The `amstext` package

Frank Mittelbach Rainer Schöpf

Version v2.01, 2021/08/26

This file is maintained by the LaTeX Project team.
Bug reports can be opened (category `amslatex`) at
`https://latex-project.org/bugs/`.

1 Introduction

This style file implements the $\mathcal{AMS}$-TeX macro `\text` for use with the new font selection scheme. The `\text` macro is a sophisticated command which allows the user to insert "normal text" into math formulas without worrying about correct sizes in sub- or superscripts. It can also be used in ordinary text; there it produces an unbreakable unit similar to `\mbox`.

Here is an example demonstrating some of its features:

$$x^{2\times\text{size of }y}\leq z_{i_{\text{upper bound of the array}}}$$

This was produced by

```
\[
   x^{2\,\times\,\text{size of $y$}}
   \leq
   z_{i_{\text{upper bound of the array}}}
\]
```

Additionally this style file redefines an internal `plain.tex` macro called `\mathhexbox` so that commands like `\dag` or `\P` will change sizes if used in math subscripts.

Package information.

```
\NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
[1994/12/01]% LaTeX date must be December 1994 or later
\ProvidesPackage{amstext}[2021/08/26 v2.01 AMS text]
```

2 The implementation

We need a few tools from `amsgen.sty`.

```
\RequirePackage{amsgen}
```

\text Now we come to the \text macro which is used to place ordinary text inside of math formulas. If it is used outside math it will produce an unbreakable unit of text.

```
\DeclareRobustCommand{\text}{%
  \ifmmode\expandafter\text@\else\expandafter\mbox\fi}
```

At the present time (late 1994) the LaTeX internal function \nfss@text is used in \ref, in font commands like \textbf, and in a few text symbol definitions like \$ and \pounds. By equating \nfss@text to \text we give it the ability of \text to change sizes properly if used in a subscript.

```
\let\nfss@text\text
```

\text@ If \text is encountered inside math mode the macro \text@ is called. It has one mandatory argument, the text which should be produced. Since we do not know in which math style we are currently in we call \mathchoice to typeset our text in all four possible styles.

```
\def\text@#1{{\mathchoice
```

To save token space we call a macro \textdef@ which takes three arguments: the current math style, the corresponding size macro and the text to typeset possibly with some additional information for typesetting.

```
    {\textdef@\displaystyle\f@size{#1}}%
```

The other three cases are similar except for the \iffirstchoice switch which we set to false. This is done to prevent LaTeX macros like \ref or \index from writing their arguments more than once.

```
    {\textdef@\textstyle\f@size{\firstchoice@false #1}}%
    {\textdef@\textstyle\sf@size{\firstchoice@false #1}}%
    {\textdef@\textstyle \ssf@size{\firstchoice@false #1}}%
```

Here we need to check whether a math size-change occurred inside the argument of \text. If so, restore

```
    \check@mathfonts
    }%
  }
```

The macros \f@size, \sf@size and \ssf@size hold the sizes which should be used when we are loading a new font for use in \textfont, \scriptfont and \scriptscriptfont. There is some question whether we should use \tf@size or \f@size for the main size, but since the primary purpose of the \text macro is to switch back to text within a display, it seems that \f@size is the better choice. (Indeed it could be said that the \text actually provides two different functions: one for escaping out of math mode in a display to print some words, and the other for handling math objects that are named by a fragment of text, when \operatorname isn't the right choice. For the latter \tf@size might be more correct but for the former \f@size is clearly better.)

