The amsopn package

Michael Downes

Version v2.04, 2022/04/08

1 Introduction

The amsopn package provides a command \DeclareMathOperator for defining new 'math operator names' similar to the standard function names \sin, \lim, \max, etc.

Standard file identification.

```
\NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
[1994/12/01]% LaTeX date must December 1994 or later
\ProvidesPackage{amsopn}[2022/04/08 v2.04 operator names]
```

What \nolimits@ does is keep a \limits typed by the user from having any effect. This is used for operatornames whose standard usage is never to have limits.

```
\def\nolimits@{\@ifnextchar\limits{\nolimits\@gobble}{\nolimits}}
```

In operator names, it is sometimes desired to have text-mode punctuation characters such as *-/:'. Because the body of an operator name is set in math mode, these few punctuation characters will not come out right (wrong symbol and/or wrong spacing). The purpose of \newmcodes@ is to make them act like their normal text versions.

Where practical, we use decimal numbers to cut down main mem usage (" not needed). Use \Umathcodedefnum with xetex and LuaTeX to avoid problems using \mathcode if - has already been defined using \Umathcode.

```
\begingroup \catcode`\"=12
\ifx\Umathcode\@undefined
\gdef\newmcodes@{\mathcode`\'39\mathcode`\*42\mathcode`\."613A%
```

Define \std@minus for \relbar use; otherwise there are problems with arrows constructed with relbar.

```
  \ifnum\mathcode`\-=45 \else
    \mathchardef\std@minus\mathcode`\-\relax
  \fi
  \mathcode`\-45\mathcode`\/47\mathcode`\:"603A\relax}
```

```
\else
\gdef\newmcodes@{\mathcode`\'39\mathcode`\*42\mathcode`\."613A%
  \ifnum\Umathcodenum`\-=45 \else
    \Umathcharnumdef\std@minus\Umathcodenum`\-\relax
  \fi
  \mathcode`\-45\mathcode`\/47\mathcode`\:"603A\relax}
\fi
\endgroup
```

The command `\operatorname` prints its argument as a 'math operator' like `\sin` or `\det`, with proper font and spacing.

```
\DeclareRobustCommand{\operatorname}{%
  \@ifstar{\qopname\newmcodes@ m}%
          {\qopname\newmcodes@ o}}%
```

In the interior of the `\mathop` we need a null object (we choose a zero kern for minimum waste of main mem) in order to guard against the case where `#3` is a single letter; TEX will seize it and center it on the math axis if there is nothing else inside the `\mathop` atom.

```
\DeclareRobustCommand{\qopname}[3]{%
  \mathop{#1\kern\z@\operator@font#3}%
  \csname n#2limits@\endcsname}
```

`\DeclareMathOperator` The command `\DeclareMathOperator` defines the first argument to be an operator name whose text is the second argument. The star form means that the operator name should take limits (like `\max` or `\lim`).

```
\newcommand{\DeclareMathOperator}{%
  \@ifstar{\@declmathop m}{\@declmathop o}}
```

