The delimset Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

30 December 2018, v1.1

Abstract

delimset is a LaTeX 2ε package to typeset and declare sets of delimiters in math mode whose size can be adjusted conveniently.

Contents

1 Introduction

In ordinary TeX and LaTeX, delimiters in math and physics expressions (brackets such as parentheses, braces, but also absolute values, sets, pairings, bra-kets and quantum expectation values, commutators, ...) are typically coded by their respective symbols. For example:

`[(ax+b)x+c]` $\to [(ax+b)x+c]$

`\{1,2,3,\ldots\}` $\to \{1,2,3,\ldots\}$

`(v,w)` $\to (v,w)$

`\langle\psi\vert M\vert\psi\rangle` $\to \langle\psi\vert M\vert\psi\rangle$

`(x^2+px+q)\vert_{-p/2}` $\to (x^2+px+q)\vert_{-p/2}$

$\ldots$

In order to adjust the size of delimiters, the modifiers `\big`, `\Big`, `\bigg`, `\Bigg` are used. More importantly the construct `\left*` ... [`\middle*`] ... `\right*` adjusts the size of delimiters to the contents:

$$\Bigg[\bigg[\Big[\big[[*]\big]\Big]\bigg]\Bigg], \qquad \left\{\frac{p}{q}\middle|p,q\in\mathbb{Z},q\neq 0\right\}$$

These modifiers allow to construct expressions involving delimiters conveniently and with a large amount of flexibility.

However, once an expression has been typeset in this way, it takes some efforts to modify, if needed. At least, one has to take care of both delimiters at the same time and change their size or type accordingly. For example, $\hat{X}$ in the expression `(A\hat{X}+B)` has an extended height that calls for bigger delimiters. Ideally, one would use `\left(...\right)` to find the correct size. Unfortunately, in this case, the smallest size of delimiters that covers the height of $\hat{X}$ is equivalent to `\Big`, which feels somewhat too big. For aesthetic reasons one might thus prefer the size `\big(...\big)`:

`(A\hat{X}+B)` $\to (A\hat{X}+B)$

`\left(A\hat{X}+B\right)` $\to \left(A\hat{X}+B\right)$

`\Big(A\hat{X}+B\Big)` $\to \Bigl(A\hat{X}+B\Bigr)$

`\big(A\hat{X}+B\big)` $\to \bigl(A\hat{X}+B\bigr)$

The package delimset provides mechanisms to declare sets of delimiters whose size can be adjusted conveniently by simple optional parameters. For example, it provides a general purpose bracket `\brk` which can be used as follows:

`\brk{A\hat{X}+B}` $\to (A\hat{X}+B)$

`\brk*{A\hat{X}+B}` $\to \left(A\hat{X}+B\right)$

`\brk2{A\hat{X}+B}` $\to \Bigl(A\hat{X}+B\Bigr)$

`\brk!{A\hat{X}+B}` $\to \bigl(A\hat{X}+B\bigr)$

It also allows to change the type of bracket conveniently:

`\brk{ax+b}` $\to (ax+b)$

`\brk[s]{ax+b}` $\to [ax+b]$

`\brk[c]{ax+b}` $\to \{ax+b\}$

`\brk[a]{ax+b}` $\to \langle ax+b\rangle$

These features can be combined and used, e.g., in nested brackets in order to distinguish the levels by shape and size:

`\brk[s]!{\brk{ax+b}x+c}` $\to \bigl[(ax+b)x+c\bigr]$

All of this can of course be achieved with the conventional tools of TEX with comparable effort, but the more complicated and nested the expressions get, the more difficult it will be to adjust them to obtain a visually acceptable result.

The main functionality of the package is based on a versatile mechanism to specify sets of delimiters, e.g.:

`\delim<>{ax+b}` → $\langle ax+b\rangle$

`\delimpair{[}{[.],}{[}!{a}{b}` → $\bigl[a, b\bigr[$

`\delimtriple<\vert\vert>*{\psi}{A^\dagger}{\psi}` → $\bigl\langle\psi\bigm|A^\dagger\bigm|\psi\bigr\rangle$

The command `\delim` can be used on the fly, but also in definitions of custom delimiter sets such as:

```
\newcommand{\comm}{\delimpair{[}{[.],}{]}}
```

Note that the definition of `\comm` does not specify any arguments. They are nevertheless read by the incomplete definition of `\delimpair` from the what follows `\comm`. In particular, this incomplete definition enables the correct parsing of the optional size specifier, e.g.:

