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`

v2.3.1: 2026-03-26

Abstract

delimset is a LaTeX 2_ε 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:

$$[(\texttt{ax+b})\texttt{x+c}] \rightarrow [(ax+b)x+c]$$
$$\texttt{\textbackslash\{1,2,3,\textbackslash ldots\textbackslash\}} \rightarrow \{1,2,3,\ldots\}$$
$$(\texttt{v,w}) \rightarrow (v,w)$$
$$\texttt{\textbackslash langle\textbackslash psi|M|\textbackslash psi\textbackslash rangle} \rightarrow \langle\psi|M|\psi\rangle$$
$$(\texttt{x\^{}2+px+q})\texttt{|\_\{-p/2\}} \rightarrow (x^2+px+q)|_{-p/2}$$
$$\ldots$$

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:

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

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):

$$(\texttt{A\textbackslash hat\{X\}+B}) \rightarrow (A\hat{X}+B)$$
$$\texttt{\textbackslash left(A\textbackslash hat\{X\}+B\textbackslash right)} \rightarrow \left(A\hat{X}+B\right)$$
$$\texttt{\textbackslash Big(A\textbackslash hat\{X\}+B\textbackslash Big)} \rightarrow \Big(A\hat{X}+B\Big)$$
$$\texttt{\textbackslash big(A\textbackslash hat\{X\}+B\textbackslash big)} \rightarrow \big(A\hat{X}+B\big)$$

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

$$\texttt{\textbackslash brk\{A\textbackslash hat\{X\}+B\}} \rightarrow (A\hat{X}+B)$$
$$\texttt{\textbackslash brk*\{A\textbackslash hat\{X\}+B\}} \rightarrow \left(A\hat{X}+B\right)$$
$$\texttt{\textbackslash brk\^{}2\{A\textbackslash hat\{X\}+B\}} \rightarrow \Big(A\hat{X}+B\Big)$$
$$\texttt{\textbackslash brk!\{A\textbackslash hat\{X\}+B\}} \rightarrow \big(A\hat{X}+B\big)$$

It also allows to change the type of bracket conveniently:

$$\texttt{\textbackslash brk\{ax+b\}} \rightarrow (ax+b)$$
$$\texttt{\textbackslash brk[s]\{ax+b\}} \rightarrow [ax+b]$$
$$\texttt{\textbackslash brk[c]\{ax+b\}} \rightarrow \{ax+b\}$$
$$\texttt{\textbackslash brk[a]\{ax+b\}} \rightarrow \langle ax+b\rangle$$

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

$$\texttt{\textbackslash brk[s]!\{\textbackslash brk\{ax+b\}x+c\}} \rightarrow \bigl[(ax + b)x + c\bigr]$$

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.:

$$\texttt{\textbackslash delim<>\{ax+b\}} \rightarrow \langle ax + b \rangle$$
$$\texttt{\textbackslash delimpair[\{[p]*,\}[!\{a\}\{b\}} \rightarrow \bigl[a, b\bigr[$$
$$\texttt{\textbackslash delimtriple<||>*\{\textbackslash psi\}\{A\^{}\textbackslash dagger\}\{\textbackslash psi\}} \rightarrow \langle \psi | A^\dagger | \psi \rangle$$

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

$$\texttt{\textbackslash newcommand\{\textbackslash comm\}\{\textbackslash delimpair\{[\}\{[p]*,\}\{]\}\}}$$

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

$$\texttt{\textbackslash comm!\{P\}\{\textbackslash psi(x)\}} \rightarrow \bigl[P, \psi(x)\bigr]$$

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

$$\texttt{\textbackslash DeclareMathDelimiterSet\{\textbackslash comm\}[2]}$$
$$\texttt{\{\textbackslash selectdeliml\{[\}\{\#1\},\{\#2\}\textbackslash selectdelimr\{]\}\}}$$

1.1 Delimiter Sizes for Math Styles

In plain $\LaTeX 2_\varepsilon$ 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 (an arguably better type-setting practice is to avoid complicated expressions in sub/superscripts in the first place), they come out way too big:

$$\texttt{e\^{}\{\textbackslash big(ax+b\textbackslash big)\}} \rightarrow e^{\bigl(ax+b\bigr)}$$

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

$$\texttt{e\^{}\{\textbackslash big(ax+b\textbackslash big)\}} \rightarrow e^{(ax+b)}$$

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:

$$\texttt{\textbackslash square(ax+b)\textbackslash square} \to \square(ax+b)\square$$
$$\texttt{\textbackslash square\textbackslash bigl(ax+b\textbackslash bigr)\textbackslash square} \to \square\bigl(ax+b\bigr)\square$$
$$\texttt{\textbackslash square\textbackslash left(ax+b\textbackslash right)\textbackslash square} \to \square\left(ax+b\right)\square$$
$$\texttt{\textbackslash square\textbackslash left(ax+\textbackslash big.b\textbackslash right)\textbackslash square} \to \square\left(ax+b\right)\square$$

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

$$\texttt{\textbackslash square(ax+b)\textbackslash square} \to \square(ax+b)\square$$
$$\texttt{\textbackslash square\textbackslash mathopen\{\}\textbackslash mathclose\{\textbackslash left(ax+b\textbackslash right)\}\textbackslash square} \to \square(ax+b)\square$$

This makes the expression look like $\texttt{\textbackslash mathopen}$ from the left and like $\texttt{\textbackslash mathclose}$ from the right. Importantly, the delimited expression should be contained in the $\texttt{\textbackslash mathclose}$ 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 (such as $\texttt{\textbackslash mathopen}$ or $\texttt{\textbackslash mathclose}$) for the delimiters, for example:

$$\texttt{\textbackslash big(-1\textbackslash big)} \to \bigl(-1\bigr)$$
$$\texttt{\textbackslash bigl(-1\textbackslash bigr)} \to \bigl(-1\bigr)$$

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

$$\texttt{\textbackslash bigl<\textbackslash psi\textbackslash big|\textbackslash psi\textbackslash bigr>} \to \bigl\langle\psi|\psi\bigr\rangle$$
$$\texttt{\textbackslash bigl<\textbackslash psi\textbackslash mathpunct\textbackslash big|\textbackslash psi\textbackslash bigr>} \to \bigl\langle\psi|\,\psi\bigr\rangle$$
$$\texttt{\textbackslash bigl<\textbackslash psi\textbackslash mathinner\textbackslash big|\textbackslash psi\textbackslash bigr>} \to \bigl\langle\psi\,|\,\psi\bigr\rangle$$
$$\texttt{\textbackslash bigl<\textbackslash psi\textbackslash mathbin\textbackslash big|\textbackslash psi\textbackslash bigr>} \to \bigl\langle\psi\,|\,\psi\bigr\rangle$$
$$\texttt{\textbackslash bigl<\textbackslash psi\textbackslash bigm|\textbackslash psi\textbackslash bigr>} \to \bigl\langle\psi\,\big|\,\psi\bigr\rangle$$

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

The package $\texttt{delimset}$ 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   Compounds and Broken Lines

The mechanism provided by the package requires the delimited expression to appear as an uninterrupted compound. In particular, the expression cannot span several lines in a multi-line equation or several columns in a matrix. The same restriction applies to the elementary $\texttt{\textbackslash left}$ and $\texttt{\textbackslash right}$ construct. In cases where delimited expressions are torn apart across several blocks, the delimiters have to be typeset individually. The package offers a mechanism to display individual delimiters of a given delimiter set. It also assists in maintaining the same type and size across blocks. Being able to display individual delimiters of defined delimiter sets offers a broad range of (ab)use.

Individual delimiters of a set can be selected and displayed by specifying appropriate flags, for example:

$$\texttt{\textbackslash brk(} \rightarrow (, \qquad \texttt{\textbackslash brk)} \rightarrow)$$

These flags can be combined with the size specifier or other bracket modifiers, e.g.:

$$\texttt{\textbackslash brk\textasciicircum 3(} \rightarrow \left(, \qquad \texttt{\textbackslash brk[s]\textasciicircum 2)} \rightarrow \right]$$

This enables to conveniently split delimited expressions over blocks of code, e.g.:

$$\texttt{\textbackslash brk[s]\textasciicircum 3(\textbackslash frac\{A\}\{B\}}\ldots /\!\!/ \ldots \texttt{\textbackslash frac\{C\}\{D\}\textbackslash brk[s]\textasciicircum 3)} \rightarrow \left[\frac{A}{B}\cdots /\!\!/ \cdots \frac{C}{D}\right]$$

Here, the type and modifiers specifying the delimiter set are repeated for all individual delimiters. To simplify such expressions and their maintenance, the package provides global registers which store the type and size of delimiters. With those, the above expression can be abbreviated as:

$$\texttt{\textbackslash brk[s]\textasciicircum 3[}\ldots /\!\!/ \ldots \texttt{\textbackslash usedelim]} \quad \text{or} \quad \texttt{\textbackslash brk[s]\textasciicircum 3[}\ldots /\!\!/ \ldots \texttt{\textbackslash brk]}$$

Here, `\usedelim]`, alternatively `\brk]`, closes the delimiter set with the type and options previously specified by `\brk...[`. This mechanism respects nesting, and for even broader control there exists a more elaborate solution using registers, see below.

