplication element, so that we have three forms a λ-construct can have. The first one is the simple one where the first element is a bound variable. The second one restricts the applicability of the bound variable via a domainofapplication element, while the third one does not have a bound variable, so it is just a function restriction operator.[4]

EdN:4

macro	args	Example	Result
\Clambda	2	\Clambda{x,y}{A}	$\lambda(x, y, A)$
\ClambdaDA	3	\ClambdaDA{x}{C}{A}	$\lambda(x, y\colon C, A)$
\Crestrict	2	\Crestrict{f}{S}	$f\vert_S$

ccinterval
cointerval
ocinterval
oointerval

The interval constructor actually represents four types of intervals in MathML. Therefore we have four semantic macros, one for each combination of open and closed endings:

macro	args	Example	Result
\Cccinterval	2	\Cccinterval{1}{2}	$[1, 2]$
\Ccointerval	2	\Ccointerval{1}{2}	$[1, 2)$
\Cocinterval	2	\Cocinterval{1}{2}	$(1, 2]$
\Coointerval	2	\Coointerval{1}{2}	$(1, 2)$

\Cpiecewise
\Cpiece
\Cotherwise

The final set of semantic macros are concerned with piecewise definition of functions.

macro	args	Example	Result
\Cpiecewise	1	see below	see below
\Cpiece	2	\Cpiece{A}{B}	A if B
\Cotherwise	1	\Cotherwise{B}	1 *else*

For instance, we could define the abstract value function on the reals with the following markup

Semantic Markup	Formatted
\Ceq{\Cabs{x}, \Cpiecewise{\Cpiece{\Cuminus{x}}{\Clt{x,0}} \Cpiece{0}{\Ceq{x,0}} \Cotherwise{x}}}	$\lvert x\rvert = \begin{cases} -(x) & \text{if } (x < 0) \\ 0 & \text{if } (x = 0) \\ x & \textit{else} \end{cases}$

[4] EdNote: need ClambdaCond

2.4 Elements for Arithmetic, Algebra, and Logic

This section introduces the infrastructure for the basic arithmetic operators. The first set is very simple

\Cquotient
\Cfactorial
\Cdivide
\Cminus
\Cplus
\Cpower
\Crem
\Ctimes
\Croot

macro	args	Example	Result
\Cquotient	2	\Cquotient{1}{2}	$\frac{1}{2}$
\Cfactorial	1	\Cfactorial{7}	$7!$
\Cdivide	2	\Cdivide{1}{2}	$1 \div 2$
\Cminus	2	\Cminus{1}{2}	$1 - 2$
\Cplus	1	\Cplus{1}	1
\Cpower	2	\Cpower{x}{2}	x^2
\Crem	2	\Crem{7}{2}	$7 \bmod 2$
\Ctimes	1	\Ctimes{1,2,3,4}	$1 \cdot 2 \cdot 3 \cdot 4$
\Croot	2	\Croot{3}{2}	$\sqrt[3]{2}$

The second batch below is slightly more complicated, since they take a set of arguments. In the `cmathml` package, we treat them like associative operators, i.e. they act on a single argument that contains a sequence of comma-separated arguments[5]

EdN:5

\Cmax
\Cmin
\Cgcd
\Clcm

macro	args	Example	Result
\Cmax	1	\Cmax{1,3,6}	$\max(1,3,6)$
\Cmin	1	\Cmin{1,4,5}	$\min(1,4,7)$
\Cgcd	1	\Cgcd{7,3,5}	$\gcd(7,3,5)$
\Clcm	1	\Clcm{3,5,4}	$\mathrm{lcm}(3,5,4)$

EdN:6

The operators for the logical connectives are associative as well[6]. Here, conjunction, (exclusive) disjunction are n-ary associative operators, therefore their semantic macro only has one TEX argument which contains a comma-separated list of subformulae.

\Cand
\Cor
\Cxor
\Cnot
\Cimplies

macro	args	Example	Result
\Cand	1	\Cand{A,B,C}	$A \wedge B \wedge C$
\Cor	1	\Cor{A,B,C}	$A \vee B \vee C$
\Cxor	1	\Cxor{A,B,C}	$A \oplus B \oplus C$
\Cnot	1	\Cnot{A}	$\neg A$
\Cimplies	2	\Cimplies{A}{B}	$A \Longrightarrow B$

The following are the corresponding big operators, where appropriate.

\CAndDA
\CAndCond
\COrDA
\COrCond
\CXorDA
\CXorCond

macro	args	Example	Result
\CAndDA	2	\CAndDA\Cnaturalnumbers\phi	$\bigwedge_{\mathbb{N}} \phi$
\CAndCond	3	\CAndCond{x}{\Cgt{x}5}{\psi(x)}	$\bigwedge_{x} x5$
\COrDA	2	\COrDA\Cnaturalnumbers\phi	$\bigvee_{\mathbb{N}} \phi$
\COrCond	3	\COrCond{x}{\Cgt{x}5}{\psi(x)}	$\bigvee_{x5} \psi(x)$
\CXorDA	2	\CXorDA\Cnaturalnumbers\phi	$\bigoplus_{\mathbb{N}} \phi$
\CXorCond	3	\CXorCond{x}{\Cgt{x}5}{\psi(x)}	$\bigoplus_{x5} \psi(x)$

The semantic macros for the quantifiers come in two forms: with- and without a condition qualifier. In a restricted quantification of the form $\forall x, C : A$, the bound variable x ranges over all values, such that C holds (x will usually occur in the condition C). In an unrestricted quantification of the form $\forall x : A$, the bound variable ranges over all possible values for x.

\Cforall
\CforallCond
\Cexists
\CexistsCond

[5] EdNote: implement this in the latexml side
[6] EdNote: maybe add some precedences here.

macro	args	Example	Result
\Cforall	2	\Cforall{x,y}{A}	$\forall x, y\colon A$
\CforallCond	3	\CforallCond{x}{C}{A}	$\forall x, C\colon A$
\Cexists	2	\Cexists{x,y}{A}	$\exists x, y\colon A$
\CexistsCond	3	\CexistsCond{x}{C}{A}	$\exists x, C\colon A$

\Cabs
\Cconjugate
\Carg
\Creal
\Cimaginary
\Cfloor
\Cceiling

The rest of the operators are very simple in structure.

macro	args	Example	Result
\Cabs	1	\Cabs{x}	$\lvert x\rvert$
\Cconjugate	1	\Cconjugate{x}	$\overline{x}$
\Carg	1	\Carg{x}	$\angle x$
\Creal	1	\Creal{x}	$\Re x$
\Cimaginary	1	\Cimaginary{x}	$\Im x$
\Cfloor	1	\Cfloor{1.3}	$\lfloor 1.3\rfloor$
\Cceiling	1	\Cceiling{x}	$\lceil x\rceil$

2.5 Relations

The relation symbols in MathML are mostly n-ary associative operators (taking a comma-separated list as an argument).

\Ceq
\Cneq
\Cgt
\Clt
\Cgeq
\Cleq
\Cequivalent
\Capprox
\Cfactorof

macro	args	Example	Result
\Ceq	1	\CeqA,B,C	$A = B = C$
\Cneq	2	\Cneq{1}{2}	$1 \neq 2$
\Cgt	1	\Cgt{A,B,C}	$A > B > C$
\Clt	1	\Clt{A,B,C}	$A < B < C$
\Cgeq	1	\Cgeq{A,B,C}	$A \geq B \geq C$
\Cleq	1	\Cleq{A,B,C}	$A \leq B \leq C$
\Cequivalent	1	\Cequivalent{A,B,C}	$A \equiv B \equiv C$
\Capprox	2	\Capprox{1}{2}	$1 \approx 1.1$
\Cfactorof	2	\Cfactorof{7}{21}	$7 \mid 21$

2.6 Elements for Calculus and Vector Calculus

The elements for calculus and vector calculus have the most varied forms.

