The jmsdelim package

Jonathan Sterling

September 14, 2019

1 Overview

Sizing delimiters using `\left` and `\right` should be outlawed! The results are nearly always unaesthetic, primarily because the correct size of a mathematical delimiter is a typesetting consideration which does *not* emanate from the physical size of the interior.

Correctly sizing delimiters is very difficult, particularly in well-architected documents: a correctly engineered mathematical document will include macros for all operations, and these macros necessarily will include delimiters (such as parentheses). However, the correct size for the delimiter cannot be chosen ahead of time, because it will depend on the arguments; two options are available:

1. Provide optional arguments to each notation macro for choosing delimiter sizes. This is nearly intractable to do in practice.
2. Ignore delimiter sizes.

With jmsdelim we offer an alternative: the correct delimiter sizes can be set at the *leaf nodes* of a mathematical expression, and magically bubble upward through the delimiters.

1.1 Basic commands

\mindelim `\mindelim{⟨intexpr$_{min}$⟩}`

This sets the minimum delimiter size to ⟨intexpr$_{min}$⟩ at a given point; delimiter sizes are represented as natural numbers, with `0` the smallest size. `\mindelim` is the work-horse of jmsdelim; let us consider an example of what one might do prior to adopting jmsdelim. Suppose we have defined a macro `\Psh` for the free co-completion, following the notation of the French school, and we wish to parenthesize an instance of it:

$$\mathrm{Hom}_{\mathbf{Cat}}(1, \widehat{\mathbb{C}})$$

```
\NewDocumentCommand\Cat{}{\mathbf{Cat}}
\NewDocumentCommand\Psh{m}{\widehat{#1}}
\NewDocumentCommand\Hom{mmm}{
  \operatorname{Hom}_{#1}(#2,#3)
}
\[ \Hom{\Cat}{1}{\Psh{\mathbb{C}}} \]
```

One might have tried to get a better result by using `\left` and `\right`:

$$\mathrm{Hom}_{\mathbf{Cat}}\left(1, \widehat{\mathbb{C}}\right)$$

```
\NewDocumentCommand\Cat{}{\mathbf{Cat}}
\NewDocumentCommand\Psh{m}{\widehat{#1}}
\NewDocumentCommand\Hom{mmm}{
  \operatorname{Hom}_{#1}\left(#2,#3\right)
}
\[ \Hom{\Cat}{1}{\Psh{\mathbb{C}}} \]
```

The above is hugely worse: the height of the hat does not in any way determine the correct size for the delimiter! The solution using jmsdelim is quite simple, however: first, we change `\Hom` to call `\parens`, and then we use `\mindelim` within the `\Psh` notation.

$$\mathrm{Hom}_{\mathbf{Cat}}(1, \widehat{\mathbb{C}})$$

```
\NewDocumentCommand\Cat{}{\mathbf{Cat}}
\NewDocumentCommand\Psh{m}{\mindelim{1}\widehat{#1}}
\NewDocumentCommand\Hom{mmm}{
  \operatorname{Hom}_{#1}\parens{#2,#3}
}
\[ \Hom{\Cat}{1}{\Psh{\mathbb{C}}} \]
```

1.1.1 Built-in delimiter commands

All commands in this section produce delimiters; they each have a similar structure:

`\command[⟨options⟩]⟨*⟩...{⟨body⟩}`

The optional ⟨options⟩ argument locally sets package options (Section 1.2) for the scope of the ⟨body⟩ argument. The starred variants set the minimum delimiter size *outside* the current delimiter to be strictly larger than the current size. For instance:

$$\{\{\{\{x\}\}\}\}$$
$$\left\{\left\{\left\{\{x\}\right\}\right\}\right\}$$
$$\{\{\{\{x\}\}\}\}$$

```
\[ \braces{\braces{\braces{\braces{x}}}}    \]
\[ \braces{\braces*{\braces*{\braces*{x}}}} \]
\[ \braces{\braces*{\braces{\braces{x}}}}   \]
```

`\delim` `\delim[⟨options⟩]⟨*⟩{⟨left⟩}{⟨right⟩}{⟨body⟩}`

Surrounds ⟨body⟩ by ⟨left⟩ and ⟨right⟩ respectively.

`\parens` `\parens[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in parentheses.

`\brackets` `\parens[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in square brackets.

`\bbrackets` `\parens[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in Scott brackets; requires that `\llbracket` and `\rrbracket` are defined, for instance by stmaryrd.

`\angles` `\angles[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in angle brackets; requires that `\langle` and `\rangle` are defined.

`\aangles` `\aangles[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in double angle brackets; requires that `\llangle` and `\rrangle` are defined, for instance by MnSymbol.

