The abraces package

Asymmetric or arbitrary braces

Werner Grundlingh latex.abraces@gmail.com

Version 2.1

November 6, 2022

1 Introduction

The abraces package provides a character key-driven interface to supplement new constructions of the traditional \overbrace and \underbrace pairs in an asymmetric or arbitrary way.

2 Basic user interface

This package defines two counterparts to the existing braces:

\aoverbrace[⟨*brace spec*⟩]{⟨*stuff*⟩}[⟨*script spec*⟩]^{⟨*upper script*⟩}_{⟨*lower script*⟩}

\aunderbrace[⟨*brace spec*⟩]{⟨*stuff*⟩}[⟨*script spec*⟩]^{⟨*upper script*⟩}_{⟨*lower script*⟩}

Note that both the ⟨*brace spec*⟩ and ⟨*script spec*⟩ arguments are optional, as well as the use of an ⟨*upper script*⟩ and ⟨*lower script*⟩. As such, in its most basic form, \aoverbrace{⟨*stuff*⟩} (and \aunderbrace{⟨*stuff*⟩}) would be similar to the traditional \overbrace{⟨*stuff*⟩} (and \underbrace{⟨*stuff*⟩}). However, if you specify a ⟨*brace spec*⟩ – a construction pattern based on the elements in Table 1 – you could adjust the shape of the brace in an arbitrary way. These definitions are robust.

The ⟨*brace spec*⟩ interface is based on a ratio-principle, allowing one to put a larger share of "filler" (the horizontal rule) at any location within the brace construction. The traditional \overbrace and \underbrace pairs have a 1:1 share between the left and right side (either side of the tip/cusp of the brace), thereby forcing the tip/cust to be placed directly in the center horizontally. With abraces, using a 1:2 ratio would place the brace cusp one third (from the left) into the brace. Similary a 3:2 ratio would place the cusp 40% (or two fifths) from the right edge of the brace. The same holds for elements specified within ⟨*script spec*⟩, except these are used to alter the location of the scrips. For more detail, see section 3 Advanced uses.

Other, more complex constructions are possible by mixing the elements presented in Table 1. See section 4 Examples for a showcase of uses.

\newbracespec{⟨*char*⟩}{⟨*brace spec*⟩}

This allows the user to define a new brace specification ⟨*char*⟩ that results in the (possibly complex) construction ⟨*brace spec*⟩. Note that ⟨*char*⟩ should be different from any already used (see Table 1). The usage is similar to that of a \newcolumntype construction provided by the array package.

⟨*spec*⟩ character	Output
`l`	⎰ (left upward brace)
`L`	⎱ (left downward brace)
`r`	⎱ (right upward brace)
`R`	⎰ (right downward brace)
`U`	⏜ (upward cusp)
`D`	⏝ (downward cusp)
`,[`⟨*len*⟩`]`	⎪ (downward end with optional ⟨*len*⟩gth control)
`'[`⟨*len*⟩`]`	⎪ (upward end with optional ⟨*len*⟩gth control)
`0`	(single) Empty fill
`1, ..., 9`	Copies of regular fill ———
`@{`⟨*stuff*⟩`}`	Places ⟨*stuff*⟩ into brace
`!{`⟨*len*⟩`}`	Regular fill of length ⟨*len*⟩
`*{`⟨*num*⟩`}{`⟨*stuff*⟩`}`	Repeat ⟨*stuff*⟩ a total of ⟨*num*⟩ times

Table 1: Character specifications within ⟨*brace spec*⟩ used to construct braces.

By default, any scripts will be placed at the cusp (`D` when providing a ⟨*lower script*⟩ via `_` or `U` when providing an ⟨*upper script*⟩ via `^`) of the brace. If you use more than one cusp within your ⟨*brace spec*⟩, you can separate scripts using `&` which will place them above/below subsequent brace cusps, similar to separating columns within a `tabular`.

`\bracecolor{`⟨*spec*⟩`}`

If you're interested in using any form of colour, `\bracecolor` will allow you to change the brace colour via an `@`-insertion (for example, `@{\bracecolor{red}}` would yield a red brace from that point onward). Regular script colouring can still be achieved using `\color` or `\textcolor`. The motivation here is that elements within the `@`-insertions are grouped; `\bracecolor` uses `\aftergroup` to re-insert the use of `\color`.

If the package is loaded with the `overload` option

```
\usepackage[overload]{abraces}
```

the traditional `\overbrace`/`\underbrace` pairs are redefined to be equivalent to `\aoverbrace` and `\aunderbrace`, respectively, via a straight-forward `\let`:

```
\let\overbrace\aoverbrace
\let\underbrace\aunderbrace
```

3 Advanced uses

There are some cases where you don't want to place elements exactly at the tip(s) of the brace(s). Maybe it's because you don't have any tips on your braces, or need to write other information around the tips. To that end, the second optional ⟨*script spec*⟩ argument is provided where you can specify a new sequence that may differ from the original brace specification. Without any tips, for example, one could write `$\aoverbrace[L1R]{a + b + \cdots +`

`z}[U]^{\text{26 terms}}$` to denote a grouping of elements:

$$\overbrace{a+b+\cdots+z}^{\text{26 terms}}$$

More detailed examples are covered in section 4 Examples.

