

The package `witharrows`*

F. Pantigny
fpantigny@wanadoo.fr

February 12, 2018

Abstract

The LaTeX package `witharrows` gives an environment `{WithArrows}` which is similar to environment `{aligned}` of `amsmath` (and `mathtools`) but gives the possibility to draw arrows on the right side of the alignment. These arrows are usually used to give explanations concerning the mathematical calculus presented.

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two compilations may be necessary. This package requires the packages `expl3`, `xparse` and `tikz`. The following Tikz libraries are also required: `calc`, `arrows.meta` and `bending`.

This package gives an environment `{WithArrows}` to construct alignments of equations with arrows for the explanations on the right side:

```
$\begin{WithArrows}
A & = (a+1)^2 \Arrow{we expand} \\
& = a^2 + 2a + 1 \\
\end{WithArrows}$
```

$$\begin{aligned} A &= (a+1)^2 \\ &= a^2 + 2a + 1 \end{aligned} \quad \begin{array}{l} \searrow \\ \text{we expand} \end{array}$$

The arrow has been drawn with the command `\Arrow` on the line from which it starts. The command `\Arrow` must be used in the second column (the best way is to put it at the end of the second cell of the line as in the previous example).

1 Options for the shape of the arrows

The commande `\Arrow` has several options. These options can be put between square brackets, before, or after the mandatory argument.

The option `jump` gives the number¹ of lines the arrow must jump (the default value is, of course, 1).

```
$\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \Arrow[jump=2]{we expand} \\
& = (a+b)^2 + 2(a+b) + 1 \\
& = a^2 + 2ab + b^2 + 2a + 2b + 1 \\
\end{WithArrows}$
```

*This document corresponds to the version 1.5 of `witharrows`, at the date of 2018/02/12.

¹It's not possible to give a non-positive value to `jump`. See below (p. 2) the way to draw an arrow which goes backwards.

$$\begin{aligned}
 A &= ((a+b)+1)^2 \\
 &= (a+b)^2 + 2(a+b) + 1 \\
 &= a^2 + 2ab + b^2 + 2a + 2b + 1
 \end{aligned}
 \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \textit{we expand}$$

It's possible to put several arrows which start from the same line.

```

 $\begin{WithArrows}$ 
A &= \bigl((a+b)+1\bigr)^2 \quad \textcolor{violet}{\Arrow{}}\textcolor{violet}{\Arrow{}}[jump=2] \\
&= (a+b)^2 + 2(a+b) + 1 \\
&= a^2 + 2ab + b^2 + 2a + 2b + 1
\end{WithArrows}

```

$$\begin{aligned}
 A &= ((a+b)+1)^2 \\
 &= (a+b)^2 + 2(a+b) + 1 \\
 &= a^2 + 2ab + b^2 + 2a + 2b + 1
 \end{aligned}
 \quad \left. \begin{array}{l} \\ \\ \end{array} \right\}$$

The option `xoffset` shift the arrows to the right (we usually don't want the arrows to be stuck on the text). The default value of `xoffset` is 3 mm.

```

 $\begin{WithArrows}$ 
A &= \bigl((a+b)+1\bigr)^2 \\
\textcolor{violet}{\Arrow[xoffset=1cm]{}}\textcolor{violet}{\texttt{\texttt{xoffset=1cm}}}} \\
&= (a+b)^2 + 2(a+b) + 1
\end{WithArrows}

```

$$\begin{aligned}
 A &= ((a+b)+1)^2 \\
 &= (a+b)^2 + 2(a+b) + 1
 \end{aligned}
 \quad \left. \right\} \textit{with } \textcolor{violet}{xoffset=1cm}$$

The arrows are drawn with Tikz. That's why the command `\Arrow` has an option `tikz` which can be used to give to the arrow (in fact, the command `\path` of Tikz) the options proposed by Tikz for such an arrow. The following example gives an thick arrow.

```

 $\begin{WithArrows}$ 
A &= (a+1)^2 \quad \textcolor{violet}{\Arrow[tikz=thick]{}}\textcolor{violet}{\texttt{we expand}} \\
&= a^2 + 2a + 1
\end{WithArrows}

```

$$\begin{aligned}
 A &= (a+1)^2 \\
 &= a^2 + 2a + 1
 \end{aligned}
 \quad \left. \right\} \textit{we expand}$$

It's also possible to change the arrowheads. For example, we can draw an arrow which goes backwards with the Tikz option `<-`.

```

 $\begin{WithArrows}$ 
A &= (a+1)^2 \quad \textcolor{violet}{\Arrow[tikz=<-]{}}\textcolor{violet}{\texttt{we factorize}} \\
&= a^2 + 2a + 1
\end{WithArrows}

```

$$\begin{aligned}
 A &= (a+1)^2 \\
 &= a^2 + 2a + 1
 \end{aligned}
 \quad \left. \right\} \textit{we factorize}$$

It's also possible to suppress both tips of the arrow with the Tikz option `-`.

```

 $\begin{WithArrows}$ 
A &= (a+1)^2 \quad \textcolor{violet}{\Arrow[tikz=-]{}}\textcolor{violet}{\texttt{very classical}} \\
&= a^2 + 2a + 1
\end{WithArrows}

```

$$A = (a + 1)^2 \\ = a^2 + 2a + 1 \quad \Bigg) \textit{ very classical}$$

In order to have straight arrows instead of curved ones, we must use the Tikz option “`bend left = 0`”.

```
$\begin{WithArrows}
A &= (a+1)^2 \Arrow[tikz={bend left=0}]{we expand} \\
&= a^2 + 2a + 1 \\
\end{WithArrows}$
```

$$A = (a + 1)^2 \\ = a^2 + 2a + 1 \quad \downarrow \textit{ we expand}$$

In fact, it’s possible to change more drastically the shape or the arrows with the option `TikzCode` presented p. 13.

One of the most useful options is “`text width`” to control the width of the text associated to the arrow.

```
$\begin{WithArrows}
A &= \bigl((a+b)+1\bigr)^2 \\
\Arrow[jump=2,tikz={text width=5.3cm}]{We have done...} \\
&= (a+b)^2 + 2(a+b) + 1 \\
&= a^2 + 2ab + b^2 + 2a + 2b + 1 \\
\end{WithArrows}$
```

$$A = ((a + b) + 1)^2 \\ = (a + b)^2 + 2(a + b) + 1 \\ = a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \Bigg) \begin{array}{l} \textit{We have done a two-stages expansion} \\ \textit{but it would have been clever to ex-} \\ \textit{pand with the multinomial theorem.} \end{array}$$

If we want to change the font of the text associated to the arrow, we can, of course, put a command like `\bfseries`, `\large` or `\sffamily` at the beginning of the text. But, by default, the texts are composed with a combination of `\small` and `\itshape`. When adding `\bfseries` at the beginning of the text, we won’t suppress the `\small` and the `\itshape` and we will consequently have a text in a bold, italic and small font.

```
$\begin{WithArrows}
A &= (a+1)^2 \Arrow{\bfseries we expand} \\
&= a^2 + 2a + 1 \\
\end{WithArrows}$
```

$$A = (a + 1)^2 \\ = a^2 + 2a + 1 \quad \Bigg) \textit{ we expand}$$

If we put commands `\` in the text to force newlines, a command of font placed in the beginning of the text will have effect only until the first command `\` (like in an environment `{tabular}`). That’s why Tikz gives a option `font` to modify the font of the whole text. Nevertheless, if we use the option `tikz={font={\bfseries}}`, the default specification of `\small` and `\itshape` will be overwritten.

```
$\begin{WithArrows}
A &= (a+1)^2 \Arrow[tikz={font={\bfseries}}]{we expand} \\
&= a^2 + 2a + 1 \\
\end{WithArrows}$
```

$$\begin{aligned}
 A &= (a+1)^2 \\
 &= a^2 + 2a + 1 \quad \searrow \text{we expand}
 \end{aligned}$$

If we want exactly the same result as previously, we have to give to the option `font` the value `\itshape\small\bfseries`.

Almost all the options can be given directly to the environment `{WithArrows}` (between square brackets). In this case, they apply to all the arrows of the environment.²

```

 $\begin{WithArrows}[tikz=blue]
A &= \bigl((a+b)+1\bigr)^2 \Arrow{First expansion.} \\
&= (a+b)^2 + 2(a+b) + 1 \Arrow{Second expansion.} \\
&= a^2 + 2ab + b^2 + 2a + 2b + 1 \\
\end{WithArrows}$ 

```

$$\begin{aligned}
 A &= ((a+b)+1)^2 \\
 &= (a+b)^2 + 2(a+b) + 1 \quad \searrow \text{First expansion.} \\
 &= a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \searrow \text{Second expansion.}
 \end{aligned}$$

The environment `{WithArrows}` has an option `displaystyle`. With this option, all the elements are composed in `\displaystyle` (like in an environment `{aligned}` of `amsmath`).

Without the option `displaystyle`:

```

 $\begin{WithArrows}
\int_0^1 (x+1)^2 dx
&= \int_0^1 (x^2+2x+1) dx \\
\Arrow{linearity of integration} \\
&= \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \\
&= \frac{1}{3} + 2\frac{1}{2} + 1 \\
&= \frac{7}{3}
\end{WithArrows}$ 

```

$$\begin{aligned}
 \int_0^1 (x+1)^2 dx &= \int_0^1 (x^2 + 2x + 1) dx \\
 &= \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \quad \searrow \text{linearity of integration} \\
 &= \frac{1}{3} + 2\frac{1}{2} + 1 \\
 &= \frac{7}{3}
 \end{aligned}$$

The same example with the option `displaystyle`:

$$\begin{aligned}
 \int_0^1 (x+1)^2 dx &= \int_0^1 (x^2 + 2x + 1) dx \\
 &= \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \quad \searrow \text{linearity of integration} \\
 &= \frac{1}{3} + 2\frac{1}{2} + 1 \\
 &= \frac{7}{3}
 \end{aligned}$$

The version 1.3 of `witharrows` give two options for a fine tuning of the arrows:

- the option `ystart` set the vertical distance between the base line of the text and the start of the arrow (default value: 0.4 ex);
- the option `ygap` set the vertical distance between two consecutive arrows (default value: 0.4 ex).

²They also apply to the nested environments `{WithArrows}` (with the logical exceptions of `interline`, `CodeBefore` and `CodeAfter`).

$$\begin{aligned}
 (\cos x + \sin x)^2 &= \cos^2 x + 2 \cos x \sin x + \sin^2 x \xrightarrow{\text{ystart}} \\
 &= \cos^2 x + \sin^2 x + 2 \sin x \cos x \xrightarrow{\text{ygap}} \\
 &= 1 + \sin(2x)
 \end{aligned}$$

Remark: It's also possible to use the options “shorten <” and “shorten >” of Tikz (via the option `tikz of witharrows`).

Almost all the options can also be set at the document level with the command `\WithArrowsOptions`. In this case, the scope of the declarations is the current TeX group (these declarations are “semi-global”). For example, if we want all the environments `{WithArrows}` composed in `\displaystyle` with blue arrows, we can write `\WithArrowsOptions{displaystyle,tikz=blue}`.³

```

\WithArrowsOptions{displaystyle,tikz=blue}
$\begin{WithArrows}
\sum_{i=1}^n (x_i+1)^2
&= \sum_{i=1}^n (x_i^2+2x_i+1) \ \Arrow{by linearity}\
&= \sum_{i=1}^n x_i^2 + 2\sum_{i=1}^n x_i + n
\end{WithArrows}$

```

$$\begin{aligned}
 \sum_{i=1}^n (x_i + 1)^2 &= \sum_{i=1}^n (x_i^2 + 2x_i + 1) \\
 &= \sum_{i=1}^n x_i^2 + 2 \sum_{i=1}^n x_i + n
 \end{aligned}$$

by linearity

The command `\Arrow` is recognized only in the environments `{WithArrows}`. If we have a command `\Arrow` previously defined, it's possible to go on using it outside the environments `{WithArrows}`. However, a previously defined command `\Arrow` may still be useful in an environment `{WithArrows}`. If we want to use it in such an environment, it's possible to change the name of the command `\Arrow` of the package `witharrows`: there is an option `CommandName` for this purpose. The new name of the command must be given to the option *without* the leading backslash.

```

\NewDocumentCommand {\Arrow} {} {\longmapsto}
$\begin{WithArrows}[CommandName=Explanation]
f &= \bigl(x \ \Arrow (x+1)^2\bigr)
\Explanation{we work directly on fonctions}\
&= \bigl(x \ \Arrow x^2+2x+1\bigr)
\end{WithArrows}$

```

$$\begin{aligned}
 f &= (x \mapsto (x+1)^2) \\
 &= (x \mapsto x^2 + 2x + 1)
 \end{aligned}$$

we work directly on fonctions

The environment `{WithArrows}` gives also two options `CodeBefore` and `CodeAfter` for LaTeX code that will be executed at the beginning and at the end of the environment. These options are not designed to be hooks (they are available only at the environment level and they are not applied to the nested environments).

```

$\begin{WithArrows}[CodeBefore = \color{blue}]
A &= (a+b)^2 \ \Arrow{we expand} \
&= a^2 + 2ab + b^2
\end{WithArrows}$

```

$$\begin{aligned}
 A &= (a+b)^2 \\
 &= a^2 + 2ab + b^2
 \end{aligned}$$

we expand

Special commands are available in `CodeAfter` : a command `\NbLines` which gives the number of lines of the current environment (this is a command and not a counter), a special form of the command `\Arrow` and the command `\MultiArrow` : these commands are described in the section concerning the nested environments, p. 10.