The integrals come in four forms: the first one is just an indefinite integral over a function, the second one specifies the bound variables, upper and lower limits. The third one specifies a set instead of an interval, and finally the last specifies a bound variable that ranges over a set specified by a condition.

\Cint
\CintLimits
\CintDA
\CintCond

macro	args	Example	Result
\Cint	1	\Cint{f}	$\int f$
\CintLimits	4	\CintLimits{x}{0}{\Cinfinit}{f(x)}	$\int_0^\infty f(x)dx$
\CintDA	2	\CintDA{\Creals}{f}	$\int_{\mathbb{R}} f$
\CintCond	3	\CintCond{x}{\Cin{x}{D}}{f(x)}	$\int_{x\in D} f(x)dx$

\Cdiff
\Cddiff

The differentiation operators are used in the usual way: simple differentiation is represented by the \Cdiff macro which takes the function to be differentiated as an argument, differentiation with the d-notation is possible by the \Cddiff, which takes the bound variable[7] as the first argument and the function expression (in the bound variable) as a second argument.

EdN:7

\Cpartialdiff

Partial Differentiation is specified by the \Cpartialdiff macro. It takes the overall degree as

[7] EdNote: really only one?

the first argument (to leave it out, just pass the empty argument). The second argument is the list of bound variables (with their degrees; see below), and the last the function expression (in these bound variables). To specify the respective degrees of differentiation on the variables, we use the `\Cdegree` macro, which takes two arguments (but no optional argument), the first one is the degree (a natural number) and the second one takes the variable. Note that the overall degree has to be the sum of the degrees of the bound variables.

\Cdegree

macro	args	Example	Result
`\Cdiff`	1	`\Cdiff{f}`	f'
`\Cddiff`	2	`\Cddiff{x}{f}`	$\frac{df(x)}{dx}$
`\Cpartialdiff`	3	`\Cpartialdiff{3}{x,y,z}{f(x,y)}`	$\frac{\partial^3}{\partial x,y,z} f(x,y)$
`\Cpartialdiff`	3	`\Cpartialdiff{7}{\Cdegree{2}{x},\Cdegree{4}{y},z}{f(x,y)}`	$\frac{\partial^7}{\partial 2^x,4^y,z} f(x,y)$

\Climit
\ClimitCond

For content MathML, there are two kinds of limit expressions: The simple one is specified by the `\Climit` macro, which takes three arguments: the bound variable, the target, and the limit expression. If we want to place additional conditions on the limit construction, then we use the `\ClimitCond` macro, which takes three arguments as well, the first one is a sequence of bound variables, the second one is the condition, and the third one is again the limit expression.

\Ctendsto
\CtendstoAbove
\CtendstoBelow

If we want to speak qualitatively about limit processes (e.g. in the condition of a `\ClimitCond` expression), then can use the MathML `tendsto` element, which is represented by the `\Ctendsto` macro, wich takes two expressions arguments. In MathML, the `tendsto` element can be further specialized by an attribute to indicate the direction from which a limit is approached. In the `cmathml` package, we supply two additional (specialized) macros for that: `\CtendstoAbove` and `\CtendstoBelow`.

macro	args	Example	Result
`\Climit`	3	`\Climit{x}{0}{\Csin{x}}`	$\lim_{x\to 0} \sin(x)$
`\ClimitCond`	3	`\ClimitCond{x}{\Ctendsto{x}{0}}{\Ccos{x}}`	$\lim_{x\to 0} \cos(x)$
`\Ctendsto`	2	`\Ctendsto{f(x)}{2}`	$f(x) \to 2$
`\CtendstoAbove`	2	`\CtendstoAbove{x}{1}`	$x \searrow 1$
`\CtendstoBelow`	2	`\CtendstoBelow{x}{2}`	$x \nearrow 2$

\Cdivergence
\Cgrad
\Ccurl
\Claplacian

macro	args	Example	Result
`\Cdivergence`	1	`\Cdivergence{A}`	$\nabla \cdot A$
`\Cgrad`	1	`\Cgrad{\Phi}`	$\nabla \Phi$
`\Ccurl`	1	`\Ccurl{\Xi}`	$\nabla \times \Xi$
`\Claplacian`	1	`\Claplacian{A}`	$\nabla^2 A$

2.7 Sets and their Operations

\Cset
\Clist
\CsetDA
\CsetRes
\CsetCond

The `\Cset` macros is used as the simple finite set constructor, it takes one argument that is a comma-separated sequence of members of the set. `\CsetRes` allows to specify a set by restricting a set of variables, and `\CsetCond` is the general form of the set construction.[8]

EdN:8

macro	args	Example	Result
`\Cset`	1	`\Cset{1,2,3}`	$\{1,2,3\}$
`\CsetRes`	2	`\CsetRes{x}{\Cgt{x}5}`	$\{x\|x5\}$
`\CsetCond`	3	`\CsetCond{x}{\Cgt{x}5}{\Cpower{x}3}`	$\{x5\|x^3\}$
`\CsetDA`	3	`\CsetDA{x}{\Cgt{x}5}{S_x}`	$\{x \in x5\|S_x\}$
`\Clist`	1	`\Clist{3,2,1}`	$\mathrm{list}(3,2,1)$

\Cunion
\Cintersect
\Ccartesianproduct
\Csetdiff
\Ccard
\Cin
\Cnotin

macro	args	Example	Result
\Cunion	1	\Cunion{S,T,L}	$S \cup T \cup L$
\Cintersect	1	\Cintersect{S,T,L}	$S \cap T \cap L$
\Ccartesianproduct	1	\Ccartesianproduct{A,B,C}	$A \times B \times C$
\Csetdiff	2	\Csetdiff{S}{L}	$S \setminus L$
\Ccard	1	\Ccard{\Cnaturalnumbers}	$\#\mathbb{N}$
\Cin	2	\Cin{a}{S}	$a \in S$
\Cnotin	2	\Cnotin{b}{S}	$b \notin S$

\CUnionDA
\CUnionCond
\CIntersectDA
\CIntersectCond
\CCartesianproductDA
\CCartesianproductCond

The following are the corresponding big operators for the first three binary ACI functions.

macro	args	Example	Result
\CUnionDA	2	\CUnionDA\Cnaturalnumbers{S_i}	$\bigwedge_{\mathbb{N}} S_i$
\CUnionCond	3	\CUnionCond{x}{\Cgt{x}5}{S_x}}	$\bigwedge_{x} x5$
\CIntersectDA	2	\CIntersectDA\Cnaturalnumbers{S_i}	$\bigvee_{\mathbb{N}} S_i$
\CIntersectCond	3	\CIntersectCond{x}{\Cgt{x}5}{S_x}	$\bigvee_{x5} S_x$
\CCartesianproductDA	2	\CCartesianproductDA\Cnaturalnumbers{S_i}	$\bigoplus_{\mathbb{N}} S_i$
\CCartesianproductCond	3	\CCartesianproductCond{x}{\Cgt{x}5}{S_x}	$\bigoplus_{x5} S_x$

