`cmath.sty`: An Infrastructure for building Inline Content Math in sTeX*

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

August 24, 2017

Abstract

The **cmath** package is a central 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 supplies an infrastructure that allows to build content math expressions (strict content MathML or OpenMath objects) in the text. This is needed whenever the head symbols of expressions are variables and can thus not be treated via the **\symdef** mechanism in sTeX.

*Version v0.1 (last revised 2016/04/07)

Contents

1 Introduction

sTeX allows to build content math expressions via the `\symdef` mechanism [KGA16] if their heads are constants. For instance, if we have defined `\symdef{lt}[2]{#1<#2}` in the module `relation1`, then an invocation of `\lt3a` will be transformed to

```
<OMA>
  <OMS cd="relation1" name="lt"/>
  <OMI>3</OMI>
  <OMV name="a"/>
</OMA>
```

If the head of the expression (i.e. the function symbol in this case) is a variable, then we cannot resort to a `\symdef`, since that would define the functional equivalent of a logical constant. Sometimes, LaTeXML can figure out that when we write $f(a,b)$ that f is a function (especially, if we declare them to be via the `functions=` key in the dominating statement environment [Koh16]). But sometimes, we want to be explicit, especially for n-ary functions and in the presence of elided elements in argument sequences. A related problem is markup for complex variable names, such as x_{left} or ST^*.

The `cmath` package supplies the LaTeX bindings that allow us to achieve this.

2 The User Interface

2.1 Variable Names

In mathematics we often use complex variable names like x', g_n, f^1, $\widetilde{\phi}_i^j$ or even foo; for presentation-oriented LaTeX, this is not a problem, but if we want to generate content markup, we must show explicitly that those are complex identifiers (otherwise the variable name foo might be mistaken for the product $f \cdot o \cdot o$). In careful mathematical typesetting, `$sin$` is distinguished from `$\sin$`, but we cannot rely on this effect for variable names.

`\vname` `\vname` identifies a token sequence as a name, and allows the user to provide an ASCII (XML-compatible) identifier for it. The optional argument is the identifier, and the second one the LaTeX representation. The identifier can also be used `\vname` with `\vnref` for referencing. So, if we have used `\vnname[xi]{x_i}`, then we can later use `\vnref{xi}` as a short name for `\vname{x_i}`. Note that in output formats that are capable of generating structure sharing, `\vnref{xi}` would be EdN:1 represented as a cross-reference.[1]

Since indexed variable names make a significant special case of complex identifiers, we provides the macros `\livar` that allows to mark up variables with lower `\livar` indices. If `\livar` is given an optional first argument, this is taken as a name. `\livar` Thus `\livar[foo]{x}1` is "short" for `\vname[foo]{x_1}`. The macros `\livar`,

[1] EdNote: DG: Do we know whether using the same name in two vname invocations, would refer to two instances of the same variable? Presumably so, since the names are the same? We should make this explicit in the text. A different variable would e.g. have a name "xi2", but the same body

`\nappa{f}{a_1,a_2,a_3}`	$f(a_1,a_2,a_3)$
`\nappe{f}{a_1}{a_n}`	$f(a_1,\ldots,a_n)$
`\symdef{eph}[1]{e_{#1}^{\varphi(#1)}}` `\nappf{g}\eph14`	$g(e_1^{\varphi(1)},\ldots,e_4^{\varphi(4)})$
`\nappli{f}a1n`	$f(a_1,\ldots,a_n)$
`\nappui{f}a1n`	$f(a^1,\ldots,a^n)$

Figure 1: Application Macros

`\ulivar` `\primvar` `\pprimvar` serve the analogous purpose for variables with upper indices, and `\ulivar` for upper and lower indices. Finally, `\primvar` and `\pprimvar` do the same for variables with primes and double primes (triple primes are bad style).

2.2 Applications

`\nappa` `\nappf` To construct a content math application of the form $f(a_1,\ldots,a_n)$ with concrete arguments a_i (i.e. without elisions), then we can use the `\nappa` macro. If we have elisions in the arguments, then we have to interpret the arguments as a sequence of argument constructors applied to the respective positional indexes. We can mark up this situation with the `\nappf` macro: `\nappf{`⟨*fun*⟩`}{`⟨*const*⟩`}{`⟨*first*⟩`}{`⟨*last*⟩`}` where ⟨*const*⟩ is a macro for the constructor is presented as ⟨*fun*⟩(⟨*const*⟩⟨*first*⟩, ..., ⟨*const*⟩⟨*last*⟩); see Figure 1 for a concrete example, and Figure 1.[2]

EdN:2

`\nappe` For a simple elision in the arguments, we can use `\nappe{`⟨*fun*⟩`}{`⟨*first*⟩`}{`⟨*last*⟩`}` will be formatted as ⟨*fun*⟩(⟨*first*⟩, ..., ⟨*last*⟩). Note that this is quite un-semantic (we have to guess the sequence), so the use of `\nappe` is discouraged.