`\comm!{P}{\psi(x)}` → $\bigl[P, \psi(x)\bigr]$

The package also provides a mechanism to declare delimited sets more flexibly. The above definition could be written as follows:

```
\DeclareMathDelimiterSet{\comm}[2]
    {\selectdelim[l]{[}{#1},{#2}\selectdelim[r]{]}}
```

1.1 Delimiter Sizes for Math Styles

In plain LATEX 2ε the size modifiers `\big`, `\Big`, `\bigg`, `\Bigg` have the shortcoming that they are based on a fixed font size of 10pt. More precisely, they use a vertical phantom of height 8.5pt, 11.5pt, 14.5pt or 17.5pt, respectively to set the height of the delimiter. The package amsmath corrects for font size by instead placing a (centred) vertical phantom of height 1.2, 1.8, 2.4 or 3 times the size of the currently selected math font (height plus depth of `\Mathstrutbox@`).

Unfortunately, it does not account for the currently selected math style. Therefore, the size of delimiters in sub/superscripts cannot be adjusted appropriately,[1] they come out way too big:

`e^{\big(ax+b\big)}` → $e^{\big(ax+b\big)}$

This package modifies the definitions of the size modifiers (of amsmath) to automatically adjust to sub/superscripts (subject to availability):

`e^{\big(ax+b\big)}` → $e^{\bigl(ax+b\bigr)}$

1.2 Spacing and Math Classes

Another shortcoming of the variable-size delimiters is that the spacing is noticeably different from their fixed-size counterparts:

`\square(ax+b)\square` → $\square(ax+b)\square$

`\square\left(ax+b\right)\square` → $\square\left(ax+b\right)\square$

`\square\bigl(ax+b\bigr)\square` → $\square\bigl(ax+b\bigr)\square$

`\square\left(ax+\big.b\right)\square` → $\square\left(ax+\big.b\right)\square$

---

[1] Arguably, it is bad typesetting practice to have too complicated expressions in sub/superscripts.

Often the construct `\left*` ... `\right*` leaves a large amount of space around it. A suitable way to fix this problem is to adjust the math class as follows:

`\square(ax+b)\square` → $\square(ax+b)\square$

`\square\mathopen{}\mathclose{\left(ax+b\right)}\square` → $\square\mathopen{}\mathclose{\left(ax+b\right)}\square$

This makes the expression look like `\mathopen` from the left and like `\mathclose` from the right. Importantly, the delimited expression should be contained in the `\mathclose` block so as to place any following sub/superscripts at the appropriate height. Unfortunately, the fix is too elaborate in comparison to the benefits of appropriate spacing. For practical purposes, consistent spacing can only be achieved by a more convenient mechanism.

On a related note, it is important to correctly specify the math class (`\mathopen`/`\mathclose`) for the delimiters, for example:

`\big(-1\big)` → $\big(-1\big)$

`\bigl(-1\bigr)` → $\bigl(-1\bigr)$

The math class can also make a major difference for intermediate delimiters, e.g.:

`\bigl\langle\psi\big\vert\psi\bigr\rangle` → $\bigl\langle\psi\big\vert\psi\bigr\rangle$

`\bigl\langle\psi\bigm\vert\psi\bigr\rangle` → $\bigl\langle\psi\bigm\vert\psi\bigr\rangle$

`\bigl\langle\psi\mathbin\big\vert\psi\bigr\rangle` → $\bigl\langle\psi\mathbin\big\vert\psi\bigr\rangle$

`\bigl\langle\psi\mathpunct\big\vert\psi\bigr\rangle` → $\bigl\langle\psi\mathpunct\big\vert\psi\bigr\rangle$

Depending on the particular situation, any of these expressions may be the most appropriate representation.

The package delimset automatically takes care of the math classes of the left and right delimiters. It also offers several choices for intermediate delimiters to take the context into account.

