

perfectcut.sty documentation

Guillaume Munch-Maccagnoni*

23rd April 2017

Contents

1 Use	1
2 Example	2
3 Advanced use	3
4 Options	4
5 Reimplementation of fixed-size delimiters	5
6 License	6

1 Use

Original use This package supplies the following commands:

Command	Produces	Command	Produces
<code>\perfectcut{#1}{#2}</code>	$\langle \#1 \parallel \#2 \rangle$	<code>\perfectcase{#1}</code>	$[\#1 \#2]$
<code>\perfectbra{#1}</code>	$\langle \#1 $	<code>\perfectbrackets{#1}</code>	$[\#1]$
<code>\perfectket{#1}</code>	$ \#1 \rangle$	<code>\perfectparens{#1}</code>	$(\#1)$

The effect of these commands is to let the delimiters grow according to the number of nested `\perfectcommands` (regardless of the size of contents). The package is originally intended for solving a notational issue regarding the representation of abstract-machine-like calculi in proof theory and computer science.

*<https://github.com/gadmm/perfectcut>

General use The package also defines `\perfectunary` and `\perfectbinary` for defining custom delimiters that behave similarly to the above ones. These commands should be considered when facing the need of nested delimiters that consistently grow regardless of the contents. (See “Advanced Use”.)

2 Example

Using `perfectcut.sty`

```
\usepackage{perfectcut}
\let\cut\perfectcut
```

The following states the idempotency of an adjunction:

$$\langle t \parallel \bar{\mu}x. \langle \mu\alpha. \langle u \parallel e \rangle \parallel e' \rangle \rangle = \langle \mu\alpha. \langle t \parallel \bar{\mu}x. \langle u \parallel e \rangle \rangle \parallel e' \rangle$$

The following states the commutativity of a strong monad:

$$\langle t \parallel \bar{\mu}x. \langle u \parallel \bar{\mu}y. \langle v \parallel e \rangle \rangle \rangle = \langle u \parallel \bar{\mu}y. \langle t \parallel \bar{\mu}x. \langle v \parallel e \rangle \rangle \rangle$$

Using `\underline` to mark redexes:

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

Using `\left`, `\middle` and `\right` instead

```
\renewcommand{\cut}[2]{\left\langle #1\middle|\mkern-2mu\middle|#2\right\rangle}
```

The following states the idempotency of an adjunction:

$$\langle t \parallel \bar{\mu}x. \langle \mu\alpha. \langle u \parallel e \rangle \parallel e' \rangle \rangle = \langle \mu\alpha. \langle t \parallel \bar{\mu}x. \langle u \parallel e \rangle \rangle \parallel e' \rangle$$

The following states the commutativity of a strong monad:

$$\langle t \parallel \bar{\mu}x. \langle u \parallel \bar{\mu}y. \langle v \parallel e \rangle \rangle \rangle = \langle u \parallel \bar{\mu}y. \langle t \parallel \bar{\mu}x. \langle v \parallel e \rangle \rangle \rangle$$

Using `\underline` to mark redexes:

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

As we can see, the legibility of the above rendering is hampered by multiple issues: the delimiters grow inconsistently, vertical bars have the wrong size, accents or underlines uselessly make the delimiters grow, and the spacing could be improved. The package is designed to fix these issues.

3 Advanced use

The package lets you define your own growing delimiters. Let us first stress that the size of these delimiters is entirely determined by the number of nestings and is insensitive to the size of the contents. If you need the size of the contents to be taken into account then it is probably sufficient to use `\left` and `\right` while tweaking `\delimitershortfall` and `\delimiterfactor`.

Example

The following displays a set $\{a \mid b\}$ with delimiters appropriately sized if there are other `\perfectcommands` inside `#1` and `#2`.

```
\def\Set#1#2{\perfectbinary{IncreaseHeight}\{||\}{#1\mathrel{}}{\mathrel{}}#2}
\[\Set{\perfectparens{a}}{\perfectparens{b}}\]
```

$$\{(a) \mid (b)\}$$

Custom delimiters

`\perfectunary#1#2#3#4` Displays $\langle \#2 \#4 \#3 \rangle$ where $\#2$ and $\#3$ are delimiters. The delimiters grow according to $\#1$ which must be one of `IncreaseHeight`, `CurrentHeight`, or `CurrentHeightPlusOne`.

`\perfectbinary#1#2#3#4#5#6` Displays $\langle \#2 \#5 \#3 \#6 \#4 \rangle$ where $\#2$, $\#3$ and $\#4$ are delimiters. The delimiters grow according to $\#1$ which must be one of `IncreaseHeight`, `CurrentHeight`, or `CurrentHeightPlusOne`.