1.4 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.

1.5 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.:

$$\texttt{\textbackslash delimpair<|>!\{\textbackslash psi\}\{\textbackslash psi\}} \rightarrow \langle \psi | \psi \rangle$$

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

$$\texttt{\textbackslash usepackage\{delimset\}}$$

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

2.1 Delimiter Sets and Presentation Flags

The package provides commands to represent various delimiter sets, see section 2.2 and section 2.3 below. Their display can be modified systematically by specifying flags. A delimiter set `\`*name*, e.g. `\brk`, is typically invoked by a sequence like:

$$\texttt{\textbackslash}name\ flags\ \{expr1\}\{expr2\} \ldots$$

where *expr(n)* are the terms to be enclosed by the delimiters and the optional *flags* adjust the presentation of delimiters. The *flags* are described as follows:

First, a size modifier flag controls the size of the delimiters to be displayed. It can take one of the following values:

`0`	default size (0.96 times default line height)
`!, +`	increase size by one step
`^1`	size `\big` (1.2 times default line height)
`^2`	size `\Big` (1.8 times default line height)
`^3`	size `\bigg` (2.4 times default line height)
`^4`	size `\Bigg` (3.0 times default line height)
`^`n, `^{`x`}`	flexible size adjustment (n=`0`...`9`; x: decimal number)
`_{`h`}`	adjustment to absolute height h (centred)
`*`	variable size `\left` ... `\right`

The intended delimiter size is stored as the dimension variable:

$$\texttt{\textbackslash delimsize}$$

It may be used to adjust spacing within the delimiter set proportional to the size of the delimiters. Note that the actual size of delimiters may deviate depending on availability within the given font. Note further that the size cannot be determined for variable-size delimiters ('*') and thus size `\Big` is assumed for this case. Note finally that the sub/superscript styles are not taken into account and `\delimsize` applies to the default case `\textstyle`. The package also supplies a macro to add horizontal glue proportional to the intended delimiter size:

$$\texttt{\textbackslash kerndelim\{}\textit{width}\texttt{\}}$$

Here, *width* is the amount of horizontal glue in units of mu scaled by the ratio of `\delimsize` and the current math font height. The macro does adjust for the selected script style.

Second, a delimiter selection flag is used to pick out and display just one individual delimiter within the set. It can take one of the following values:

(	left delimiter
|, ,	first intermediate delimiter
?n	n-th delimiter (starting with 0: left)
)	right delimiter

Third, register flags store and retrieve delimiter sets for splitting them across blocks or lines:

[	push delimiter type to register stack, display left delimiter (similar to '>.(')
]	pop delimiter type from stack, display right delimiter (similar to '<.)')
>r, >{*reg*}	store delimiter type to register r or *reg*
<r, <{*reg*}	retrieve delimiter type from register r or *reg*

A register stores the type of delimiter set (*name*), the desired size (except for variable size '*') as well as the math class of the compound (''' vs. '"', see below). The register stack enables convenient access to nested delimiter sets, and the special register '.' points to the top of the stack. A stored register can be retrieved and displayed by the macro:

$$\texttt{\textbackslash usedelim}\,r\;\textit{flags}\;\textit{sel}\quad\text{or}\quad\texttt{\textbackslash usedelim\{}\textit{reg}\texttt{\}}\,\textit{flags}\;\textit{sel}\quad\text{or}\quad\texttt{\textbackslash usedelim}\,\textit{sel}$$

Here, *sel* is a flag to select an individual delimiter ('(', '|', ',', '?n', ')' or ']'), see above. The abbreviated form without register specification operates on the top of the register stack.

In addition, there are some general purpose flags:

.	terminates flag processing
:	enclose delimiter set in block (for using total box dimensions)
'	render delimiter set as `\mathopen` and `\mathclose`
"	render delimiter set as `\mathinner`
–	display phantom delimiters (for individual delimiters)

Please note the following:

- A delimiter selector (including the combined flags '[' and ']') terminates flag processing, therefore all relevant flags must be specified before it.

- Parsing also stops with the opening of the first term argument by means of the opening brace '{'.

- Parsing can also be terminated manually by specifying the flag '.' (in case the opening brace '{' for arguments is missing for whatever reason).

- Any other letter or token at the location of *flags* is interpreted as an unrecognised flag and triggers a compiler error. For instance, `\brk{x}` must not be abbreviated as `\brk x` because 'x' would be interpreted as a flag (it might be abbreviated as `\brk.x`).

- The legacy size modifier flags '1' . . . '4' provided for backward compatibility have the same effect as '^1'. . . '^4'.

- The phantom delimiter flag '-' does not work for variable size. It is intended for reserving space for individual delimiters when they are composed manually into sets (e.g. across lines).

- Technically, individual delimiters can be produced in the variable size version (even though it may defeat the purpose of the delimiter selection mechanism). One has to pay attention when using only the opening or only the closing delimiters from the package and counterbalancing it by direct code for the opposite side. This is because '*(' and '*)' do not directly translate to `\left`*delim* and `\right`*delim* due to proper spacing, see section 1.2. Instead, they translate to:

$$*(\to \mathtt{\backslash mathopen\{\}\backslash mathclose\backslash bgroup\backslash left}\mathit{delim}$$
$$*) \to \mathtt{\backslash right}\mathit{delim}\mathtt{\backslash egroup}$$

  One should use analogous code to complete the manual variable size block.
  Note that similar restrictions apply to using the flag ':' for block enclosure which uses `\bgroup` and `\egroup` as well.
  As it does not make much sense to use variable-size ('*') and block (':') compounds for individual delimiter selection, the state of these flags is not stored in the global registers.

- The delimiter selectors '[' and ']' implement a stack, and they must be properly balanced. Technically, they access the register '.*n*' with the stack counter *n*=1,2,3. . . a global variable which is increased or decreased, respectively. The special register '.' points to the current top of the stack to enable access to intermediate delimiters.

2.2 Inline Declarations

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

$$\mathtt{\backslash delim}\{l\}\{r\}\mathit{flags}\{\mathit{expr}\}$$
$$\mathtt{\backslash delimpair}\{l\}\{m\}\{r\}\ \mathit{flags}\{\mathit{expr1}\}\{\mathit{expr2}\}$$
$$\mathtt{\backslash delimtriple}\{l\}\{m\}\{n\}\{r\}\ \mathit{flags}\{\mathit{expr1}\}\{\mathit{expr2}\}\{\mathit{expr3}\}$$

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*:

$$l\ expr\ r, \qquad l\ expr1\ m\ expr2\ r, \qquad l\ expr1\ m\ expr2\ n\ expr3\ r$$

Here, *l*, *r*, *m*, *n* should be math delimiters *delim* (elementary symbols which can be used for `\left` and `\right`) or the dot '.' for the null delimiter. The delimiter set expression can be adjusted in two ways:

8

First, size adjustment for the delimiters l, r, m, n can be suppressed by a starred variant:

$$\{*token\} \qquad \text{or} \qquad \{*\{expr\}\}$$

This allows to use an arbitrary *token* or a compound expression *expr* in place of a delimiter *delim*. This is useful when a delimiter character should not adjust in size (e.g. commas), or if a flexible-size version of the character is not available. Compounds can be used to design size-adjusting delimiters more flexibly using `\selectdelim` and `\doubledelim` (see below). One might also use a compound expression to create a null delimiter with non-zero width (the width may be specified by `\kerndelim` to make it proportional to the delimiter size).

