The **underoverlap** package[*]

Michiel Helvensteijn
mhelvens+latex@gmail.com

February 3, 2013

Development of this package is organized at latex-underoverlap.googlecode.com.
I am happy to receive feedback there!

---

1 Introduction

TeX and LaTeX provide several commands for decorating parts of our equations—to direct attention or to indicate some special meaning. For example:

```
\[ n \times x = \overbrace{x + \cdots + x}^n \]
```

$$n \times x = \overbrace{x + \cdots + x}^{n}$$

I'm refering to the `\overbrace`, of course. You can even nest such commands:

```
\[ n \times x =
   \overbrace{x + \underbrace{x + \cdots + x}_{n-2}{} + x}^n \]
```

$$n \times x = \overbrace{x + \underbrace{x + \cdots + x}_{n-2}{} + x}^{n}$$

But they have an inherent limitation: they cannot overlap partially. For example, you cannot do the following out-of-the-box:

$$a + \overbrace{b + \mathrlap{\underbrace{}_{y}}c + d}^{x} + e + f$$

That's what this package is all about. It allows you to define new decorator commands that can partially overlap like this, as well as augment existing commands.

---

[*] This document corresponds to **underoverlap** 0.0.1-r1, dated 2013/02/03.

2 Usage

⟨*UOL decorator*⟩ `{`⟨*non-overlapping content*⟩`}` `[`⟨*overlapping content*⟩`]`

An ⟨*UOL decorator*⟩ is a decorator command prepared by this package, such as `\UOLoverbrace` and `\UOLunderbrace` in the examples below.

If you never use the optional argument at all, they should behave exactly like the simple TEX commands `\overbrace` and `\underbrace`.

As far as the 'current ⟨*UOL decorator*⟩' is concerned, the optional content is placed directly to the right of the mandatory content and both are decorated as a whole. But a subsequent ⟨*UOL decorator*⟩ will be expected directly following this one. Its decorator will cover also the optional content from the current command. That's how we can typeset the example from the introduction:

```
\[ a + \UOLoverbrace{b +}[c + d]^x \UOLunderbrace{+ e}_y + f \]
```

$$a + \UOLoverbrace{b +}[c + d]^x \UOLunderbrace{+ e}_y + f$$

`\UOLoverbrace`
`\UOLunderbrace`
`\UOLoverline`
`\UOLunderline`

The following ⟨*UOL decorator*⟩ commands are predefined by the package, with obvious meaning:

- `\UOLoverbrace`
- `\UOLunderbrace`
- `\UOLoverline`
- `\UOLunderline`

Note that you are not limited to *two* overlapping decorators:

```
\[ a + \UOLoverbrace{b +}[c]^x \UOLunderbrace{+}[d]_y
   \UOLoverbrace{+ e}^z + f \]
```

$$a + \UOLoverbrace{b +}[c]^x \UOLunderbrace{+}[d]_y \UOLoverbrace{+ e}^z + f$$

`\newUOLdecorator` `{`⟨*command sequence*⟩`}` `{`⟨*definition containing* `#1`⟩`}`

Using this command you can create a new ⟨*UOL decorator*⟩. The first argument is the desired command sequence and the second contains its definition.

Define it as if a simple `\newcommand` were used for a traditional decorator. The package will automatically augment it to allow it to overlap. You can use `#1` to stand for the content provided by the user. The `\UOLoverline` decorator, for example, was defined as follows:

```
\newUOLdecorator\UOLoverline{\overline{#1}}
```

But note that the result will not be satisfactory unless the decorator requires no *horizontal space* to be allocated other than for its content. For the commands listed above, this is no problem. But this may not be true for all native decorations.

For example, one may naively try to define a `\UOLfbox` command as follows:

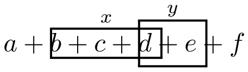
```
\newUOLdecorator\UOLfbox{{%
    \fboxsep=1.2pt \fboxrule=.8pt \fbox{$#1$}%
}}
```

To test it, we define the following command as well:

```
\newcommand\bigstrut{\vrule height 10pt depth 4pt width 0pt}
```

The result will look like this:

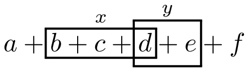
```
\[ a + \UOLfbox{b +c +}[d]^x \UOLfbox{\bigstrut + e}^y + f \]
```



The spacing is incorrect because `\fbox` allocates horizontal space for itself in addition to the space for its content and our package cannot take that into account. We'll have to correct for this manually:

```
\newUOLdecorator\UOLfbox{{%
    \fboxsep=1.2pt \fboxrule=.8pt%
    \kern-\fboxrule \kern-\fboxsep%
    \fbox{$#1$}%
    \kern-\fboxsep  \kern-\fboxrule%
}}
```

```
\[ a + \UOLfbox{b +c +}[d]^x \UOLfbox{\bigstrut + e}^y + f \]
```



Much better!

