The **mathstyle** package

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

2021/10/28 0.98l

### 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 ETeX 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
    1
  \else
    #1\end{\%
\}
```
The same definition using \currentmathstyle from this package.

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

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

\providecommand*{\mathrlap}{\usepackage[mathactivechars]{mathstyle}}

\providecommand*{\mathrlap}{\usepackage[activechars]{mathstyle}}

\providecommand*{\mathrlap}{\usepackage[noactivechars]{mathstyle}}
\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, Lua\TeX{} 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 Lua\TeX{} use.)

\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 Lua\TeX{} there is a primitive to do...
the same so it just has to be enabled. Note that in all cases we use \LaTeX-like numbering for the states.

\begin{verbatim}
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
\chardef\mathstyle=2\relax
\chardef\mathstyledenom=0\relax
\else
\directlua{tex.enableprimitives("", \{"mathstyle"\})}
\endverbatim

Save the four style changing primitives, \texttt{\mathchoice} and the fraction commands.

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

Then we redeclare the four style changing primitives: set the value of \texttt{\mathstyle} if Lua\TeX{} is not in use.

\begin{verbatim}
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
\DeclareRobustCommand{\displaystyle}{\@@displaystyle \chardef\mathstyle0\relax}
\DeclareRobustCommand{\textstyle}{\@@textstyle \chardef\mathstyle2\relax}
\DeclareRobustCommand{\scriptstyle}{\@@scriptstyle \chardef\mathstyle4\relax}
\DeclareRobustCommand{\scriptscriptstyle}{\@@scriptscriptstyle \chardef\mathstyle6\relax}
\fi
\end{verbatim}

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.

\begin{verbatim}
\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=_
\}%
\end{verbatim}

If we enter a sub- or superscript the \texttt{\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 \texttt{Lua\TeX} is in use, in which case this command is a no-op.

81 \begin{group}
82 \expandafter\expandafter\expandafter\endgroup
83 \expandafter\ifx\csname directlua\endcsname\relax
84 \def\subsupstyle{%
85 \ifnum\mathstyle<4\relax
86 \chardef\mathstyle=\numexpr4+\mathstyledenom\relax
87 \else
88 \chardef\mathstyle=\numexpr6+\mathstyledenom\relax
89 \fi
90 }
91 \else
92 \def\subsupstyle{}
93 \fi

Provide commands with meaningful names for the two primitives, cf. \texttt{\mathrel}.

93 \let\mathsup=\@@superscript
94 \let\mathsub=\@@subscript
\sb and \sp are then defined as macros.
95 \def\sb#1{\mathsub{\protect\subsupstyle#1}}%
96 \def\sp#1{\mathsup{\protect\subsupstyle#1}}%

\texttt{\mathchoice} is now just a switch. Note that this redefinition does not allow the arbitrary \texttt{\langle filler\rangle} of the \TeX primitive. Very rarely used anyway.

97 \def\mathchoice{%
98 \relax\ifcase\numexpr\mathstyle\relax
99 \expandafter\@firstoffour % Display
100 \or
101 \expandafter\@firstoffour % Cramped display
102 \or
103 \expandafter\@secondoffour % Text
104 \or
105 \expandafter\@secondoffour % Cramped text
106 \or
107 \expandafter\@thirdoffour % Script
108 \or
109 \expandafter\@thirdoffour % Cramped script
110 \else
111 \expandafter\@fourthoffour % (Cramped) Scriptscript
112 \fi
113 }

Helper macros.
114 \providecommand\@firstoffour[4]{#1}
115 \providecommand\@secondoffour[4]{#2}
116 \providecommand\@thirdoffour[4]{#3}
117 \providecommand\@fourthoffour[4]{#4}

\texttt{\textbackslash genfrac} The \texttt{amsmath} definition:
with arguments:

- left-delim
- right-delim
- line thickness (default if empty)
- mathstyle override
- numerator
- denominator

The fractions. Note that this uses the same names as in `amsmath`. Much the same except here they call \fracstyle.
The \fracstyle command is a switch to go one level down but no further than three.

\begin{group}
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname directlua\endcsname\relax
\def\fracstyle{\
\ifcase\numexpr\mathstyle\relax
\@@displaystyle 
\or
\chardef\mathstyle=1\relax % 1
\or
\chardef\mathstyle=2\relax % 2
\or
\chardef\mathstyle=3\relax % 3
\else
\chardef\mathstyle=3\relax % 4 or more
\fi}
\else
\def\fracstyle{}
\fi
\end{group}
\expandafter\ifx\csname currentmathstyle\endcsname\relax
\def\currentmathstyle{\
\ifcase\numexpr\mathstyle\relax
\@@displaystyle \or
\chardef\mathstyle=1\relax % 1
\or
\chardef\mathstyle=2\relax % 2
\or
\chardef\mathstyle=3\relax % 3
\else
\chardef\mathstyle=3\relax % 4 or more
\fi}
\else
\def\currentmathstyle{}
\fi

The \currentmathstyle checks the value of \mathstyle and switches to it so it is in essence the opposite of \displaystyle and friends.
Finally, we declare the package options.
\begin{verbatim}
\DeclareOption{mathactivechars}{%
  \catcode\^=12\relax
  \catcode\_=12\relax
  \AtBeginDocument{\catcode\^=12\relax \catcode\_=12\relax}%
}\end{verbatim}
\begin{verbatim}
\DeclareOption{activechars}{%
  \catcode\^=13\relax
  \catcode\_=13\relax
  \AtBeginDocument{\catcode\^=13\relax \catcode\_=13\relax}%
}\end{verbatim}
\begin{verbatim}
\DeclareOption{noactivechars}{%
  \catcode\^=7\relax
  \catcode\_=8\relax
  \AtBeginDocument{\catcode\^=7\relax \catcode\_=8\relax}%
}\end{verbatim}
\begin{verbatim}
\ExecuteOptions{mathactivechars}
\ProcessOptions\relax
\end{verbatim}
WSPR: Set up the active behaviours: (this is set even in the noactivechars case but they are never activated. no worries?)
\begin{verbatim}
\ifnum\catcode\^=13\relax
  \let^=\sp \let_=#1\sb
\else
  \mathcode\^="8000\relax
  \mathcode\_="8000\relax
  \begingroup
    \catcode\^=\active
    \catcode\_=\active
    \global\let^=\sp
    \global\let_=#1\sb
  \endgroup
\fi
\end{verbatim}