Second, the intended math class of the intermediate delimiters m, n may be adjusted in order to achieve a more appropriate automatic spacing by using `{[`*class*`]`*delim*`}`. Here, *class* specifies the intended math class of the delimiter *delim* according to:

delim ⎫	⎧ no glue for size-adjusted delimiters (same as `[o]`)
`{`*delim*`}` ⎭	⎩ determined by expression for unadjusted size (∗ variant)
`{[o]`*delim*`}`	no glue, similar to `\big`, `\mathord`, `\mathopen`, `\mathclose`
`{[p]`*delim*`}`	followed by `\thinmuskip`, similar to `\bigp`, `\mathpunct`
`{[i]`*delim*`}`	surrounded by `\thinmuskip`, similar to `\bigi`, `\mathinner`
`{[b]`*delim*`}`	surrounded by `\medmuskip`, similar to `\bigb`, `\mathbin`
`{[m]`*delim*`}`	surrounded by `\thickmuskip`, similar to `\bigm`, `\mathrel`

Please note the following:

- The left and right delimiters l and r are assigned to special classes: They leave no initial and final glue on the enclosed expressions (same as `\mathopen` and `\mathclose`). From the outside, the delimiter set appears either as a combination of `\mathopen`+`\mathclose` (flag '') or as `\mathinner` (flag '"').

- The legacy class `[.]` provided for backward compatibility specifies delimiters with unadjusted size: the compound `{[.]`*expr*`}` has the same effect as `{*{`*expr*`}}`.

- The legacy class `[c]` provided for backward compatibility has the same effect as `[o]`.

- Unadjusted-size delimiters can be combined with math classes as:

$$\{[class]*token\} \qquad \text{or} \qquad \{[class]*\{expr\}\}$$

- The opening square bracket '[' takes the dual role of indicating the optional [*class*] argument. To avoid conflicts on intermediate delimiters, opening square bracket delimiters should be encoded as '`{{[}}`' (for `\delim...`) or '`{[}`' (for `\selectdelim`) rather than '['.

2.3 Declarations

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

$$\texttt{\textbackslash newcommand\{}\textbackslash name\texttt{\}\{}\textbackslash\texttt{delim\{}l\texttt{\}\{}r\texttt{\}} \ flags\texttt{\}}$$
$$\texttt{\textbackslash newcommand\{}\textbackslash name\texttt{\}\{}\textbackslash\texttt{delimpair\{}l\texttt{\}\{}m\texttt{\}\{}r\texttt{\}} \ flags\texttt{\}}$$
$$\texttt{\textbackslash newcommand\{}\textbackslash name\texttt{\}\{}\textbackslash\texttt{delimtriple\{}l\texttt{\}\{}m\texttt{\} \{}n\texttt{\}\{}r\texttt{\}}flags\texttt{\}}$$

Here it makes sense to drop all arguments starting at the optional *flags* argument from the definition. The TEX parsing mechanism will then automatically use the tokens following *name* including the optional size modifier. If any of the encapsulated expression(s) are to

be passed as explicit arguments to *name*, one will have to find an alternative way to pass *flags*. However, it is possible to predefine *flags* which can be overridden by further *flags* specified by the user.

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

$$\text{\\DeclareMathDelimiterSet}\{\text{\\}name\}[narg]\{compositor\}$$

The syntax of this command is equivalent to the one of \\newcommand. The difference is that the command *name* first looks for the *flags* argument as described above in section 2.2. It remembers the desired size for evaluating the macro expression *compositor*. Then it parses the arguments as if the command was declared by \\newcommand.

As usual, the macro expression *compositor* 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:

$$\text{\\selectdeliml}[*]\{delim\}$$
$$\text{\\selectdelim}[class][*]\{delim\}$$
$$\text{\\selectdelimr}[*]\{delim\}$$

The three commands must be in proper sequence starting with \\selectdeliml followed by arbitrarily many \\selectdelim and terminated by \\selectdelimr. The math classes *class* and starred variants are defined analogously to section 2.2. Here, the sizes are adjusted automatically according to the previously specified modifier. Note that the expressions for the starred forms may contain \\selectdelim to produce delimiters in the desired size. Compounds of two identical delimiters with horizontal kerning adjustment *width* (default -2) are produced by:

$$\text{\\doubledelim}[width]\,delim$$

Picking out individual delimiters from a set declared by \\DeclareMathDelimiterSet cannot be achieved automatically. Instead, a selector method may be specified manually (where needed) by:

$$\text{\\DeclareMathDelimiterSel}\{\text{\\}name\}$$
$$\{\text{\\selectdeliml}\{l\}[\text{\\or}\text{\\selectdelim}\{m\}...]\}$$
$$\{\text{\\selectdelimr}\{r\}\}$$

As many intermediate delimiters as needed can be specified at the end of the first argument; each one must start with '\\or'.

For example, one might declare a set of double square brackets $[\![*]\!]$ using:

```
\DeclareMathDelimiterSet{\dsb}[1]
    {\selectdeliml{[}\kerndelim{-1.75}\selectdelim[o]{[}{#1}
    \selectdelim[o]{]}}\kerndelim{-1.75}\selectdelimr{]}}
```

Here, the command \\kerndelim{-1.75} reduces the spacing between the two square brackets by approximately 1.75 mu in default size. This could be used for:

$$\text{\\dsb\textasciicircum4\{\\dsb\textasciicircum3\{\\dsb\textasciicircum2\{\\dsb\textasciicircum1\{\\dsb\{*\}\}\}\}\}} \to \left[\!\left[\left[\!\left[\left[\!\left[[\![[\![*]\!]]\!]\right]\!\right]\right]\!\right]\right]\!\right] \qquad \text{\\dsb*...} \to \left[\!\left[\frac{x}{y}\right]\!\right]$$

An appropriate selector method for displaying individual delimiters reads:

```
\DeclareMathDelimiterSel{\dsb}
  {\selectdeliml{[}\kerndelim{-1.75}\selectdelim[o]{[}}
  {\selectdelim[o]]\kerndelim{-1.75}\selectdelimr{]}}
```

An inline version of the above definitions using compound expressions with manual resizing of the delimiters (via `\selectdelim` and `\doubledelim`) is achieved by:

```
\delim{*{\doubledelim[-1.75](}}{*{\doubledelim[-1.75])}}
```

Finally, towards a very flexible manual composition of delimiter sets, there is the macro:

$$\text{\texttt{\textbackslash parsedelimflags}}\{\textit{compositor}\}\{\textit{selector}\}\textit{flags}$$

It parses the modifier flags as described in section 2.1. The argument *compositor* is code that displays the composited delimiter set whereas *selector* is a macro that displays the delimiter number `\selecteddelim` of the set. The *flags* follow immediately and should be terminated by an opening brace '{', the terminating flag '.' or a delimiter selection flag.

2.4   Default Declarations

The package predefines four commonly used sets of delimiters:

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

		single		double
round	`r:`	(x)	`R:`	$((x))$
square	`s:`	$[x]$	`S:`	$[[x]]$
curly	`c:`	$\{x\}$	`C:`	$\{\{x\}\}$
angle	`a:`	$\langle x \rangle$	`A:`	$\langle\langle x \rangle\rangle$
bar	`v:`	$|x|$	`V:`	$\|x\|$

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

$$\textit{empty or }[\texttt{v}]\text{: } f(x)\big|_a, \qquad [\texttt{s}]\text{: } \big[f(x)\big]_a^b$$

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

The above definitions can be suppressed by setting the package option `stddef` to `false`, see section 2.6. The package also defines some extended sets of delimiters as follows:

- `\pair[`*type sep*`]`*flags*`{`*expr1*`}{`*expr2*`}` represents a pair(ing) $(\textit{expr1}, \textit{expr2})$. The type of bracket can be altered by the optional argument *type* as for `\brk`. The separator *sep* can be adjusted to ',' (default) for comma, ';' for semi-colon and '|' for vertical bar.
- `\set`*flags*`{`*expr*`}` represents the set $\{\textit{expr}\}$.
- `\setcond`*flags*`{`*expr*`}{`*cond*`}` represents a set with condition $\{\textit{expr}|\textit{cond}\}$.

- `\intv[`*type*`]`*flags*`{`*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`*flags*`{`*expr*`}` represents some average $\langle expr \rangle$.
- `\corr`*flags*`{`*expr*`}` represents some correlator $\langle expr \rangle$.
- `\comm`*flags*`{`*expr1*`}{`*expr2*`}` represents the commutator $[expr1, expr2]$.
- `\acomm`*flags*`{`*expr1*`}{`*expr2*`}` represents the anti-commutator $\{expr1, expr2\}$.
- `\bra`*flags*`{`*expr*`}` represents a bra-vector $\langle expr |$ in quantum mechanics.
- `\ket`*flags*`{`*expr*`}` represents a ket-vector $| expr \rangle$ in quantum mechanics.
- `\braket`*flags*`{`*expr1*`}{`*expr2*`}` represents a bra-ket contraction $\langle expr1 | expr2 \rangle$.
- `\lfrac[`*type*`]`*flags*`{`*expr1*`}{`*expr2*`}` describes the linear representation *expr1*/*expr2* of a fraction. An optional argument *type* taking values `[r]`, `[s]`, `[c]`, `[a]` encloses the fraction in round, square, curly or angle brackets as for `\brk`.

