The mathstyle package

Authors: Michael J. Downes, Morten Høgholm
Maintained by Morten Høgholm, Will Robertson
Feedback: https://github.com/wspr/breqn/issues

2020/02/18 0.98i

User's guide

This package exists for two reasons:

- The primitive operations for creating a super- or subscript in TEX work almost as if ^ and _ are macros taking an argument. However, that is not quite the case, and some things that you'd expect to work don't (e.g., ^\cong) whereas others which you'd think shouldn't work actually do (such as ^\mathsf{s}). We do everyone a favor if it behaves consistently, i.e., if the superscript and subscript operations act as if they are macros taking exactly one argument.
- Because the TEX math typesetting engine uses infix notation for fractions, one has to use \mathchoice or \mathpalette whenever trying to do anything requiring boxing or measuring math. This creates problems for loading fonts on demand as the font loading mechanism has to load fonts for all styles without even knowing if the font is going to be used. Getting the timing of \mathchoice right can be tricky as well. Since LATEX does not promote the primitive infix notation, this package keeps track of a current mathstyle parameter.

1 Some usage tips

If you want to use this package with amsmath, it is important mathstyle is loaded *after* amsmath.

The current mathstyle is stored in the variable \mathstyle. The command \currentmathstyle can be used to switch to the mode currently active. Below is shown how the macro \mathrlap from mathtools is implemented without knowing about the current mathstyle using \mathpalette.

```
\providecommand*\mathrlap[1][]{%
  \ifx\@empty#1\@empty
```

```
    \expandafter \mathpalette \expandafter \@mathrlap
  \else
    \expandafter \@mathrlap \expandafter #1%
  \fi}
\providecommand*\@mathrlap #1#2{{}\rlap{$\m@th#1{#2}$}}
```

The same definition using `\currentmathstyle` from this package.

```
\providecommand*\mathrlap[2][]{%
  #1 {}\rlap{$\m@th \currentmathstyle {#2}$}}
```

1.1 Package options

This package has one set of options affecting the `_` and `^` characters:

- `\usepackage[mathactivechars]{mathstyle}`

  This is the default behaviour. Here, `_` and `^` are made into harmless characters in text mode and behave as expected (for entering sub/superscript) when inside math mode. Certain code that assumes the catcodes of these characters may get confused about this; see below for a possible fix.

- `\usepackage[activechars]{mathstyle}`

  With this option, `_` and `^` are made into active characters for entering sub/superscript mode in all cases—therefore, in text mode they will produce a regular error ('Missing $ inserted') indicating they are being used out of place.

- `\usepackage[noactivechars]{mathstyle}`

  This is the option most like to solve any compatibility problems. Here, `_` and `^` retain their regular catcodes at all times and behave in their default fashion. **However**, certain other features of this package (such as `\currentmathstyle` inside a subscript) will then fail to work, so only use this option as a last resort.

Implementation

```
1 ⟨*package⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \RequirePackage{expl3}
4 \ProvidesExplPackage{mathstyle}{2020/02/18}{0.98i}{Tracking mathstyle implicitly}
5 \ExplSyntaxOff
```

`\@saveprimitive` A straight copy from breqn, see implementation details there. Of course, with a recent pdfTEX (v1.40+), one can just use `\primitive` to get the original. We will implement that some day.

```
\providecommand\@saveprimitive[2]{%
  \begingroup
  \edef\@tempa{\string#1}\edef\@tempb{\meaning#1}%
  \ifx\@tempa\@tempb \global\let#2#1%
  \else
    \edef\@tempb{\meaning#2}%
    \ifx\@tempa\@tempb
    \else \@saveprimitive@a#1#2%
    \fi
  \fi
  \endgroup
}
\providecommand\@saveprimitive@a[2]{%
  \begingroup
  \def\@tempb##1#1##2{\edef\@tempb{##2}\@car{}}%
  \@tempb\nullfont{select font nullfont}%
    \topmark{\string\topmark:}%
    \firstmark{\string\firstmark:}%
    \botmark{\string\botmark:}%
    \splitfirstmark{\string\splitfirstmark:}%
    \splitbotmark{\string\splitbotmark:}%
    #1{\string#1}%
    \@nil % for the \@car
  \edef\@tempa{\expandafter\strip@prefix\meaning\@tempb}%
  \edef\@tempb{\meaning#1}%
  \ifx\@tempa\@tempb \global\let#2#1%
  \else
    \PackageError{mathstyle}%
      {Unable to properly define \string#2; primitive
      \noexpand#1no longer primitive}\@eha
    \fi
  \fi
  \endgroup
}
```