³Since version 1.4 of `witharrows`, it's no longer possible to give these options directly when loading the package, *i.e.* with the command `\usepackage` in the preamble.

2 Precise positioning of the arrows

The environment `{WithArrows}` defines, during the composition of the array, two series of nodes materialized in red in the following example.⁴

$$\begin{aligned}
 I &= \int_{\frac{\pi}{4}}^0 \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right)(-du) \quad , \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right) du \quad , \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(1 + \frac{1 - \tan u}{1 + \tan u}\right) du \quad , \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(\frac{1 + \tan u + 1 - \tan u}{1 + \tan u}\right) du \quad , \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(\frac{2}{1 + \tan u}\right) du \quad , \\
 &= \int_0^{\frac{\pi}{4}} (\ln 2 - \ln(1 + \tan u)) du \quad , \\
 &= \frac{\pi}{4} \ln 2 - \int_0^{\frac{\pi}{4}} \ln(1 + \tan u) du \quad , \\
 &= \frac{\pi}{4} \ln 2 - I \quad ,
 \end{aligned}$$

The nodes of the left are at the end of each line of text. These nodes will be called *left nodes*. The nodes of the right side are aligned vertically on the right side of the array. These nodes will be called *right nodes*.

By default, the arrows use the right nodes. We will say that they are in **rr** mode (*r* for *right*). These arrows are **vertical** (we will say that an arrow is *vertical* when its two ends have the same abscissa).

However, it's possible to use the left nodes, or a combination of left and right nodes, with one of the options **lr**, **rl** and **ll** (*l* for *left*). Those arrows are, usually, not vertical.

Therefore $I = \int_{\frac{\pi}{4}}^0 \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right)(-du)$ This arrow uses the **lr** option.

$$\begin{aligned}
 &= \int_0^{\frac{\pi}{4}} \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right) du \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(1 + \frac{1 - \tan u}{1 + \tan u}\right) du \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(\frac{1 + \tan u + 1 - \tan u}{1 + \tan u}\right) du \\
 &= \int_0^{\frac{\pi}{4}} \ln\left(\frac{2}{1 + \tan u}\right) du \\
 &= \int_0^{\frac{\pi}{4}} (\ln 2 - \ln(1 + \tan u)) du \quad \leftarrow \text{This arrow uses a **ll** option and a jump equal to 2} \\
 &= \frac{\pi}{4} \ln 2 - \int_0^{\frac{\pi}{4}} \ln(1 + \tan u) du \\
 &= \frac{\pi}{4} \ln 2 - I
 \end{aligned}$$

There is also an option called **i** (*i* for *intermediate*). With this option, the arrow is vertical and at the leftmost position.

⁴The option `shownodes` can be used to materialize the nodes. The nodes are in fact Tikz nodes of shape “rectangle”, but with zero width. An arrow between two nodes starts at the *south* anchor of the first node and arrives at the *north* anchor of the second node.

```

 $\begin{WithArrows}$ 
(a+b)(a+ib)(a-b)(a-ib)
& = (a+b)(a-b)\cdot(a+ib)(a-ib) \\\
& = (a^2-b^2)(a^2+b^2) \xrightarrow[i]{\text{because } (x-y)(x+y)=x^2-y^2}} \\\
& = a^4-b^4
\end{WithArrows}

```

$$\begin{aligned}
 (a+b)(a+ib)(a-b)(a-ib) &= (a+b)(a-b) \cdot (a+ib)(a-ib) \\
 &= (a^2-b^2)(a^2+b^2) \quad \searrow \text{because } (x-y)(x+y)=x^2-y^2 \\
 &= a^4-b^4
 \end{aligned}$$

The environment `{WithArrows}` gives also a `group` option. With this option, *all* the arrows of the environment are grouped on a same vertical line and at a leftmost position.

```

 $\begin{WithArrows}[displaystyle,group]$ 
2xy'-3y=\sqrt{x}
& \Longleftarrow 2x(K'y_0+Ky_0')-3Ky_0 = \sqrt{x} \\\
& \Longleftarrow 2xK'y_0 + K(2xy_0'-3y_0) = \sqrt{x} \\\
& \Longleftarrow 2xK'y_0 = \sqrt{x} \xrightarrow{\dots} \\\
\ldots
\end{WithArrows}

```

$$\begin{aligned}
 2xy' - 3y &= \sqrt{x} \iff 2x(K'y_0 + Ky_0') - 3Ky_0 = \sqrt{x} \\
 &\iff 2xK'y_0 + K(2xy_0' - 3y_0) = \sqrt{x} \\
 &\iff 2xK'y_0 = \sqrt{x} \\
 &\iff 2xK'x^{\frac{3}{2}} = x^{\frac{1}{2}} \quad \searrow \text{We replace } y_0 \text{ by its value.} \\
 &\iff K' = \frac{1}{2x^2} \quad \searrow \text{simplification of the } x \\
 &\iff K = -\frac{1}{2x} \quad \searrow \text{antiderivation}
 \end{aligned}$$

The environment `{WithArrows}` gives also a `groups` option (with a *s* in the name). With this option, the arrows are divided into several “groups”. Each group is a set of connected⁵ arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

$$\begin{aligned}
 A &= B \\
 &= C + D \quad \searrow \text{one} \\
 &= D' \quad \searrow \text{two} \\
 &= E + F + G + H + I \\
 &= K + L + M \\
 &= N \quad \searrow \text{three} \\
 &= O \quad \searrow \text{four}
 \end{aligned}$$

If desired, the option `group` or the option `groups` can be given to the command `\WithArrowsOptions` so that it will become the default value. In this case, it's still possible to come back to the default behaviour for a given environment `{WithArrows}` with the option `rr`: `\begin{WithArrows}{rr}`

3 Comparison with the environment `{aligned}`

`{WithArrows}` bears similarities with the environment `{aligned}` of the extension `amsmath`. These are only similarities because `{WithArrows}` has not been written upon the environment `{aligned}`.⁶

As in the environments of `amsmath`, it's possible to change the spacing between two given lines with the option of the command `\` of end of line (it's also possible to use `*` but it has exactly the same effect as `\` since an environment `{WithArrows}` is always unbreakable).

⁵More precisely: for each arrow *a*, we note *i(a)* the number of its initial line and *f(a)* the number of its final line ; for two arrows *a* and *b*, we say that *a* ~ *b* when $\llbracket i(a), f(a) \rrbracket \cap \llbracket i(b), f(b) \rrbracket \neq \emptyset$; the groups are the equivalence classes of the transitive closure of ~.

⁶In fact, it's possible to use the package `witharrows` without the package `amsmath`.

```

 $\begin{WithArrows}$ 
 $A = (a+1)^2 \quad \text{\Arrow{we expand} \text{\texttt{\\[2ex]}}}$ 
 $\quad = a^2 + 2a + 1$ 
 $\end{WithArrows}$ 

```

$$\begin{aligned}
 A &= (a+1)^2 \\
 &\quad \quad \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{we expand} \\
 &= a^2 + 2a + 1
 \end{aligned}$$

In the environments of `amsmath` (or `mathtools`), the spacing between lines is fixed by a parameter called `\jot` (it's a dimension and not a skip). That's also the case for the environment `{WithArrows}`. An option `jot` has been given to the environment `{WithArrows}` in order to change the value of this parameter `\jot` for an given environment.⁷

```

 $\begin{WithArrows}[displaystyle,jot=2ex]$ 
 $F = \frac{1}{2}G \quad \text{\Arrow{we expand} \text{\texttt{\\}}}$ 
 $\quad = H + \frac{1}{2}K \quad \text{\Arrow{we go on} \text{\texttt{\\}}}$ 
 $\quad = K$ 
 $\end{WithArrows}$ 

```

$$\begin{aligned}
 F &= \frac{1}{2}G \\
 &\quad \quad \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{we expand} \\
 &= H + \frac{1}{2}K \\
 &\quad \quad \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{we go on} \\
 &= K
 \end{aligned}$$

However, this new value of `\jot` will also be used in other alignments included in the environment `{WithArrows}`:

```

 $\begin{WithArrows}[jot=2ex]$ 
 $\varphi(x,y) = 0 \quad \& \quad \text{\Leftrightarrow} \quad (x+y)^2 + (x+2y)^2 = 0$ 
 $\text{\Arrow{\$x\$ and \$y\$ are real} \text{\texttt{\\}}}$ 
 $\& \quad \text{\Leftrightarrow} \quad \text{\left\{}$ 
 $\begin{aligned}$ 
 $x+y \quad \& \quad = 0 \quad \text{\texttt{\\}}$ 
 $x+2y \quad \& \quad = 0 \quad \text{\texttt{\\}}$ 
 $\end{aligned}$ 
 $\text{\right.}$ 
 $\end{WithArrows}$ 

```

$$\begin{aligned}
 \varphi(x,y) = 0 &\Leftrightarrow (x+y)^2 + (x+2y)^2 = 0 \\
 &\quad \quad \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} x \text{ and } y \text{ are real} \\
 &\Leftrightarrow \begin{cases} x+y = 0 \\ x+2y = 0 \end{cases}
 \end{aligned}$$

Maybe this doesn't correspond to the desired outcome. That's why an option `interline` is proposed. It's possible to use a skip (`=glue`) for this option.

```

 $\begin{WithArrows}[interline=2ex]$ 
 $\varphi(x,y) = 0 \quad \& \quad \text{\Leftrightarrow} \quad (x+y)^2 + (x+2y)^2 = 0$ 
 $\text{\Arrow{\$x\$ and \$y\$ are real} \text{\texttt{\\}}}$ 
 $\& \quad \text{\Leftrightarrow} \quad \text{\left\{}$ 
 $\begin{aligned}$ 
 $x+y \quad \& \quad = 0 \quad \text{\texttt{\\}}$ 
 $x+2y \quad \& \quad = 0 \quad \text{\texttt{\\}}$ 
 $\end{aligned}$ 
 $\text{\right.}$ 
 $\end{WithArrows}$ 

```

⁷It's also possible to change `\jot` with the environment `{spreadlines}` of `mathtools`.

$$\varphi(x, y) = 0 \Leftrightarrow (x + y)^2 + (x + 2y)^2 = 0 \quad \left. \vphantom{\varphi(x, y) = 0} \right\} x \text{ and } y \text{ are real}$$

$$\Leftrightarrow \begin{cases} x + y = 0 \\ x + 2y = 0 \end{cases}$$

Like the environment `{aligned}`, `{WithArrows}` has an option of placement which can assume the values `t`, `c` or `b`. However, the default value is not `c` but `t`. If desired, it's possible to have the `c` value as the default with the command `\WithArrowsOptions{c}` at the beginning of the document.

```
So\enskip
$\begin{WithArrows}
A \& = (a+1)^2 \Arrow{we expand} \\
& = a^2 + 2a + 1 \\
\end{WithArrows}$
```

$$\text{So } A = (a + 1)^2 \quad \left. \vphantom{A = (a + 1)^2} \right\} \text{we expand}$$

$$= a^2 + 2a + 1$$

The value `c` may be useful, for example, if we want to add curly braces:

```
On pose\enskip $\left\{
\begin{WithArrows}[c]
f(x) \& = 3x^3+2x^2-x+4 \\
\Arrow{tikz=-}{both are polynoms} \\
g(x) \& = 5x^2-5x+6 \\
\end{WithArrows}
\right.$
```

$$\text{On pose } \left\{ \begin{array}{l} f(x) = 3x^3 + 2x^2 - x + 4 \\ g(x) = 5x^2 - 5x + 6 \end{array} \right\} \text{ both are polynoms}$$

Unlike `{aligned}`, the environment `{WithArrows}` uses `\textstyle` by default. Once again, it's possible to change this behaviour with `\WithArrowsOptions{displaystyle}`.

The following example is composed with `{aligned}`:

$$\left\{ \begin{array}{l} \sum_{i=1}^n (x_i + 1)^2 = \sum_{i=1}^n (x_i^2 + 2x_i + 1) \\ \phantom{\sum_{i=1}^n (x_i + 1)^2} = \sum_{i=1}^n x_i^2 + 2 \sum_{i=1}^n x_i + n \end{array} \right.$$

The following is composed with `{WithArrows}[c,displaystyle]`. The results are stricly identical.⁸

$$\left\{ \begin{array}{l} \sum_{i=1}^n (x_i + 1)^2 = \sum_{i=1}^n (x_i^2 + 2x_i + 1) \\ \phantom{\sum_{i=1}^n (x_i + 1)^2} = \sum_{i=1}^n x_i^2 + 2 \sum_{i=1}^n x_i + n \end{array} \right.$$

⁸In versions of `amsmath` older than the 5 nov. 2016, an thin space was added on the left of an environment `{aligned}`. The new versions do not add this space and neither do `{WithArrows}`.