\Csubset
\Cprsubset
\Cnotsubset
\Cnotprsubset

For the set containment relations, we are in a somewhat peculiar situation: content MathML only supplies the subset side of the reations and leaves out the superset relations. Of course they are not strictly needed, since they can be expressed in terms of the subset relation with reversed argument order. But for the `cmathml` package, the macros have a presentational side (for the LaTeX workflow) and a content side (for the LaTeXML converter) therefore we will need macros for both relations.

macro	args	Example	Result
\Csubset	1	\Csubset{S,T,K}	$S \subseteq T \subseteq K$
\Cprsubset	1	\Cprsubset{S,T,K}	$S \subset T \subset K$
\Cnotsubset	2	\Cnotsubset{S}{K}	$S \not\subseteq K$
\Cnotprsubset	2	\Cnotprsubset{S}{L}	$S \not\subset L$

\Csupset
\Cprsupset
\Cnotsupset
\Cnotprsupset

The following set of macros are presented in LaTeX as their name suggests, but upon transformation will generate content markup with the MathML elements (i.e. in terms of the subset relation).

macro	args	Example	Result
\Csupset	1	\Csupset{S,T,K}	$S \supseteq T \supseteq K$
\Cprsupset	1	\Cprsupset{S,T,K}	$S \supset T \supset K$
\Cnotsupset	2	\Cnotsupset{S}{K}	$S \not\supseteq K$
\Cnotprsupset	2	\Cnotprsupset{S}{L}	$S \not\supset L$

2.8 Sequences and Series

\CsumLimits
\CsumCond
\CsumDA
\CprodLimist
\CprodCond
\CprodDA

macro	args	Example	Result
\CsumLimits	4	\CsumLimits{i}{0}{50}{x^i}	$\sum_{i=0}^{50} x^i$
\CsumCond	3	\CsumCond{i}{\Cintegers}{i}	$\sum_{i\in\mathbb{Z}} i$
\CsumDA	2	\CsumDA{\Cintegers}{f}	$\sum_{\mathbb{Z}} f$
\CprodLimits	4	\CprodLimits{i}{0}{20}{x^i}	$\prod_{i=20}2^{20}x^i$
\CprodCond	3	\CprodCond{i}{\Cintegers}{i}	$\prod_{i\in\mathbb{Z}} i$
\CprodDA	2	\CprodDA{\Cintegers}{f}	$\prod_{f}$

[8] EdNote: need to do this for lists as well? Probably

2.9 Elementary Classical Functions

\Csin
\Ccos
\Ctan
\Csec
\Ccsc
\Ccot

macro	args	Example	Result
\Csin	1	\Csin{x}	$\sin(x)$
\Ccos	1	\Ccos{x}	$\cos(x)$
\Ctan	1	\Ctan{x}	$\tan(x)$
\Csec	1	\Csec{x}	$\sec(x)$
\Ccsc	1	\Ccsc{x}	$\csc(x)$
\Ccot	1	\Ccot{x}	$\cot(x)$

\Csinh
\Ccosh
\Ctanh
\Csech
\Ccsch
\Ccoth

macro	args	Example	Result
\Csinh	1	\Csinh{x}	$\sinh(x)$
\Ccosh	1	\Ccosh{x}	$\cosh(x)$
\Ctanh	1	\Ctanh{x}	$\tanh(x)$
\Csech	1	\Csech{x}	$\operatorname{sech}(x)$
\Ccsch	1	\Ccsch{x}	$\operatorname{csch}(x)$
\Ccoth	1	\Ccoth{x}	$\coth(x)$

\Carcsin
\Carccos
\Carctan
\Carcsec
\Carccsc
\Carccot

macro	args	Example	Result
\Carcsin	1	\Carcsin{x}	$\arcsin(x)$
\Carccos	1	\Carccos{x}	$\arccos(x)$
\Carctan	1	\Carctan{x}	$\arctan(x)$
\Carccosh	1	\Carccosh{x}	$\operatorname{arccosh}(x)$
\Carccot	1	\Carccot{x}	$\operatorname{arccot}(x)$

\Carcsinh
\Carccosh
\Carctanh
\Carcsech
\Carccsch
\Carccoth

macro	args	Example	Result
\Carccoth	1	\Carccoth{x}	$\operatorname{arccoth}(x)$
\Carccsc	1	\Carccsc{x}	$\operatorname{arccsc}(x)$
\Carcsinh	1	\Carcsinh{x}	$\operatorname{arcsinh}(x)$
\Carctanh	1	\Carctanh{x}	$\operatorname{arctanh}(x)$
\Cexp	1	\Cexp{x}	$\exp(x)$
\Cln	1	\Cln{x}	$\ln(x)$
\Clog	2	\Clog{5}{x}	$\log_5(x)$

2.10 Statistics

The only semantic macro that is non-standard in this module is the one for the `moment` and `momentabout` elements in MathML. They are combined into the semantic macro `CmomentA`; its first argument is the degree, its second one the point in the distribution, the moment is taken about, and the third is the distribution.

\Cmean
\Csdev
\Cvar
\Cmedian
\Cmode
\Cmoment
\CmomentA

macro	args	Example	Result
\Cmean	1	\Cmean{X}	$\operatorname{mean}(X)$
\Csdev	1	\Csdev{X}	$\operatorname{std}(X)$
\Cvar	1	\Cvar{X}	$\operatorname{var}(X)$
\Cmedian	1	\Cmedian{X}	$\operatorname{median}(X)$
\Cmode	1	\Cmode{X}	$\operatorname{mode}(X)$
\Cmoment	3	\Cmoment{3}{X}	$\langle X^3 \rangle$
\CmomentA	3	\CmomentA{3}{p}{X}	$\langle p^3 \rangle X$

2.11 Linear Algebra

In these semantic macros, only the matrix constructor is unusual; instead of constructing a matrix from `matrixrow` elements like MathML does, the macro follows the TEX/LATEX tradition allows to give a matrix as an array. The first argument of the macro is the column specification (it will only be used for presentation purposes), and the second one the rows.

\Cvector
\Cmatrix
\Cdeterminant
\Ctranspose
\Cselector
\Cvectorproduct
\Cscalarproduct
\Couterproduct

macro	args	Example	Result
\Cvector	1	\Cvector{1,2,3}	$(1,2,3)$
\Cmatrix	2	\Cmatrix{ll}{1 & 2\\ 3 & 4}	$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$
\Cdeterminant	1	\Cdeterminant{A}	$\lvert A \rvert$
\Ctranspose	1	\Ctranspose{A}	$A^\top$
\Cselector	2	\Cselector{A}{2}	A_2
\Cvectproduct	2	\Cvectproduct{\phi}{\psi}	$\phi \cdot \psi$
\Cscalarproduct	2	\Cscalarproduct{\phi}{\psi}	$\phi\psi$
\Couterproduct	2	\Couterproduct{\phi}{\psi}	$\phi \times \psi$

2.12 Constant and Symbol Elements

The semantic macros for the MathML constant and symbol elements are very simple, they do not take any arguments, and their name is just the MathML element name prefixed by a capital C.