`\everydisplay` We need to keep track of whether we're in inline or display maths, and the only way to do that is to add a switch inside `\everydisplay`. We act sensibly and preserve any of the previous contents of that token register before adding our own code here. As we'll see in a second, LuaTEX provides a native mechanism for this so we don't need any action in that case. (Various other parts of the code also need to have different paths for LuaTEX use.)

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \everydisplay=\expandafter{\the\everydisplay\chardef\mathstyle\z@}
\fi
```

`\mathstyle` A counter for the math style: 0–display, 2–text, 4–script, 6–scriptscript. The logic is that display maths will explicitly set `\mathstyle` to zero (see above), so by default it is set to the 'text' maths style. With LuaTEX there is a primitive to do

the same so it just has to be enabled. Note that in all cases we use LuaTEX-like numbering for the states.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \chardef\mathstyle\@ne
\else
  \directlua{tex.enableprimitives("", {"mathstyle"})}
\fi
```

Save the four style changing primitives, `\mathchoice` and the fraction commands.

```
\@saveprimitive\displaystyle\@@displaystyle
\@saveprimitive\textstyle\@@textstyle
\@saveprimitive\scriptstyle\@@scriptstyle
\@saveprimitive\scriptscriptstyle\@@scriptscriptstyle
\@saveprimitive\mathchoice\@@mathchoice
\@saveprimitive\over\@@over
\@saveprimitive\atop\@@atop
\@saveprimitive\above\@@above
\@saveprimitive\overwithdelims\@@overwithdelims
\@saveprimitive\atopwithdelims\@@atopwithdelims
\@saveprimitive\abovewithdelims\@@abovewithdelims
```

Then we redeclare the four style changing primitives: set the value of `\mathstyle` if LuaTEX is not in use.q

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \DeclareRobustCommand{\displaystyle}{%
    \@@displaystyle \chardef\mathstyle\z@}
  \DeclareRobustCommand{\textstyle}{%
    \@@textstyle \chardef\mathstyle\tw@}
  \DeclareRobustCommand{\scriptstyle}{%
    \@@scriptstyle \chardef\mathstyle4 }
  \DeclareRobustCommand{\scriptscriptstyle}{%
    \@@scriptscriptstyle \chardef\mathstyle6 }
\fi
```

First we get the primitive operations. These should have been control sequences in TEX just like operations for begin math, end math, begin display, end display.

```
\begingroup \catcode`\^=7\relax \catcode`\_=8\relax % just in case
\lowercase{\endgroup
\let\@@superscript=^ \let\@@subscript=_
}%
\begingroup \catcode`\^=12\relax \catcode`\_=12\relax % just in case
\lowercase{\endgroup
\let\@@superscript@other=^ \let\@@subscript@other=_
}%
```

If we enter a sub- or superscript the `\mathstyle` must be adjusted. Since all is happening in a group, we do not have to worry about resetting. We can't tell the

difference between cramped and non-cramped styles unless LuaTEX is in use, in which case this command is a no-op.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \def\subsupstyle{%
    \ifnum\mathstyle<5 \chardef\mathstyle4 %
    \else \chardef\mathstyle6 %
    \fi
  }
\else
  \def\subsupstyle{}
\fi
```

Provide commands with meaningful names for the two primitives, cf. `\mathrel`.

```
\let\mathsup=\@@superscript
\let\mathsub=\@@subscript
```

`\sb` and `\sp` are then defined as macros.

```
\def\sb#1{\mathsub{\protect\subsupstyle#1}}%
\def\sp#1{\mathsup{\protect\subsupstyle#1}}%
```