4 Arrows in nested environments

The environments `{WithArrows}` can be nested. In this case, the options given to the encompassing environment applies also to the inner ones (with logical exceptions for `interline`, `CodeBefore` and `CodeAfter`). The command `Arrow` can be used as usual in each environment `{WithArrows}`.

```

 $\begin{WithArrows}$ 
\varphi(x,y)=0
& \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \Arrow{the numbers are real}
& \Leftrightarrow
\left\{ \begin{array}{l} x+2y = 0 \\ 2x+4y = 0 \end{array} \right.
\end{WithArrows} \right.
& \Leftrightarrow
\left\{ \begin{array}{l} x+2y = 0 \\ x+2y = 0 \end{array} \right. \Arrow{tikz=-}{the same equation}
\end{WithArrows} \right.
& \Leftrightarrow x+2y=0
\end{WithArrows}
```

$$\begin{aligned}
\varphi(x,y) = 0 &\Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0 \\
&\Leftrightarrow \left\{ \begin{array}{l} x+2y = 0 \\ 2x+4y = 0 \end{array} \right. \quad \left. \vphantom{\left\{ \begin{array}{l} x+2y = 0 \\ 2x+4y = 0 \end{array} \right.}} \right) \text{the numbers are real} \\
&\Leftrightarrow \left\{ \begin{array}{l} x+2y = 0 \\ x+2y = 0 \end{array} \right. \quad \left. \vphantom{\left\{ \begin{array}{l} x+2y = 0 \\ x+2y = 0 \end{array} \right.}} \right) \text{the same equation} \\
&\Leftrightarrow x+2y = 0
\end{aligned}$$

However, one may want to draw an arrow between lines that are not in the same environment. For example, one may want to draw the following arrow :

$$\begin{aligned}
\varphi(x,y) = 0 &\Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0 \\
&\Leftrightarrow \left\{ \begin{array}{l} x+2y = 0 \\ 2x+4y = 0 \end{array} \right. \\
&\Leftrightarrow \left\{ \begin{array}{l} x+2y = 0 \\ x+2y = 0 \end{array} \right. \quad \left. \vphantom{\left\{ \begin{array}{l} x+2y = 0 \\ x+2y = 0 \end{array} \right.}} \right) \text{Division by 2} \\
&\Leftrightarrow x+2y = 0
\end{aligned}$$

Such a construction is possible by using `\Arrow` in the `CodeAfter` option. Indeed, in `CodeAfter`, a special version of `\Arrow` is available (we will call it “`\Arrow` in `CodeAfter`”).

A command `\Arrow` in `CodeAfter` takes three arguments :

- a specification of the start line of the arrow ;
- a specification of the end line of the arrow ;
- the label of the arrow.

As usual, it’s also possible to give options within square brackets before or after the three arguments. However, these options are limited (see below).

The specification of the line is constructed with the position of the concerned environment in the nesting tree, followed (after an hyphen) by the number of the line.

In the previous example, there are two environments `{WithArrows}` nested in the main environment `{WithArrows}`.

$$\begin{aligned}
\varphi(x, y) = 0 &\Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0 \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ 2x + 4y = 0 \end{cases} \quad \text{environment number 1} \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ x + 2y = 0 \end{cases} \quad \text{environment number 2} \\
&\Leftrightarrow x + 2y = 0
\end{aligned}$$

The arrow we want to draw starts in the line 2 of the sub-environment number 1 (and therefore, the specification is 1-2) and ends in the line 2 of the sub-environment number 2 (and therefore, the specification is 2-2). We can draw the arrow with the following command `\Arrow` in `CodeAfter` :

```

 $\begin{WithArrows}[CodeAfter = {\Arrow{1-2}{2-2}{Division by $2$}}]
\varphi(x,y)=0
& \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \\\
.....
\end{WithArrows}$ 

```

$$\begin{aligned}
\varphi(x, y) = 0 &\Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0 \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ 2x + 4y = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ x + 2y = 0 \end{cases} \quad \left. \begin{array}{l} \nearrow \\ \nwarrow \end{array} \right\} \text{Division by 2} \\
&\Leftrightarrow x + 2y = 0
\end{aligned}$$

The options allowed for a command `\Arrow` in `CodeAfter` are : `ll`, `lr`, `rl`, `rr`, `v`, `xoffset`, `tikz` and `TikzCode`. Except `v`, which is specific to `\Arrow` in `CodeAfter`, all these options have their usual meaning.

With the option `v`, the arrow drawn is vertical to an abscissa computed with the start line and the end line only : the intermediate lines are not taken into account unlike with the option `i`. Currently, the option `i` is not available for the command `\Arrow` in `CodeAfter`. However, it's always possible to translate an arrow with `xoffset` (or `xshift` of `Tikz`).

```

 $\begin{WithArrows}[CodeAfter = {\Arrow[v]{1-2}{2-2}{Division by $2$}}]
\varphi(x,y)=0
& \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \\\
.....
\end{WithArrows}$ 

```

$$\begin{aligned}
\varphi(x, y) = 0 &\Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0 \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ 2x + 4y = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} x + 2y = 0 \\ x + 2y = 0 \end{cases} \quad \left. \begin{array}{l} \nearrow \\ \nwarrow \end{array} \right\} \text{Division by 2} \\
&\Leftrightarrow x + 2y = 0
\end{aligned}$$

The package `witharrows` gives also another command available only in `CodeAfter` : the command `\MultiArrow`. This commands draws a “rak”. The list of the lines of the environment concerned by this rak are given in the first argument of the command `\MultiArrow`. This list is given with the syntax of the list in a `\foreach` command of `pgfkeys`.

```

 $\begin{WithArrows}[tikz = rounded corners,
CodeAfter = {\MultiArrow{1,...,4}{text}}]
A & = B \\\
& = C \\\$ 
```

$$\begin{array}{l} A = B \\ = C \\ = D \\ = E \\ = F \end{array} \left. \begin{array}{l} \leftarrow \\ \leftarrow \\ \leftarrow \\ \leftarrow \end{array} \right\} \textit{text}$$

5 Arrows from outside environments WithArrows

The Tikz name of a node created by `witharrows` is prefixed by `wa-`. Then, we have a list of numbers which give the position in the nesting tree and the number of the line in the environment. At the end, we have the suffixe `l` for a “left node” and `r` for a “right node”.

$$\begin{aligned}
A &\triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B_{\text{wa-37-1}} \\
&\triangleleft \begin{cases} C \triangleleft D_{\text{wa-37-1-1}} \\ E \triangleleft F_{\text{wa-37-1-2}} \end{cases} \quad \text{wa-37-2} \\
&\triangleleft \begin{cases} G \triangleleft H + H + H + H + H + H + H_{\text{wa-37-2-1}} \\ I \triangleleft \begin{cases} J \triangleleft K_{\text{wa-37-2-1-1}} \\ L \triangleleft M_{\text{wa-37-2-1-2}} \end{cases} \end{cases} \quad \begin{matrix} \text{wa-37-3} \\ \text{wa-37-2} \end{matrix} \\
&\triangleleft \begin{cases} N \triangleleft O_{\text{wa-37-3-1}} \\ P \triangleleft Q_{\text{wa-37-3-2}} \end{cases} \quad \text{wa-37-4}
\end{aligned}$$

- the command `\WithArrowsLastEnv` gives the number of the last environment of level 0 ;
- the Tikz style `WithArrows/arrow` is the style used by `witharrows` when drawing an arrow¹⁰ ;
- the Tikz style `WithArrows/arrow/tips` is the style for the style of the arrow (loaded by `WithArrows/arrow`).

```
\begin{tikzpicture}[remember picture,overlay]
\draw [WithArrows/arrow]
      ($(\wa-\WithArrowsLastEnv-2-1-2-r.south)+(3mm,0)$)
      to ($(\wa-\WithArrowsLastEnv-3-2-r.north)+(3mm,0)$) ;
\end{tikzpicture}
```

¹⁰ More precisely, this style is given to the Tikz option “**every path**” before drawing the arrow with the code of the option `TikzCode`. This style is modified (in TeX scopes) by the option `tikz` of `witharrows`.

$$\begin{array}{l}
A \triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B + B + B \\
\triangleleft \left\{ \begin{array}{l} C \triangleleft D \\ E \triangleleft F \end{array} \right. \\
\triangleleft \left\{ \begin{array}{l} G \triangleleft H + H + H + H + H + H + H \\ I \triangleleft \left\{ \begin{array}{l} J \triangleleft K \\ L \triangleleft M \end{array} \right. \end{array} \right. \\
\triangleleft \left\{ \begin{array}{l} N \triangleleft O \\ P \triangleleft Q \end{array} \right. \quad \leftarrow \quad \text{curved arrow from } (I, J, K, L, M) \text{ to } (P, Q)
\end{array}$$

In this case, it would be easier to use a command `\Arrow` in `CodeAfter` but this is an example to explain how the Tikz nodes created by `witharrows` can be used.

6 Advanced features

6.1 The option TikzCode : how to change the shape of the arrows

The option `TikzCode` allows the user to change the shape of the arrow.

The value of this option must be an valid Tikz drawing instruction (with the final semi-colon) with three markers **#1**, **#2** and **#3** for the start point, the end point and the label of the arrow.

By default, the value is the following :

```
\draw (#1) to node {#3} (#2) ;
```

In the following example, we replace this default path by a path with three segments (and the node overwriting the second segment).

```

\begin{WithArrows}[ygap=5pt,interline=4mm,
    TikzCode = {\draw[rounded corners]
        (#1) -- ($(#1) + (5mm,0)$)
        -- node[circle,
            draw,
            auto = false,
            fill = gray!50,
            inner sep = 1pt] {\tiny #3}
        ($(#2) + (5mm,0)$)
        -- (#2) ; }
E & \Longleftarrow 3(2x+4) = 6 & \Arrow{$\div 3$} \\\
& \Longleftarrow 2x+4 = 2 & \Arrow{$-4$} \\\
& \Longleftarrow 2x = -2 & \Arrow{$\div 2$} \\\
& \Longleftarrow 2x = -1 \\
\end{WithArrows}

```

$$\begin{array}{lcl}
 E \iff 3(2x+4) = 6 & \xrightarrow{\div 3} & \\
 \iff 2x+4 = 2 & \xrightarrow{-4} & \\
 \iff 2x = -2 & \xrightarrow{\div 2} & \\
 \iff 2x = -1 & &
 \end{array}$$

6.2 Footnotes in the environment WithArrows

If you want to put footnotes in an environment `{WithArrows}`, you can use a pair `\footnotemark-`
`\footnotetext`.

It's also possible to extract the footnotes with the help of the package `footnote` or the package `footnotehyper`.

If `witharrows` is loaded with the option `footnote` (with `\usepackage[footnote]{witharrows}` or with `\PassOptionsToPackage`), the package `footnote` is loaded (if it is not yet loaded) and it is used to extract the footnotes.

If `witharrows` is loaded with the option `footnotehyper`, the package `footnotehyper` is loaded (if it is not yet loaded) and it is used to extract footnotes.

Caution: The packages `footnote` and `footnotehyper` are incompatible. The package `footnotehyper` is the successor of the package `footnote` and should be used preferently. The package `footnote` has some drawbacks, in particular: it must be loaded after the package `xcolor` and it is not perfectly compatible with `hyperref`.

In this document, the package `witharrows` has been loaded with the option `footnotehyper` and we give an example with a footnote in the label of an arrow:

$$\begin{aligned} A &= (a+b)^2 \\ &= a^2 + b^2 + 2ab \end{aligned} \quad \downarrow \textit{We expand}^{\textcolor{red}{11}}$$

7 Examples

7.1 With only one column

It's possible to use the environment `{WithArrows}` with making use of the left column only, or the right column only.

```
\begin{WithArrows}
&f(x) \geq g(x) \ \Arrow{by squaring both sides} \\
&f(x)^2 \geq g(x)^2 \ \Arrow{by moving to left side} \\
&f(x)^2 - g(x)^2 \geq 0 \\
\end{WithArrows}
```

$$\begin{aligned} f(x) &\geq g(x) \\ f(x)^2 &\geq g(x)^2 \\ f(x)^2 - g(x)^2 &\geq 0 \end{aligned} \quad \begin{array}{l} \downarrow \textit{by squaring both sides} \\ \downarrow \textit{by moving to left side} \end{array}$$

7.2 MoveEqLeft

It's possible to use `\MoveEqLeft` of `mathtools` (if we don't want ampersand on the first line):

```
\begin{WithArrows}[interline=0.5ex]
\MoveEqLeft \arccos(x) = \arcsin \frac{4}{5} + \arcsin \frac{5}{13}
\Arrow{because both are in $[-\frac{\pi}{2}, \frac{\pi}{2}]$} \\
&\Leftrightarrow x = \sin(\arcsin \frac{4}{5} + \arcsin \frac{5}{13}) \\
&\Leftrightarrow x = \frac{4}{5} \cos \arcsin \frac{5}{13} + \frac{5}{13} \cos \arcsin \frac{4}{5} \\
\Arrow{\$forall x \in [-1, 1], \cos(\arcsin x) = \sqrt{1-x^2}$} \\
&\Leftrightarrow x = \frac{4}{5} \sqrt{1 - (\frac{5}{13})^2} + \frac{5}{13} \sqrt{1 - (\frac{4}{5})^2} \\
+ \frac{5}{13} \sqrt{1 - (\frac{4}{5})^2} \\
\end{WithArrows}
```

$$\begin{aligned} \arccos(x) &= \arcsin \frac{4}{5} + \arcsin \frac{5}{13} \\ \Leftrightarrow x &= \sin \left(\arcsin \frac{4}{5} + \arcsin \frac{5}{13} \right) \\ \Leftrightarrow x &= \frac{4}{5} \cos \arcsin \frac{5}{13} + \frac{5}{13} \cos \arcsin \frac{4}{5} \\ \Leftrightarrow x &= \frac{4}{5} \sqrt{1 - \left(\frac{5}{13}\right)^2} + \frac{5}{13} \sqrt{1 - \left(\frac{4}{5}\right)^2} \end{aligned} \quad \begin{array}{l} \downarrow \textit{because both are in } [-\frac{\pi}{2}, \frac{\pi}{2}] \\ \downarrow \forall x \in [-1, 1], \cos(\arcsin x) = \sqrt{1-x^2} \end{array}$$


¹¹ A footnote.