`\verts` `\verts[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in vertical bars.

`\vverts` `\verts[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in double vertical bars.

`\braces` `\verts[⟨options⟩]⟨*⟩{⟨body⟩}`

Surrounds ⟨body⟩ in curly braces.

1.2 Configuration and options

`\jmsdelimsetup` `\jmsdelimsetup{⟨options⟩}`

jmsdelim can be customized along a few axes.

`delimiters` The option `delimiters` is a comma-separated list which contains a list of sizing commands for delimiters, from smallest to largest.

`formatters` The option `formatters` is a comma-separated list which contains a list of formatting commands for delimiters, from outermost to innermost; the formatters are cycled repeatedly as the depth of delimiters exceeds the provided formatters. The `formatters` option can, for instance, be used to implement "rainbow delimiters":

```
\jmsdelimsetup{
  formatters =
    {\color{blue},
     \color{red},
     \color{green},
     \color{violet}}
}
\[ \verts*{\verts*{\verts*{\verts*{x}}}} \]
```

1.3 Advanced commands

`\delimsep` `\delimsep{⟨sep⟩}`

This command can be used to insert a separator in a multi-place operation; this can be used to notate cuts in sequent calculus as in Munch-Maccagnoni [Mun13; Mun17], for instance:

$$\left\langle t \,\middle\|\, \tilde{\mu}x.\left\langle \mu\alpha.\langle u \parallel e\rangle \,\middle\|\, e'\right\rangle\right\rangle$$

```
\NewDocumentCommand\Cut{mm}{
  \angles*{#1 \mathrel{\delimsep{\Vert}} #2}
}
\NewDocumentCommand\Mu{mm}{\mu #1.#2}
\NewDocumentCommand\MuTilde{mm}{\tilde\mu #1.#2}
\[
  \Cut{t}{
    \MuTilde{x}{
      \Cut{\Mu{\alpha}{\Cut{u}{e}}}{e'}
    }
  }
\]
```

1.4 Extended example from perfectcut

The following states the idempotency of an adjunction:

$$\left\langle t \,\middle\|\, \tilde{\mu}x.\left\langle \mu\alpha.\langle u \parallel e\rangle \,\middle\|\, e'\right\rangle\right\rangle = \left\langle \mu\alpha.\left\langle t \,\middle\|\, \tilde{\mu}x.\langle u \parallel e\rangle\right\rangle \,\middle\|\, e'\right\rangle$$

The following states the commutativity of a strong monad:

$$\left\langle t \,\middle\|\, \tilde{\mu}x.\left\langle u \,\middle\|\, \tilde{\mu}y.\langle v \parallel e\rangle\right\rangle\right\rangle = \left\langle u \,\middle\|\, \tilde{\mu}y.\left\langle t \,\middle\|\, \tilde{\mu}x.\langle v \parallel e\rangle\right\rangle\right\rangle$$

Using \underline to mark redexes:

$$
\begin{aligned}
&\delta(V, x.y, x.y)\\
&= \mu\star.\left\langle V \,\middle\|\, \left[\tilde{\mu}x.\underline{\langle y \parallel \star\rangle} \,\middle|\, \tilde{\mu}x.\underline{\langle y \parallel \star\rangle}\right]\right\rangle\\
&= \mu\star.\left\langle V \,\middle\|\, \underline{\left[\tilde{\mu}x.\left\langle \iota_1(x) \,\middle\|\, \tilde{\mu}z.\langle y \parallel \star\rangle\right\rangle \,\middle|\, \tilde{\mu}x.\left\langle \iota_2(x) \,\middle\|\, \tilde{\mu}z.\langle y \parallel \star\rangle\right\rangle\right]}\right\rangle\\
&= \mu\star.\left\langle V \,\middle\|\, \underline{\tilde{\mu}z.}\langle y \parallel \star\rangle\right\rangle\\
&= \mu\star.\langle y \parallel \star\rangle = y
\end{aligned}
$$

```
\NewDocumentCommand\Cut{mm}{
  \angles*{#1 \mathrel{\delimsep{\Vert}} #2}
}
\NewDocumentCommand\mt{}{\tilde\mu}
\NewDocumentCommand\Case{mm}{
  \brackets{#1 \mathrel{\delimsep{\vert}} #2}
}
The following states the idempotency of an adjunction:
\[
\Cut{t}{\mt x.\Cut{\mu\alpha.\Cut{u}{e}}{e'}}=\Cut{\mu\alpha.\Cut{t}{\mt x.\Cut{u}{e}}}{e'}
\]

The following states the commutativity of a strong monad:
\[
\Cut t{\mt x.\Cut u{\mt y.\Cut ve}}=\Cut u{\mt y.\Cut t{\mt x.\Cut ve}}
\]
Using \cs{underline} to mark redexes:
\begin{align*}
  & \delta(V,x.y,x.y)\\
  & = \mu{\star}.
      \Cut{V}{
        \Case{
          \mt x.\underline{\Cut y{\star}}
        }{
          \mt x.\underline{\Cut y{\star}}
        }
      }\\
  & = \mu{\star}.
      \Cut{V}{
        \underline{
          \Case{
            \mt x.\Cut{\iota_{1}(x)}{\mt z.\Cut{y}{\star}}
          }{
            \mt x.\Cut{\iota_{2}(x)}{\mt z.\Cut{y}{\star}}
          }
        }
      }\\
  & = \mu{\star}.\Cut{V}{\underline{\mt z.}\Cut{y}{\star}}\\
  & = \mu{\star}.\Cut{y}{\star}=y
\end{align*}
```