Note that subscripts / superscripts will be placed directly below / above the content, even if the original decorator did not support that.

`\UOLaugment` {⟨*command sequence*⟩}

This command can augment an existing native decorator command to allow overlapping with its original name:

```
\UOLaugment\overline
\UOLaugment\overbrace
\[ a + \overline{b +}[c + d] \overbrace{\bigstrut + e}^y + f \]
```

$$a + \overline{b +} \overbrace{\overline{c + d} + e}^{y} + f$$

Note, however, the following assumptions on the original command:

- It should not be a native TEX command such as `\underline`. Augmenting `\underline` like this actually seems to work fine, but the TEX specification can give us no guarantees about its behavior if we do.
- It should originally only support one standard mandatory argument. If it already supported optional arguments or other code structures then that behavior will be lost after augmentation.

If these requirements are satisfied then the new command should be *almost* backwards compatible. The exceptions being, of course, when you supply an optional argument or when it follows another ⟨*UOL decorator*⟩ that does. The other exception is the behavior of subscripts and superscripts if the original decorator did not place them directly below / above the content.

Of course, you can manually correct for these exceptions, but be aware that backwards compatibility is not completely guaranteed if you simply augment a decorator without any other measures.

This package does not 'pre-augment' any commands.

`\UOLunaugment` `{`⟨*command sequence*⟩`}`

Assuming that ⟨*command sequence*⟩ has been augmented by the `\UOLaugment` command, using `\UOLunaugment` reverses the effects and ⟨*command sequence*⟩ will once again have its original meaning.

3 Acknowledgements

I originaly read the following question on http://tex.stackexchange.com:

- http://tex.stackexchange.com/questions/12963

The commands of this package are partially based on the technique described by Herbert and the command definitions proposed by Martin Scharrer.

I got a lot of useful help from the answers to my own questions too:

- http://tex.stackexchange.com/questions/95786
- http://tex.stackexchange.com/questions/95897

Thanks in particular to David Carlisle and Hendrik Vogt.

4 Implementation

We now show and explain the entire implementation from `underoverlap.sty`.

4.1 Package Info

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{underoverlap}[2013/02/03 0.0.1-r1
    construct for partly overlapping math decorations]
```

4.2 Packages

These are the packages we'll need.

```
\RequirePackage{etoolbox}
\RequirePackage{mathtools}
\RequirePackage{xparse}
```

4.3 Public Macros

`\newUOLdecorator` {⟨*control sequence*⟩} {⟨*definition*⟩}

This defines the macro ⟨*control sequence*⟩ as a new decorator command that is capable of overlapping with others.

```
\NewDocumentCommand{\newUOLdecorator}{mm}{%
    \newcommand{#1}[1]{#2}%
    \UOLaugment{#1}%
}
```

`\UOLaugment` {⟨*control sequence*⟩}

This augments the macro ⟨*control sequence*⟩ to allow it to be overlapped.

```
\NewDocumentCommand{\UOLaugment}{m}{%
    \cslet{\uol@oldcsname{#1}}{#1}%
    \renewcommand{#1}{%
        \ifbool{mmode}{%
            \uol@construct{\csuse{\uol@oldcsname{#1}}}%
        }{%
            \csuse{\csuse{\uol@oldcsname{#1}}}%
        }%
    }%
}
```

`\UOLunaugment` {⟨*original cs*⟩}

This unaugments the macro ⟨*original cs*⟩ so it will once again have its original meaning. This will only make sense if you're previously augmented that command with `\UOLaugment`.

```
\NewDocumentCommand{\UOLunaugment}{m}{%
    \letcs{#1}{\uol@oldcsname{#1}}%
}
```

4.4 Private Macros

The content inside a decoration has three parts:

- A ‘left’ part that may overlap with the previous construct.
- A ‘middle’ part that does not overlap.
- A ‘right’ part that overlaps with the next construct.

Only the ‘middle’ and ‘right’ parts are passed as parameters throughout most of the following commands. The ‘left’ part is defined by the previous ⟨*UOL decorator*⟩ inside the `\uol@overlap@content` macro.

`\uol@oldcsname` `{`⟨*control sequence*⟩`}`

Given a control sequence, this derives the corresponding ’old csname’ used by this package when augmenting it.

```
\newcommand{\uol@oldcsname}[1]{%
    uol@old@\expandafter\@gobble\string#1%
}
```

`\uol@construct` `{`⟨*original macro*⟩`}` `{`⟨*middle*⟩`}` `{`⟨*right*⟩`}`

Scans ahead for subscript or superscript.