\Cintegers
\Creals
\Crationals
\Ccomplexes
\Cprimes

macro	args	Example	Result
\Cintegers		\Cintegers	$\mathbb{Z}$
\Creals		\Creals	$\mathbb{R}$
\Crationals		\Crationals	$\mathbb{Q}$
\Cnaturalnumbers		\Cnaturalnumbers	$\mathbb{N}$
\Ccomplexes		\Ccomplexes	$\mathbb{C}$
\Cprimes		\Cprimes	$\mathbb{P}$

\Cexponentiale
\Cimaginaryi
\Ctrue
\Cfalse
\Cemptyset
\Cpi
\Ceulergamma
\Cinfinit

macro	args	Example	Result
\Cexponemtiale		\Cexponemtiale	e
\Cimaginaryi		\Cimaginaryi	i
\Cnotanumber		\Cnotanumber	NaN
\Ctrue		\Ctrue	true
\Cfalse		\Cfalse	false
\Cemptyset		\Cemptyset	$\emptyset$
\Cpi		\Cpi	π
\Ceulergamma		\Ceulergamma	γ
\Cinfinit		\Cinfinit	∞

2.13 Extensions

Content MathML does not (even though it claims to cover M-14 Math) symbols for all the common mathematical notions. The `cmathmlx` attempts to collect these and provide TEX/LATEX and LATEXML bindings.

\Ccomplement

macro	args	Example	Result
\Ccomplement	1	\Ccomplement{\Cnaturalnumbers}	$\mathbb{N}^c$

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX GitHub repository [sTeX].

1. none reported yet

4 The Implementation

4.1 Initialization and auxiliary functions

We first make sure that the SᴛEX `presentation` package is loaded.

```
⟨*sty | styx⟩
\RequirePackage{presentation}
⟨/sty | styx⟩
```

The structural macros are rather simple:

```
⟨*sty⟩
\newcommand\Capply[3][]{#2(#3)}
%    after this, the implementation will always have the same form. We will first
%    implement a block of {\LaTeX} macros via a |\newcommand| and then specify the
%    corresponding {\latexml} bindings for them.
%
% \subsection{The Token Elements}\label{impl:tokens}
%
%    \begin{macrocode}
\def\CMathML@cn#1{#1}
\newcommand\Ccn[2][]{\CMathML@cn{#2}}
\def\CMathML@ci#1{#1}
\newcommand\Cci[2][]{\CMathML@ci{#2}}
\def\CMathML@csymbol#1{#1}
\newcommand\Ccsymbol[2][]{\CMathML@csymbol{#2}}
```

4.2 The Basic Elements

```
\def\CMathML@ccinterval#1#2{[#1,#2]}
\newcommand\Cccinterval[3][]{\CMathML@ccinterval{#2}{#3}}
\def\CMathML@cointerval#1#2{[#1,#2)}
\newcommand\Ccointerval[3][]{\CMathML@cointerval{#2}{#3}}
\def\CMathML@ocinterval#1#2{(#1,#2]}
\newcommand\Cocinterval[3][]{\CMathML@ocinterval{#2}{#3}}
\def\CMathML@oointerval#1#2{(#1,#2)}
\newcommand\Coointerval[3][]{\CMathML@oointerval{#2}{#3}}

\newcommand\Cinverse[2][]{#2^{-1}}
% what about separator

\def\CMathML@lambda#1#2{\lambda({#1},{#2})}
\newcommand\Clambda[3][]{\CMathML@lambda{#2}{#3}}
\def\CMathML@lambdaDA#1#2#3{\lambda({#1}\colon{#2},#3)}
\newcommand\ClambdaDA[4][]{\CMathML@lambdaDA{#2}{#3}{#4}}
\def\CMathML@restrict#1#2{\left.#1\right|_{#2}}
\newcommand\Crestrict[3][]{\CMathML@restrict{#2}{#3}}
%\ednote{need do deal with multiple variables!}

\def\CMathML@composeOp{\circ}
\newcommand\CcomposeOp{\CMathML@composeOp}
\def\CMathML@compose#1{\assoc[p=500,pi=500]{\CMathML@composeOp}{#1}}
\newcommand\Ccompose[2][]{\CMathML@compose{#2}}
\def\CMathML@ident#1{\mathrm{id}}
\newcommand\Cident[1][]{\CMathML@ident{#1}}
\def\CMathML@domain#1{\mbox{dom}(#1)}
\newcommand\Cdomain[2][]{\CMathML@domain{#2}}
\def\CMathML@codomain#1{\mbox{codom}(#1)}
\newcommand\Ccodomain[2][]{\CMathML@codomain{#2}}
\def\CMathML@image#1{{\mathbf{Im}}(#1)}
\newcommand\Cimage[2][]{\CMathML@image{#2}}
```

```
\def\CMathML@piecewise#1{\left\{\begin{array}{ll}#1\end{array}\right.}
\newcommand\Cpiecewise[2][]{\CMathML@piecewise{#2}}
\def\CMathML@piece#1#2{#1&{\mathrm{if}}\;{#2}\\}
\newcommand\Cpiece[3][]{\CMathML@piece{#2}{#3}}
\def\CMathML@otherwise#1{#1&else\\}
\newcommand\Cotherwise[2][]{\CMathML@otherwise{#2}}
```

4.3 Elements for Arithmetic, Algebra, and Logic

```
\def\CMathML@quotient#1#2{\frac{#1}{#2}}
\newcommand\Cquotient[3][]{\CMathML@quotient{#2}{#3}}
\def\CMathML@factorialOp{!}
\newcommand\CfactorialOp{\CMathML@factorialOp}
\def\CMathML@factorial#1{#1{\CMathML@factorialOp}}
\newcommand\Cfactorial[2][]{\CMathML@factorial{#2}}
\def\CMathML@divideOp{\div}
\newcommand\CdivideOp{\CMathML@divideOp}
\def\CMathML@divide#1#2{\infix[p=400]{\CMathML@divideOp}{#1}{#2}}
\newcommand\Cdivide[3][]{\CMathML@divide{#2}{#3}}
\def\CMathML@maxOp{\mathrm{max}}
\newcommand\CmaxOp{\CMathML@maxOp}
\def\CMathML@max#1{{\CMathML@maxOp}(#1)}
\newcommand\Cmax[2][]{\CMathML@max{#2}}
\def\CMathML@minOp{\mathrm{min}}
\newcommand\CminOp{\CMathML@minOp}
\def\CMathML@min#1{{\CMathML@minOp}(#1)}
\newcommand\Cmin[2][]{\CMathML@min{#2}}
\def\CMathML@minusOp{-}
\newcommand\CminusOp{\CMathML@minusOp}
\def\CMathML@minus#1#2{\infix[p=500]{\CMathML@minusOp}{#1}{#2}}
\newcommand\Cminus[3][]{\CMathML@minus{#2}{#3}}
\def\CMathML@uminus#1{\prefix[p=200]{\CMathML@minusOp}{#1}}
\newcommand\Cuminus[2][]{\CMathML@uminus{#2}}
\def\CMathML@plusOp{+}
\newcommand\CplusOp{\CMathML@plusOp}
\def\CMathML@plus#1{\assoc[p=500]{\CMathML@plusOp}{#1}}
\newcommand\Cplus[2][]{\CMathML@plus{#2}}
\def\CMathML@power#1#2{\infix[p=200]{^}{#1}{#2}}
\newcommand\Cpower[3][]{\CMathML@power{#2}{#3}}
\def\CMathML@remOp{\bmod}
\newcommand\CremOp{\CMathML@remOp}
\def\CMathML@rem#1#2{#1 \CMathML@remOp #2}
\newcommand\Crem[3][]{\CMathML@rem{#2}{#3}}
\def\CMathML@timesOp{\cdot}
\newcommand\CtimesOp{\CMathML@timesOp}
\def\CMathML@times#1{\assoc[p=400]{\CMathML@timesOp}{#1}}
\newcommand\Ctimes[2][]{\CMathML@times{#2}}
\def\CMathML@rootOp{\sqrt}
\newcommand\CrootOp{\CMathML@rootOp{}}
\def\CMathML@root#1#2{\CMathML@rootOp[#1]{#2}}
\newcommand\Croot[3][]{\CMathML@root{#2}{#3}}
\def\CMathML@gcd#1{\gcd(#1)}
\newcommand\Cgcd[2][]{\CMathML@gcd{#2}}
\def\CMathML@andOp{\wedge}
\newcommand\CandOp{\CMathML@andOp}
\def\CMathML@and#1{\assoc[p=400]{\CMathML@andOp}{#1}}
\newcommand\Cand[2][]{\CMathML@and{#2}}
\def\CMathML@orOp{\vee}
\newcommand\CorOp{\CMathML@orOp}
```

```
\def\CMathML@or#1{\assoc[p=500]{\CMathML@orOp}{#1}}
\newcommand\Cor[2][]{\CMathML@or{#2}}
\def\CMathML@xorOp{\oplus}
\newcommand\CxorOp{\CMathML@xorOp}
\def\CMathML@xor#1{\assoc[p=400]{\CMathML@xorOp}{#1}}
\newcommand\Cxor[2][]{\CMathML@xor{#2}}
\def\CMathML@notOp{\neg}
\newcommand\CnotOp{\CMathML@notOp}
\def\CMathML@not#1{\CMathML@notOp{#1}}
\newcommand\Cnot[2][]{\CMathML@not{#2}}
\def\CMathML@impliesOp{\Longrightarrow}
\newcommand\CimpliesOp{\CMathML@impliesOp}
\def\CMathML@implies#1#2{#1\CMathML@impliesOp{#2}}
\newcommand\Cimplies[3][]{\CMathML@implies{#2}{#3}}
```