7.3 Modifying the shape of the nodes

It's possible to change the shape of the labels, which are Tikz nodes, by modifying the key “every node” of Tikz.

```
\begin{WithArrows}[%
  interline = 4mm,
  tikz = {every node/.style = {circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt,
                                font = \tiny}}]

E & \Longleftarrow 3 (2x+4) = 6
\Arrow{$\div 3$}\\
& \Longleftarrow 2x+4 = 2
\Arrow{$-4$}\\
& \Longleftarrow 2x = -2
\Arrow{$\div 2$} \\
& \Longleftarrow 2x = -1
\end{WithArrows}
```

$$\begin{array}{l}
 E \Longleftarrow 3(2x+4) = 6 \\
 \Longleftarrow 2x+4 = 2 \\
 \Longleftarrow 2x = -2 \\
 \Longleftarrow 2x = -1
 \end{array}$$


7.4 Examples with the option TikzCode

We recall that the option `TikzCode` is the Tikz code used by `witharrows` to draw the arrows.

The value by default of `TikzCode` is `\draw (#1) to node {#3} (#2)` ; where the three markers #1, #2 and #3 represent the start line, the end line and the label of the arrow.

7.4.1 Example 1

In the following example, we define the value of `TikzCode` with two instructions `\path` : the first instruction draw the arrow itself and the second puts the label in a Tikz node in the rectangle delimited by the arrow.

```
$\begin{WithArrows}[
  displaystyle,
  ygap = 2mm,
  ystart = 0mm,
  TikzCode = {\path[draw] (#1) -- ++(4.5cm,0) |- (#2) ;
              \path (#1) -- (#2)
              node[text width = 4.2cm, right, midway] {#3} ;}]

S_n
& = \frac{1}{n} \sum_{k=0}^{n-1} \cos \bigl( \frac{\pi}{2} \cdot k \bigr)
.....
```

$$\begin{aligned}
S_n &= \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) & \overline{\cos x = \Re(e^{ix})} \\
&= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i \frac{k\pi}{2n}}\right) & \overline{\leftarrow} \\
&= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right) & \overline{\Re(z + z') = \Re(z) + \Re(z')} \\
&= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right) & \overline{\leftarrow} \\
&= \frac{1}{n} \Re\left(\frac{1 - \left(e^{i \frac{\pi}{2n}}\right)^n}{1 - e^{i \frac{\pi}{2n}}}\right) & \overline{\begin{array}{l} \text{exp is a morphism for } \times \text{ et } + \\ \text{sum of terms of a geometric} \\ \text{progression of ratio } e^{i \frac{2\pi}{n}} \end{array}} \\
&= \frac{1}{n} \Re\left(\frac{1 - i}{1 - e^{i \frac{\pi}{2n}}}\right) & \overline{\leftarrow}
\end{aligned}$$

7.4.2 Example 2

In the following example, we change the shape of the arrow depending on whether the start line is longer than the end line or not.

```

\begin{WithArrows}[11,interline=5mm,xoffset=5mm,
  TikzCode = {\draw[rounded corners,
    every node/.style = {circle,
      draw,
      auto = false,
      inner sep = 1pt,
      fill = gray!50,
      font = \tiny }]}

    let \p1 = (#1),
        \p2 = (#2)
    in \ifdim \x1 > \x2
      (\p1) -- node [#3] (\x1,\y2) -- (\p2)
    \else
      (\p1) -- (\x2,\y1) -- node [#3] (\p2)
    \fi ;}]

E & \Longleftarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\Arrow{$\times 15$}\\
& \Longleftarrow 5(x+4) + 3(5x+3) = 105 \\
& \Longleftarrow 5x+20 + 15x+9 = 105 \\
& \Longleftarrow 20x+29 = 105
\Arrow{$-29$}\\
& \Longleftarrow 20x = 76
\Arrow{$\div 20$}\\
& \Longleftarrow x = \frac{38}{10}
\end{WithArrows}

```


$$\begin{aligned}
E &\iff \frac{(x+4)}{3} + \frac{5x+3}{5} = 7 && \xrightarrow{\quad \text{---} \quad} \textcircled{\times 15} \\
&\iff 5(x+4) + 3(5x+3) = 105 && \downarrow \\
&\iff 5x + 20 + 15x + 9 = 105 \\
&\iff 20x + 29 = 105 && \textcircled{-29} \\
&\iff 20x = 76 && \leftarrow \textcircled{\div 20} \\
&\iff x = \frac{38}{10} && \leftarrow
\end{aligned}$$

7.5 Automatic numbered loop

Assume we want to draw a loop of numbered arrows. In this purpose, it's possible to write a dedicated command `\NumberedLoop` which will do the job when used in `CodeAfter`. In the following example, we write this command with `\NewDocumentCommand` of `xparse` and `\foreach` of `pgffor` (both packages are loaded when `witharrows` is loaded).

```

\NewDocumentCommand \NumberedLoop {}
{
  \foreach \j in {2,...,\NbLines}
  {
    \pgfmathtruncatemacro{\i}{\j-1}
    \Arrow[rr]{\i}{\j}{\i} }
  \Arrow[rr,xoffset=1cm,tikz=<-]{1}{\NbLines}{\NbLines}}

```

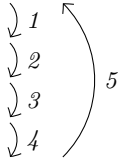
The command `\NbLines` is a command available in `CodeAfter` which gives the total number of lines of the current environment (it's a command and not a counter).

```

$\begin{WithArrows}[CodeAfter = {\NumberedLoop}]
a.\;& f \text{ est continuous on } E \\\
b.\;& f \text{ est continuous in } 0 \\\
c.\;& f \text{ is bounded on the unit sphere} \\\
d.\;& \exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K \|x\| \\\
e.\;& f \text{ is lipschitzian} \\
\end{WithArrows}$

```

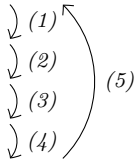
$a.$ f est continuous on E
 $b.$ f est continuous in 0
 $c.$ f is bounded on the unit sphere
 $d.$ $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K \|x\|$
 $e.$ f is lipschitzian



As usual, it's possible to change the characteristic of both arrows and nodes with the option `tikz`. However, if we want to change the style and to have, for example, numbers in parenthesis, the best way is to change the value of `TikzCode`:

```
TikzCode = {\draw (#1) to node {\footnotesize (#3)} (#2) ;}}
```

$a.$ f est continuous on E
 $b.$ f est continuous in 0
 $c.$ f is bounded on the unit sphere
 $d.$ $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K \|x\|$
 $e.$ f is lipschitzian



8 Implementation

8.1 Declaration of the package and extensions loaded

First, `tikz` and some Tikz libraries are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.¹²

```
1 \RequirePackage{tikz}
2 \usetikzlibrary{calc,arrows.meta,bending}
```

Then, we can give the traditionnal declaration of a package written with `expl3`:

```
3 \RequirePackage{l3keys2e}
4 \ProvidesExplPackage
5   {witharrows}
6   {\myfiledate}
7   {\myfileversion}
8   {Draws arrows for explanations on the right}
```

The package `xparse` will be used to define the environment `{WithArrows}` and the document-level commands `\Arrow` and `\WithArrowsOptions`.

```
9 \RequirePackage{xparse}
```

8.2 The packages `footnote` and `footnotehyper`

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 1.5), there are two such options: `footnote` and `footnotehyper`. With the option `footnote`, `witharrows` loads `footnote` and uses it to extract the footnotes from the environments `{WithArrows}`. Idem for the option `footnotehyper`.

The boolean `\g_@@_footnotehyper_bool` will indicate if the option `footnotehyper` is used.

```
10 \bool_new:N \g_@@_footnotehyper_bool
```

The boolean `\g_@@_footnote_bool` will indicate if the option `footnote` is used, but quickly, it will also be set to true if the option `footnotehyper` is used.

```
11 \bool_new:N \g_@@_footnote_bool
```

We define a set of keys `WithArrows/package` for these options. However, first, we define a “level of options” `\l_@@_level_int` even if, in the version 1.5 of `witharrows`, this integer is not used by the options of the set `WithArrows/package`.

```
12 \int_new:N \l_@@_level_int
13 \keys_define:nn {WithArrows/package}
14   {footnote      .bool_gset:N = \g_@@_footnote_bool,
15   footnotehyper .bool_gset:N = \g_@@_footnotehyper_bool}
```

We process the options when the package is loaded (with `\usepackage`).

```
16 \ProcessKeysOptions {WithArrows/package}

17 \msg_new:nnn {witharrows}
18   {Option-incompatible-with-Beamer}
19   {The-option-"\tl_use:N \l_keys_key_tl" is-incompatible-
20     with-Beamer-because-Beamer-has-its-own-system-to-extract-footnotes.}

21 \msg_new:nnn {witharrows}
22   {footnote-with-footnotehyper-package}
23   {You-can't-use-the-option-footnote-because-the-package-
24     footnotehyper-has-already-been-loaded.~
25     If-you-want,~you-can-use-the-option-"footnotehyper"~and-the-footnotes~
26     within-the-environments-{WithArrows}~will-be-extracted-with-the-tools~
27     of-the-package-footnotehyper.}
```

¹²cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

```

28 \msg_new:nnn {witharrows}
29     {footnotehyper~with~footnote~package}
30     {You~can't~use~the~option~"footnotehyper"~because~the~package~
31       footnote~has~already~been~loaded.~
32       If~you~want,~you~can~use~the~option~"footnote"~and~the~footnotes~
33       within~the~environments~{WithArrows}~will~be~extracted~with~the~tools~
34       of~the~package~footnote.}

35 \bool_if:NT \g_@@_footnote_bool
36     {\@ifclassloaded {beamer}
37       {\msg_fatal:nn {witharrows}
38         {Option~incompatible~with~Beamer}}}
39     {}
40     \@ifpackageloaded{footnotehyper}
41       {\msg_fatal:nn {witharrows}
42         {footnote~with~footnotehyper~package}}
43     {}
44     \usepackage{footnote}}

45 \bool_if:NT \g_@@_footnotehyper_bool
46     {\@ifclassloaded {beamer}
47       {\msg_fatal:nn {witharrows}
48         {Option~incompatible~with~Beamer}}}
49     {}
50     \@ifpackageloaded{footnote}
51       {\msg_fatal:nn {witharrows}
52         {footnotehyper~with~footnote~package}}
53     {}
54     \usepackage{footnotehyper}
55     \bool_gset_true:N \g_@@_footnote_bool

```

The flag `\g_@@_footnote_bool` is raised and so, we will only have to test `\g_@@_footnote_bool` in order to know if we have to insert an environment `{savenotes}` (see the definition of environment `{WithArrows}`).

8.3 Some technical definitions

We define a Tikz style `@@_node_style` for the nodes that will be created in the `\halign`. The nodes are Tikz nodes of shape “rectangle” but with zero width. An arrow between two nodes starts from the *south* anchor of the first node and arrives at the *north* anchor of the second node.

```

56 \tikzset{@@_node_style/.style= {
57     above = \l_@@_ystart_dim,
58     inner~sep = 0 pt,
59     minimum~width = 0pt,
60     minimum~height = \l_@@_ygap_dim,
61     red,
62     \bool_if:NT \l_@@_shownodes_bool {draw} }}

```

The color of the nodes is red, but in fact, the nodes will be drawn only when the option `shownodes` is used (this option is useful for debugging).

The style `@@_standard` is load in standard in the `{tikzpicture}` we need. The names of the nodes are prefixed by `wa` (by security) but also by a prefix which is the position-in-the-tree of the nested environments.

```

63 \tikzset{@@_standard/.style= { remember~picture,
64     overlay,
65     name~prefix = wa-\l_@@_prefix_str- }}

```

We also define a style for the tips of arrow. The final user of the extension `witharrows` will use this style if he wants to draw an arrow directly with a Tikz command in his document (probably using the Tikz nodes created by `{WithArrows}` in the `\halign`).

```

66 \tikzset{WithArrows/arrow/tips/.style = { > = {Straight~Barb[scale=1.2,bend]} }}

```

```

67 \tikzset{WithArrows/arrow/.style = { align = left,
68                                     auto = left,
69                                     font = {\small\itshape},
70                                     WithArrows/arrow/tips,
71                                     bend~left = 45,
72                                     -> }}

```

In order to increase the interline in the environments `{WithArrows}`, we will use the command `\spread@equation` of `amsmath`. When used, this command becomes no-op (in the current TeX group). Therefore, it will be possible to use the environments of `amsmath` (e.g. `{aligned}`) in an environment `{WithArrows}`.

Nevertheless, we want the extension `witharrows` available without `amsmath`. That's why we give a definition of `\spread@equation` (this definition will be loaded only if `amsmath` — or `mathtools` — has not been loaded yet).

```

73 \cs_if_free:NT \spread@equation
74     {\cs_set:Npn \spread@equation{\openup\jot
75                                     \cs_set:Npn \spread@equation {}}}

```

Don't put `\cs_set:Npn \spread@equation \prog_do_nothing:` in the last line because this would raise errors with nested environments.