```
\NewDocumentCommand{\uol@construct}{mmO{}}{%
    \let\uol@sup\empty%
    \let\uol@sub\empty%
    \@ifnextchar^{%
        \uol@construct@sup{#1}{#2}{#3}%
    }{%
        \@ifnextchar_%
            {\uol@construct@sub  {#1}{#2}{#3}}%
            {\uol@construct@final{#1}{#2}{#3}}%
    }%
}
```

`\uol@construct@sup` `{`⟨*original macro*⟩`}` `{`⟨*middle*⟩`}` `{`⟨*right*⟩`}` `^` `{`⟨*superscript*⟩`}`

Processes superscript and scans ahead for subscript.

```
\def\uol@construct@sup#1#2#3^#4{%
    \def\uol@sup{#4}%
    \@ifnextchar_%
        {\uol@construct@sup@sub{#1}{#2}{#3}}%
        {\uol@construct@final  {#1}{#2}{#3}}%
}
```

`\uol@construct@sub` {⟨*original macro*⟩} {⟨*middle*⟩} {⟨*right*⟩} _ {⟨*subscript*⟩}

Processes subscript and scans ahead for superscript.

```
\def\uol@construct@sub#1#2#3_#4{%
    \def\uol@sub{#4}%
    \@ifnextchar^%
        {\uol@construct@sub@sup{#1}{#2}{#3}}%
        {\uol@construct@final  {#1}{#2}{#3}}%
}
```

`\uol@construct@sub@sup` {⟨*original macro*⟩} {⟨*middle*⟩} {⟨*right*⟩} ^ {⟨*superscript*⟩}

Processes superscript and calls the 'final' command.

```
\def\uol@construct@sub@sup#1#2#3^#4{%
    \def\uol@sup{#4}%
    \uol@construct@final{#1}{#2}{#3}%
}
```

`\uol@construct@sup@sub` {⟨*original macro*⟩} {⟨*middle*⟩} {⟨*right*⟩} _ {⟨*subscript*⟩}

Processes subscript and calls the 'final' command.

```
\def\uol@construct@sup@sub#1#2#3_#4{%
    \def\uol@sub{#4}%
    \uol@construct@final{#1}{#2}{#3}%
}
```

`\uol@overlap@content` This macro stores the 'overlapped' part of the math content between two augmented macros. We make sure that it is empty at the beginning of a new equation.

```
\let\uol@overlap@content\@empty
```

`\uol@construct@final` {⟨*original macro*⟩} {⟨*middle*⟩} {⟨*right*⟩}

This command typesets the result and reserves space for the overlap.

```
\newcommand{\uol@construct@final}[3]{
```

The following block will set the visible ink of the decoration without visibly setting the content itself and without moving our current position (using `\mathrlap`).

```
    \mathrlap{%
```

We do several things next. Using `\mathop` we allow any subscript and superscript to be positioned directly under or over the construct, even if the original command doesn't natively support it.

We call the original command with `#1` and make it initially invisible using `\phantom`.

Using braces we provide the spacing most likely to be satisfactory. If it's not, the user should manually correct with math-spacing commands.

```
61        \displaystyle%
62        \mathop{#1{\phantom{%
63            {\uol@overlap@content}{{}#2{}}{#3}%
64        }}}%
```

The following two lines set the subscript and superscript that we collected during the previous phases.

```
65        ^{\mathclap{\uol@sup}}%
66        _{\mathclap{\uol@sub}}%
```

```
67    }%
```

Next, we optionally print the debug-shadow (disabled right now).

```
68    %\mathrlap{\textcolor{green}{\uol@overlap@content#2#3}}%
```

The following code visibly typesets the ‘left’ and ‘middle’ parts of the content.

```
69    {\uol@overlap@content}{{}#2{}}%
```

We assume that the ‘right’ part (that overlaps with the next construct) will be visibly set by the following command, which will be *its* ‘left’ part. We store it in the `\uol@overlap@content` macro now.

```
70    \def\uol@overlap@content{#3}%
```

```
71 }
```

`\UOLoverbrace`
`\UOLunderbrace`
`\UOLoverline`
`\UOLunderline`

We predefine the following commands in a straightforward way:

```
72 \newUOLdecorator{\UOLoverbrace} {\overbrace {#1}}
73 \newUOLdecorator{\UOLunderbrace}{\underbrace{#1}}
74 \newUOLdecorator{\UOLoverline}  {\overline  {#1}}
75 \newUOLdecorator{\UOLunderline} {\underline {#1}}
```

Change History

0.0.1

General: initial version 1

0.0.1-r1

General: several improvements in the documentation 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.