EdN:9 [9]

```
\def\CMathML@AndDA#1#2{\bigwedge_{#1}{#2}} % set, scope
\newcommand\CAndDA[3][]{\CMathML@AndDA{#2}{#3}}
\def\CMathML@AndCond#1#2#3{\bigwedge_{#2}{#3}} % bvars,condition, scope
\newcommand\CAndCond[4][]{\CMathML@AndCond{#2}{#2}{#3}}
\def\CMathML@OrDA#1#2{\bigvee_{#1}{#2}} % set, scope
\newcommand\COrDa[3][]{\CMathML@OrDA{#2}{#3}}
\def\CMathML@OrCond#1#2#3{\bigvee_{#2}{#3}}% bvars,condition, scope
\newcommand\COrCond[4][]{\CMathML@OrCond{#2}{#3}{#4}}
\def\CMathML@XorDA#1#2{\bigoplus_{#1}{#2}} % set, scope
\newcommand\CXorDA[3][]{\CMathML@XorDA{#2}{#3}}
\def\CMathML@XorCond#1#2#3{\bigoplus_{#2}{#3}}% bvars,condition, scope
\newcommand\CXorCond[4][]{\CMathML@XorCond{#2}{#3}{#4}}
%
\def\CMathML@forall#1#2{\forall{#1}\colon{#2}}
\newcommand\Cforall[3][]{\CMathML@forall{#2}{#3}}
\def\CMathML@forallCond#1#2#3{\forall{#1},{#2}\colon{#3}}  % list), condition, scope
\newcommand\CforallCond[4][]{\CMathML@forallCond{#2}{#3}{#4}}
\def\CMathML@exists#1#2{\exists{#1}\colon{#2}}
\newcommand\Cexists[3][]{\CMathML@exists{#2}{#3}}
\def\CMathML@esistsCont#1#2#3{\exists{#1},{#2}\colon{#3}}
\newcommand\CexistsCond[4][]{\CMathML@esistsCont{#2}{#3}{#4}}
\def\CMathML@abs#1{\left|#1\right|}
\newcommand\Cabs[2][]{\CMathML@abs{#2}}
\def\CMathML@conjugate#1{\overline{#1}}
\newcommand\Cconjugate[2][]{\CMathML@conjugate{#2}}
\def\CMathML@arg#1{\angle #1}
\newcommand\Carg[2][]{\CMathML@arg{#2}}
\def\CMathML@real#1{\Re #1}
\newcommand\Creal[2][]{\CMathML@real{#2}}
\def\CMathML@imaginary#1{\Im #1}
\newcommand\Cimaginary[2][]{\CMathML@imaginary{#2}}
\def\CMathML@lcm#1{\mbox{lcm}(#1)}
\newcommand\Clcm[2][]{\CMathML@lcm{#2}}
\def\CMathML@floor#1{\left\lfloor{#1}\right\rfloor}
\newcommand\Cfloor[2][]{\CMathML@floor{#2}}
\def\CMathML@ceiling#1{\left\lceil{#1}\right\rceil}
\newcommand\Cceiling[2][]{\CMathML@ceiling{#2}}
```