1.5 Internals

The internals of jmsdelim are implemented in expl3.

jmsdelim_make:nnnn `\jmsdelim_make:nnnn {⟨boolexpr$_{bump}$⟩} {⟨left⟩} {⟨right⟩} {⟨body⟩}`

This routine renders ⟨body⟩ into a scratch hbox to determine the minimum size of delimiter that can surround it. Then, it renders it again for real, delimiting it accordingly by ⟨left⟩ and ⟨right⟩ respectively. If ⟨boolexpr$_{bump}$⟩ is true, then it will increase the minimum delimiter size outside the current scope.

2 jmsdelim implementation

```
⟨*package⟩
\RequirePackage{expl3}
\RequirePackage{l3keys2e}
\RequirePackage{xparse}
\ProvidesExplPackage {jmsdelim} {2019/09/14} {0.1}
  {Compositional delimiter sizing}
⟨@@=jmsdelim⟩
```

We first declare the options for the jmsdelim module, together with their default valeus.

```
\keys_define:nn { jmsdelim } {
  delimiters .clist_set:N = \l_jmsdelim_size_cmds,
  formatters .clist_set:N = \l_jmsdelim_fmt_cmds,
}
%
\keys_set:nn { jmsdelim } {
  delimiters = {{},\big,\Big,\bigg,\Bigg},
  formatters = {{}}
}
\int_new:N \l_jmsdelim_depth
```

__jmsdelim_fmt_delim:n

```
\cs_new:Npn \__jmsdelim_fmt_delim:n #1 {
  \clist_item:Nn \l_jmsdelim_fmt_cmds {
    \int_mod:nn
      { \int_max:nn \l_jmsdelim_depth 0 }
      { \clist_count:N \l_jmsdelim_fmt_cmds }
    + 1
  }
  \clist_item:Nn \l_jmsdelim_size_cmds {
    \int_min:nn
      { \int_max:nn {#1 + 1} {1} }
      { \clist_count:N \l_jmsdelim_size_cmds }
  }
}
```

(*End definition for* `\__jmsdelim_fmt_delim:n`.)

```
\int_new:N \l_jmsdelim_current_delim_size
\int_new:N \g_jmsdelim_min_delim_size
\bool_new:N \l_jmsdelim_counting_mode
```

__jmsdelim_draw:nnn

```
\cs_new:Npn \__jmsdelim_draw:nnn #1 #2 #3 {
  \bool_if:nT \l_jmsdelim_counting_mode {
    \msg_fatal:nn {jmsdelim} {
      \__jmsdelim_draw:nnn called during counting mode. this is a bug
    }
  }
%
  \group_begin:
    \int_set:Nn \l_jmsdelim_current_delim_size \g_jmsdelim_min_delim_size
    \mathopen{}
    {\__jmsdelim_fmt_delim:n \l_jmsdelim_current_delim_size #1}
    { #3 }
    {\__jmsdelim_fmt_delim:n \l_jmsdelim_current_delim_size #2}
    \mathclose{}
  \group_end:
}
```

(*End definition for* __jmsdelim_draw:nnn.)

__jmsdelim_int_gset_monotone:Nn

```
\cs_new:Npn \__jmsdelim_int_gset_monotone:Nn #1 #2 {
  \int_gset:Nn #1 {
    \int_max:nn {#1} {#2}
  }
}
```

(*End definition for* __jmsdelim_int_gset_monotone:Nn.)