1.3 Philosophy

Semantic typesetting is one of the philosophies behind LaTeX: The author should focus on the content while the layout is taken care of by the engine. The (body of a) source file largely codes the contents of the document, while the layout is largely specified by the kernel, classes, styles and macro definitions (in the preamble). In order for the separation of content and layout to work well, the meaning of the content must be accurately specified by the source file so that the appropriate layout can be applied to it. For example, a left bracket '`(`' can have many meanings, which the engine could not possibly guess. Even a simple compound expression such as `[A,B]` could have different meanings depending on context such as a compact interval or a commutator. A semantic coding of the latter two concepts such as `\intv{A}{B}` vs. `\comm{A}{B}` clearly distinguishes between them. This allows the typesetting engine to represent them appropriately in every situation. It also allows to consistently define or adjust the typeset representation globally according to one's taste, such as $[A,B]$ vs. $[A;B]$. The price to pay is a larger number of abstract commands (which possibly evaluate to the same expression) and using them to specify the semantics throughout the source file (or at least where practical and useful). Conversely, the price to pay for an immediate typesetting scheme is that all notations need to be fixed at the start, and later adjustments require an elaborate search and replacement of (somewhat ambiguous) patterns like `[`x`,`y`]`.

Another distinction between TeX and LaTeX is that the former frequently uses free-format expressions such as `{x\over y}` whereas the latter normally uses structured commands with arguments such as `\frac{x}{y}`. In that sense, the construct `\left(ax+b\right)` belongs to the world of TeX, whereas an expression like `\delim()*{ax+b}` fits the LaTeX framework better.

1.4 Related CTAN Packages

There are at least three other packages which offer a similar functionality:

- The package delim supplies a command `\delimdef` to declare a set of delimiters which is similar to the present `\DeclareMathDelimiterSet`. The size of delimiters to be used in each case is then specified by a prefix command such as `\mbig` or `\mauto`.
- The package mathtools supplies commands `\DeclarePairedDelimiter...` (among many other things) which are similar to the present `\DeclareMathDelimiterSet`. The size of delimiters to be used in each case is then specified by an optional argument such as '`*`' or `[\big]`.
- The package delimseasy defines a collection of useful delimiters such as `\prn` for round parentheses or `\sqpr` square parentheses. Modifier letters can be prepended and appended to adjust their size.

A functionality of the present package not offered by any of the above packages is to typeset delimiters on the fly, e.g.:

`\delim<\vert>!{\psi}{\psi}` $\to \langle\psi|\psi\rangle$

The mechanism to specify the size is leaner in the sense that it uses only a single character and a single command.

2 Usage

To use the package delimset add the command

`\usepackage{delimset}`

to the preamble of your LaTeX document. If not yet present, the package amsmath will be loaded automatically.

2.1 Inline Usage

The package provides three general purpose commands to compose delimiter sets with one, two or three encapsulated expressions:

`\delim{`*l*`}{`*r*`}`*size*`{`*expr*`}`
`\delimpair{`*l*`}{`*m*`}{`*r*`}`*size*`{`*expr1*`}{`*expr2*`}`
`\delimtriple{`*l*`}{`*m1*`}{`*m2*`}{`*r*`}`*size*`{`*expr1*`}{`*expr2*`}{`*expr3*`}`

The expression(s) *expr(n)* will be surrounded by the delimiters *l* and *r* and, in the case of more than one expression, they will be separated by the delimiters *m(n)*:

l expr r, *l expr1 m expr2 r*, *l expr1 m1 expr2 m2 expr3 r*

Here, *l* and *r* have to be math delimiters (symbols which can be used for `\left` and `\right`) or the dot '.' for the null delimiter. The separators *m(n)* can be plain math delimiters as well, but they can also be compound expression of the form `[`*class*`]`*delim* where *class* specifies the intended math class of the delimiter *delim*:

delim
`{`*delim*`}`
`{[]`*delim*`}`
	class `\mathord` (similar to `\big`)

`{[l]`*delim*`}`	class `\mathopen` (similar to `\bigl`, not for use in `\delim...`)
`{[r]`*delim*`}`	class `\mathclose` (similar to `\bigr`, not for use in `\delim...`)
`{[o]`*delim*`}`	class `\mathopen` (similar to `\bigl`)
`{[c]`*delim*`}`	class `\mathclose` (similar to `\bigr`)
`{[m]`*delim*`}`	class `\mathrel` (similar to `\bigm`)
`{[p]`*delim*`}`	class `\mathpunct` (similar to `\bigp`)
`{[b]`*delim*`}`	class `\mathbin` (similar to `\bigb`)
`{[.]`*expr*`}`	no size adjustment, *expr* can be any expression

Note that the left and right delimiters l and r effectively have `[l]` and `[r]` prepended, respectively, and therefore they must be a plain math delimiter. Conversely, the immediate delimiters *m(n)* must not have the `[l]` and `[r]` classes because the latter must appear precisely once and they are already taken by l and r; use classes `[o]` and `[c]` instead.