4.4 Relations

```
\def\CMathML@eqOp{=}
```

[9] EdNote: need to do something about the associative things in ltxml

```
\newcommand\CeqOp{\CMathML@eqOp}
\def\CMathML@eq#1{\assoc[p=700]{\CMathML@eqOp}{#1}}
\newcommand\Ceq[2][]{\CMathML@eq{#2}}
\def\CMathML@neqOp{\neq}
\newcommand\CneqOp{\CMathML@neqOp}
\def\CMathML@neq#1#2{\infix[p=700]{\CMathML@neqOp}{#1}{#2}}
\newcommand\Cneq[3][]{\CMathML@neq{#2}{#3}}
\def\CMathML@gtOp{>}
\newcommand\CgtOp{\CMathML@gtOp}
\def\CMathML@gt#1{\assoc[p=700]{\CMathML@gtOp}{#1}}
\newcommand\Cgt[2][]{\CMathML@gt{#2}}
\def\CMathML@ltOp{<}
\newcommand\CltOp{\CMathML@ltOp}
\def\CMathML@lt#1{\assoc[p=700]{\CMathML@ltOp}{#1}}
\newcommand\Clt[2][]{\CMathML@lt{#2}}
\def\CMathML@geqOp{\geq}
\newcommand\CgeqOp{\CMathML@geqOp}
\def\CMathML@geq#1{\assoc[p=700]{\CMathML@geqOp}{#1}}
\newcommand\Cgeq[2][]{\CMathML@geq{#2}}
\def\CMathML@leqOp{\leq}
\newcommand\CleqOp{\CMathML@leqOp}
\def\CMathML@leq#1{\assoc[p=700]{\CMathML@leqOp}{#1}}
\newcommand\Cleq[2][]{\CMathML@leq{#2}}
\def\CMathML@equivalentOp{\equiv}
\newcommand\CequivalentOp{\CMathML@equivalentOp}
\def\CMathML@equivalent#1{\assoc[p=700]{\CMathML@equivalentOp}{#1}}
\newcommand\Cequivalent[2][]{\CMathML@equivalent{#2}}
\def\CMathML@approxOp{\approx}
\newcommand\CapproxOp{\CMathML@approxOp}
\def\CMathML@approx#1#2{#1\CMathML@approxOp{#2}}
\newcommand\Capprox[3][]{\CMathML@approx{#2}{#3}}
\def\CMathML@factorofOp{\mid}
\newcommand\CfactorofOp{\CMathML@factorofOp}
\def\CMathML@factorof#1#2{#1\CMathML@factorofOp{#2}}
\newcommand\Cfactorof[3][]{\CMathML@factorof{#2}{#3}}

\def\CMathML@intOp{\int}
\newcommand\CintOp{\CMathML@intOp}
\def\CMathML@int#1{\CMathML@intOp{#1}}
\newcommand\Cint[2][]{\CMathML@int{#2}}
\def\CMathML@intLimits#1#2#3#4{\CMathML@intOp_{#2}^{#3}{#4}d{#1}} %bvars,llimit, ulimit,body
\newcommand\CintLimits[5][]{\CMathML@intLimits{#2}{#3}{#4}{#5}}
\def\CMathML@intSet#1#2{\CMathML@intOp_{#1}{#2}}% set,function
\newcommand\CintDA[3][]{\CMathML@intSet{#2}{#3}}
\def\CMathML@intCond#1#2#3{\CMathML@intOp_{#2}{#3}d{#1}} %bvars, condition, body
\newcommand\CintCond[4][]{\CMathML@intCond{#2}{#3}{#4}}

\def\CMathML@diff#1{#1'}
\newcommand\Cdiff[2][]{\CMathML@diff{#2}}
\def\CMathML@ddiff#1#2{{d{#2}(#1)\over{d{#1}}}}
\newcommand\Cddiff[3][]{\CMathML@ddiff{#2}{#3}}
\def\CMathML@partialdiff#1#2#3{{\partial^{#1}\over\partial{#2}}{#3}}% degree, bvars, body
\newcommand\Cpartialdiff[4][]{\CMathML@partialdiff{#2}{#3}{#4}}
\newcommand\Cdegree[2]{#1^{#2}}

\def\CMathML@limit#1#2#3{\lim_{#1\rightarrow{#2}}{#3}}
\newcommand\Climit[4][]{\CMathML@limit{#2}{#3}{#4}}   % bvar, lowlimit, scope
\def\CMathML@limitCond#1#2#3{\lim_{#2}{#3}}
```

```
\newcommand\ClimitCond[4][]{\CMathML@limitCond{#2}{#3}{#4}}   % bvars, condition, scope

\def\CMathML@tendstoOp{\rightarrow}
\newcommand\CtendstoOp{\CMathML@tendstoOp}
\def\CMathML@tendsto#1#2{#1\CMathML@tendstoOp{#2}}
\newcommand\Ctendsto[3][]{\CMathML@tendsto{#2}{#3}}
\def\CMathML@tendstoAboveOp{\searrow}
\newcommand\CtendstoAboveOp{\CMathML@tendstoAboveOp}
\def\CMathML@tendstoAbove#1#2{#1\searrow{#2}}
\newcommand\CtendstoAbove[3][]{\CMathML@tendstoAbove{#2}{#3}}
\def\CMathML@tendstoBelowOp{\nearrow}
\newcommand\CtendstoBelowOp{\CMathML@tendstoBelowOp}
\def\CMathML@tendstoBelow#1#2{#1\CMathML@tendstoBelowOp{#2}}
\newcommand\CtendstoBelow[3][]{\CMathML@tendstoBelow{#2}{#3}}

\def\CMathML@divergence#1{\nabla\cdot{#1}}
\newcommand\Cdivergence[2][]{\CMathML@divergence{#2}}
\def\CMathML@grad#1{\nabla{#1}}
\newcommand\Cgrad[2][]{\CMathML@grad{#2}}
\def\CMathML@curl#1{\nabla\times{#1}}
\newcommand\Ccurl[2][]{\CMathML@curl{#2}}
\def\CMathML@laplacian#1{\nabla^2#1}
\newcommand\Claplacian[2][]{\CMathML@laplacian{#2}}
```

4.5 Sets and their Operations

```
\def\CMathML@set#1{\left\{#1\right\}}
\newcommand\Cset[2][]{\CMathML@set{#2}}
\def\CMathML@setRes#1#2{\{#1|#2\}}
\newcommand\CsetRes[3][]{\CMathML@setRes{#2}{#3}}
\def\CMathML@setCond#1#2#3{\{#2|#3\}}
\newcommand\CsetCond[4][]{\CMathML@setCond{#2}{#3}{#4}}
\def\CMathML@setDA#1#2#3{\{#1\in{#2}|#3\}}
\newcommand\CsetDA[4][]{\CMathML@setDA{#2}{#3}{#4}}
\def\CMathML@listOp{\mbox{list}}
\newcommand\ClistOp{\CMathML@listOp}
\def\CMathML@list#1{\CMathML@listOp({#1})}
\newcommand\Clist[2][]{\CMathML@list{#2}}
\def\CMathML@unionOp{\cup}
\newcommand\CunionOp{\CMathML@unionOp}
\def\CMathML@union#1{\assoc[p=500]{\CMathML@unionOp}{#1}}
\newcommand\Cunion[2][]{\CMathML@union{#2}}
\def\CMathML@intersectOp{\cap}
\newcommand\CintersectOp{\CMathML@intersectOp}
\def\CMathML@intersect#1{\assoc[p=400]{\CMathML@intersectOp}{#1}}
\newcommand\Cintersect[2][]{\CMathML@intersect{#2}}
\def\CMathML@inOp{\in}
\newcommand\CinOp{\CMathML@inOp}
\def\CMathML@in#1#2{#1\CMathML@inOp{#2}}
\newcommand\Cin[3][]{\CMathML@in{#2}{#3}}
\def\CMathML@notinOp{\notin}
\newcommand\CnotinOp{\CMathML@notinOp}
\def\CMathML@notin#1#2{#1\CMathML@notinOp{#2}}
\newcommand\Cnotin[3][]{\CMathML@notin{#2}{#3}}
\def\CMathML@setdiffOp{\setminus}
\newcommand\CsetdiffOp{\CMathML@setdiffOp}
\def\CMathML@setdiff#1#2{#1\CMathML@setdiffOp{#2}}
\newcommand\Csetdiff[3][]{\CMathML@setdiff{#2}{#3}}
\def\CMathML@cardOp{\#}
\newcommand\CcardOp{\CMathML@cardOp}
```

```
\def\CMathML@card#1{\CMathML@cardOp #1}
\newcommand\Ccard[2][]{\CMathML@card{#2}}
\def\CMathML@cartesianproductOp{\times}
\newcommand\CcartesianproductOp{\CMathML@cartesianproductOp}
\def\CMathML@cartesianproduct#1{\assoc[p=400]{\CMathML@cartesianproductOp}{#1}}
\newcommand\Ccartesianproduct[2][]{\CMathML@cartesianproduct{#2}}
\def\CMathML@subsetOp{\subseteq}
\newcommand\CsubsetOp{\CMathML@subsetOp}
\def\CMathML@subset#1{\assoc[p=700]{\CMathML@subsetOp}{#1}}
\newcommand\Csubset[2][]{\CMathML@subset{#2}}
\def\CMathML@prsubsetOp{\subset}
\newcommand\CprsubsetOp{\CMathML@prsubsetOp}
\def\CMathML@prsubset#1{\assoc[p=700]{\CMathML@prsubsetOp}{#1}}
\newcommand\Cprsubset[2][]{\CMathML@prsubset{#2}}
\def\CMathML@notsubsetOp{\not\subseteq}
\newcommand\CnotsubsetOp{\CMathML@notsubsetOp}
\def\CMathML@notsubset#1#2{#1\CMathML@notsubsetOp{#2}}
\newcommand\Cnotsubset[3][]{\CMathML@notsubset{#2}{#3}}
\def\CMathML@notprsubsetOp{\not\subset}
\newcommand\CnotprsubsetOp{\CMathML@notprsubsetOp}
\def\CMathML@notprsubset#1#2{#1\CMathML@notprsubsetOp{#2}}
\newcommand\Cnotprsubset[3][]{\CMathML@notprsubset{#2}{#3}}
```