__jmsdelim_set_min:n

```
\cs_new:Npn \__jmsdelim_set_min:n #1 {
  \bool_if:nT \l_jmsdelim_counting_mode {
    \__jmsdelim_int_gset_monotone:Nn \g_jmsdelim_min_delim_size {#1}
  }
}
```

(*End definition for* __jmsdelim_set_min:n.)

```
\bool_new:N \g_jmsdelim_should_bump
```

\jmsdelim_make:nnnn

```
\cs_new:Npn \jmsdelim_make:nnnn #1 #2 #3 #4 {
  \bool_if:nTF \l_jmsdelim_counting_mode {
```

First, we recall whether we have already received a *bump* instruction at the current level.

```
    \bool_set:Nn \l_tmpa_bool \g_jmsdelim_should_bump
    \hbox_set:Nn \l_tmpa_box { $#4$ }
```

If typesetting #4 inside the scratch box resulted in the first *bump* instruction, then we will increase the minimum delimiter size. Otherwise, we know that this is either unnecessary or has already happened.

```
    \bool_if:nT {\g_jmsdelim_should_bump && ! \l_tmpa_bool } {
      \int_gincr:N \g_jmsdelim_min_delim_size
    }
    \bool_gset:Nn \g_jmsdelim_should_bump {#1}
  }{
    \group_begin:
      \bool_set:Nn \l_jmsdelim_counting_mode \c_true_bool
      \bool_gset:Nn \g_jmsdelim_should_bump \c_false_bool
      \int_gset:Nn \g_jmsdelim_min_delim_size {0}
      \jmsdelim_make:nnnn {#1} {#2} {#3} {#4}
    \group_end:
%
    \__jmsdelim_draw:nnn {#2} {#3} {
      \int_incr:N \l_jmsdelim_depth
      #4
    }
  }
}
```

(*End definition for* `\jmsdelim_make:nnnn`. *This function is documented on page 6.*)

`\jmsdelimsetup`

```
\NewDocumentCommand\jmsdelimsetup{+m}{
  \keys_set:nn {jmsdelim} {#1}
}
```

(*End definition for* `\jmsdelimsetup`. *This function is documented on page 3.*)

`\mindelim`

```
\NewDocumentCommand\mindelim{m}{
  \__jmsdelim_set_min:n {#1}
}
```

(*End definition for* `\mindelim`. *This function is documented on page 2.*)

`\delimsep`

```
\NewDocumentCommand\delimsep{m}{
  {\__jmsdelim_fmt_delim:n {\l_jmsdelim_current_delim_size} {#1}}
}
```

(*End definition for* `\delimsep`. *This function is documented on page 4.*)

`\delim`

```
\NewDocumentCommand\delim{+O{}smmm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} {#3} {#4} {#5}
}
```

(*End definition for* \delim. *This function is documented on page 3.*)

\parens

```
\NewDocumentCommand\parens{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} () {#3}
}
```

(*End definition for* \parens. *This function is documented on page 3.*)

\brackets

```
\NewDocumentCommand\brackets{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} [] {#3}
}
```

(*End definition for* \brackets. *This function is documented on page 3.*)

\bbrackets

```
\NewDocumentCommand\bbrackets{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \llbracket\rrbracket {#3}
}
```

(*End definition for* \bbrackets. *This function is documented on page 3.*)

\angles

```
\NewDocumentCommand\angles{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \langle\rangle {#3}
}
```

(*End definition for* \angles. *This function is documented on page 3.*)

\aangles

```
\NewDocumentCommand\aangles{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \llangle\rrangle {#3}
}
```

(*End definition for* \aangles. *This function is documented on page 3.*)

\verts

```
\NewDocumentCommand\verts{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \lvert\rvert {#3}
}
```

(*End definition for* `\verts`. *This function is documented on page 3.*)

`\vverts`

```
\NewDocumentCommand\vverts{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \lVert\rVert {#3}
}
```

(*End definition for* `\vverts`. *This function is documented on page 3.*)

`\braces`

```
\NewDocumentCommand\braces{+O{}sm}{
  \keys_set:nn {jmsdelim} {#1}
  \jmsdelim_make:nnnn {#2} \{\} {#3}
}
```

(*End definition for* `\braces`. *This function is documented on page 3.*)

```
\ProcessKeysPackageOptions {jmsdelim}
```

References

[Mun13] Guillaume Munch-Maccagnoni. "Syntax and Models of a non-Associative Composition of Programs and Proofs". PhD thesis. Univ. Paris Diderot, 2013 (cit. on p. 4).

[Mun17] Guillaume Munch-Maccagnoni. `perfectcut` – *Nested delimiters that consistently grow regardless of the contents*. Sept. 3, 2017. URL: https://www.ctan.org/pkg/perfectcut (cit. on p. 4).