4 Examples

Some basic examples of the types of braces that can be constructed using `abraces`:

```
\newcommand{\FnD}{%
  \textrm{The quick brown fox jumped over the lazy dog}}
```

- `\aoverbrace{\FnD}` (traditional `\overbrace`):
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace{\FnD}` (traditional `\underbrace`):
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aoverbrace[L3U1R]{\FnD}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aoverbrace[*{6}{0}l1D1r*{5}{0}]{\FnD}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace[l2D1r000@{\bracecolor{blue!70!black}}l1D2r]{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace[l1D2U2D1r]{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aoverbrace[L1R]{\FnD}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace[L1U3R]{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace['6,0l3D3r0,6']{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aoverbrace[L5*{3}{01}05U50*{3}{10}5R]{\FnD}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace[l1@{\hspace{5em}}2D2@{~\ldots~}1r]{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aunderbrace[l1R@{\bracecolor{red!80!white}}L1r]{\FnD}`:
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}$
- `\aoverbrace[,1D!{5em},]{\FnD}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}$

Some more advanced techniques of adding content to the brace cusps:

- `\aoverbrace[L1U2R]{\FnD}^{\text{one-third of the way}}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}^{\text{one-third of the way}}$

- `\aoverbrace[L1U1D1U1R]{\FnD}^{\text{left} & \text{right}}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}^{\text{left}\qquad\qquad\qquad\qquad\text{right}}$
- `\aoverbrace[L1U1D1U1R]{\FnD}`
  `[L1U1U1U1R]^{\text{left} & \text{middle} & \text{right}}`:
  $\overbrace{\text{The quick brown fox jumped over the lazy dog}}^{\text{left}\qquad\text{middle}\qquad\text{right}}$
- ```
  \newbracespec{u}
     {@{\hspace{-.5\bracecusplen}}U@{\hspace{-.5\bracecusplen}}}
  \newbracespec{d}
     {@{\hspace{-.5\bracecusplen}}D@{\hspace{-.5\bracecusplen}}}
  $\aunderbrace
     [0012@{\hspace{-\bracecusplen}}1r]{% \aunderbrace brace script
        \aoverbrace
           [L1@{\hspace{-\bracecusplen}}1R000]% \aoverbrace brace script
           {\FnD}% stuff
           [1u13]% \aoverbrace script spec
           ^{2/5}% \aoverbrace upper script
     }
  [43d3]% \aunderbrace script spec
  _{3/5}$% \aunderbrace lower script:
  ```
  $\underbrace{\overbrace{\text{The quick brown fox jumped over the lazy dog}}^{2/5}}_{3/5}$
- ```
  \newbracespec{a}{@{\hspace{-.5\bracecusplen}}D}
  \newbracespec{z}{D@{\hspace{-.5\bracecusplen}}}
  \aunderbrace[l1r]{\FnD}
  [a1z]_{\text{\rlap{far left}} & \text{\llap{far right}}}:
  ```
  $\underbrace{\text{The quick brown fox jumped over the lazy dog}}_{\text{far left}\qquad\qquad\qquad\qquad\qquad\qquad\text{far right}}$

Here is a real-world example where "breaking" a brace across lines is required to indicate a continuous grouping of objects. This example[1] constructs two open-ended `\aoverbrace`s that "span" multiple lines:

$$f(x) = a_0 + a_1x + a_2x^2 + \overbrace{a_3x^3 + a_4x^4 + \cdots + a_{i-1}x^{i-1}}^{\text{some text}} + \cdots \\ \cdots \overbrace{a_ix^i + a_{i+1}x^{i+1}} + \cdots + a_{n-1}x^{n-1}$$

```
\usepackage{amsmath}% http://ctan.org/pkg/amsmath
%...
\begin{multline*}
  f(x) = a_0 + a_1 x + a_2 x^2 +
    \aoverbrace[L1U1@{~\ldots}]{a_3 x^3 + a_4 x^4 + \dots +
      a_{i - 1} x^{i - 1} + \quad}^
```

[1] Taken from the question `\overbrace` split across multiple lines on the TeX StackExchange network.

```
      {\text{some text}} \\[\jot]
    \aoverbrace[@{\ldots~}1R]{\quad a_i x^i + a_{i + 1} x^{i + 1}} +
      \dots + a_{n - 1} x^{n - 1}
\end{multline*}
```

As a final example, consider a brace that should include a dashed component. Using `\newbracespec` one can define your own dashed component:

```
\newbracespec{d}{%
  5@{\hspace{4pt}}1@{\hspace{4pt}}!{2em}@{\hspace{4pt}}1@{\hspace{4pt}}5%
}
```

and then use

```
\[
  \aunderbrace[l*{3}{d}D*{3}{d}r]{\FnD}_{\text{What happened to the cat?}}
\]
```

$$\underbrace{\text{The quick brown fox jumped over the lazy dog}}_{\text{What happened to the cat?}}$$

5 Terms of reference

This package originated from a question on the TeX StackExchange network called Asymmetric overbrace. Some code was taken from the `mathtools` package.
This material is released under and subject to the LaTeX Project Public Licence.

6 Acknowledgements

Thanks to Frank Mittlebach who stepped in and suggested an improvement in the original way `abraces` functioned. Expansions included the use of LaTeX3 command interface (via the `xparse` package).

7 Change log

- v2.1 (2022-11-06) Minor update
  - `\Repeat` is replaced by something more obscure (`\Repeat@br@ces`) to avoid other package conflicts.
- v2.0 (2021-03-31) Major update
  - The package now uses `xparse` for macro definitions.
  - Included an automated way for handling elementary/default scripts. The placement of scripts can still be modified using the `\bracescript` interface.
  - The documentation was also updated to reflect the changes.
- v1.0 (2012-08-31) Initial release