Stock delimiters

The stock commands behave as follow:

Command	Produces	Growth	Inserts skips
<code>\perfectcut{\#1}{\#2}</code>	$\langle \#1 \parallel \#2 \rangle$	<code>IncreaseHeight</code>	Yes
<code>\perfectbra{\#1}</code>	$\langle \#1 $	<code>IncreaseHeight</code>	Yes
<code>\perfectket{\#1}</code>	$ \#1 \rangle$	<code>IncreaseHeight</code>	Yes
<code>\perfectcase{\#1}</code>	$[\#1 \#2]$	<code>CurrentHeightPlusOne</code>	Yes
<code>\perfectbrackets{\#1}</code>	$[\#1]$	<code>CurrentHeightPlusOne</code>	Only inside
<code>\perfectparens{\#1}</code>	$(\#1)$	<code>CurrentHeight</code>	Only inside
<code>\perfectunary{\#1}</code> <code>{\#2}{\#3}{\#4}</code>	$\#2 \#4 \#3$	$\#1$	No
<code>\perfectbinary{\#1}</code> <code>{\#2}{\#3}{\#4}{\#5}{\#6}</code>	$\#2 \#5 \#3 \#6 \#4$	$\#1$	No

4 Options

Option `mathstyle`

With this option, the command `\currentmathstyle` from the package `mathstyle` is used instead of the command `\ThisStyle` from the `scalere1` package. The latter (default) uses `\mathchoice` which has exponential time complexity in the number of nestings. The former does not use `\mathchoice` but redefines many primitives and is therefore a source of incompatibilities.

Option `realVert`

With the option `realVert`, the double bars are obtained with the `\Vert` command. But without it, two `\vert` symbols are used and their spacing is controlled with `\cutinterbarskip`. Also, if `realVert` is not activated, then a penalty (`binoppenalty`) is added, such that $\langle \mu\alpha.\langle a \parallel b \rangle \mid \bar{\mu}x.\langle c \parallel d \rangle \rangle$ splits across lines.

Option `fixxits`

For some reason that the author was unable to identify, the vertical alignment is wrong with the Opentype XITS math font with XeTeX. The option `fixxits` fixes this behaviour.

Customisation

The following mu-skips can be redefined in your preamble:

Command	Defines the spacing...
<code>\cutbarskip=5.0mu plus 8mu minus 2.0mu</code>	around bars
<code>\cutangleskip=0.0mu plus 8mu minus 1.0mu</code>	around delimiters (inside)
<code>\cutangleouterskip=0.0mu plus 8mu minus 0mu</code>	around delimiters (outside)
<code>\cutinterbarskip=0.8mu plus 0mu minus 0mu</code>	between bars (excl. <code>realVert</code>)

(1 mu equals 1/18 of an em in the current math font.)

5 Reimplementation of fixed-size delimiters

In addition, I provide the following corrections and generalisations of `\big`, `\bigg`, etc. Why not using the latter? Because both the plain T_EX and the `amsmath` versions can be incorrect when changing the math font, the font size, the math style or `\delimitershortfall`. Moreover, Opentype math fonts sometimes offer more than five sizes. For this package we need a robust solution.

Command	Example
<i>#1-th size of delimiter #2</i>	
<code>\nthleft{#1}{#2}</code>	<code>\nthleft{2}{(</code> (
<code>\nthmiddle{#1}{#2}</code>	<code>\nthmiddle{2}{\Vert}</code>
<code>\nthright{#1}{#2}</code>	<code>\nthright{2}{)}</code>)
<i>delimiter #2 of height at least #1</i>	
<code>\lenleft{#1}{#2}</code>	<code>\lenleft{3.2mm}{[</code> [
<code>\lenmiddle{#1}{#2}</code>	<code>\lenmiddle{3.2mm}{ </code>
<code>\lenright{#1}{#2}</code>	<code>\lenright{3.2mm}{]</code>]
<i>delimiter #2 of height exactly #1 obtained by scaling the above one</i>	
<code>\reallenleft{#1}{#2}</code>	<code>\reallenleft{3.2mm}{[</code> [
<code>\reallenmiddle{#1}{#2}</code>	<code>\reallenmiddle{3.2mm}{ </code>
<code>\reallenright{#1}{#2}</code>	<code>\reallenright{3.2mm}{]</code>]

Example with `\nthleft`

`\nrthleft0(\nthleft1(\nthleft2(\nthleft3(\nthleft4(\nthleft5(\nthleft6(`



Example with `\big`, `\Big`, `\bigg`, `\Bigg`

`(\big(\Big(\bigg(\Bigg(`



The above uses the `\big` commands from the `amsmath` package. The `amsmath` package corrects issues with the original `TEX` commands, but I could still notice inconsistencies, such as `\big` starting at size 2, under some font combinations. `\nthleft`, `\nthright` and `\nthmiddle` are implemented in a more robust way.

6 License

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version. Refer to the README file.