The extended definitions need to be activated by the package option `extdef`, see section 2.6.

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

2.5   Auxiliary Commands

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

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 2.6.

2.6   Package Options

Options can be passed to the package by:

$$\text{\usepackage[}\textit{opts}\text{]\{delimset\}} \qquad \text{or} \qquad \text{\PassOptionsToPackage\{}\textit{opts}\text{\}\{delimset\}}$$

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

- `stddef[`=`true`|`false]` controls the activation of standard delimiter definitions specified in section 2.4. 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 2.4. If no value is given `true` is assumed; initially set to `false`.
- `scriptstyle[`=`true`|`false]` controls the overwriting of size modifiers explained in section 2.5. If no value is given `true` is assumed; initially set to `true`.

3 Information

3.1 Copyright

Copyright © 2016–2026 Niklas Beisert

This work may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in `https://www.latex-project.org/lppl.txt` and version 1.3c or later is part of all distributions of LaTeX version 2008 or later.

This work has the LPPL maintenance status 'maintained'.

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 listed below.

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
`delimset-samp.tex`	sample file
`delimset.pdf`	documentation

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

- Run LaTeX on `delimset.ins` to create the package `delimset.sty` and further documentation files.
- Copy the file `delimset.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/delimset`.
- Alternatively, you may copy `delimset.sty` to your project directory.
- Run (pdf)LaTeX on `delimset.dtx` to compile the documentation `delimset.pdf` (this file).

3.3 Interaction with CTAN Packages

The package is related to other packages available at CTAN:

- Compatibility with the amsmath package has been tested with v2.15d (2016-06-28) and v2.17t (2024-11-05).
- 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).
- The package icomma modifies the spacing behaviour of the comma character in math mode which leads to inadequate spacing when it is used as a middle delimiter. To achieve proper spacing in sample code, use `{[p]*,}` rather than `{*,}`. The extended commands `\intv`, `\comm`, `\acomm` and `\pair` are compatible with the icomma package has been tested with v2.0 (2002-03-10).

- This package cooperates with the package beamer in not changing colours for delimiters. Compatibility with the beamer package has been tested with v3.77 (2026-01-22).

3.4 Revision History

v2.3.1: 2026-03-26

- fixed vertical bars in conjunction with unicode-math (thanks to Ulrike Fischer for pointing out the issue)

v2.3: 2026-03-23

- added more types to \brk and \pair including double delimiters
- added \doubledelim for compounds of two identical delimiters
- modifiers '!'/'+' can accumulate to increase size further
- provide new \big, \Big, \bigg, \Bigg if scriptstyle is active
- fix for not applying math text colour for delimiters in beamer
- modifier '[' now throws an error when used with variable size
- package maintenance

v2.2.2: 2025-03-27

- parameter fetching mechanism for \usedelim changed

v2.2.1: 2025-03-25

- maintenance and manual update

v2.2: 2025-02-27

- replicate the amsmath mechanism for tracking the vertical height of a default delimiter in order to be independent of amsmath

v2.1: 2025-01-05

- the flags '[' and ']' now implement a register stack for convenient individual access to nested delimiter sets
- syntax of \usedelim extended for direct access to top of register stack

v2.0: 2024-07-17

- option to display individual delimiters and to store them in global registers across blocks (columns, lines, etc.)
- more general handling of unadjusted-size delimiters
- more general size adjustments
- added inner math class for intermediate delimiters

- added flag to enclose by open/close and inner math class
- added flag to display phantom delimiters
- selected size accessible by `\delimsize` and add proportional kerning by `\kerndelim`.
- added extended definition `\lfrac` for plain inline fractions
- null delimiter now properly has zero width for variable size
- internal mechanisms revised
- compatibility with icomma package (thanks to Olivier Godin for pointing out the issue)

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

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

Preamble and beginning of document body:

```
1 \documentclass[12pt]{article}
2
3 \usepackage[margin=2cm]{geometry}
4 \usepackage{amsfonts}
5 \usepackage{delimset}
6
7 \begin{document}
```

sizes for default brackets:

```
 8 \[
 9 \brk^0{x},\quad
10 \brk^1{x},\quad
11 \brk^2{x},\quad
12 \brk^3{x},\quad
13 \brk^4{x}
14 \]
```

styles for default brackets:

```
15 \[
16 \brk[r]{x},\quad
17 \brk[s]{x},\quad
18 \brk[c]{x},\quad
19 \brk[a]{x}
20 \]
```

nested brackets:

```
21 \[
22 \brk[c]!!{\brk[s]!{\brk{ax+b}x+c}x+d}
23 \]
```

default absolute value, norm and default evaluations:

```
24 \[
25 \abs*{\frac{ax+b}{cx+d}},\qquad
26 \norm*{\frac{ax+b}{cx+d}},\qquad
27 \eval*{\frac{ax+b}{cx+d}}_{x=0},\qquad
28 \eval[s]*{\frac{ax+b}{cx+d}}_{x=0}^{x=\infty}
29 \]
```

outer delimiter spacing:

```
30 \[\begin{array}{ll}
31 \square\brk^0{x}\square,&\square\brk^1{A^k}\square,
32 \\
33 \square\brk*{x}\square,&\square\brk*{A^k}\square
34 \end{array}\]
```

delimiter sizes in exponents:

```
35 \[
36 e^{\brk{ax+b}},\qquad
37 e^{\brk!{ax+b}}
38 \]
```

delimiter declaration:

```
39 \DeclareMathDelimiterSet{\braket}[2]
40   {\selectdeliml<#1\selectdelim|#2\selectdelimr>}
41 \[
42 \braket!{\psi}{\psi},
43 \quad
44 \braket*{\psi}{\psi\big.}
45 \]
```

delimiter usage:

```
46 \[
47 \delimpair<|>!{\psi}{\psi}
48 \]
```

conditional set, alternative layouts:

```
49 \[
50 \delimpair\{{[m]|}\}!{2n}{n\in\mathbb{Z}},
51 \quad
52 \delimpair\{{[b]|}\}!{2n}{n\in\mathbb{Z}},
53 \quad
54 \delimpair\{{[i]|}\}!{2n}{n\in\mathbb{Z}},
```

```
55 \quad
56 \delimpair\{{[p]|}\}!{2n}{n\in\mathbb{Z}},
57 \quad
58 \delimpair\{|\}!{2n}{n\in\mathbb{Z}},
59 \quad
60 \delimpair\{{*;}\}!{2n}{n\in\mathbb{Z}}
61 \]
62 conditional set, alternative layouts with variable size:
63 \[
64 \delimpair\{{[m]|}\}*{2n}{n\in\mathbb{Z}\big.},
65 \quad
66 \delimpair\{{[b]|}\}*{2n}{n\in\mathbb{Z}\big.},
67 \quad
68 \delimpair\{{[i]|}\}*{2n}{n\in\mathbb{Z}\big.},
69 \quad
70 \delimpair\{{[p]|}\}*{2n}{n\in\mathbb{Z}\big.},
71 \quad
72 \delimpair\{|\}*{2n}{n\in\mathbb{Z}\big.},
73 \quad
74 \delimpair\{{*;}\}*{2n}{n\in\mathbb{Z}\big.}
75 \]
```

delimiter definition:

```
76 \newcommand{\comm}{\delimpair[{[p]*,}]}
77 \[
78 \comm!{\comm{A}{B}}{C}
79 +\comm!{\comm{B}{C}}{A}
80 +\comm!{\comm{C}{A}}{B}
81 =0
82 \]
```

alternative representation:

```
83 \renewcommand{\comm}{\delimpair[{[p]*;}]}
84 \[
85 \comm!{\comm{A}{B}}{C}
86 +\comm!{\comm{B}{C}}{A}
87 +\comm!{\comm{C}{A}}{B}
88 =0
89 \]
```

display individual delimiters of a set with nesting:

```
90 \renewcommand{\braket}{\delimpair<|>}
91 \[
92 \braket{A}{B}
93 \to \braket( A \braket| B \braket),
94 \quad
95 \braket*( A\big. \braket*| B_{} \braket*),
96 \quad
97 \braket^1( A \braket^3| B \braket^2),
98 \quad
99 \braket^2[ \brk[s]^1[ A \brk] \usedelim| B \usedelim]
100 \]
```

placing indices before a bracket (does not work in variable-size mode because the final size is not available for the enclosed expressions):

```
101 \DeclareMathDelimiterSet{\quadindex}[5]
102   {\selectdeliml.^{#2}_{#3}\mathord{}\selectdelim[o][
```

```
103   {#1}\selectdelim[o]]^{#4}_{#5}\selectdelimr.}
104 \[
105 \quadindex^2{\frac{x}{y}}{1}{2}{3}{4}
106 \]
```