\textdef@ To typeset the argument of \text correctly we have to make several actions. We start by placing everything inside an \hbox. But this is not enough: we need one extra level of grouping. These extra braces are necessary because of the new font

selection scheme which might produce an `\aftergroup` to globally restore some font values after the current group. To prevent any damage by this mechanism we add the braces thereby bringing the token inserted by `\aftergroup` inside the `\hbox`.[1]

```
\def\textdef@#1#2#3{\hbox{{%
```

Since text typeset inside an `\hbox` always stays in the size of the text surrounding the formula we have to adjust this for script and scriptscript sizes. For any math formula inside this argument this will be achieved by setting `\everymath` to the first argument of `\textdef@` since this argument contains the math size in the current typeset case of `\mathchoice`. Since LaTeX also knows about `\parbox`es and the `minipage` environment it might be neccessary to adjust `\everydisplay` too but this has to be tested further.

```
\everymath{#1}%
```

The next line of code changes locally (i.e. inside the current `\hbox`) the value of `\f@size`. This macro holds the size for typesetting ordinary text (e.g. loading or selecting a new font via `\selectfont`). By changing it to a smaller value a following `\selectfont` will switch to the wanted size.

```
\let\f@size#2\selectfont
```

Now we simply call the third argument and close all open groups.

```
#3}}}
```

`\iffirstchoice@` Here is the switch that we use to decide if `\ref` etc. should print its warnings. The default is true since normally these warnings shouldn't be suppressed.

```
\newif\iffirstchoice@
\firstchoice@true
```

2.1 Re-definition of LaTeX macros to work with `\text`

If a counter-changing command occurs inside the argument of `\text`, we don't want the counter to be changed four times because `\stepcounter` and `\addtocounter` have global effect. So we add the `\iffirstchoice@` test to make the counter operations execute only once.

`\stepcounter` Use `\def` rather than `\renewcommand*` because the star-form (for non-`\long` definitions) doesn't work with the June 1994 release of LaTeX.

```
\def\stepcounter#1{%
  \iffirstchoice@
     \addtocounter{#1}\@ne
     \begingroup \let\@elt\@stpelt \csname cl@#1\endcsname \endgroup
  \fi
}
```

`\addtocounter`

```
\def\addtocounter#1#2{%
```

[1] The mechanism will not produce a second `\aftergroup`. For more details see the technical documentation for NFSS2.

```
\iffirstchoice@
\@ifundefined {c@#1}{\@nocounterr {#1}}%
  {\global \advance \csname c@#1\endcsname #2\relax}%
\fi}
```

For \ref, \pageref, and indeed anything else that issues a warning or error, \text will produce four copies of the warning/error message. To suppress the last three copies, we change \GenericInfo, \GenericWarning, \GenericError.

```
\let\m@gobble\@empty
\@xp\let\csname m@gobble4\endcsname\@gobblefour
\long\@xp\def\csname m@gobble6\endcsname#1#2#3#4#5#6{}

\toks@{%
  \csname m@gobble\iffirstchoice@\else 4\fi\endcsname
  \protect}
\edef\GenericInfo{\the\toks@
  \@xp\@nx\csname GenericInfo \endcsname}
%
\edef\GenericWarning{\the\toks@
  \@xp\@nx\csname GenericWarning \endcsname}
%
\toks@{%
  \csname m@gobble\iffirstchoice@\else 6\fi\endcsname
  \protect}
\edef\GenericError{\the\toks@
  \@xp\@nx\csname GenericError \endcsname}
```

At one time \label, \@wrindex and \@wrglossary were changed here too to use the \iffirstchoice@ test but it seems that was a mistake because those are non-immediate writes. Something like

```
\text{something \index{foo}}
```

within a math formula would therefore *lose the index term* if the surrounding context was not displaystyle. (Unlikely in practice, but not impossible.) [mjd,1994/12/09]

2.2 Applications of \text

\mathhexbox We start with a re-definition of the plain.tex macro \mathhexbox. (Although M. Spivak in $\mathcal{AMS}$-TEX uses the name \mathhexbox@ for this purpose, I [FMi] don't see any reason to use a new name since the new definition is superior, has the same syntax and is used for the same purpose.)

```
\begingroup \catcode`\"=12
\gdef\mathhexbox#1#2#3{\text{$\m@th\mathchar"#1#2#3$}}
\endgroup
```

This redefinition means that now symbols like §, ¶, †, . . . , which are defined via \mathhexbox in plain.tex or elsewhere now correctly change sizes if they are used in math mode.

The usual `\endinput` to ensure that random garbage at the end of the file doesn't get copied by `docstrip`.

```
\endinput
```