The optional size modifier *size* should take one of the following values:

empty, *anything else*, `0`	default size
`!`, `+`, `1`	size `\big` (`!` is preferred)
`2`	size `\Big`
`3`	size `\bigg`
`4`	size `\Bigg`
`*`	variable size `\left...\right`

2.2 Declarations

The above constructs can be used to define new delimiter commands via:

`\newcommand{\`*name*`}{\delim...{`*l*`}...{`*r*`}}`

Here it makes sense to drop all arguments starting with the size modifier *size* from the definition. The TeX parsing mechanism will then automatically use the tokens following `\`*name* including the optional size modifier. If the encapsulated expression(s) are to be passed as explicit arguments to `\`*name*, one will have to find an alternative way to pass the optional size argument.

The above declarations via `\delim...` should be sufficient for almost all situations. However, there is an even more flexible way to declare delimiter sets:

`\DeclareMathDelimiterSet{\`*name*`}[`*narg*`]{`*expr*`}`

The syntax of this command is equivalent to the one of `\newcommand`. The difference is that the command `\`*name* first looks for the optional size modifier *size* as described above in section 2.1. It remembers the desired size for evaluating the macro *expr*. Then it parses the arguments as if the command was declared by `\newcommand`.

As usual, the macro *expr* contains the command arguments specified by `#1`, `#2`, ... Note that these should be encapsulated in groups `{#1}`, `{#2}`, ..., in order to prevent them from overwriting definitions at the level of the current group. It should also contain the desired math delimiters specified by:

`\selectdelim[`*class*`]{`*delim*`}`

Here, the sizes are adjusted automatically according to the previously specified modifier *size*. The math classes must be in proper sequence, i.e. the first and last ones must be `l` and `r`, respectively, while the intermediate ones can be anything by `l` and `r`, see section 2.1.

2.3 Default Declarations

The package predefines four commonly used sets of delimiters:

- `\brk[`*type*`]`*size*`{`*expr*`}` represents a standard bracket around a single expression *expr*. The type of bracket can be specified by the optional argument *type*:

  empty or `[r]`: round (x), `[s]`: square $[x]$, `[c]`: curly $\{x\}$, `[a]`: angle $\langle x\rangle$

- `\eval[`*type*`]`*size*`{`*expr*`}` represents evaluation of a functional expression *expr*. The type of bracket can be specified by the optional argument *type*:

  empty or `[v]`: $f(x)|_a$, `[s]`: $[f(x)]_a^b$

- `\abs`*size*`{`*expr*`}` represents the absolute value $|\mathit{expr}|$.
- `\norm`*size*`{`*expr*`}` represents the norm $\|\mathit{expr}\|$.

The above definitions can be suppressed by setting the package option `stddef` to `false`, see section 2.5. The package also defines some extended sets of delimiters as follows:

- `\pair`*size*`{`*expr1*`}{`*expr2*`}` represents a pair(ing) $(\mathit{expr1}, \mathit{expr2})$.
- `\set`*size*`{`*expr*`}` represents the set $\{\mathit{expr}\}$.
- `\setcond`*size*`{`*expr*`}{`*cond*`}` represents a set with condition $\{\mathit{expr}|\mathit{cond}\}$.
- `\intv[`*type*`]`*size*`{`*expr1*`}{`*expr2*`}` represents an interval from *expr1* to *expr2*. The in/exclusion of the bounds can be specified by the optional argument *type*:

  empty or `[c]`: closed $[a,b]$, `[o]`: open $]a,b[$, `[l]`: left-open $]a,b]$
  `[r]`: right-open $[a,b[$

- `\avg`*size*`{`*expr*`}` represents some average $\langle\mathit{expr}\rangle$.
- `\corr`*size*`{`*expr*`}` represents some correlator $\langle\mathit{expr}\rangle$.
- `\comm`*size*`{`*expr1*`}{`*expr2*`}` represents the commutator $[\mathit{expr1}, \mathit{expr2}]$.
- `\acomm`*size*`{`*expr1*`}{`*expr2*`}` represents the anti-commutator $\{\mathit{expr1}, \mathit{expr2}\}$.
- `\bra`*size*`{`*expr*`}` represents a bra-vector $\langle\mathit{expr}|$ in quantum mechanics.
- `\ket`*size*`{`*expr*`}` represents a ket-vector $|\mathit{expr}\rangle$ in quantum mechanics.
- `\setcond`*size*`{`*expr1*`}{`*expr2*`}` represents a bra-ket contraction $\langle\mathit{expr1}|\mathit{expr2}\rangle$.