End of document body:

```
107 \end{document}
```

B Implementation

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

Required Packages.   The package loads the package `keyval` if not yet present. `keyval` is used for extended options processing.

```
108 \RequirePackage{keyval}
```

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

```
109 \newif\ifdlm@std\dlm@stdtrue
110 \newif\ifdlm@ext\dlm@extfalse
111 \newif\ifdlm@script\dlm@scripttrue
112
113 \def\dlm@group{dlm@}
114 \define@key{\dlm@group}{stddef}[true]{\csname dlm@std#1\endcsname}
115 \define@key{\dlm@group}{extdef}[true]{\csname dlm@ext#1\endcsname}
116 \define@key{\dlm@group}{scriptstyle}[true]{\csname dlm@script#1\endcsname}
117
118 \DeclareOption*{\expandafter\setkeys\expandafter\dlm@group
119   \expandafter{\CurrentOption}}
120 \ProcessOptions
```

Interaction with beamerbasecolor.   Suppress changing colors in inner math structures when `beamer` is loaded:

```
121 \let\dlm@beamerbasecolor@fix\@empty
122 \AddToHook{package/beamerbasecolor/after}{%
123   \def\dlm@beamerbasecolor@fix{\donotcoloroutermaths}}
```

Improved Size Adjustments.   Overwrite the macros `\big`, `\Big`, `\bigg`, `\Bigg` and 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`.

```
124 \ifdlm@script
125 \def\dlm@big@s#1#2{{\delimsize#1\dlm@size\dlm@big@{#2}}}
126 \DeclareRobustCommand{\big}{\dlm@big@s{}}
127 \DeclareRobustCommand{\Big}{\dlm@big@s{1.5}}
128 \DeclareRobustCommand{\bigg}{\dlm@big@s{2}}
129 \DeclareRobustCommand{\Bigg}{\dlm@big@s{2.5}}
130 \ifdefined\bBigg@
131 \def\bBigg@choice#1#2#3#4{%
132   {\dlm@beamerbasecolor@fix\@mathmeasure\z@{\nulldelimiterspace\z@}%
```

```
133        {#1\left#4\vcenter to#3\dimexpr#2\big@size\relax{}\right.}%
134      \box\z@}}
135 \def\bBigg@#1#2{\leavevmode@ifvmode{\mathchoice
136    {\bBigg@choice{\displaystyle}{1}{#1}{#2}}%
137    {\bBigg@choice{\textstyle}{1}{#1}{#2}}%
138    {\bBigg@choice{\scriptstyle}{0.7}{#1}{#2}}%
139    {\bBigg@choice{\scriptscriptstyle}{0.5}{#1}{#2}}}}
140 \fi
141 \fi
```

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

```
142 \DeclareRobustCommand{\bigp}{\mathpunct\big}
143 \DeclareRobustCommand{\Bigp}{\mathpunct\Big}
144 \DeclareRobustCommand{\biggp}{\mathpunct\bigg}
145 \DeclareRobustCommand{\Biggp}{\mathpunct\Bigg}
146 \DeclareRobustCommand{\bigb}{\mathbin\big}
147 \DeclareRobustCommand{\Bigb}{\mathbin\Big}
148 \DeclareRobustCommand{\biggb}{\mathbin\bigg}
149 \DeclareRobustCommand{\Biggb}{\mathbin\Bigg}
150 \DeclareRobustCommand{\bigi}{\mathinner\big}
151 \DeclareRobustCommand{\Bigi}{\mathinner\Big}
152 \DeclareRobustCommand{\biggi}{\mathinner\bigg}
153 \DeclareRobustCommand{\Biggi}{\mathinner\Bigg}
```

Size Adjustment Definitions.   Replicate the amsmath implementation for tracking the vertical size of a default delimiter in the current font. Our `\dlm@size` matches with `\big@size` in amsmath which is 1.2 times the vertical size of '(':

```
154 \newlength\dlm@size
```

The macros `\dlm@size@setchar` and `\dlm@size@scan` scan the font and code point in unicode (xetex/luatex) vs. TEX (etex) engines:

```
155 \ifdefined\Umathcharnumdef
156   \def\dlm@size@setchar{\Umathcharnumdef\dlm@size@char\Umathcodenum`\(\relax}
157   \def\dlm@size@scan#1"#2"#3"#4\relax{\dlm@size@get{#3}{#4}}
158 \else
159   \def\dlm@size@setchar{\mathchardef\dlm@size@char\mathcode`\(\relax}
160   \def\dlm@size@scan#1"#2#3#4\relax{\dlm@size@get{#3}{#4}}
161 \fi
```

The macro `\dlm@size@get` computes 1.2 times the vertical height of '(':

```
162 \def\dlm@size@get#1#2{%
163   \dlm@size\fontcharht\textfont"#1"#2\relax
164   \advance\dlm@size\fontchardp\textfont"#1"#2\relax
165   \global\dlm@size1.2\dlm@size}
```

The macro `\dlm@size@reset` updates `\dlm@size`:

```
166 \newcommand{\dlm@size@reset}{%
167   \dlm@size@setchar\expandafter\dlm@size@scan\meaning\dlm@size@char\relax}
```

We can hook onto the amsmath mechanism if already present:

```
168 \ifdefined\big@size
169 \def\dlm@size{\big@size}
170 \else
```

```
171 \addto@hook\every@math@size{\dlm@size@reset}
172 \fi
```

The macro `\dlm@setvar` enables variable size, the macro `\dlm@setsize` sets a fixed size. Delimiter sizes are implented by `\dlm@big@`. The macro `\kerndelim` adds some kerning proportional to the chosen fixed size of the delimiters (the spacing is merely an approximation, and it does not actually scale for flexible size).

```
173 \newlength\delimsize
174 \newcommand{\dlm@setvar}{\let\dlm@ifvar\@firstoftwo\delimsize1.5\dlm@size}
175 \newcommand{\dlm@resetsize}{\let\dlm@ifvar\@secondoftwo\delimsize0.8\dlm@size}
176 \newcommand{\dlm@setsize}[1]{\let\dlm@ifvar\@secondoftwo
177   \delimsize\dimexpr(#1\dlm@size+\dlm@size)/2\relax}
178 \newcommand{\dlm@addsize}[1]{%
179   \ifx\dlm@ifvar\@firstoftwo\delimsize0.8\dlm@size\fi
180   \ifdim\delimsize=0.8\dlm@size\delimsize0.5\dlm@size\fi
181   \advance\delimsize\dimexpr(#1\dlm@size)/2\relax}
182 \newcommand{\dlm@setabssize}[1]{\let\dlm@ifvar\@secondoftwo
183   \delimsize\glueexpr#1\relax}
184 \newcommand{\dlm@big@}[1]{{\mathchoice
185   {\dlm@big@choice\displaystyle{1}{#1}}%
186   {\dlm@big@choice\textstyle{1}{#1}}%
187   {\dlm@big@choice\scriptstyle{0.7}{#1}}%
188   {\dlm@big@choice\scriptscriptstyle{0.5}{#1}}}}
189 \newcommand{\dlm@big@choice}[3]{\hbox{\dlm@beamerbasecolor@fix%
190   $\m@th\nulldelimiterspace\z@
191   #1\left#3\vcenter to#2\delimsize{}\right.$}}
192 \newcommand{\kerndelim}[1]{\mkern#1\muexpr\ifdim\delimsize>2.5\dlm@size3mu%
193   \else1.2mu*\delimsize/\dlm@size\fi\relax}
```

Math Classes Processing.   Define class selectors for fixed and unadjusted sizes.

```
194 \newcommand{\dlm@big}[2]{\dlm@plain{#1}{\dlm@big@#2}}
195 \newcommand{\dlm@plain}[2]{\dlm@class{#1}{\dlm@phantom{#2}}}
196 \newcommand{\dlm@class}[1]{\csname dlm@class@#1\endcsname}
197 \let\dlm@class@\@firstofone
198 \let\dlm@class@o\mathord
199 \let\dlm@class@c\mathord
200 \let\dlm@class@p\mathpunct
201 \let\dlm@class@i\mathinner
202 \let\dlm@class@b\mathbin
203 \let\dlm@class@m\mathrel
```