In the basic set of operator names (below) we did not use `\DeclareRobustCommand` because of the hash table cost. But we use it here to minimize the chances of trouble, since we are producing a user-defined command.

```
\long\def\@declmathop#1#2#3{%
  \@ifdefinable{#2}{%
    \DeclareRobustCommand{#2}{\qopname\newmcodes@#1{#3}}}}
%
\@onlypreamble\DeclareMathOperator
\@onlypreamble\@declmathop

\protected\def\arccos{\qopname\relax o{arccos}}
\protected\def\arcsin{\qopname\relax o{arcsin}}
\protected\def\arctan{\qopname\relax o{arctan}}
\protected\def\arg{\qopname\relax o{arg}}
\protected\def\cos{\qopname\relax o{cos}}
\protected\def\cosh{\qopname\relax o{cosh}}
\protected\def\cot{\qopname\relax o{cot}}
\protected\def\coth{\qopname\relax o{coth}}
\protected\def\csc{\qopname\relax o{csc}}
\protected\def\deg{\qopname\relax o{deg}}
```

```
\protected\def\det{\qopname\relax m{det}}
\protected\def\dim{\qopname\relax o{dim}}
\protected\def\exp{\qopname\relax o{exp}}
\protected\def\gcd{\qopname\relax m{gcd}}
\protected\def\hom{\qopname\relax o{hom}}
\protected\def\inf{\qopname\relax m{inf}}
\protected\def\injlim{\qopname\relax m{inj\,lim}}
\protected\def\ker{\qopname\relax o{ker}}
\protected\def\lg{\qopname\relax o{lg}}
\protected\def\lim{\qopname\relax m{lim}}
\protected\def\liminf{\qopname\relax m{lim\,inf}}
\protected\def\limsup{\qopname\relax m{lim\,sup}}
\protected\def\ln{\qopname\relax o{ln}}
\protected\def\log{\qopname\relax o{log}}
\protected\def\max{\qopname\relax m{max}}
\protected\def\min{\qopname\relax m{min}}
\protected\def\Pr{\qopname\relax m{Pr}}
\protected\def\projlim{\qopname\relax m{proj\,lim}}
\protected\def\sec{\qopname\relax o{sec}}
\protected\def\sin{\qopname\relax o{sin}}
\protected\def\sinh{\qopname\relax o{sinh}}
\protected\def\sup{\qopname\relax m{sup}}
\protected\def\tan{\qopname\relax o{tan}}
\protected\def\tanh{\qopname\relax o{tanh}}
```

`\operator@font` `\operatorfont` This command is provided to allow the document styles to decide in which way math operators like 'max' or 'log' are typeset. The default is to set them in ⟨*math group*⟩ zero of the current math version. The original name was `\operator@font`, retained for compatibility; the second name was added to make it more accessible so that users can call this font for use in special constructs that are not ordinary operator names but conceptually related.

`\operator@font` is also declared by the LaTeX kernel (for at least 14 years), thus defining it here effectively means "resetting it" to the kernel value, which is counterproductive in situations where the user (or a class) has altered its definition and at a later point `amsopn` got added.

```
%\def\operator@font{\mathgroup\symoperators} % commented out in 2.03
\def\operatorfont{\operator@font}
```

For backwards compatibility we keep this old command name for the time being:

```
\def\operatornamewithlimits{\operatorname*}
```

These macros use `\mathpalette`s so that they will change size in script and scriptscript styles, though it's hard to imagine they will ever be used there (the arrows, particularly, look bad in subscript sizes). Notice that the use of `\ex@` means that the vertical spacing may not be optimal in script and scriptscript sizes. Unfortunately TeX provides no easy way to do math mode vertical spacing that varies with current math style like mu units.

```
\protected\def\varlim@#1#2{%
```

```
  \vtop{\m@th\ialign{##\cr
    \hfil$#1\operator@font lim$\hfil\cr
    \noalign{\nointerlineskip\kern1.5\ex@}#2\cr
    \noalign{\nointerlineskip\kern-\ex@}\cr}}%
}
\protected\def\varinjlim{%
  \mathop{\mathpalette\varlim@{\rightarrowfill@\textstyle}}\nmlimits@
}
\protected\def\varprojlim{%
  \mathop{\mathpalette\varlim@{\leftarrowfill@\textstyle}}\nmlimits@
}
\protected\def\varliminf{\mathop{\mathpalette\varliminf@{}}\nmlimits@}
\def\varliminf@#1{\@@underline{\vrule\@depth.2\ex@\@width\z@
   \hbox{$#1\m@th\operator@font lim$}}}
\protected\def\varlimsup{\mathop{\mathpalette\varlimsup@{}}\nmlimits@}
\def\varlimsup@#1{\@@overline{\hbox{$#1\m@th\operator@font lim$}}}
```

```
\let\nmlimits@\displaylimits
\DeclareOption{namelimits}{\let\nmlimits@\displaylimits}
\DeclareOption{nonamelimits}{\let\nmlimits@\nolimits}
\ProcessOptions\relax
```

If we don't load the `amsgen` package then the use of `\ex@` in `\varlim@` will lead to trouble.

```
\RequirePackage{amsgen}\relax
```

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

```
\endinput
```