8.4 Variables

The following sequence is the position of the last environment `{WithArrows}` in the tree of the nested environments `{WithArrows}`.

```

76 \seq_new:N \g_@@_position_in_the_tree_seq
77 \seq_gput_right:Nn \g_@@_position_in_the_tree_seq 1

```

The following counter will give the number of the last environment `{WithArrows}` of level 0. This counter will be used only in the definition of `\WithArrowsLastEnv`.

```

78 \int_new:N \g_@@_last_env_int

```

The following skip (`=glue`) is the vertical space inserted between two lines of the `\halign`.

```

79 \skip_new:N \l_@@_interline_skip

```

If the following flag is raised, then the user can use more than two columns.

```

80 \bool_new:N \l_@@_MoreColumns_bool

```

The following integer indicates the position of the box that will be created: 0 (`=t=\vtop`), 1 (`=c=\vcenter`) or 2 (`=b=\vbox`).

```

81 \int_new:N \l_@@_pos_env_int

```

```

82 \dim_new:N \l_@@_xoffset_dim

```

```

83 \dim_set:Nn \l_@@_xoffset_dim {3mm}

```

The integer `\l_@@_pos_arrows_int` indicates the position of the arrows with the following code (the option `v` is accessible only for the arrows in `CodeAfter` where the options `i`, `group` et `groups` are not available).

option	rr	ll	rl	lr	v	i	group	groups
<code>\l_@@_pos_arrows_int</code>	0	1	2	3	4	5	6	7

```

84 \int_new:N \l_@@_pos_arrows_int

```

When we scan a list of options, we want to be able to raise an error if two options of position of the arrows are present. That's why we keep the code of the first option of position in a variable called `\l_@@_previous_pos_arrows_int`. This variable will be set to -1 each time we start the scanning of a list of options.

```
85 \int_new:N \l_@@_previous_pos_arrows_int
```

At each possible level for the options (*global*, *environment* or *local*: see below), the new values will be appended on the right of this token list.

The dimension `\g_@@_x_dim` will be used to compute the x -value for some vertical arrows when one of the options `i`, `group` and `groups` (values 5, 6 and 7 of `\l_@@_pos_arrows_int`) is used.

```
86 \dim_new:N \g_@@_x_dim
```

In the `\halign` of an environment `{WithArrows}`, we will have to use three counters:

- `\g_@@_arrow_int` to count the arrows created in the environment ;
- `\g_@@_line_int` to count the lines of the `\halign` ;
- `\g_@@_line_bis_int` to count the lines of the `\halign` which have a second column.¹³

These three counters will be incremented in a cell of the `\halign` and, therefore, the incrementation must be global. However, we want to be able to include a `{WithArrows}` in another `{WithArrows}`. To do so, we must restore the previous value of these counters at the end of an environment `{WithArrows}` and we decide to manage a stack for each of these counters.

```
87 \seq_new:N \g_@@_arrow_int_seq
88 \int_new:N \g_@@_arrow_int
89 \seq_new:N \g_@@_line_int_seq
90 \int_new:N \g_@@_line_int
91 \seq_new:N \g_@@_line_bis_int_seq
92 \int_new:N \g_@@_line_bis_int
```

8.5 The definition of the options

There are four levels where options can be set:

- in the options of the `\usepackage`: this level will be called *package* level (number 0);
- with `\WithArrowsOptions{...}`: this level will be called *global* level (number 1);
- with `\begin{WithArrows}[...]`: this level will be called *environment* level (number 2);
- with `\Arrow[...]` (included in `CodeAfter`): this level will be called *local* level (number 3).

The level is specified in the variable `\l_@@_level_int` and the code attached to the options can use this information to alter their actions.

```
93 \int_set:Nn \l_@@_level_int 1
```

We go on with a submodule which will be loaded only at the global or the environment level.

The options `t`, `c` and `b` indicate if we will create a `\vtop`, a `\vcenter` or a `\vbox`. This information is stored in the variable `\l_@@_pos_env_int`.

```
94 \keys_define:nn {WithArrows/GlobalOrEnv}
95 { t .code:n = \int_set:Nn \l_@@_pos_env_int 0,
96   t .value_forbidden:n = true,
97   c .code:n = \int_set:Nn \l_@@_pos_env_int 1,
98   c .value_forbidden:n = true,
99   b .code:n = \int_set:Nn \l_@@_pos_env_int 2,
100  b .value_forbidden:n = true,
```

¹³This counter is used in order to raise an error if there is a line without the second column (such an situation could raise a PGF error for an undefined node).

The gap between two consecutive arrows.

```

101     ygap .dim_set:N          = \l_@@_ygap_dim,
102     ygap .value_required:n = true,
103     ygap .initial:n         = 0.4 ex,

```

The vertical position of the start point of an arrow.

```

104     ystart .dim_set:N        = \l_@@_ystart_dim,
105     ystart .value_required:n = true,
106     ystart .initial:n        = 0.4 ex,

```

Usually, the number of columns in a `{WithArrows}` environment is limited to 2. Nevertheless, it's possible to have more columns with the option `MoreColumns`.

```

107     MoreColumns .bool_set:N      = \l_@@_MoreColumns_bool,
108     MoreColumns .value_forbidden:n = true,

```

If the user wants to give a new name to the `\Arrow` command (and the name `\Arrow` remains free).

```

109     CommandName .tl_set:N        = \l_@@_CommandName_tl,
110     CommandName .initial:n       = Arrow ,
111     CommandName .value_required:n = true,

112     TikzCode .tl_set:N          = \l_@@_tikz_code_tl,
113     TikzCode .initial:n         = \draw~(#1)~to~node{#3}~(#2)~; ,
114     TikzCode .value_required:n = true,

```

With the option `displaystyle`, the environments will be composed in `\displaystyle`.

```

115     displaystyle .bool_set:N      = \l_@@_displaystyle_bool,
116     displaystyle .initial:n       = false,

```

With the option `shownodes`, the nodes will be drawn in red (useful only for debugging).

```

117     shownodes .bool_set:N        = \l_@@_shownodes_bool,
118     shownodes .initial:n         = false,

```

With the option `shownodenames`, the name of the “right nodes” will be written in the document (useful only for debugging).

```

119     shownodenames .bool_set:N     = \l_@@_shownodenames_bool,
120     shownodenames .initial:n      = false,

```

With the option `group`, *all* the arrows of the environment are vertical with the same abscissa and at a leftmost position.

```

121     group .code:n      = {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
122                           {\msg_error:nn {witharrows}
123                               {Two~options~are~incompatible}}}
124                           \int_set:Nn \l_@@_previous_pos_arrows_int 6
125                           \int_set:Nn \l_@@_pos_arrows_int 6} ,
126     group .value_forbidden:n = true,

```

With the option `groups` (with a *s*), the arrows of the environment are divided in groups by an argument of connexity, and, in each group, the arrows are vertical with the same abscissa and at a leftmost position. When the option `group` or `groups` is used, it's not possible to another option of position like `ll`, `lr`, etc. for a individual key.

```

127     groups .code:n      = {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
128                           {\msg_error:nn {witharrows}
129                               {Two~options~are~incompatible}}}
130                           \int_set:Nn \l_@@_previous_pos_arrows_int 7
131                           \int_set:Nn \l_@@_pos_arrows_int 7} ,
132     groups .value_forbidden:n = true,

```

The option `CodeBefore` gives a code that is executed at the beginning of the environment `{WithArrows}` (after the eventual `\begin{savenotes}`).

```

133     CodeBefore .code:n = {\int_compare:nNnTF \l_@@_level_int = 1
134                          {\msg_error:nn {witharrows} {Option~will~be~ignored}}
135                          {\tl_put_right:Nn \l_@@_code_before_tl {#1}}} ,

```

The option `CodeAfter` gives a code that is executed at the end of the environment `{WithArrows}` (after the eventual `\end{savenotes}`).

```

136     CodeAfter .code:n = {\int_compare:nNnTF \l_@@_level_int = 1
137                          {\msg_error:nn {witharrows} {Option~will~be~ignored}}
138                          {\tl_put_right:Nn \l_@@_code_after_tl {#1}}} ,
139     unknown .code:n = \msg_error:nn {witharrows} {Option~unknown}
140 }

```

Then we define the main module called `WithArrows/General` which will be loaded at all the levels.

The option `tikz` gives Tikz parameters that will be given to the arrow when it is drawn (more precisely, the parameters will be given to the command `\path` of Tikz).

```

141 \keys_define:nn {WithArrows/General}
142 {tikz .code:n = \tikzset {WithArrows/arrow/.append~style = {#1}},
143  tikz .initial:n = {},
144  tikz .value_required:n = true,

```

The other options are for the position of the arrows. The treatment is the same for the options `ll`, `rr`, `lr`, `lr` and `i` and that's why a dedicated function `\@@_analyze_option_position:n` has been written (see below).

```

145     rr .value_forbidden:n = true,
146     rr .code:n = \@@_analyze_option_position:n 0 ,
147     ll .value_forbidden:n = true,
148     ll .code:n = \@@_analyze_option_position:n 1 ,
149     rl .value_forbidden:n = true,
150     rl .code:n = \@@_analyze_option_position:n 2 ,
151     lr .value_forbidden:n = true,
152     lr .code:n = \@@_analyze_option_position:n 3 ,
153     i .value_forbidden:n = true,
154     i .code:n = \@@_analyze_option_position:n 5 ,

```

The option `xoffset` change the x -offset of the arrows (towards the right). It's a dimension and not a skip. It's not possible to change the value of this parameter for a individual arrow if the option `group` or the option `groups` is used. When we will treat an individual arrow, we will give it the option `tikz={xshift=\l_@@_xoffset_dim}` (we can't to it at the global or the environment level because the Tikz options `xshift` are cumulative).

```

155     xoffset .code:n = {\bool_if:nTF {\int_compare_p:nNn \l_@@_level_int = 3 &&
156                                     \int_compare_p:nNn \l_@@_pos_arrows_int > 5}
157                          {\msg_error:nn {witharrows}
158                           {Option~incompatible~with~"group(s)"}}
159                          {\dim_set:Nn \l_@@_xoffset_dim {#1}}} ,
160     xoffset .value_required:n = true,

```

The option `jot` exists for compatibility. It changes directly the value of the parameter `\jot`, which is a LaTeX parameter and not a parameter specific to `witharrows`. It's allowed only at the level of the environment (maybe we should suppress completely this option in the future).

```

161     jot .code:n = {\int_compare:nNnTF \l_@@_level_int = 2
162                  {\dim_set:Nn \jot {#1}}
163                  {\msg_error:nn {witharrows}
164                   {Option~will~be~ignored}}} ,
165     jot .value_required:n = true,

```

The option `interline` gives the vertical skip (=glue) inserted between two lines (independently of `\jot`). It's accepted only at the level of the environment (this last point is a kind of security). Furthermore, this option has a particular behaviour: it applies only to the current environment and doesn't apply to the nested environments.

```

166   interline .code:n      = {\int_compare:nNnTF \l_@@_level_int = 2
167                               {\skip_set:Nn \l_@@_interline_skip {#1}}
168                               {\msg_error:nn {witharrows}
169                                           {Option~will~be~ignored}}} ,
170   interline .value_required:n = true,

```

Eventually, a key `jump` (see below) and a key for unknown keys.

```

171   jump .code:n = \msg_error:nn {witharrows} {Option~will~be~ignored} ,
172   unknown .code:n = \msg_error:nn {witharrows} {Option~unknown}
173 }

```

The key `jump` indicates the number of lines jumped by the arrow (1 by default). This key will be extracted when the command `\Arrow` will be executed. That's why there is a special module for this key. The key `jump` is extracted in the command `\Arrow` because we want to compute right away the final line of the arrow (this will be useful for the options `group` and `groups`).

```

174 \keys_define:nn {WithArrows/jump}
175   {jump .code:n      = {\int_set:Nn \l_@@_jump_int {#1}
176                           \int_compare:nNnF \l_@@_jump_int > 0
177                           {\msg_error:nn {witharrows}
178                                           {The~option~"jump"~must~be~non~negative}}}} ,
179   jump .value_required:n = true}

```

The following command is for technical reasons. It's used for the following options of position: `ll`, `lr`, `rl`, `rr` and `i`. The argument is the corresponding code for the position of the arrows.

```

180 \cs_new_protected:Nn \@@_analyze_option_position:n
181   {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
182     {\msg_error:nn {witharrows}
183       {Two~options~are~incompatible}}}
184   \int_set:Nn \l_@@_previous_pos_arrows_int {#1}

```

It's not possible to use one of the considered options at the level of an arrow (level 2) when the option `group` or the option `groups` is used. However, if we are at the level of an environment, it's possible to override a previous option `group` or `groups` (this previous option `group` or `groups` would necessarily have been set at a global level by `\WithArrowsOptions`).

```

185   \bool_if:nTF { \int_compare_p:nNn \l_@@_level_int = 3 &&
186                 \int_compare_p:nNn \l_@@_pos_arrows_int > 5}
187     {\msg_error:nn {witharrows}
188       {Option~incompatible~with~"group(s)"}}
189     {\int_set:Nn \l_@@_pos_arrows_int {#1}}

```

`\WithArrowsOptions` is the command of the `witharrows` package to fix options at the document level.

```

190 \NewDocumentCommand \WithArrowsOptions {m}
191   {\int_set:Nn \l_@@_previous_pos_arrows_int {-1}
192     \keys_set_known:nnN {WithArrows/General} {#1} \l_tmpa_tl
193     \keys_set:nV {WithArrows/GlobalOrEnv} \l_tmpa_tl}

```

8.6 The command `Arrow`

In fact, the internal command is not named `\Arrow` but `\@@_Arrow`. Usually, at the beginning of an environment `{WithArrows}`, `\Arrow` is set to be equivalent to `\@@_Arrow`. However, the user can change the name with the option `CommandName` and the user command for `\@@_Arrow` will be different. This mechanism can be useful when the user has already a command named `\Arrow` he wants to still be able to use in the environment `{WithArrows}`.

```

194 \NewDocumentCommand \@@_Arrow {0{} m 0{}}
195   {

```


The counter `\g_@@_arrow_int` counts the arrows in the environment. The incrementation must be global (`gincr`) because the command `\Arrow` will be used in the cell of a `\halign`. It's recalled that we manage a stack for this counter.

```
196 \int_gincr:N \g_@@_arrow_int
```

We decide to extract immediatly the key `jump` in order to compute the end line. That's the reason why there is a module `WithArrows/jump` with this sole key. The remained key-value pairs are stored in `\l_tmpa_tl` and will be stored further in the properly list of the arrow.

```
197 \int_zero_new:N \l_@@_jump_int
198 \int_set:Nn \l_@@_jump_int 1
199 \keys_set_known:nnN {WithArrows/jump} {#1,#3} \l_tmpa_tl
```

We will construct a global property list to store the informations of the considered arrow. The four fields of this property list are “initial”, “final”, “options” and “label”.