`\nappli` `\nappui` A solution to this situation is if we can think of the arguments as a finite sequence $a =: (a_i)_{l\leq i\leq h}$, then we can use `\nappli{`⟨*fun*⟩`}{`⟨*seq*⟩`}{`⟨*start*⟩`}{`⟨*end*⟩`}`, where ⟨*seq*⟩ is the sequence, and the remaining arguments are the start and end index. The works like `\nappli`, but uses upper indices in the presentation.

2.3 Binders

EdN:3

[3]

2.4 Sharing

We (currently) use the

[2] EdNote: MK@MK: we need a meta-cd `cmath` with the respective notation definition here. It is very frustrating that we cannot even really write down the axiomatization of flexary constants in OpenMath.

[3] EdNote: MK: document

```
\symdef{eph}[1]{e_{#1}^{\phi(#1)}}
\nappf{g}\eph14
```

currently generates

```
<OMA>
  <OMS cd="cmath" name="apply-from-to"/>
  <OMV name="g"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR><OMV name="x"/></OMBVAR>
    <OMA><OMS cd="???" name="eph"/><OMV name="x"/></OMA>
  </OMBIND>
  <OMI>1</OMI>
  <OMI>4</OMI>
</OMA>
```

Example 1: Application Macros

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 Package Options

The `cmath` package does not take options (at the moment), but we pass any we get to the `presentation` package.

```
⟨*package⟩
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{presentation}}
\ProcessOptions
```

The next measure is to ensure that some sTeX packages are loaded. For LaTeXML, we also initialize the package inclusions, there we do not need `ntheorem`, since the XML does not do the presentation.

```
\RequirePackage{presentation}
```

4.2 Variable Names

`\vname` a name macro; the first optional argument is an identifier $\langle id\rangle$, this is standard for LaTeX, but for LaTeXML, we want to generate attributes `xml:id="cvar.`$\langle id\rangle$`"`

and `name="`⟨*id*⟩`"`. However, if no id was given in we default them to `xml:id="cvar.`⟨*count*⟩`"` and `name="name.cvar.`⟨*count*⟩`"`.

```
\newcommand\vname[2][]{#2%
\def\@opt{#1}%
\ifx\@opt\@empty\else\expandafter\gdef\csname MOD@name@#1\endcsname{#2}\fi}
```

`\vnref`

```
\def\vnref#1{\csname MOD@name@#1\endcsname}
```

EdN:4 [4]

`\uivar` constructors for variables.

```
\newcommand\primvar[2][]{\vname[#1]{#2^\prime}}
\newcommand\pprimvar[2][]{\vname[#1]{#2^{\prime\prime}}}
\newcommand\uivar[3][]{\vname[#1]{{#2}^{#3}}}
\newcommand\livar[3][]{\vname[#1]{{#2}_{#3}}}
\newcommand\ulivar[4][]{\vname[#1]{{#2}^{#3}_{#4}}}
```

4.3 Applications

EdN:5 `\napp*` [5]

```
\newcommand\nappa[3][]{\prefix[#1]{#2}{#3}}
\newcommand\nappe[4][]{\nappa[#1]{#2}{#3,\ldots,#4}}
\newcommand\nappf[5][]{\nappe[#1]{#2}{#3{#4}}{#3{#5}}}
\newcommand\nappli[5][]{\nappe[#1]{#2}{#3_{#4}}{#3_{#5}}}
\newcommand\nappui[5][]{\nappe[#1]{#2}{#3^{#4}}{#3^{#5}}}
```

EdN:6 `\anapp*` [6]

```
\newcommand\anappa[3][]{\assoc[#1]{#2}{#3}}
\newcommand\anappe[4][]{\anappa[#1]{#2}{#3,\ldots,#4}}
\newcommand\anappf[5][]{\anappe[#1]{#2}{#3{#4}}{#3{#5}}}
\newcommand\anappli[5][]{\anappe[#1]{#2}{#3_{#4}}{#3_{#5}}}
\newcommand\anappui[5][]{\anappe[#1]{#2}{#3^{#4}}{#3^{#5}}}
```

4.4 Binders

4.5 Sharing

These macros are lifted from Bruce Miller's `latexml.sty`, we do not want the rest.

`\LXMID`

```
\def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
```

[4] EdNote: the following macros are just ideas, they need to be implemented and documented

[5] EdNote: document keyval args above and implement them in LaTeXML

[6] EdNote: document anapp* and implement in LaTeXML (i.e. get the presentation information into the OM/MathML).

\LXMRef

```
\def\LXMRef#1{\csname xmarg#1\endcsname}
⟨/package⟩
```

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.

References

[KGA16] Michael Kohlhase, Deyan Ginev, and Rares Ambrus. *`modules.sty`: Semantic Macros and Module Scoping in sTeX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2016. URL: `http://www.ctan.org/get/macros/latex/contrib/stex/modules/modules.pdf`.

[Koh16] Michael Kohlhase. *`omtext`: Semantic Markup for Mathematical Text Fragments in LATEX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2016. URL: `http://mirror.ctan.org/macros/latex/contrib/stex/sty/omtext/omtext.pdf`.

[sTeX] *KWARC/sTeX*. URL: `https://github.com/KWARC/sTeX` (visited on 05/15/2015).