The next set of macros are needed, since they are presentational.

```
\def\CMathML@supsetOp{\supseteq}
\newcommand\CsupsetOp{\CMathML@supsetOp}
\def\CMathML@supset#1{\assoc[p=700]{\CMathML@supsetOp}{#1}}
\newcommand\Csupset[2][]{\CMathML@supset{#2}}
\def\CMathML@prsupsetOp{\supset}
\newcommand\CprsupsetOp{\CMathML@prsupsetOp}
\def\CMathML@prsupset#1{\assoc[p=700]{\CMathML@prsupsetOp}{#1}}
\newcommand\Cprsupset[2][]{\CMathML@prsupset{#2}}
\def\CMathML@notsupsetOp{\not\supseteq}
\newcommand\CnotsupsetOp{\CMathML@notsupsetOp}
\def\CMathML@notsupset#1#2{#1\CMathML@notsupsetOp{#2}}
\newcommand\Cnotsupset[3][]{\CMathML@notsupset{#2}{#3}}
\def\CMathML@notprsupsetOp{\not\supset}
\newcommand\CnotprsupsetOp{\CMathML@notprsupsetOp}
\def\CMathML@notprsupset#1#2{#1\CMathML@notprsupsetOp{#2}}
\newcommand\Cnotprsupset[3][]{\CMathML@notprsupset{#2}{#3}}
```

On the semantic side (in LaTeXML), we need to implement them in terms of the MathML elements. Fortunately, we can just turn them around. [10]

EdN:10

```
\def\CMathML@UnionDAOp{\bigwedge}
\newcommand\CUnionDAOp{\CMathML@UnionDAOp}
\def\CMathML@UnionDA#1#2{\CMathML@UnionDAOp_{#1}{#2}} % set, scope
\newcommand\CUnionDA[3][]{\CMathML@UnionDA{#2}{#3}}
\def\CMathML@UnionCond#1#2#3{\CMathML@UnionDAOp_{#2}{#3}} % bvars,condition, scope
\newcommand\CUnionCond[4][]{\CMathML@UnionCond{#2}{#2}{#3}}
\def\CMathML@IntersectDAOp{\bigvee}
\newcommand\CIntersectDAOp{\CMathML@IntersectDAOp}
\def\CMathML@IntersectDA#1#2{\CMathML@IntersectDAOp_{#1}{#2}} % set, scope
\newcommand\CIntersectDa[3][]{\CMathML@IntersectDA{#2}{#3}}
\def\CMathML@IntersectCond#1#2#3{\CMathML@IntersectDAOp_{#2}{#3}}% bvars,condition, scope
\newcommand\CIntersectCond[4][]{\CMathML@IntersectCond{#2}{#3}{#4}}
\def\CMathML@CartesianproductDAOp{\bigoplus}
\newcommand\CCartesianproductDAOp{\CMathML@CartesianproductDAOp}
\def\CMathML@CartesianproductDA#1#2{\CMathML@CartesianproductDAOp_{#1}{#2}} % set, scope
```

[10] EdNote: ooooops, this does not work for the associative ones.

```
\newcommand\CCartesianproductDA[3][]{\CMathML@CartesianproductDA{#2}{#3}}
\def\CMathML@CartesianproductCond#1#2#3{\CMathML@CartesianproductDAOp_{#2}{#3}}% bvars,condition, scope
\newcommand\CCartesianproductCond[4][]{\CMathML@CartesianproductCond{#2}{#3}{#4}}
```

4.6 Sequences and Series

```
\def\CMathML@sumOp{\sum}
\newcommand\CsumOp{\CMathML@sumOp}
\def\CMathML@sumLimits#1#2#3#4{\CMathML@sumOp_{#1=#2}^{#3}#4}% bvar, llimit, ulimit, body
\newcommand\CsumLimits[5][]{\CMathML@sumLimits{#2}{#3}{#4}{#5}}
\def\CMathML@sumCond#1#2#3{\CMathML@sumOp_{#1\in{#2}}#3}   % bvar, condition, body
\newcommand\CsumCond[4][]{\CMathML@sumCond{#2}{#3}{#4}}
\def\CMathML@sumDA#1#2{\CMathML@sumOp_{#1}#2}            % set, body
\newcommand\CsumDA[3][]{\CMathML@sumDA{#2}{#3}}
```

EdN:11 [11] EdN:12 [12]

```
\def\CMathML@prodOp{\prod}
\newcommand\CprodOp{\CMathML@prodOp}
\def\CMathML@prodLimits#1#2#3#4{\CMathML@prodOp_{#1=#32^{#3}#4}}% bvar, llimit, ulimit, body
\newcommand\CprodLimits[5][]{\CMathML@prodLimits{#2}{#3}{#4}{#5}}
\def\CMathML@prodCond#1#2#3{\CMathML@prodOp_{#1\in{#2}}#3}   % bvar, condition, body
\newcommand\CprodCond[4][]{\CMathML@prodCond{#2}{#3}{#4}}
\def\CMathML@prodDA#1#2{\CMathML@prodOp_{#1}#2}            % set, body
\newcommand\CprodDA[3]{\CMathML@prodDA{#2}{#3}}
```

EdN:13 [13]

4.7 Elementary Classical Functions

```
\def\CMathML@sin#1{\sin(#1)}
\newcommand\Csin[2][]{\CMathML@sin{#2}}
\def\CMathML@cos#1{\cos(#1)}
\newcommand\Ccos[2][]{\CMathML@cos{#2}}
\def\CMathML@tan#1{\tan(#1)}
\newcommand\Ctan[2][]{\CMathML@tan{#2}}
\def\CMathML@sec#1{\sec(#1)}
\newcommand\Csec[2][]{\CMathML@sec{#2}}
\def\CMathML@csc#1{\csc(#1)}
\newcommand\Ccsc[2][]{\CMathML@csc{#2}}
\def\CMathML@cot#1{\cot(#1)}
\newcommand\Ccot[2][]{\CMathML@cot{#2}}
\def\CMathML@sinh#1{\sinh(#1)}
\newcommand\Csinh[2][]{\CMathML@sinh{#2}}
\def\CMathML@cosh#1{\cosh(#1)}
\newcommand\Ccosh[2][]{\CMathML@cosh{#2}}
\def\CMathML@tanh#1{\tanh(#1)}
\newcommand\Ctanh[2][]{\CMathML@tanh{#2}}
\def\CMathML@sech#1{\mbox{sech}(#1)}
\newcommand\Csech[2][]{\CMathML@sech{#2}}
\def\CMathML@csch#1{\mbox{csch}(#1)}
\newcommand\Ccsch[2][]{\CMathML@csch{#2}}
\def\CMathML@coth#1{\mbox{coth}(#1)}
\newcommand\Ccoth[2][]{\CMathML@coth{#2}}
\def\CMathML@arcsin#1{\arcsin(#1)}
\newcommand\Carcsin[2][]{\CMathML@arcsin{#2}}
\def\CMathML@arccos#1{\arccos(#1)}
\newcommand\Carccos[2][]{\CMathML@arccos{#2}}
\def\CMathML@arctan#1{\arctan(#1)}
```