The extended definitions need to be activated by the package option `extdef`, see section 2.5.

If the representations of the above delimiters do not suit the purpose or taste of the user, they can be redefined with `\renewcommand`.

2.4 Auxiliary Commands

In addition to the `\bigl`, `\bigr` and `\bigm` commands (as well as their `\Big.`, `\bigg.` and `\Bigg.` counterparts), the package defines two additional sets `\bigp` and `\bigb` (and counterparts). Here `\bigp` implies the math class `\mathpunct` and `\bigb` the class `\mathbin`.

Furthermore, the package overloads the size calculation in the `\big...` commands to properly account for the math styles in sub/superscripts (`\scriptstyle`) and nested sub/superscripts (`\scriptscriptstyle`). The latter behaviour can be controlled by the package option `scriptstyle`, see section 2.5.

2.5 Package Options

Options can be passed to the package by

`\usepackage[`*opts*`]{delimset}` or `\PassOptionsToPackage{`*opts*`}{delimset}`

Here *opts* is a comma-separated list of the available options:

- `stddef`[`=true`|`false`] controls the activation of standard delimiter definitions specified in section 2.3. If no value is given `true` is assumed; initially set to `true`.
- `extdef`[`=true`|`false`] controls the activation of extended delimiter definitions specified in section 2.3. If no value is given `true` is assumed; initially set to `false`.
- `scriptstyle`[`=true`|`false`] controls the overwriting of size modifiers explained in section 2.4. If no value is given `true` is assumed; initially set to `true`.

3 Information

3.1 Copyright

The Current Maintainer of this work is Niklas Beisert.

This work consists of the files `README.txt`, `delimset.ins` and `delimset.dtx` as well as the derived files `delimset.sty`, `dlmssamp.tex` and `delimset.pdf`.

3.2 Files and Installation

The package consists of the files

`README.txt`	readme file
`delimset.ins`	installation file
`delimset.dtx`	source file
`delimset.sty`	package file
`dlmssamp.tex`	sample file
`delimset.pdf`	manual

The distribution consists of the files `README.txt`, `delimset.ins` and `delimset.dtx`.

- Run (pdf)LaTeX on `delimset.dtx` to compile the manual `delimset.pdf` (this file).
- Run LaTeX on `delimset.ins` to create the package `delimset.sty` and the sample `dlmssamp.tex`. Copy the file `delimset.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/delimset`.

3.3 Interaction with CTAN Packages

The package relies on other packages:

- This package relies on some functionality of the package amsmath by using and overwriting some native code. Compatibility with the amsmath package has been tested with v2.15d (2016/06/28).
- This package uses the package keyval from the graphics bundle to process optional arguments to the package options. Compatibility with the keyval package has been tested with v1.15 (2014/10/28).

3.4 Revision History

v1.1: 2018/12/30

- classes added, class and size selection mechanism simplified

v1.01: 2018/01/17

- manual rearranged

v1.0: 2016/11/01

- extended standard definitions
- manual and installation package added
- first version published on CTAN

v0.5–0.7: 2016/05/08 – 2016/09/04

- basic functionality
- standard definitions

A Sample File

In this section we provide a LaTeX example how to use some of the delimset features.

Preamble and beginning of document body:

```
\documentclass[12pt]{article}

\usepackage[margin=2cm]{geometry}
\usepackage{amsmath,amsfonts}
```

```
\usepackage{delimset}

\begin{document}
```

sizes for default brackets:

```
\[
\brk0{x},\quad
\brk1{x},\quad
\brk2{x},\quad
\brk3{x},\quad
\brk4{x}
\]
```

styles for default brackets:

```
\[
\brk[r]{x},\quad
\brk[s]{x},\quad
\brk[c]{x},\quad
\brk[a]{x}
\]
```