1. First, the line from which the arrow starts:

```
200 \prop_put:NnV \l_tmpa_prop {initial} \g_@@_line_int
```

2. The line where the arrow ends (that's why it was necessary to extract the key `jump`):

```
201 \int_set:Nn \l_tmpa_int {\g_@@_line_int + \l_@@_jump_int}
202 \prop_put:NnV \l_tmpa_prop {final} \l_tmpa_int
```

3. All the options of the arrow (it's a token list):

```
203 \prop_put:NnV \l_tmpa_prop {options} \l_tmpa_tl
```

4. The label of the arrow (it's also a token list):

```
204 \prop_put:Nnn \l_tmpa_prop {label} {#2}
```

The property list has been created in a local variable for convenience. Now, it will be stored in a global variable indicating both the position-in-the-tree and the number of the arrow.

```
205 \prop_gclear_new:c
206 {g_@@_arrow_\l_@@_prefix_str _\int_use:N\g_@@_arrow_int _prop}
207 \prop_gset_eq:cN
208 {g_@@_arrow_\l_@@_prefix_str _\int_use:N\g_@@_arrow_int _prop}
209 \l_tmpa_prop
210 }
```

```
211 \cs_new_protected:Nn \@@_Arrow_first_column:
```

All messages of LaTeX3 must be *fully expandable* and that's why we do the affectation (necessary for a comparison) before the `\msg_error:nn`.

```
212 {\tl_set:Nn \l_tmpa_tl {Arrow}
213 \msg_error:nn {witharrows} {Arrow-in-first-column}
214 \@@_Arrow}
```

8.7 The environment `{WithArrows}`

The environment `{WithArrows}` starts with the initialisation of the three counters `\g_@@_arrow_int`, `\g_@@_line_int` and `\g_@@_line_bis_int`. However, we have to save their previous values with the three stacks created for this end.

```
215 \NewDocumentEnvironment {WithArrows} {0{}}
216 { \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
217 \int_gzero:N \g_@@_arrow_int
218 \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
219 \int_gzero:N \g_@@_line_int
220 \seq_gput_right:NV \g_@@_line_bis_int_seq \g_@@_line_bis_int
221 \int_gzero:N \g_@@_line_bis_int
```

We also have to update the position on the nesting tree.

```
222 \seq_gput_right:Nn \g_@@_position_in_the_tree_seq 1
```

The nesting tree is used to create a prefix which will be used in the names of the Tikz nodes and in the names of the arrows (each arrow is a property list of four fields). If we are in the second environment `{WithArrows}` nested in the third environment `{WithArrows}` of the document, the prefix will be 3-2 (although the position in the tree is [3,2,1] since such a position always ends with a 1). First, we do a copy of the position-in-the-tree and then we pop the last element of this copy (in order to drop the last 1).

```
223 \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
224 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
225 \str_clear_new:N \l_@@_prefix_str
226 \str_set:Nx \l_@@_prefix_str {\seq_use:Nnnn \l_tmpa_seq {-} {-} {-}}
```

The environment `{WithArrows}` must be used in math mode.

```
227 \reverse_if:N \if_mode_math:
228 \msg_error:nn {witharrows}
229 {{WithArrows}~used~outside~math~mode}
230 \fi:
```

We define the command `\` to be the command `\@@_cr:` (defined below).

```
231 \cs_set_eq:NN \ \@@_cr:
232 \mathsurround = \c_zero_dim
```

These counters will be used later as variables.

```
233 \int_zero_new:N \l_@@_initial_int
234 \int_zero_new:N \l_@@_final_int
235 \int_zero_new:N \l_@@_arrow_int
```

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.¹⁴

```
236 \skip_zero:N \l_@@_interline_skip
```

The value corresponding to the key `CodeBefore` is put to nil before the treatment of the options of the environment, because, of course, we don't want the code executed at the beginning of all the nested environments `{WithArrows}`. Idem for `CodeAfter`.

```
237 \tl_clear_new:N \l_@@_code_before_tl
238 \tl_clear_new:N \l_@@_code_after_tl
```

We process the options given to the `{WithArrows}` environment. The level of options is set to 1.

```
239 \int_set:Nn \l_@@_previous_pos_arrows_int {-1}
240 \int_set:Nn \l_@@_level_int 2
241 \keys_set_known:nnN {WithArrows/General} {#1} \l_tmpa_tl
242 \keys_set:nV {WithArrows/GlobalOrEnv} \l_tmpa_tl
```

If the option `footnote` or the option `footnotehyper` is used, then we extract the footnotes with an environment `{savenotes}` (of the package `footnote` or the package `footnotehyper`).

```
243 \bool_if:NT \g_@@_footnote_bool {\begin{savenotes}}
```

We execute the code `\l_@@_code_before_tl` of the option `CodeBefore` of the environment after the eventual `\begin{savenotes}` and, symetrically, we will execute the `\l_@@_code_after_tl` before the eventual `\end{savenotes}` (we have a good reason for the last point : we want to extract the footnotes of the arrows executed in the `CodeAfter`).

```
244 \l_@@_code_before_tl
```

¹⁴It's recalled that, by design, the option `interline` of an environment doesn't apply in the nested environments.

If the user has given a value for the option `CommandName` (at the global or at the *environment* level), a command with this name is defined locally in the environment with meaning `\@@_Arrow`. The default value of the option `CommandName` is “`Arrow`” and thus, by default, the name of the command will be `\Arrow`.

```
245 \cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow
```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`¹⁵ depending of the value of `\l_@@_pos_env_int` (fixed by the options `t`, `c` or `b`). The environment `{WithArrows}` must be used in math mode¹⁶ and therefore, we can use `\vcenter`.

```
246 \int_case:nn \l_@@_pos_env_int
247 {0 {\vtop}
248 1 {\vcenter}
249 2 {\vbox}}
250 \bgroup
```

The command `\spread@equation` is the command used by `amsmath` in the beginning of an alignment to fix the interline. When used, it becomes no-op. However, it’s possible to use `witharrows` without `amsmath` since we have redefined `\spread@equation` (if it is not defined yet).

```
251 \spread@equation
```

We begin the `\halign` and the preamble.

```
252 \ialign\bgroup
```

We increment the counter `\g_@@_line_int` which will be used in the names of the Tikz nodes created in the array. This incrementation must be global (`gincr`) because we are in the cell of a `\halign`. It’s recalled that we manage a stack for this counter.

```
253 \int_gincr:N \g_@@_line_int
254 \cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow_first_column:
255 \strut\hfil
256 $\bool_if:NT \l_@@_displaystyle_bool \displaystyle {##}$
257 &
```

In the second column, we increment the counter `\g_@@_line_bis_int` because we want to count the lines with a second column and raise an error if there is lines without a second column. Once again, the incrementation must be global and it’s recalled that we manage a stack for this counter too.

```
258 \int_gincr:N \g_@@_line_bis_int
259 $\bool_if:NT \l_@@_displaystyle_bool \displaystyle {#}##$
```

We create the “left node” of the line (when using macros in Tikz node names, the macros have to be fully expandable: here, `\tl_use:N` and `\int_use:N` are fully expandable).

```
260 \tikz [@@_standard] \node [@@_node_style] (\int_use:N\g_@@_line_int-l) {} ;
261 \hfil
```

Now, after the `\hfil`, we create the “right node” and, if the option `shownodenames` is raised, the name of the node is written in the document (useful for debugging).

```
262 \tikz [@@_standard] \node [@@_node_style] (\int_use:N\g_@@_line_int-r) {} ;
263 \bool_if:NT \l_@@_shownodenames_bool
264 {\hbox_overlap_right:n {\small wa-\l_@@_prefix_str
265 -\int_use:N\g_@@_line_int}}
```

Usually, the `\halign` of an environment `{WithArrows}` will have exactly two columns. Nevertheless, if the user wants to use more columns (without arrows) it’s possible with the option `MoreColumns`.

```
266 && \bool_if:NF \l_@@_MoreColumns_bool
267 {\msg_error:nn {witharrows} {Third-column-in-a-{WithArrows}-environment}}
268 $\bool_if:NT \l_@@_displaystyle_bool \displaystyle {##}$
269 \cr
270 }
```

¹⁵Notice that the use of `\vtop` seems color-safe here...

¹⁶An error is raised if the environment is used outside math mode.

We begin the second part of the environment `{WithArrows}`. We have two `\egroup`: one for the `\halign` and one for the `\vtop` (or `\vcenter` or `\vbox`).

```
271      {\crrc
272       \egroup
273       \egroup
```

If there is a line without the second column, we raise an error (a line without the second column could generate an PGF error for an unknown node since the nodes are created in the second column).

```
274      \int_compare:nNnT \g_@@_line_bis_int < \g_@@_line_int
275      {\msg_error:nn {witharrows} {All~lines~must~have~an~ampersand}}
```

If there is really arrows in the environment, we draw the arrows:

- if neither option `group` or `groups` is used, we can draw directly ;
- if option `group` or option `groups` is used (`\l_@@_pos_arrows_int > 5`), we have to draw the arrows group by group ; the macro `\@@_draw_arrows:` does the work.

```
276      \int_compare:nNnT \g_@@_arrow_int > 0
277      {\int_compare:nNnTF \l_@@_pos_arrows_int > 5
278       \@@_draw_arrows:
279       {\@@_draw_arrows:nn 1 \g_@@_arrow_int}}
```

We will execute the code specified in the option `CodeAfter`, after some settings.

```
280      \group_begin:
281      \tikzset{every-picture/.style = @@_standard}
```

The command `\NbLines` is not used by `witharrows`. It's only a convenience given to the user.

```
282      \cs_set:Npn \NbLines {\int_use:N \g_@@_line_int}
```

The command `\MultiArrow` is available in `CodeAfter`, and we have a special version of `\Arrow`, called “`\Arrow` in `CodeAfter`” in the documentation.¹⁷

```
283      \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
284      \cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow_code_after
285      \l_@@_code_after_tl
286      \group_end:
```

If the option `footnote` or the option `footnotehyper` is used, then we extract the footnotes with an environment `{footnote}` (of the package `footnote` or the package `footnotehyper`).

```
287      \bool_if:NT \g_@@_footnote_bool {\end{savenotes}}
```

We update the position-in-the-tree. First, we drop the last component and then we increment the last element.

```
288      \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
289      \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
290      \seq_gput_right:Nx \g_@@_position_in_the_tree_seq {\int_eval:n {\l_tmpa_tl+1}}
```

We update the value of the counter `\g_@@_last_env_int`. This counter is used only by the user function `\WithArrowsLastEnv`.

```
291      \int_compare:nNnT {\seq_count:N \g_@@_position_in_the_tree_seq} = 1
292      {\int_gincr:N \g_@@_last_env_int}
```

¹⁷As for now, `\MultiArrow` has no option, and that's why its internal name is a name of `expl3` with the signature `:nn` whereas `\Arrow` in `CodeAfter` provides options and has the name of a function defined with `\NewDocumentCommand`.

Finally, we restore the previous values of the three counters `\g_@@_arrow_int`, `\g_@@_line_int` and `\g_@@_line_bis_int`. It is recalled that we manage three stacks in order to be able to do such a restoration.

```

293         \seq_gpop_right:NN \g_@@_arrow_int_seq {\l_tmpa_tl}
294         \int_gset:Nn \g_@@_arrow_int {\l_tmpa_tl}
295         \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
296         \int_gset:Nn \g_@@_line_int {\l_tmpa_tl}
297         \seq_gpop_right:NN \g_@@_line_bis_int_seq \l_tmpa_tl
298         \int_gset:Nn \g_@@_line_bis_int {\l_tmpa_tl}
299     }

```

That's the end of the environment `{WithArrows}`.

We give now the definition of `\@@_cr`: which is the definition of `\\` in an environment `{WithArrows}`. The two `expl3` commands `\group_align_safe_begin:` and `\group_align_safe_end:` are specifically designed for this purpose: test the token that follows in a `\halign` structure.

First, we remove an eventual token `*` since the commands `\\` and `*` are equivalent in an environment `{WithArrows}` (an environment `{WithArrows}`, like an environment `{aligned}` of `amsmath` is always unbreakable).

```

300 \cs_set_protected:Nn \@@_cr:
301     {\scan_stop:
302      \group_align_safe_begin:
303      \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:}

```

Then, we peek the next token to see if it's a `[`. In this case, the command `\\` has an optional argument which is the vertical skip (`=glue`) to put.

```

304 \cs_set_protected:Nn \@@_cr_i:
305     {\peek_meaning:NTF [ {\@@_cr_ii:} {\@@_cr_ii: [\c_zero_dim]} }
306 \cs_new_protected:Npn \@@_cr_ii: [#1]
307     {\group_align_safe_end:
308      \cr\noalign{\skip_vertical:n {#1 + \l_@@_interline_skip}}
309      \scan_stop:}}

```

According of the documentation of `expl3`, the previous addition in `"#1 + \l_@@_interline_skip"` is really an addition of skips (`=glues`).