Define class selector for variable size. The macro `\dlm@var@null` removes the extra space from variable-size null delimiters (`\left.`, `\middle.` and `\right.` apply the value of `\nulldelimiterspace` only at the end of math processing, even across blocks). The macro `\dlm@var@kern` applies kerning unless in the script styles.

```
204 \newcommand{\dlm@var@null}[1]{\if.#1\kern-\nulldelimiterspace\fi}
205 \newcommand{\dlm@var@kern}[1]{\nonscript\mkern#1}
206 \newcommand{\dlm@var}[1]{\csname dlm@var@#1\endcsname}
207 \newcommand{\dlm@var@}[1]{\dlm@var@null#1\middle#1}
208 \let\dlm@var@o\dlm@var@
209 \let\dlm@var@c\dlm@var@
210 \newcommand{\dlm@var@p}[1]{\dlm@var@#1\dlm@var@kern\thinmuskip}
211 \newcommand{\dlm@var@i}[1]{\dlm@var@kern\thinmuskip
212   \dlm@var@#1\dlm@var@kern\thinmuskip}
213 \newcommand{\dlm@var@b}[1]{\dlm@var@kern\medmuskip
```

```
214    \dlm@var@#1\dlm@var@kern\medmuskip}
215 \newcommand{\dlm@var@m}[1]{\dlm@var@kern\thickmuskip
216    \dlm@var@#1\dlm@var@kern\thickmuskip}
```

Definitions for selecting outer math class \mathopen+\mathclose vs. \mathinner.

```
217 \newcommand{\dlm@open@i}{\mathinner{}\mathclose{}\mathopen}
218 \newcommand{\dlm@close@i}{\mathclose{}\mathopen{}\mathinner}
219 \newcommand{\dlm@inner@oc}{\mathopen{}\mathclose}
220 \newcommand{\dlm@enclose@openclose}{%
221    \let\dlm@open\mathopen\let\dlm@close\mathclose\let\dlm@inner\dlm@inner@oc}
222 \newcommand{\dlm@enclose@inner}{%
223    \let\dlm@open\dlm@open@i\let\dlm@close\dlm@close@i\let\dlm@inner\mathinner}
```

Definitions for selecting inline vs. block insertion.

```
224 \newcommand{\dlm@inline@l}[1]{\dlm@open{#1}}
225 \newcommand{\dlm@inline@r}[1]{\dlm@close{#1}}
226 \newcommand{\dlm@block@l}[1]{\dlm@inner\bgroup\mathopen{#1}}
227 \newcommand{\dlm@block@r}[1]{\mathclose{#1}\egroup}
228 \newcommand{\dlm@enclose@inline}{%
229    \let\dlm@class@l\dlm@inline@l\let\dlm@class@r\dlm@inline@r}
230 \newcommand{\dlm@enclose@block}{%
231    \let\dlm@class@l\dlm@block@l\let\dlm@class@r\dlm@block@r}
```

Opening and closing definitions.

```
232 \newcommand{\dlm@plain@l}[1]{\dlm@class@l{\dlm@phantom{#1}}}
233 \newcommand{\dlm@plain@r}[1]{\dlm@class@r{\dlm@phantom{#1}}}
234 \newcommand{\dlm@big@l}[1]{\dlm@plain@l{\dlm@big@#1}}
235 \newcommand{\dlm@big@r}[1]{\dlm@plain@r{\dlm@big@#1}}
236 \newcommand{\dlm@var@l}[1]{\dlm@inner\bgroup\dlm@var@null#1\left#1}
237 \newcommand{\dlm@var@r}[1]{\right#1\dlm@var@null#1\egroup}
238 \newcommand{\dlm@var@pl}[1]{\dlm@var@l.#1\mathclose{}\mathopen{}}
239 \newcommand{\dlm@var@pr}[1]{\mathclose{}\mathopen{}#1\dlm@var@r.}
```

Delimiter Storage.   Macros for storing and retrieving delimiter types using global registers.

```
240 \newcount\dlm@reg@lvl
241 \dlm@reg@lvl\z@
242 \def\dlm@reg@dot{.}
243 \newcommand{\dlm@reg@set}[1]{\edef\dlm@reg@cur{#1}\ifx\dlm@reg@cur\dlm@reg@dot
244    \edef\dlm@reg@cur{\dlm@reg@dot\the\dlm@reg@lvl}\fi}
245 \newcommand{\dlm@reg@step}[1]{\global\advance\dlm@reg@lvl#1%
246    \ifnum\dlm@reg@lvl<\z@\PackageError{delimset}{register stack exhausted}{}%
247    \dlm@reg@lvl\z@\fi}
248 \newcommand{\dlm@reg@save@init}[1]{%
249    \xdef#1{\delimsize\the\delimsize}\ifx\dlm@inner\mathinner
250    \expandafter\gdef\expandafter#1\expandafter{#1\dlm@enclose@inner}\fi}
251 \newcommand{\dlm@reg@save}[2]{\dlm@reg@set{#1}%
252    \expandafter\dlm@reg@save@init\csname dlm@reg@init@\dlm@reg@cur\endcsname
253    \expandafter\gdef\csname dlm@reg@sel@\dlm@reg@cur\endcsname{#2}}
254 \newcommand{\dlm@reg@init}[1]{\dlm@reg@set{#1}%
255    \csname dlm@reg@init@\dlm@reg@cur\endcsname}
256 \newcommand{\dlm@reg@sel}{\csname dlm@reg@sel@\dlm@reg@cur\endcsname}
```

Flags Processing.   The macro \parsedelimflags parses the optional argument(s) following \delim... commands to adjust the presentation. Parsing is terminated if the

next character in line begins a group ('{') and the delimiter set is composed by executing argument #1. A size flag is stored and parsing continues. A delimiter selector flag immediately displays the desired delimiter by executing argument #2. Unknown flags produce an error message.

```
257 \newcommand{\parsedelimflags}{\dlm@resetsize\let\dlm@phantom\@firstofone
258   \dlm@enclose@openclose\dlm@enclose@inline\dlm@parseflags}
259 \newcommand{\dlm@parseflags}[2]{%
260   \@ifnextchar\bgroup{#1}{\dlm@parseflag{#1}{#2}}}
261 \newcommand{\dlm@parseflag}[3]{\begingroup
262   \ifcsname dlm@parseflag@\string#3\endcsname
263   \def\dlm@do{\csname dlm@parseflag@\string#3\endcsname{#1}{#2}}\else
264   \def\dlm@do{\PackageError{delimset}%
265   {unknown delimiter set flag '\string#3'}{}#1}%
266   \fi\expandafter\endgroup\dlm@do}
267 \newcommand{\dlm@parsedef}[3]{%
268   \expandafter\def\csname dlm@parseflag@\string#1\endcsname ##1##2#2{#3}}
```

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

Flags to adjust size. Note that sizes 0 and 1 amount to 0.96 and 1.2 times the current empty math box height.

```
269 \dlm@parsedef{!}{}{\dlm@addsize{1}\dlm@parseflags{#1}{#2}}
270 \dlm@parsedef{+}{}{\dlm@addsize{1}\dlm@parseflags{#1}{#2}}
271 \dlm@parsedef{0}{}{\dlm@resetsize\dlm@parseflags{#1}{#2}}
272 \dlm@parsedef{1}{}{\dlm@setsize{1}\dlm@parseflags{#1}{#2}}
273 \dlm@parsedef{2}{}{\dlm@setsize{2}\dlm@parseflags{#1}{#2}}
274 \dlm@parsedef{3}{}{\dlm@setsize{3}\dlm@parseflags{#1}{#2}}
275 \dlm@parsedef{4}{}{\dlm@setsize{4}\dlm@parseflags{#1}{#2}}
276 \dlm@parsedef{^}{#3}{\dlm@setsize{#3}\dlm@parseflags{#1}{#2}}
277 \dlm@parsedef{_}{#3}{\dlm@setabssize{#3}\dlm@parseflags{#1}{#2}}
278 \dlm@parsedef{*}{}{\dlm@setvar\dlm@parseflags{#1}{#2}}
```