[11] EdNote: complete the other cases
[12] EdNote: add a keyword argument to all newcommands
[13] EdNote: complete the other cases

```
\newcommand\Carctan[2][]{\CMathML@arctan{#2}}
\def\CMathML@arccosh#1{\mbox{arccosh}(#1)}
\newcommand\Carccosh[2][]{\CMathML@arccosh{#2}}
\def\CMathML@arccot#1{\mbox{arccot}(#1)}
\newcommand\Carccot[2][]{\CMathML@arccot{#2}}
\def\CMathML@arccoth#1{\mbox{arccoth}(#1)}
\newcommand\Carccoth[2][]{\CMathML@arccoth{#2}}
\def\CMathML@arccsc#1{\mbox{arccsc}(#1)}
\newcommand\Carccsc[2][]{\CMathML@arccsc{#2}}
\def\CMathML@arcsinh#1{\mbox{arcsinh}(#1)}
\newcommand\Carcsinh[2][]{\CMathML@arcsinh{#2}}
\def\CMathML@arctanh#1{\mbox{arctanh}(#1)}
\newcommand\Carctanh[2][]{\CMathML@arctanh{#2}}

\def\CMathML@exp#1{\exp(#1)}
\newcommand\Cexp[2][]{\CMathML@exp{#2}}
\def\CMathML@ln#1{\ln(#1)}
\newcommand\Cln[2][]{\CMathML@ln{#2}}
\def\CMathML@log#1#2{\log_{#1}(#2)}
\newcommand\Clog[3][]{\CMathML@log{#2}{#3}}
```

4.8 Statistics

```
\def\CMathML@mean#1{\mbox{mean}(#1)}
\newcommand\Cmean[2][]{\CMathML@mean{#2}}
\def\CMathML@sdev#1{\mbox{std}(#1)}
\newcommand\Csdev[2][]{\CMathML@sdev{#2}}
\def\CMathML@var#1{\mbox{var}(#1)}
\newcommand\Cvar[2][]{\CMathML@var{#2}}
\def\CMathML@median#1{\mbox{median}(#1)}
\newcommand\Cmedian[2][]{\CMathML@median{#2}}
\def\CMathML@mode#1{\mbox{mode}(#1)}
\newcommand\Cmode[2][]{\CMathML@mode{#2}}
\def\CMathML@moment#1#2{\langle{#2}^{#1}\rangle}% degree, momentabout, scope
\newcommand\Cmoment[3][]{\CMathML@moment{#2}{#3}}
\def\CMathML@momentA#1#2{\langle{#2}^{#1}\rangle}% degree, momentabout, scope
\newcommand\CmomentA[4][]{\CMathML@momentA{#2}{#3}{#4}}
```

EdN:14
EdN:15

4.9 Linear Algebra

```
\def\CMathML@vector#1{(#1)}
\newcommand\Cvector[2][]{\CMathML@vector{#2}}
\def\CMathML@matrix#1#2{\left(\begin{array}{#1}#2\end{array}\right)}% row pattern, body
\newcommand\Cmatrix[3][]{\CMathML@matrix{#2}{#3}}
\def\CMathML@determinant#1{\left|#1\right|}
\newcommand\Cdeterminant[2][]{\CMathML@determinant{#2}}
\def\CMathML@transpose#1{#1^\top}
\newcommand\Ctranspose[2][]{\CMathML@transpose{#2}}
\def\CMathML@selector#1#2{#1_{#2}}
\newcommand\Cselector[3][]{\CMathML@selector{#2}{#3}}
\def\CMathML@vectproductOp{\cdot}
\newcommand\CvectproductOp{\CMathML@vectproductOp}
\def\CMathML@vectproduct#1#2{#1\CMathML@vectproductOp{#2}}
\newcommand\Cvectproduct[3][]{\CMathML@vectproduct{#2}{#3}}
\def\CMathML@scalarproduct#1#2{{#1}#2}
\newcommand\Cscalarproduct[3][]{\CMathML@scalarproduct{#2}{#3}}
```

[14] EdNote: we do not seem to need the momentabout.
[15] EdNote: moment and momentA have funny elided arguments

```
\def\CMathML@outerproductOp{\times}
\newcommand\CouterproductOp{\CMathML@outerproductOp}
\def\CMathML@outerproduct#1#2{#1\CMathML@outerproductOp{#2}}
\newcommand\Couterproduct[3][]{\CMathML@outerproduct{#2}{#3}}
```

4.10 Constant and Symbol Elements

```
\def\CMathML@integers{{\mathbb{Z}}}
\newcommand\Cintegers[1][]{\CMathML@integers}
\def\CMathML@reals{{\mathbb{R}}}
\newcommand\Creals[1][]{\CMathML@reals}
\def\CMathML@rationals{{\mathbb{Q}}}
\newcommand\Crationals[1][]{\CMathML@rationals}
\def\CMathML@naturalnumbers{{\mathbb{N}}}
\newcommand\Cnaturalnumbers[1][]{\CMathML@naturalnumbers}
\def\CMathML@complexes{{\mathbb{C}}}
\newcommand\Ccomplexes[1][]{\CMathML@complexes}
\def\CMathML@primes{{\mathbb{P}}}
\newcommand\Cprimes[1][]{\CMathML@primes}
\def\CMathML@exponemtiale{e}
\newcommand\Cexponemtiale[1][]{\CMathML@exponemtiale}
\def\CMathML@imaginaryi{i}
\newcommand\Cimaginaryi[1][]{\CMathML@imaginaryi}
\def\CMathML@notanumber{{\mathrm{NaN}}}
\newcommand\Cnotanumber[1][]{\CMathML@notanumber}
\def\CMathML@true{{\mathrm{true}}}
\newcommand\Ctrue[1][]{\CMathML@true}
\def\CMathML@false{{\mathrm{false}}}
\newcommand\Cfalse[1][]{\CMathML@false}
\def\CMathML@emptyset{\emptyset}
\newcommand\Cemptyset[1][]{\CMathML@emptyset}
\def\CMathML@pi{\pi}
\newcommand\Cpi[1][]{\CMathML@pi}
\def\CMathML@eulergamma{\gamma}
\newcommand\Ceulergamma[1][]{\CMathML@eulergamma}
\def\CMathML@infinit{\infty}
\newcommand\Cinfinit[1][]{\CMathML@infinit}
⟨/sty⟩
```

4.11 Extensions

\Ccomplement

```
⟨*styx⟩
\def\CMathML@complement#1{#1^c}
\newcommand\Ccomplement[2][]{\CMathML@complement{#2}}
⟨/styx⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

*, 5, 8

argument
  order, 5
associative
  operator, 5

big
  operator, 5
binding
  operator, 5

LaTexML, 1, 3–5, 11, 13, 20

MathML, 1, 3–7, 9–13, 20

operator
  associative, 5
  big, 5
  binding, 5
order
  argument, 5

XML, 3

References

[sTeX] *KWARC/sTeX*. URL: `https://svn.kwarc.info/repos/stex` (visited on 05/15/2015).