8.8 We draw the arrows

`\@@_draw_arrows:` draws the arrows when the option `group` or the option `groups` is used. In both cases, we have to compute the x -value of a group of arrows before actually drawing the arrows of that group. The arrows will actually be drawn by the macro `\@@_draw_arrows:nn`.

```

310 \cs_new_protected:Nn \@@_draw_arrows:
311     { \group_begin:

```

`\l_@@_first_arrow_of_group_int` will be the first arrow of the current group.

`\l_@@_first_line_of_group_int` will be the first line involved in the group of arrows (equal to the initial line of the first arrow of the group because the option `jump` is always positive).

`\l_@@_last_line_of_group_int` will be the last line involved in the group (impossible to guess in advance).

```

312     \int_zero_new:N \l_@@_first_arrow_of_group_int
313     \int_zero_new:N \l_@@_first_line_of_group_int
314     \int_zero_new:N \l_@@_last_line_of_group_int
315     \bool_set_true:N \l_@@_new_group_bool

```

We begin a loop over all the arrows of the environment. Inside this loop, if a group is finished, we will draw the lines of that group.

```

316     \int_set:Nn \l_@@_arrow_int 1
317     \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
318     {

```

We extract from the property list of the current arrow the fields “initial” and “final” and we store these values in `\l_@@_initial_int` and `\l_@@_final_int`. However, we have to do a conversion because the components of a property list are token lists.

```

319     \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
320             {initial} \l_tmpa_tl
321     \int_set:Nn \l_@@_initial_int {\l_tmpa_tl}
322     \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
323             {final} \l_tmpa_tl
324     \int_set:Nn \l_@@_final_int {\l_tmpa_tl}

```

We test if the previous arrow was in fact the last arrow of a group. In this case, we have to draw all the arrows of that group (with the x -value computed in `\g_@@_x_dim`).

```

325     \bool_if:nT {      \int_compare_p:nNn \l_@@_pos_arrows_int = 7
326                     && \int_compare_p:nNn \l_@@_arrow_int > 1
327                     && \int_compare_p:nNn
328                             \l_@@_initial_int > \l_@@_last_line_of_group_int}
329     {\@@_draw_arrows:nn \l_@@_first_arrow_of_group_int {\l_@@_arrow_int - 1}
330      \bool_set_true:N \l_@@_new_group_bool}

```

The flag `\l_@@_new_group_bool` indicates if we have to begin a new group of arrows. In fact, We have to begin a new group in two circumstances: if we are at the first arrow of the environment (that’s why the flag is raised before the beginning of the loop) and if we have just finished a group (that’s why the flag is raised in the previous conditionnal). At the beginning of a group, we have to initialize four variables: `\l_@@_first_arrow_int`, `\l_@@_first_line_of_group_int`, `\l_@@_last_line_of_group` and `\g_@@_x_dim` (global for technical reasons). The last two will evolve during the construction of the group.

```

331     \bool_if:nTF \l_@@_new_group_bool
332     {\bool_set_false:N \l_@@_new_group_bool
333      \int_set:Nn \l_@@_first_arrow_of_group_int \l_@@_arrow_int
334      \int_set:Nn \l_@@_first_line_of_group_int \l_@@_initial_int
335      \int_set:Nn \l_@@_last_line_of_group_int \l_@@_final_int
336      \tikz [@@_standard]
337      \path let \p1=(\int_use:N\l_@@_initial_int-1)
338              in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\x1}} ;
339     }

```

If we are not at the beginning of a new group, we actualize `\l_@@_last_line_of_group_int`.

```

340     {\int_set:Nn \l_@@_last_line_of_group_int
341      {\int_max:nn \l_@@_last_line_of_group_int \l_@@_final_int}}

```

We actualise the current x -value (in `\g_@@_x_dim`) even if we are at the beginning of a group. Indeed, the previous initialisation of `\g_@@_x_dim` only considers the initial line of the arrows and now we consider all the lines between the initial and the final line. This is done with `\@@_actualise_x_value:nn`. We have written a command for this because it is also used with the option `i` (`\l_@@_pos_arrows_int = 5`).

```

342     \@@_actualise_x_value:nn \l_@@_initial_int \l_@@_final_int

```

Incrementation of the index of the loop (and end of the loop).

```

343     \int_incr:N \l_@@_arrow_int
344 }

```

After the last arrow of the environment, you have to draw the last group of arrows.

```

345     \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int
346     \group_end:
347 }

```

The following code is necessary because we will have to expand an argument exactly 3 times.

```

348 \cs_generate_variant:Nn \keys_set:nn {no}
349 \cs_new_protected:Nn \@@_keys_set: {\keys_set:no {WithArrows/General}}

```

The macro `\@@_draw_arrows:nn` draws all the arrows whose numbers are between `#1` and `#2`. `#1` and `#2` must be expressions that expands to an integer (they are expanded in the beginning of the macro).

```

350 \cs_new_protected:Nn \@@_draw_arrows:nn
351   {\group_begin:
352     \int_zero_new:N \l_@@_first_arrow_int
353     \int_set:Nn \l_@@_first_arrow_int {#1}
354     \int_zero_new:N \l_@@_last_arrow_int
355     \int_set:Nn \l_@@_last_arrow_int {#2}

```

We begin a loop over the arrows of the environment. The variable `\l_@@_arrow_int` (local in the environment `{WithArrows}`) will be used as index for the loop.

```

356     \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
357     \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
358     {

```

We extract from the property list of the current arrow the fields “initial” and “final” and we store these values in `\l_@@_initial_int` and `\l_@@_final_int`. However, we have to do a conversion because the components of a property list are token lists.

```

359         \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
360             {initial} \l_tmpa_tl
361         \int_set:Nn \l_@@_initial_int {\l_tmpa_tl}
362         \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
363             {final} \l_tmpa_tl
364         \int_set:Nn \l_@@_final_int {\l_tmpa_tl}

```

If the arrow ends after the last line of the environment, we raise an error.

```

365         \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
366             {\msg_error:nn {witharrows} {Too~few~lines~for~an~arrow}}
367             {\@@_draw_arrows_i:}
368         \int_incr:N \l_@@_arrow_int
369     }
370 \group_end:
371 }

```

The macro `\@@_draw_arrows_i:` is only for the lisibility of the code. This macro will draw the current arrow if the arrow is not impossible (that is to say if the Tikz node exists). The first `\group_begin:` is for the options of the arrow.

```

372 \cs_new:Nn \@@_draw_arrows_i:
373   {\group_begin:
374     \int_set:Nn \l_@@_previous_pos_arrows_int {-1}
375     \int_set:Nn \l_@@_level_int 3

```

We process the options of the current arrow. The second argument of `\keys_set:nn` must be expanded exactly three times. An x-expansion is not possible because there can be tokens like `\bfseries` in the option `font` of the option `tikz`. This expansion is a bit tricky.

```

376     \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str
377                 _\int_use:N\l_@@_arrow_int _prop} {options} \l_tmpa_tl
378     \exp_args:NNo \exp_args:No
379     \@@_keys_set: {\l_tmpa_tl,tikz={xshift = \l_@@_xoffset_dim}}

```

We create two booleans to indicate the position of the initial node and final node of the arrow in cases of options `rr`, `rl`, `lr` or `ll`:

```

380     \bool_set_false:N \l_@@_initial_r_bool
381     \bool_set_false:N \l_@@_final_r_bool
382     \int_case:nn \l_@@_pos_arrows_int
383         {0 {\bool_set_true:N \l_@@_initial_r_bool
384           \bool_set_true:N \l_@@_final_r_bool}
385          2 {\bool_set_true:N \l_@@_initial_r_bool}
386          3 {\bool_set_true:N \l_@@_final_r_bool}}

```

option	rr	ll	rl	lr	v	i	group	groups
\l_@@_pos_arrows_int	0	1	2	3	4	5	6	7

The option v can be used only in \Arrow in CodeAfter (see below).

In case of option i (\l_@@_pos_arrows_int = 5), we have to compute the x -value of the arrow (which is vertical). The computed x -value is stored in \g_@@_x_dim (the same variable used when the option group or the option groups is used). This variable is global for technical reasons: we have to do assignments in a Tikz node.

```

387     \int_compare:nNnT \l_@@_pos_arrows_int = 5
388     {

```

First, we calculate the initial value for \g_@@_x_dim. We use a Tikz command, but, in fact, nothing is drawn. We use this Tikz command only to read the abscissa of a Tikz node.

```

389         \tikz [@@_standard]
390         \path let \p1 = (\int_use:N\l_@@_initial_int-1)
391         in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\x1}} ;

```

A global assignment is necessary because of Tikz.

Then, we will loop to determine the maximal length of the lines between the lines \l_@@_initial_int and \l_@@_final_int... but we have written a command dedicated to this work because it will also be used in @@_draw_arrows:.

```

392     @@_actualise_x_value:nn \l_@@_initial_int \l_@@_final_int
393 }

```

\l_@@_initial_tl contains the name of the Tikz node from which the arrow starts (in normal cases... because with the option i, group and groups, the point will perhaps have another x -value — but always the same y -value). Idem for \l_@@_final_tl.

```

394     \tl_set:Nx \l_@@_initial_tl
395     {\int_use:N\l_@@_initial_int-\bool_if:NTF\l_@@_initial_r_bool rl .south}
396     \tl_set:Nx \l_@@_final_tl
397     {\int_use:N\l_@@_final_int-\bool_if:NTF\l_@@_final_r_bool rl .north}

```

We use “.south” and “.north” because we want a small gap between two consecutive arrows (and the Tikz nodes created have the shape of small vertical segments: use option shownodes to visualize the nodes).

The label of the arrow will be stored in \l_tmpa_tl.

```

398     \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
399     {label}
400     \l_tmpa_tl

```

We have to compute the coordinates of the extremities of the arrow. We use the library calc to define two points \p1 and \p2 and we retrieve the coordinates in \g_tmpa_tl and \g_tmpb_tl. This extraction of the coordinates is necessary because we must give coordinates and not nodes (even anchored) to @@_draw_arrow:nnn to have the xshift correct.

```

401     \int_compare:nNnTF \l_@@_pos_arrows_int < 5
402     {\tikz [@@_standard]
403     \path let \p1 = (\l_@@_initial_tl),
404     \p2 = (\l_@@_final_tl)
405     in \pgfextra { \tl_gset:Nx \g_tmpa_tl {\p1}
406     \tl_gset:Nx \g_tmpb_tl {\p2}} ; }

```

If we use option i or group or groups, we use the abscissa specially computed in \g_@@_x_dim.

```

407     {\tikz [@@_standard]
408     \path let \p1 = (\l_@@_initial_tl),
409     \p2 = (\l_@@_final_tl)
410     in \pgfextra { \tl_gset:Nx \g_tmpa_tl {\dim_use:N \g_@@_x_dim , \y1}
411     \tl_gset:Nx \g_tmpb_tl {\dim_use:N \g_@@_x_dim , \y2}} ; }

```


Eventually, we can draw the arrow with the code in `\l_@@_tikz_code_tl`. We recall that the value by default for this token list is : “`\draw (#1) to node {#3} (#2) ;`”. This value can be modified with the option `TikzCode`.

```
412 \@@_draw_arrow:nnn {\g_tmpa_tl} {\g_tmpb_tl} {\l_tmpa_tl}
```

We close the TeX group opened for the options given to `\Arrow[...]` (local level of the options).

```
413 \group_end: }
```

The function `@@_tmpa:nnn` will draw the arrow. It’s merely an environment `{tikzpicture}`. However, the Tikz instruction in this environment must be inserted from `\l_@@_tikz_code_tl` with the markers `#1`, `#2` and `#3`. That’s why we create a function `\@@_def_function_tmpa:n` which will create the function `\@@_tmpa:nnn`.

```
414 \cs_new_protected:Nn \@@_def_function_tmpa:n
415   {\cs_set:Nn \@@_tmpa:nnn
416     {\begin{tikzpicture}[@@_standard,every-path/.style = {WithArrows/arrow}]
417       #1
418     \end{tikzpicture}}}
```

When we draw the arrow (with `\@@_draw_arrow:nnn`), we create first the function `\@@_tmpa:nnn` and, then, we use the function `\@@_tmpa:nnn` :

```
419 \cs_new_protected:Nn \@@_draw_arrow:nnn
420   {\exp_args:No \@@_def_function_tmpa:n \l_@@_tikz_code_tl
421     \@@_tmpa:nnn {#1} {#2} {#3} }
```

The command `\@@_actualise_x_value:nn` will analyze the lines between `#1` and `#2` in order to modify `\g_@@_x_dim` in consequence. More precisely, `\g_@@_x_dim` is increased if a line longer than the current value of `\g_@@_x_dim` is found. `\@@_actualise_x_value:nn` is used in `\@@_draw_arrows:` (for options group and groups) and in `\@@_draw_arrows:nn` (for option `i`).

```
422 \cs_new_protected:Nn \@@_actualise_x_value:nn
423   {\int_step_inline:nnnn {#1} \c_one {#2}
424     {\tikz [@@_standard]
425       \path let \p1 = (#1-1)
426       in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\dim_max:nn \g_@@_x_dim {\x1}}}; }}
```

The command `\WithArrowsLastEnv` is not used by the package `witharrows`. It’s only a facility given to the final user. It gives the number of the last environment `{WithArrows}` at level 0 (to the sens of the nested environments). This macro is fully expandable and, thus, can be used directly in the name of a Tikz node.

```
427 \cs_new:Npn \WithArrowsLastEnv {\int_use:N \g_@@_last_env_int}
```