Flags to select individual delimiters.

```
279 \dlm@parsedef{(}{}{\def\selecteddelim{0}#2}
280 \dlm@parsedef{|}{}{\def\selecteddelim{1}#2}
281 \dlm@parsedef{,}{}{\def\selecteddelim{1}#2}
282 \dlm@parsedef{)}{}{\def\selecteddelim{9}#2}
283 \dlm@parsedef{?}{#3}{\def\selecteddelim{#3}#2}
```

Flags to access registers.

```
284 \dlm@parsedef{[}{}{\dlm@ifvar{%
285   \PackageError{delimset}{cannot use variable size with '['}{}}{%
286   \dlm@reg@step\@ne\dlm@reg@save\dlm@reg@dot{#2}\def\selecteddelim{0}#2}}
287 \dlm@parsedef{]}{}{\dlm@reg@init\dlm@reg@dot\dlm@reg@step\m@ne
288   \def\selecteddelim{9}\dlm@reg@sel}
289 \dlm@parsedef{>}{#3}{\dlm@reg@save{#3}{#2}}
290 \dlm@parsedef{<}{#3}{\dlm@reg@init{#3}\dlm@parseflags
291   {\PackageError{delimset}{must select delimiter}{}}{\dlm@reg@sel}}
```

Further flags.

```
292 \dlm@parsedef{.}{}{#1}
293 \dlm@parsedef{:}{}{\dlm@enclose@block\dlm@parseflags{#1}{#2}}
294 \dlm@parsedef{'}{}{\dlm@enclose@openclose\dlm@parseflags{#1}{#2}}
295 \dlm@parsedef{"}{}{\dlm@enclose@inner\dlm@parseflags{#1}{#2}}
296 \dlm@parsedef{-}{}{\let\dlm@phantom\phantom\dlm@parseflags{#1}{#2}}
```

Choice Processing. Delimiter definitions may accommodate different representations via optional argument. The following commands define the options and select the chosen one.

```
297 \newcommand{\dlm@choicedef}[3]{%
298   \expandafter\def\csname dlm@#1@\string#2@\endcsname{#3}}
299 \newcommand{\dlm@choicelet}[3]{%
300   \expandafter\let\csname dlm@#1@\string#2@\expandafter\endcsname
301   \csname dlm@#1@\string#3@\endcsname}
302 \newcommand{\dlm@choice}[1]{%
303   \@ifnextchar[{\dlm@choice@opt{#1}}{\csname dlm@#1@@@\endcsname}}
304 \def\dlm@choice@opt#1[#2]{\csname dlm@#1@%
305   \ifcsname dlm@#1@\string#2@\endcsname\string#2@\else @@\fi\endcsname}
306 \newcommand{\dlm@choice@arg}[2]{\begingroup
307   \expandafter\let\expandafter\dlm@do\csname dlm@#1@\string#2@\endcsname
308   \ifx\dlm@do\relax\def\dlm@do{\csname dlm@#1@@@\endcsname{#2}}\fi
309   \expandafter\endgroup\dlm@do}
```

Delimiter Display. The command \selectdelim reproduces the delimiter in argument #2 using the math class given in argument #1 and the previously stored size. If the class identifier is '.', just return the delimiter argument as is. The commands \selectdeliml and \selectdelimr implement the left and right delimiter classes, respectively. The command \doubledelim displays a compound of two identical delimiters with optional kerning adjustment.

```
310 \newcommand{\selectdelim}[1][]{\begingroup\def\dlm@do{\@ifstar{\dlm@plain{#1}}%
311   {\dlm@ifvar{\dlm@var{#1}}{\dlm@big{#1}}}}%
312   \if.#1\let\dlm@do\@empty\fi\expandafter\endgroup\dlm@do}
313 \newcommand{\selectdeliml}{\@ifstar
314   {\dlm@ifvar\dlm@var@pl\dlm@plain@l}{\dlm@ifvar\dlm@var@l\dlm@big@l}}
315 \newcommand{\selectdelimr}{\@ifstar
316   {\dlm@ifvar\dlm@var@pr\dlm@plain@r}{\dlm@ifvar\dlm@var@r\dlm@big@r}}
317 \newcommand{\doubledelim}[2][-2]{%
318   \selectdelim{#2}\kerndelim{#1}\selectdelim{#2}}
```

Declaration of New Delimiter Commands. The macro \DeclareMathDelimiterSet declares a new set of delimiters as the macro '*name*'. This macro checks for optional flags and stores the desired size. It then passes on to a second macro '\dlm@dcl@*name*', which takes the actual code. If the flags select an individual delimiter, the macro '\dlm@sel@*name*' is called instead. The latter macro should be defined via \DeclareMathDelimiterSel such that it produces the desired delimiter number \selecteddelim.

```
319 \newcommand{\DeclareMathDelimiterSet}[1]{\expandafter\dlm@declare
320   \csname dlm@dcl@\expandafter\@gobble\string#1\endcsname{#1}}
321 \def\dlm@declare#1#2{\expandafter\dlm@declare@
322   \csname dlm@sel@\expandafter\@gobble\string#2\endcsname{#1}{#2}}
323 \def\dlm@declare@#1#2#3{\newcommand{#3}{\parsedelimflags{#2}{#1}}%
324   \providecommand{#1}{}\newcommand{#2}}
325 \newcommand{\DeclareMathDelimiterSel}[3]{\expandafter\def
326   \csname dlm@sel@\expandafter\@gobble\string#1\endcsname
327   {\ifcase\selecteddelim#2\else#3\fi}}
```

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.

```
328 \newcommand{\delim}[2]{%
329   \parsedelimflags{\dlm@dcl@delim{#1}{#2}}{\dlm@sel@delim{#1}{#2}}}
330 \newcommand{\dlm@dcl@delim}[3]{%
331   \selectdeliml#1{#3}\selectdelimr#2}
332 \newcommand{\dlm@sel@delim}[2]{\ifcase\selecteddelim\selectdeliml#1%
333   \else\selectdelimr#2\fi}
```

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

```
334 \newcommand{\delimpair}[3]{%
335   \parsedelimflags{\dlm@dcl@delimpair{#1}{#2}{#3}}%
336   {\dlm@sel@delimpair{#1}{#2}{#3}}}
337 \newcommand{\dlm@dcl@delimpair}[5]{%
338   \selectdeliml#1{#4}\selectdelim#2{#5}\selectdelimr#3}
339 \newcommand{\dlm@sel@delimpair}[3]{\ifcase\selecteddelim\selectdeliml#1%
340   \or\selectdelim#2\else\selectdelimr#3\fi}
```

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

```
341 \newcommand{\delimtriple}[4]{%
342   \parsedelimflags{\dlm@dcl@delimtriple{#1}{#2}{#3}{#4}}%
343   {\dlm@sel@delimtriple{#1}{#2}{#3}{#4}}}
344 \newcommand{\dlm@dcl@delimtriple}[7]{%
345   \selectdeliml#1{#5}\selectdelim#2{#6}\selectdelim#3{#7}\selectdelimr#4}
346 \newcommand{\dlm@sel@delimtriple}[4]{\ifcase\selecteddelim\selectdeliml#1%
347   \or\selectdelim#2\or\selectdelim#3\else\selectdelimr#4\fi}
```

`\usedelim` retrieves a stored delimiter type. The arguments ']', '(', '|', ',', '?n', ')' immediately select the delimiter using the stack register. Otherwise the desired register must be followed by the delimiter selection flag.

```
348 \dlm@choicedef{usedelim}{]}{\parsedelimflags{}{}]}
349 \dlm@choicedef{usedelim}{(}{\parsedelimflags{}{}<.(}
350 \dlm@choicedef{usedelim}{|}{\parsedelimflags{}{}<.|}
351 \dlm@choicedef{usedelim}{,}{\parsedelimflags{}{}<.,}
352 \dlm@choicedef{usedelim}{)}{\parsedelimflags{}{}<.)}
353 \dlm@choicedef{usedelim}{?}{\parsedelimflags{}{}<.?}
354 \dlm@choicedef{usedelim}{@}{\parsedelimflags{}{}<}
355 \newcommand{\usedelim}{\dlm@choice@arg{usedelim}}
```

Vertical Bar Definitions. Define basic macros for left/right vertical bars and overwrite them later if defined otherwise.

```
356 \def\dlm@lvert{\vert}\let\dlm@rvert\dlm@lvert
357 \def\dlm@lVert{\Vert}\let\dlm@rVert\dlm@lVert
358 \AddToHook{begindocument/end}{%
359   \ifdefined\lvert\def\dlm@lvert{\lvert}\fi
360   \ifdefined\rvert\def\dlm@rvert{\rvert}\fi
361   \ifdefined\lVert\def\dlm@lVert{\lVert}\fi
362   \ifdefined\rVert\def\dlm@rVert{\rVert}\fi
363 }
```