`\mathchoice` `\mathchoice` is now just a switch. Note that this redefinition does not allow the arbitrary ⟨*filler*⟩ of the TEX primitive. Very rarely used anyway.

```
\def\mathchoice{%
  \relax\ifcase\numexpr\mathstyle\relax
    \expandafter\@firstoffour % Display
  \or
    \expandafter\@firstoffour % Cramped display
  \or
    \expandafter\@secondoffour % Text
  \or
    \expandafter\@secondoffour % Cramped text
  \or
    \expandafter\@thirdoffour % Script
  \or
    \expandafter\@thirdoffour % Cramped script
  \else
    \expandafter\@fourthoffour % (Cramped) Scriptscript
  \fi
}
```

Helper macros.

```
\providecommand\@firstoffour[4]{#1}
\providecommand\@secondoffour[4]{#2}
\providecommand\@thirdoffour[4]{#3}
\providecommand\@fourthoffour[4]{#4}
```

`\genfrac` The fractions. Note that this uses the same names as in amsmath. Much the same except here they call `\fracstyle`.

```
\DeclareRobustCommand\genfrac[6]{%
```

```
  {#1\fracstyle
    {\begingroup #5\endgroup
      \csname @@\ifx\maxdimen#4\maxdimen over\else above\fi
        \if @#2@\else withdelims\fi\endcsname #2#3\relax
     #6}%
  }%
}
\renewcommand{\frac}{\genfrac{}{}{}{}}
\providecommand{\dfrac}{}
\providecommand{\tfrac}{}
\renewcommand{\dfrac}{\genfrac\displaystyle{}{}{}}
\renewcommand{\tfrac}{\genfrac\textstyle{}{}{}}
\providecommand{\binom}{}
\providecommand{\tbinom}{}
\providecommand{\dbinom}{}
\renewcommand{\binom}{\genfrac{}(){0pt}}
\renewcommand{\dbinom}{\genfrac\displaystyle(){0pt}}
\renewcommand{\tbinom}{\genfrac\textstyle(){0pt}}
```

The `\fracstyle` command is a switch to go one level down but no further than three.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
  \def\fracstyle{%
    \ifcase\numexpr\mathstyle\relax
      \chardef\mathstyle=\@ne
    \or
      \chardef\mathstyle=\@ne
    \or
      \chardef\mathstyle=\tw@
    \or
      \chardef\mathstyle=\tw@
    \else
      \chardef\mathstyle=\thr@@
    \fi
  }
\else
  \def\fracstyle{}
\fi
```

The `\currentmathstyle` checks the value of `\mathstyle` and switches to it so it is in essence the opposite of `\displaystyle` and friends.

```
\def\currentmathstyle{%
  \ifcase\numexpr\mathstyle\relax
    \@@displaystyle
  \or
    \@@displaystyle
  \or
    \@@textstyle
  \or
```

```
    \@@textstyle
  \or
    \@@scriptstyle
  \or
    \@@scriptstyle
  \else
    \@@scriptscriptstyle
  \fi}
```

Finally, we declare the package options.

```
\DeclareOption{mathactivechars}{%
 %  \catcode`\^=12\relax
 %  \catcode`\_=12\relax
\AtBeginDocument{\catcode`\^=12\relax \catcode`\_=12\relax}%
}
\DeclareOption{activechars}{%
 %  \catcode`\^=13\relax
 %  \catcode`\_=13\relax
\AtBeginDocument{\catcode`\^=13\relax \catcode`\_=13\relax}%
}
\DeclareOption{noactivechars}{%
 %  \catcode`\^=7\relax
 %  \catcode`\_=8\relax
\AtBeginDocument{\catcode`\^=7\relax \catcode`\_=8\relax}%
}
\ExecuteOptions{mathactivechars}
\ProcessOptions\relax
```

WSPR: Set up the active behaviours: (this is set even in the `noactivechars` case but they are never activated. no worries?)

```
\ifnum\catcode`\^=13\relax
  \let^=\sp \let_=\sb
\else
  \mathcode`\^="8000\relax
  \mathcode`\_="8000\relax
  \begingroup
    \catcode`\^=\active
    \catcode`\_=\active
    \global\let^=\sp
    \global\let_=\sb
  \endgroup
\fi
```

⟨/package⟩