\texttt{cmath.sty}: An Infrastructure for building Inline Content Math in \TeX\

Michael Kohlhase
FAU Erlangen-Nürnberg
http://kwarc.info/kohlhase

Deyan Ginev
Authorea

March 20, 2019

Abstract

The \texttt{cmath} package is a central part of the \TeX\ 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 \texttt{symdef} mechanism in \TeX.

\footnote{Version v0.1 (last revised 2019/03/20)}
1 Introduction

STEx allows to build content math expressions via the \symdef mechanism \cite{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

\[
\OMA \\
\OMS cd="relation1" name="lt"/> \\
\OMI>3</OMI> \\
\OMI name="a"/> \\
\OMA
\]

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 \cite{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 \(x_{\text{left}}\) or \(ST^*\).

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

2 The User Interface

2.1 Variable Names

In mathematics we often use complex variable names like \(x', g_n, f^1, \tilde{\phi}_i^j\) or even \(foo\); for presentation-oriented PDFX, 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.

\texttt{\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 with \texttt{\vname} for referencing. So, if we have used \vname{xi}{x_i}, then we can later use \vname{xi}{x_i} as a short name for \vname{x_i}. Note that in output formats that are capable of generating structure sharing, \vname{xi} would be represented as a cross-reference.\footnote{\texttt{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 “\texttt{xi2}”, but the same body.}}

Since indexed variable names make a significant special case of complex identifiers, we provides the macros \texttt{\livar} that allows to mark up variables with lower indices. If \texttt{\livar} is given an optional first argument, this is taken as a name.

\texttt{\livar} Thus \texttt{\livar[foo]}\{x\}1 is “short” for \vname{foo}{x_1}. The macros \texttt{\livar},...
\[ \text{appa}(a_1, a_2, a_3) \quad f(a_1, a_2, a_3) \]

\[ \text{appa}(a_1, a_2, \ldots, a_n) \quad f(a_1, \ldots, a_n) \]

\[ \text{symdef}(e\varphi)[1](e\varphi(#1)) \quad g(e^{\varphi(1)}, \ldots, e^{\varphi(4)}) \]

\[ \text{appf}(g)_{\text{eph}14} \]

\[ \text{appf}(a_1 \ldots a_n) \quad f(a_1, \ldots, a_n) \]

\[ \text{appui}(a_1 \ldots a_n) \quad f(a^1, \ldots, a^n) \]

Figure 1: Application Macros

\ulivar 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

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:

\[ \text{nappf}(\text{fun})\{\text{const}\}\{\text{first}\}\{\text{last}\} \]

where \( \langle \text{const} \rangle \) is a macro for the constructor is presented as \( \langle \text{fun} \rangle(\langle \text{const} \rangle(\langle \text{first} \rangle), \ldots, \langle \text{const} \rangle(\langle \text{last} \rangle)) \); see Figure 1 for a concrete example, and Figure 1.

\nappf For a simple elision in the arguments, we can use \nappa{\langle \text{fun} \rangle}{\langle \text{first} \rangle}{\langle \text{last} \rangle}

will be formatted as \( \langle \text{fun} \rangle(\langle \text{first} \rangle, \ldots, \langle \text{last} \rangle) \). Note that this is quite un-semantic (we have to guess the sequence), so the use of \nappa is discouraged.

A solution to this situation is if we can think of the arguments as a finite sequence \( a =: (a_i)_{i \leq h} \), then we can use \nappi{\langle \text{fun} \rangle}{\langle \text{seq} \rangle}{\langle \text{start} \rangle}{\langle \text{end} \rangle},

where \( \langle \text{seq} \rangle \) is the sequence, and the remaining arguments are the start and end index. The works like \nappi, but uses upper indices in the presentation.

### 2.3 Binders

### 2.4 Sharing

We (currently) use the

\text{EdNote: MK: document}
\SymbolDef{\text{eph}}[1]{e_{#1}^\text{\Phi(#1)}}
\NameDef{g}{\text{eph}14}

currently generates

\begin{OMA}
  \OMS{cd}{\text{cmath}}{name}{apply-from-to}/
  \OMV{name}{g}/
  \OMBIND
    \OMS{cd}{\text{fns1}}{name}{\lambda}/
    \OMBVAR\OMV{name}{x}/</OMBVAR
    \OMA \OMS{cd}{\?}\OMV{name}{\text{eph}}/\OMV{name}{x}/</OMA
  </OMBIND
  \OMI{1}/</OMI
  \OMI{4}/</OMI
\end{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 $\text{\LaTeX}$ GitHub repository [STeX].

1. none reported yet

4 The Implementation

4.1 Package Options

The $\text{cmath}$ package does not take options (at the moment), but we pass any we get to the presentation package.

1 \texttt{\langle\ast package\rangle}
2 \texttt{\DeclareOption{*}{\PassOptionsToPackage{\CurrentOption}{presentation}}}\texttt{\ProcessOptions}
3

The next measure is to ensure that some $\text{\LaTeX}$ packages are loaded. For LA-TeXML, we also initialize the package inclusions, there we do not need $\text{ntheorem}$, since the XML does not do the presentation.

4 \texttt{\RequirePackage{presentation}}

4.2 Variable Names

\NameDef{a name macro}{id}{the first optional argument is an identifier (id), this is standard for $\text{\LaTeX}$, but for LA-TeXML, we want to generate attributes $\text{xml:id=\text{cvar.(id)}}$}
and name="\langle \text{id} \rangle". However, if no id was given in we default them to \text{xml:id}="cvar.(\text{count})" and name="name.cvar.(\text{count})".

\begin{verbatim}
\newcommand\vnref[1]{\csname MOD@name@#1\endcsname}
\end{verbatim}

### 4.3 Applications

\begin{verbatim}
\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\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}}} 
\end{verbatim}

### 4.4 Binders

### 4.5 Sharing

These macros are lifted from Bruce Miller's \texttt{latexml.sty}, we do not want the rest.

\begin{verbatim}
\def\LXMID#1#2{\expandafter\gdef\csname xmarg#1\endcsname{#2}\csname xmarg#1\endcsname}
\end{verbatim}
\LXMRef

25 \def\LXMRef#1{\csname xmargin#1\endcsname}
26 (/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.

\LaTeX{}XML, 3

XML, 8
Change History

v0.2

General: First Version with Documentation, extracted variables stuff from

reinstating id macros from

References


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