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`.

```
364 \ifdlm@std
365 \dlm@choicedef{brk}{r}{\delim()}
366 \dlm@choicedef{brk}{s}{\delim[]}
367 \dlm@choicedef{brk}{c}{\delim\{\}}
368 \dlm@choicedef{brk}{a}{\delim<>}
369 \dlm@choicedef{brk}{v}{\delim\dlm@lvert\dlm@rvert}
370 \dlm@choicedef{brk}{R}{\delim{*{\doubledelim(}}{*{\doubledelim)}}}
371 \dlm@choicedef{brk}{S}{%
372   \delim{*{\doubledelim[-1.75][}}{*{\doubledelim[-1.75]]}}}
373 \ifdefined\llbracket\dlm@choicedef{brk}{S}{\delim\llbracket\rrbracket}\fi
374 \dlm@choicedef{brk}{C}{%
375   \delim{*{\doubledelim[-2.5]\{}}{*{\doubledelim[-2.5]\}}}}
376 \dlm@choicedef{brk}{A}{\delim{*{\doubledelim<}}{*{\doubledelim>}}}
377 \dlm@choicedef{brk}{V}{\delim\dlm@lVert\dlm@rVert}
378 \dlm@choicelet{brk}{@}{r}
379 \providecommand{\brk}{\dlm@choice{brk}}
380 \dlm@choicedef{eval}{v}{\delim.\dlm@rvert}
381 \dlm@choicedef{eval}{s}{\delim[]}
382 \dlm@choicelet{eval}{@}{v}
383 \providecommand{\eval}{\dlm@choice{eval}}
384 \providecommand{\abs}{\delim\dlm@lvert\dlm@rvert}
385 \providecommand{\norm}{\delim\dlm@lVert\dlm@rVert}
386 \fi
```

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

```
387 \ifdlm@ext
388 \dlm@choicedef{pair}{r,}{\delimpair({[p]*,})}
389 \dlm@choicedef{pair}{r;}{\delimpair({[p]*;})}
390 \dlm@choicedef{pair}{r|}{\delimpair(|)}
391 \dlm@choicedef{pair}{s,}{\delimpair[{[p]*,}]}
392 \dlm@choicedef{pair}{s;}{\delimpair[{[p]*;}]}
393 \dlm@choicedef{pair}{s|}{\delimpair[|]}
394 \dlm@choicedef{pair}{c,}{\delimpair\{{[p]*,}\}}
395 \dlm@choicedef{pair}{c;}{\delimpair\{{[p]*;}\}}
396 \dlm@choicedef{pair}{c|}{\delimpair\{|\}}
397 \dlm@choicedef{pair}{a,}{\delimpair<{[p]*,}>}
398 \dlm@choicedef{pair}{a;}{\delimpair<{[p]*;}>}
399 \dlm@choicedef{pair}{a|}{\delimpair<|>}
400 \dlm@choicedef{pair}{v,}{\delimpair\dlm@lvert{[p]*,}\dlm@rvert}
401 \dlm@choicedef{pair}{v;}{\delimpair\dlm@lvert{[p]*;}\dlm@rvert}
402 \dlm@choicedef{pair}{v|}{\delimpair\dlm@lvert|\dlm@rvert}
403 \dlm@choicedef{pair}{R,}{\delimpair{*{\doubledelim(}}{[p]*,}{*{\doubledelim)}}}
404 \dlm@choicedef{pair}{R;}{\delimpair{*{\doubledelim(}}{[p]*;}{*{\doubledelim)}}}
405 \dlm@choicedef{pair}{R|}{\delimpair{*{\doubledelim(}}\Vert{*{\doubledelim)}}}
406 \dlm@choicedef{pair}{S,}{%
407   \delimpair{*{\doubledelim[-1.75][}}{[p]*,}{*{\doubledelim[-1.75]]}}}
408 \dlm@choicedef{pair}{S;}{%
409   \delimpair{*{\doubledelim[-1.75][}}{[p]*;}{*{\doubledelim[-1.75]]}}}
410 \dlm@choicedef{pair}{S|}{%
411   \delimpair{*{\doubledelim[-1.75][}}\Vert{*{\doubledelim[-1.75]]}}}
412 \ifdefined\llbracket
413   \dlm@choicedef{pair}{S,}{\delimpair\llbracket{[p]*,}\rrbracket}
414   \dlm@choicedef{pair}{S;}{\delimpair\llbracket{[p]*;}\rrbracket}
415   \dlm@choicedef{pair}{S|}{\delimpair\llbracket\Vert\rrbracket}
416 \fi
417 \dlm@choicedef{pair}{C,}{%
```

```
418    \delimpair{*{\doubledelim[-2.5]\{}}{[p]*,}{*{\doubledelim[-2.5]\}}}}
419 \dlm@choicedef{pair}{C;}{%
420    \delimpair{*{\doubledelim[-2.5]\{}}{[p]*;}{*{\doubledelim[-2.5]\}}}}
421 \dlm@choicedef{pair}{C|}{%
422    \delimpair{*{\doubledelim[-2.5]\{}}\Vert{*{\doubledelim[-2.5]\}}}}
423 \dlm@choicedef{pair}{A,}{\delimpair{*{\doubledelim<}}{[p]*,}{*{\doubledelim>}}}
424 \dlm@choicedef{pair}{A;}{\delimpair{*{\doubledelim<}}{[p]*;}{*{\doubledelim>}}}
425 \dlm@choicedef{pair}{A|}{\delimpair{*{\doubledelim<}}\Vert{*{\doubledelim>}}}
426 \dlm@choicedef{pair}{V,}{\delimpair\dlm@lVert{[p]*,}\dlm@rVert}
427 \dlm@choicedef{pair}{V;}{\delimpair\dlm@lVert{[p]*;}\dlm@rVert}
428 \dlm@choicedef{pair}{V|}{\delimpair\dlm@lVert\Vert\dlm@rVert}
429 \dlm@choicelet{pair}{r}{r,}
430 \dlm@choicelet{pair}{s}{s,}
431 \dlm@choicelet{pair}{c}{c,}
432 \dlm@choicelet{pair}{a}{a,}
433 \dlm@choicelet{pair}{v}{v,}
434 \dlm@choicelet{pair}{R}{R,}
435 \dlm@choicelet{pair}{S}{S,}
436 \dlm@choicelet{pair}{C}{C,}
437 \dlm@choicelet{pair}{A}{A,}
438 \dlm@choicelet{pair}{V}{V,}
439 \dlm@choicelet{pair}{,}{r,}
440 \dlm@choicelet{pair}{;}{r;}
441 \dlm@choicelet{pair}{|}{r|}
442 \dlm@choicelet{pair}{@}{r}
443 \providecommand{\pair}{\dlm@choice{pair}}
444 \providecommand{\set}{\delim\{\}}
445 \providecommand{\setcond}{\delimpair\{|\}}
446 \dlm@choicedef{intv}{c}{\delimpair[{[p]*,}]}
447 \dlm@choicedef{intv}{l}{\delimpair]{[p]*,}]}
448 \dlm@choicedef{intv}{r}{\delimpair[{[p]*,}[}
449 \dlm@choicedef{intv}{o}{\delimpair]{[p]*,}[}
450 \dlm@choicelet{intv}{@}{c}
451 \providecommand{\intv}{\dlm@choice{intv}}
452 \providecommand{\avg}{\delim<>}
453 \providecommand{\corr}{\delim<>}
454 \providecommand{\comm}{\delimpair[{[p]*,}]}
455 \providecommand{\acomm}{\delimpair\{{[p]*,}\}}
456 \providecommand{\bra}{\delim<\dlm@rvert}
457 \providecommand{\ket}{\delim\dlm@lvert>}
458 \providecommand{\braket}{\delimpair<|>}
459 \dlm@choicedef{lfrac}{.}{\delimpair./.}
460 \dlm@choicedef{lfrac}{r}{\delimpair(/)}
461 \dlm@choicedef{lfrac}{s}{\delimpair[/]}
462 \dlm@choicedef{lfrac}{c}{\delimpair\{/\}}
463 \dlm@choicedef{lfrac}{a}{\delimpair</>}
464 \dlm@choicelet{lfrac}{}{.}
465 \dlm@choicelet{lfrac}{@}{.}
466 \providecommand{\lfrac}{\dlm@choice{lfrac}}
467 \fi
```