nested brackets:

```
\[
\brk[c]2{\brk[s]!{\brk{ax+b}x+c}x+d}
\]
```

default absolute value, norm and default evaluations:

```
\[
\abs*{\frac{ax+b}{cx+d}},\qquad
\norm*{\frac{ax+b}{cx+d}},\qquad
\eval*{\frac{ax+b}{cx+d}}_{x=0},\qquad
\eval[s]*{\frac{ax+b}{cx+d}}_{x=0}^{x=\infty}
\]
```

outer delimiter spacing:

```
\begin{align*}
&\square\brk0{x}\square,&&\square\brk1{A^k}\square,
\\
&\square\brk*{x}\square,&&\square\brk*{A^k}\square
\end{align*}
```

delimiter sizes in exponents:

```
\[
e^{\brk{ax+b}},\qquad
e^{\brk!{ax+b}}
\]
```

delimiter declaration:

```
\DeclareMathDelimiterSet{\braket}[2]
  {\selectdelim[l]<#1\selectdelim|#2\selectdelim[r]>}
\[
\braket!{\psi}{\psi},
\quad
\braket*{\psi}{\psi\big.}
\]
```

delimiter usage:

```
\[
\delimpair<|>!{\psi}{\psi}
\]
```

conditional set, alternative layouts:

```
\[
\delimpair\{{[m]|}\}!{2n}{n\in\mathbb{Z}},
\qquad
\delimpair\{{[b]|}\}!{2n}{n\in\mathbb{Z}},
\qquad
\delimpair\{{[p]|}\}!{2n}{n\in\mathbb{Z}},
\qquad
\delimpair\{|\}!{2n}{n\in\mathbb{Z}},
\qquad
\delimpair\{{[.];}\}!{2n}{n\in\mathbb{Z}}
\]
conditional set, alternative layouts with variable size:
\[
\delimpair\{{[m]|}\}*{2n}{n\in\mathbb{Z}\big.},
\qquad
\delimpair\{{[b]|}\}*{2n}{n\in\mathbb{Z}\big.},
\qquad
\delimpair\{{[p]|}\}*{2n}{n\in\mathbb{Z}\big.},
\qquad
\delimpair\{|\}*{2n}{n\in\mathbb{Z}\big.},
\qquad
\delimpair\{{[.];}\}*{2n}{n\in\mathbb{Z}\big.}
\]
```

delimiter definition:

```
\newcommand{\comm}{\delimpair[{[.],}]}
\[
\comm!{\comm{A}{B}}{C}
+\comm!{\comm{B}{C}}{A}
+\comm!{\comm{C}{A}}{B}
=0
\]
```

alternative representation:

```
\renewcommand{\comm}{\delimpair[{[.];}]}
\[
\comm!{\comm{A}{B}}{C}
+\comm!{\comm{B}{C}}{A}
+\comm!{\comm{C}{A}}{B}
=0
\]
```

end of document body:

```
\end{document}
```

B Implementation

In this section we describe the package `delimset.sty`.

Required Packages. The package loads the packages amsmath and keyval if not yet present. amsmath is used for basic delimiter size functionality. keyval is used for extended options processing.

```
\RequirePackage{amsmath}
\RequirePackage{keyval}
```

Package Options. The package has some boolean keyval options which can be set to `true` or `false`.

```
\newif\ifdlm@std\dlm@stdtrue
\newif\ifdlm@ext\dlm@extfalse
\newif\ifdlm@script\dlm@scripttrue

\def\dlm@group{dlm@}
\define@key{\dlm@group}{stddef}[true]{\csname dlm@std#1\endcsname}
\define@key{\dlm@group}{extdef}[true]{\csname dlm@ext#1\endcsname}
\define@key{\dlm@group}{scriptstyle}[true]{\csname dlm@script#1\endcsname}

\DeclareOption*{\expandafter\setkeys\expandafter\dlm@group%
 \expandafter{\CurrentOption}}
\ProcessOptions
```

Internal Definitions. Overwrite the amsmath command `\bBigg@` to select the size according to the present math style (uses the amsmath definitions `\@mathmeasure` and `\big@size`). This code is activated only if the package option `scriptstyle` is set to `true`.

```
\ifdlm@script
\def\bBigg@choice#1#2#3#4{%
 {\@mathmeasure\z@{\nulldelimiterspace\z@}%
    {\big@size#2\big@size#1\left#4\vcenter to#3\big@size{}\right.}%
  \box\z@}}
\def\bBigg@#1#2{{\mathchoice%
 {\bBigg@choice{\displaystyle}{1}{#1}{#2}}%
 {\bBigg@choice{\textstyle}{1}{#1}{#2}}%
 {\bBigg@choice{\scriptstyle}{0.7}{#1}{#2}}%
 {\bBigg@choice{\scriptscriptstyle}{0.5}{#1}{#2}}}}
\fi
```