8.9 The command `Arrow` in `CodeAfter`

The option `CodeAfter` is an option of the environment `{WithArrows}` (this option is only available at the environment level). In the option `CodeAfter`, one can use the command `Arrow` but it’s a special version of the command `Arrow`. For this special version (internally called `\@@_Arrow_code_after`), we define a special set of keys called `WithArrows/CodeAfter`.

```
428 \keys_define:nn {WithArrows/CodeAfter}
429   {tikz      .code:n          = \tikzset {WithArrows/arrow/.append-style = {#1}} ,
430   tikz      .value_required:n = true,
431   rr        .value_forbidden:n = true,
432   rr        .code:n          = \@@_analyze_option_position:n 0 ,
433   ll        .value_forbidden:n = true,
434   ll        .code:n          = \@@_analyze_option_position:n 1 ,
435   rl        .value_forbidden:n = true,
436   rl        .code:n          = \@@_analyze_option_position:n 2 ,
437   lr        .value_forbidden:n = true,
438   lr        .code:n          = \@@_analyze_option_position:n 3 ,
439   v         .value_forbidden:n = true,
```

```

440 v .code:n = \@@_analyze_option_position:n 4 ,
441 TikzCode .tl_set:N = \l_@@_tikz_code_tl,
442 TikzCode .value_required:n = true,
443 xoffset .dim_set:N = \l_@@_xoffset_dim,
444 xoffset .value_required:n = true}

```

```

445 \NewDocumentCommand \@@_Arrow_code_after {0{}} mmm 0{}
446 {\int_set:Nn \l_@@_pos_arrows_int 1
447 \int_set:Nn \l_@@_previous_pos_arrows_int {-1}
448 \group_begin:

```

Even if `\Arrow` in `CodeAfter` has its own set of options (`WithArrows/CodeAfter`), we set the level of the options to 3 (as with the classical command `\Arrow`) because of the error messages.

```

449 \int_set:Nn \l_@@_level_int 3
450 \keys_set:nn {WithArrows/CodeAfter}
451 {#1,#5,tikz={xshift = \l_@@_xoffset_dim}}
452 \bool_set_false:N \l_@@_initial_r_bool
453 \bool_set_false:N \l_@@_final_r_bool
454 \int_case:nn \l_@@_pos_arrows_int
455 {0 {\bool_set_true:N \l_@@_initial_r_bool
456 \bool_set_true:N \l_@@_final_r_bool}
457 2 {\bool_set_true:N \l_@@_initial_r_bool}
458 3 {\bool_set_true:N \l_@@_final_r_bool}}

```

We test whether the two Tikz nodes (#2-1) and (#3-1) really exist. If not, the arrow won't be drawn.

```

459 \cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-#2-1}
460 {\msg_error:nnn {witharrows} {Wrong-line-specification-in-Arrow} {#2}}
461 {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-#3-1}
462 {\msg_error:nnn {witharrows} {Wrong-line-specification-in-Arrow} {#3}}
463 {\int_compare:nNnTF \l_@@_pos_arrows_int = 4
464 {\tikz [@@_standard]
465 \path let \p1 = (#2-1.south),
466 \p2 = (#3-1.north),
467 \p3 = (\dim_max:nn {\x1} {\x2} , \y1),
468 \p4 = (\dim_max:nn {\x1} {\x2} , \y2)
469 in \pgfextra { \tl_gset:Nx \g_tmpa_tl {\p3}
470 \tl_gset:Nx \g_tmpb_tl {\p4}} ; }
471 {\tikz [@@_standard]
472 \path let \p1 = (#2-\bool_if:NNTF\l_@@_initial_r_bool rl .south),
473 \p2 = (#3-\bool_if:NNTF\l_@@_final_r_bool rl .north)
474 in \pgfextra { \tl_gset:Nx \g_tmpa_tl {\p1}
475 \tl_gset:Nx \g_tmpb_tl {\p2}} ; }
476 \@@_draw_arrow:nnn {\g_tmpa_tl} {\g_tmpb_tl} {#4} }}
477 \group_end:
478 }

```

8.10 MultiArrow

The command `\@@_MultiArrow:nn` will be linked to `\MultiArrow` when the `CodeAfter` is executed.

```

479 \cs_new_protected:Nn \@@_MultiArrow:nn
480 {

```

The user of the command `\MultiArrow` (in `CodeAfter`) will be able to specify the list of lines with the same syntax as the loop `\foreach` of `pgffor`. That's why we construct a “clist” of `expl3` from the specification of list given by the user. The construction of the “clist” must be global in order to exit the `\foreach` and that's why we construct the list in `\g_tmpa_clist`.

```

481 \foreach \x in {#1} {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-\x-1}
482 {\msg_error:nnx {witharrows}
483 {Wrong-line-specification-in-MultiArrow} {\x}}
484 {\clist_gput_right:Nx \g_tmpa_clist {\x}}

```

We sort the list `\g_tmpa_clist` because we want to extract the minimum and the maximum.

```

485 \int_compare:nNnTF {\clist_count:N \g_tmpa_clist} < 2
486   {\msg_error:nn {witharrows} {Too-small-specification-for-MultiArrow}}
487   {\clist_sort:Nn \g_tmpa_clist
488     {\int_compare:nNnTF {##1} > {##2}
489       {\sort_return_swapped:}
490       {\sort_return_same:}}}
```

We extract the minimum in `\l_tmpa_tl` (it must be a integer but we store it in a token list of `expl3`).

```

491 \clist_pop:NN \g_tmpa_clist \l_tmpa_tl
```

We extract the maximum in `\l_tmpb_tl`. The remaining list (in `\g_tmpa_clist` will be sorted in decreasing order but never mind...).

```

492 \clist_reverse:N \g_tmpa_clist
493 \clist_pop:NN \g_tmpa_clist \l_tmpb_tl
```

We draw the teeth of the rak (except the first one and the last one) with the auxiliary function `\@@_MultiArrow_i:n`. This auxiliary function is necessary to expand the specification of the list in the `\foreach` loop. The first and the last teeth of the rak can't be drawn the same way as the others (think, for example, to the case of the option "rounded corners" is used).

```

494 \exp_args:Nx \@@_MultiArrow_i:n {\g_tmpa_clist}
```

Now, we draw the rest of the structure.

```

495 \begin{tikzpicture}[@@_standard,every~path/.style={WithArrows/arrow}]
496   \draw [<->] ($(\l_tmpa_tl-r.south)+(\l_@@_xoffset_dim,0)$)
497     -- ++(5mm,0)
498     -- node {#2} ($(\l_tmpb_tl-r.south)+(\l_@@_xoffset_dim+5mm,0)$)
499     -- ($(\l_tmpb_tl-r.south)+(\l_@@_xoffset_dim,0)$) ;
500 \end{tikzpicture} } }
501
502 \cs_new_protected:Nn \@@_MultiArrow_i:n
503   {\begin{tikzpicture}[@@_standard,every~path/.style={WithArrows/arrow}]
504     \foreach \k in {#1}
505       {\draw [<-] ($(\k-r.south)+(\l_@@_xoffset_dim,0)$) -- ++(5mm,0) ;} ;
506   \end{tikzpicture}}
```

8.11 The error messages of the package

```

507 \msg_new:nnn {witharrows}
508   {Third-column-in-a-{WithArrows}-environment}
509   {By~default,~a~\{WithArrows\}-environment~can~only~have~two~columns.~
510     Maybe~you~have~forgotten~a~\str_use:N \c_backslash_str
511     \str_use:N \c_backslash_str.~If~you~really~want~more~than~two~columns,~
512     you~should~use~the~option~"MoreColumns"~at~a~global~level~or~for~
513     an~environment.~However,~you~can~go~one~for~this~time.}

514 \msg_new:nnn {witharrows}
515   {The~option~"jump"~must~be~non-negative}
516   {You~can't~use~a~strictly~negative~value~for~the~option~"jump"~of~command~
517     \token_to_str:N\Arrow.~ You~can~create~an~arrow~going~backwards~with~
518     the~option~"<-"~of~Tikz.}

519 \msg_new:nnn {witharrows}
520   {Too~few~lines~for~an~arrow}
521   {An~arrow~specified~in~line~\int_use:N \l_@@_initial_int\ can't~be~drawn~
522     because~it~arrives~after~the~last~line~of~the~environment.~If~you~go~on,~
523     this~arrow~will~be~ignored.}

524 \msg_new:nnn {witharrows}
525   {\{WithArrows\}~used~outside~math~mode}
526   {The~environment~\{WithArrows\}~should~be~used~only~in~math~mode.~
527     Nevertheless,~you~can~go~on.}

528 \msg_new:nnn {witharrows}
529   {Two~options~are~incompatible}
```

```

530 {You-try-to-use-the-option~"\tl_use:N\l_keys_key_tl"~but~
531 this-option-is~incompatible-or-redundant-with-the-option~"
532 \int_case:nn\l_@@_previous_pos_arrows_int
533 {0 {rr}
534 1 {ll}
535 2 {rl}
536 3 {lr}
537 4 {i}
538 5 {group}
539 6 {groups}
540 7 {v}}"}~
541 previously-set~in~the~same~
542 \int_case:nn\l_@@_level_int
543 {1 {command~\token_to_str:N\WithArrowsOptions}
544 2 {declaration-of~options~of~the~environment~\{WithArrows\}}
545 3 {command~\token_to_str:N\Arrow}}.~
546 If~you-go-on,~I~will~overwrite~the~first~option.}

547 \msg_new:nnn {witharrows}
548 {All~lines-must-have-an~ampersand}
549 {All~lines-of~an~environment~\{WithArrows\}~must-have-an~second-column~
550 (because~the~nodes-are~created~in~the~second~column).~
551 You-can-go-on-but~maybe-you-will~have-an~pgf~error~for~
552 an~undefined~shape.}
553 {The~ampersand~can~be~implicit~
554 (e.g.~if~you-use~\token_to_str:N\MoveEqLeft\ of~mathtools).}

555 \msg_new:nnn {witharrows}
556 {Option-incompatible-with~"group(s)"}
557 {You-try-to-use-the-option~"\tl_use:N\l_keys_key_tl"~while~
558 you-are~using~the~option~"
559 \int_compare:nNnTF \l_@@_pos_arrows_int = 5
560 {group}
561 {groups}}.~
562 It's~incompatible.~You-can-go-on~ignoring~this~option~
563 "\tl_use:N\l_keys_key_tl"~but~you~should~correct~your~code.}

564 \msg_new:nnn {witharrows}
565 {Option-will-be-ignored}
566 {The-option~"\tl_use:N\l_keys_key_tl"~can't~be-used~here.~
567 If~you-go-on,~it~will~be~ignored.}

568 \msg_new:nnn {witharrows}
569 {Option-unknown}
570 {The-option~"\tl_use:N\l_keys_key_tl"~is~unknown~or~
571 meaningless~in~the~context.~If~you-go-on,~it~will~be~ignored.}

572 \msg_new:nnn {witharrows}
573 {Arrow-in-first-column}
574 {You-should-not-use~the~command~\token_to_str:N\Arrow\
575 \tl_if_eq:NnF \l_@@_CommandName_tl \l_tmpa_tl
576 {(renamed~in~\str_use:N \c_backslash_str
577 \tl_use:N \l_@@_CommandName_tl)~}
578 ~in~the~first~column~but~only~in~the~second~column.~
579 This~is~a~restriction~of~the~version~1.3~of~the~
580 package~witharrows~(in~the~aim~of~developping~further~
581 ~a~new~functionality~with~\token_to_str:N\Arrow\ in~the~
582 first~column).\}
583 However~you~can~go~on~for~this~time.}

584 \msg_new:nnn {witharrows}
585 {Wrong-line-specification~in~Arrow}
586 {The-specification~of~line~"#1"~you-use~in~\token_to_str:N\Arrow\
587 ~doesn't~exist.\}
588 If~you-go-on,~the~arrow~will~be~ignored.}

589 \msg_new:nnn {witharrows}
590 {Wrong-line-specification~in~MultiArrow}

```

```

591      {The~specification~of~line~"#1"~doesn't~exist.\\
592      If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.}
593 \msg_new:nnn {witharrows}
594      {Too~small~specification~for~MultiArrow}
595      {The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
596      is~too~small:~we~need~at~least~two~lines.~If~you~go~on,~the~
597      command~\token_to_str:N\MultiArrow\ ~will~be~ignored.}

```

9 History

9.1 Changes between versions 1.0 and 1.1

Option for the command `\\` and option `interline`
 Compatibility with `\usetikzlibrary{babel}`
 Possibility of nested environments `{WithArrows}`
 Better error messages
 Creation of a DTX file

9.2 Changes between versions 1.1 and 1.2

The package `witharrows` can now be loaded without having loaded previously `tikz` and the Tikz libraries `calc`, `arrow.meta` and `bending`.
 New option groups (with a *s*)
 Better error messages

9.3 Changes between versions 1.2 and 1.3

New options `ygap` and `ystart` for fine tuning.
 Minor bugs.

9.4 Changes between versions 1.3 and 1.4

The package `footnote` is no longer loaded by default. Instead, two options `footnote` and `footnotehyper` have been added. In particular, `witharrows` becomes compatible with `beamer`.

9.5 Changes between versions 1.4 and 1.5

The Tikz code used to draw the arrows can be changed with the option `TikzCode`.
 Two new options `CodeBefore` and `CodeAfter` have been added at the environment level.
 A special version of `\Arrow` is available in `CodeAfter` in order to draw arrows in nested environments.
 A command `\MultiArrow` is available in `CodeAfter` to draw arrows of other shapes.