Define punctuation marks (`\bigp`, etc.) and binary operators (`\bigb`, etc.).

```
\newcommand{\bigp}{\mathpunct\big}
\newcommand{\Bigp}{\mathpunct\Big}
\newcommand{\biggp}{\mathpunct\bigg}
\newcommand{\Biggp}{\mathpunct\Bigg}
\newcommand{\bigb}{\mathbin\big}
\newcommand{\Bigb}{\mathbin\Big}
\newcommand{\biggb}{\mathbin\bigg}
\newcommand{\Biggb}{\mathbin\Bigg}
```

Define size selectors for standard size (`dlm@norm`), variable size (`dlm@var`).

```
\newcommand{\dlm@class}[1]{\csname dlm@class@#1\endcsname}
\newcommand{\dlm@class@}{}
\newcommand{\dlm@class@l}{\mathopen}
\newcommand{\dlm@class@r}{\mathclose}
\newcommand{\dlm@class@o}{\mathopen}
\newcommand{\dlm@class@c}{\mathclose}
```

```
\newcommand{\dlm@class@m}{\mathrel}
\newcommand{\dlm@class@p}{\mathpunct}
\newcommand{\dlm@class@b}{\mathbin}
\newcommand{\dlm@norm}[1]{\dlm@class{#1}\bBigg@{0.0}}
\newcommand{\dlm@big}[1]{\dlm@class{#1}\big}
\newcommand{\dlm@Big}[1]{\dlm@class{#1}\Big}
\newcommand{\dlm@bigg}[1]{\dlm@class{#1}\bigg}
\newcommand{\dlm@Bigg}[1]{\dlm@class{#1}\Bigg}
\newcommand{\dlm@var}[1]{\csname dlm@var@#1\endcsname}
\newcommand{\dlm@var@}[1]{\middle#1}
\newcommand{\dlm@var@l}[1]{\mathopen{}\mathclose\bgroup\left#1}
\newcommand{\dlm@var@r}[1]{\right#1\egroup}
\newcommand{\dlm@var@o}[1]{\mathopen{}\middle#1\mathopen{}}
\newcommand{\dlm@var@c}[1]{\mathclose{}\middle#1\mathclose{}}
\newcommand{\dlm@var@m}[1]{\mathrel{}\middle#1\mathrel{}}
\newcommand{\dlm@var@p}[1]{\middle#1\mathpunct{}}
\newcommand{\dlm@var@b}[1]{\mathbin{}{}\mkern-\medmuskip%
  \middle#1\mkern-\medmuskip{}\mathbin{}}
```

Optional Size Argument Processing. Parse the optional argument following `\delim`... commands to specify the size of delimiters. If the next character in line begins a group (‘`{`’) it is assumed that there is no optional argument. Otherwise the selected size is written to `\dlm@arg`. Unspecified or unknown sizes default to normal size. Finally, execute the code passed as an explicit argument to `\dlm@parsesize`.

```
\newcommand{\dlm@parsesize}[1]{\@ifnextchar\bgroup%
  {\dlm@parsesize@{#1}{0}}{\dlm@parsesize@{#1}}}
\newcommand{\dlm@parsesize@}[2]{\let\dlm@arg\dlm@norm%
  \if0#2\let\dlm@arg\dlm@norm\fi%
  \if1#2\let\dlm@arg\dlm@big\fi%
  \if+#2\let\dlm@arg\dlm@big\fi%
  \if!#2\let\dlm@arg\dlm@big\fi%
  \if2#2\let\dlm@arg\dlm@Big\fi%
  \if3#2\let\dlm@arg\dlm@bigg\fi%
  \if4#2\let\dlm@arg\dlm@Bigg\fi%
  \if*#2\let\dlm@arg\dlm@var\fi%
  #1}
```

Note that the delimited expression should be contained within a group such that nested delimiters will not overwrite the outer size definition.

Size Selection. The command `\selectdelim` reproduces the delimiter in the second argument using the math class given in the first argument using the previously selected size stored in `\dlm@arg`. It appends the class identifier to the command stored in `\dlm@arg` and calls the latter with the delimiter as the argument. If the class identifier is ‘`.`’, just return the delimiter argument as is.

```
\newcommand{\selectdelim}[2][]{\if.#1#2\else\dlm@arg{#1}#2\fi}
```

Declaration of New Delimiter Commands. Declares a new set of delimiters as the macro ‘\name’. This macro checks for an optional size argument and stores it in `\dlm@arg`. It then passes on to a second macro ‘`\dlm@dcl@`*name*’, which takes the actual code.

```
\newcommand{\DeclareMathDelimiterSet}[1]{\expandafter\dlm@declare%
  \csname dlm@dcl@\expandafter\@gobble\string#1\endcsname{#1}}
\def\dlm@declare#1#2{\newcommand{#2}{\dlm@parsesize{#1}}\newcommand{#1}}
```

Inline Delimiter Declarations. Inline declaration for delimiters via `\delim...`. The following code is similar to the one produced by `\DeclareMathDelimiterSet`, but the delimiter arguments are processed *before* the optional size modifier.

`\delim` is used for a single delimited expression.

```
\newcommand{\delim}[2]
  {\dlm@parsesize{\dlm@dcl@delim{#1}{#2}}}
\newcommand{\dlm@dcl@delim}[3]
  {\selectdelim[l]#1{#3}\selectdelim[r]#2}
```

`\delimpair` is used for two delimited expressions separated by an intermediate delimiter.

```
\newcommand{\delimpair}[3]
  {\dlm@parsesize{\dlm@dcl@delimpair{#1}{#2}{#3}}}
\newcommand{\dlm@dcl@delimpair}[5]
  {\selectdelim[l]#1{#4}\selectdelim#2{#5}\selectdelim[r]#3}
```

`\delimtriple` is used for three delimited expressions separated by two intermediate delimiters.

```
\newcommand{\delimtriple}[4]
  {\dlm@parsesize{\dlm@dcl@delimtriple{#1}{#2}{#3}{#4}}}
\newcommand{\dlm@dcl@delimtriple}[7]
  {\selectdelim[l]#1{#5}\selectdelim#2{#6}\selectdelim#3{#7}\selectdelim[r]#4}
```

Standard Definitions. Define some common delimiters (by `providecommand` so as not to overwrite previously existing commands). This code is activated only if the package option `stddef` is set to `true`.

```
\ifdlm@std
\providecommand{\brk}[1][r]{\begingroup\def\dlm@use{\delim()}%
  \if r#1 \def\dlm@use{\delim()}\fi%
  \if s#1 \def\dlm@use{\delim[]}\fi%
  \if c#1 \def\dlm@use{\delim\{\}}\fi%
  \if a#1 \def\dlm@use{\delim<>}\fi%
  \expandafter\endgroup\dlm@use}
\providecommand{\eval}[1][v]{\begingroup\def\dlm@use{\delim.\rvert}%
  \if v#1 \def\dlm@use{\delim.\rvert}\fi%
  \if s#1 \def\dlm@use{\delim[]}\fi%
  \expandafter\endgroup\dlm@use}
\providecommand{\abs}{\delim\lvert\rvert}
\providecommand{\norm}{\delim\lVert\rVert}
\fi
```

Extended Definitions. Define some extended delimiters. This code is activated only if the package option `extdef` is set to `true`.

```
\ifdlm@ext
\providecommand{\pair}{\delimpair({[.],})}
\providecommand{\set}{\delim\{\}}
\providecommand{\setcond}{\delimpair\{|\}}
\providecommand{\intv}[1][c]{\begingroup\def\dlm@use{\delimpair[{[.],}]}%
  \if c#1 \def\dlm@use{\delimpair[{[.],}]}\fi%
  \if l#1 \def\dlm@use{\delimpair]{[.],}]}\fi%
  \if r#1 \def\dlm@use{\delimpair[{[.],}[}\fi%
  \if o#1 \def\dlm@use{\delimpair]{[.],}[}\fi%
  \expandafter\endgroup\dlm@use}
\providecommand{\avg}{\delim<>}
```

```
\providecommand{\corr}{\delim<>}
\providecommand{\comm}{\delimpair[{[.],}]}
\providecommand{\acomm}{\delimpair\{{[.],}\}}
\providecommand{\bra}{\delim<|}
\providecommand{\ket}{\delim|>}
\providecommand{\braket}{\delimpair<|>}
\fi
```