

The package `witharrows` for plain-TeX and LaTeX*

F. Pantigny
fpantigny@wanadoo.fr

July 30, 2019

Abstract

The LaTeX package `witharrows` provides environments `{WithArrows}` and `{DispWithArrows}` similar to the environments `{aligned}` and `{align}` of `amsmath` but with 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.

In this document, we describe the LaTeX extension `witharrows` (however, `witharrows` can also be used with plain-TeX: see p. 21). This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). This package loads the packages `expl3`, `l3keys2e`, `xparse`, `tikz` and the Tikz libraries `arrows.meta` and `bending`. The arrows are drawn with Tikz and that's why several compilations may be necessary.

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

```


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


```

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

The arrow has been drawn with the command `\Arrow` on the row 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 row as in the previous example).

The environment `{WithArrows}` bears similarities with the environment `{aligned}` of `amsmath` (and `mathtools`). The extension `witharrows` also provides an environment `{DispWithArrows}` which is similar to the environment `{align}` of `amsmath`: cf. p. 16.

1 Options for the shape of the arrows

The command `\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 rows the arrow must jump (the default value is, of course, 1).

```


$$A = \bigl((a+b)+1\bigr)^2 \quad \text{we expand}$$


```

*This document corresponds to the version 2.0 of `witharrows`, at the date of 2019/07/30.

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.

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

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

```

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

```

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

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

```

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

```

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

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.

```

 $\begin{WithArrows}
A &= (a+1)^2 \ \Arrowleft[tikz=thick]{we expand} \ \backslash
&= a^2 + 2a + 1
\end{WithArrows}$ 

```

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

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

```

 $\begin{WithArrows}
A &= (a+1)^2 \ \Arrowleft[tikz=<-]{we factorize} \ \backslash
&= a^2 + 2a + 1
\end{WithArrows}$ 

```

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

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

```

 $\begin{WithArrows}
A &= (a+1)^2 \ \Arrowleft[tikz=--]{very classical} \ \backslash
&= a^2 + 2a + 1
\end{WithArrows}$ 

```

$$A = (a + 1)^2 \quad \left. \begin{array}{l} \\ \\ \end{array} \right) \textit{very classical}$$

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

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}

```

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

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

In fact, it’s possible to change more drastically the shape or the arrows with the option `tikz-code` (presented p. 21).

It’s possible to use the Tikz option “`text width`” to control the width of the text associated to the arrow.²

```

 $\begin{WithArrows}$ 
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}

```

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

$$= (a + b)^2 + 2(a + b) + 1$$

$$= a^2 + 2ab + b^2 + 2a + 2b + 1$$

In the environments `{DispWithArrows}` and `{DispWithArrows*}`, there is an option `wrap-lines`. With this option, the lines of the labels are automatically wrapped on the right: see p. 18.

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}

```

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

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

It’s possible to put commands `\` in the text to force new lines³. However, if we put a `\`, 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 an 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.

²It’s possible to avoid the hyphenations of the words with the option “`align = flush left`” of Tikz.

³By default, this is not possible in a Tikz node. However, in `witharrows`, the nodes are created with the option `align=left`, and, thus, it becomes possible.

```

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

```

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

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

The options can be given directly between square brackets to the environment `{WithArrows}`. There must be no space between the `\begin{WithArrows}` and the opening bracket (`[`) of the options of the environment. Such options apply to all the arrows of the environment.⁴

```

 $\begin{WithArrows}[tikz=blue]$ 
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}

```

$$\begin{aligned}
A &= ((a+b)+1)^2 \\
&= (a+b)^2 + 2(a+b) + 1 \quad \left. \vphantom{A} \right\} \text{first expansion.} \\
&= a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \left. \vphantom{A} \right\} \text{second expansion.}
\end{aligned}$$

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}

```

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

The same example with the option `displaystyle`:

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

⁴They also apply to the nested environments `{WithArrows}` (with the logical exceptions of `interline`, `code-before` and `code-after`).

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}`.⁵

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

$$\begin{aligned} \sum_{i=1}^n (x_i + 1)^2 &= \sum_{i=1}^n (x_i^2 + 2x_i + 1) \\ &= \sum_{i=1}^n x_i^2 + 2 \sum_{i=1}^n x_i + n \end{aligned} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \textit{by linearity}$$

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 `command-name`⁶ for this purpose. The new name of the command must be given to the option *without* the leading backslash.

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

$$\begin{aligned} f &= (x \mapsto (x + 1)^2) \\ &= (x \mapsto x^2 + 2x + 1) \end{aligned} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \textit{we work directly on fonctions}$$

The environment `{WithArrows}` provides also two options `code-before` and `code-after`⁷ 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 do not apply to the nested environments).

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

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

Special commands are available in `code-after`: a command `\WithArrowsNbLines` which gives the number of lines (=rows) 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. 12.

⁵It’s also possible to configure `witharrows` by modifying the Tikz style `WithArrows/arrow` which is the style used by `witharrows` when drawing an arrow. For example, to have the labels in blue with roman (upright) types, one can use the following instruction: `\tikzset{WithArrows/arrow/.append style = {blue,font = {}}}`.

⁶For historical reasons, there is an alias for this option: `CommandName`.

⁷For historical reasons, there are aliases for these options: `CodeBefore` and `CodeAfter`.

2 Numbers of columns

So far, we have used the environment `{WithArrows}` with two columns. However, it's possible to use the environment with an arbitrary number of columns with the option `format`. The value given to this option is like the preamble of an environment `{array}`, that is to say a sequence of letters `r`, `c` and `l`. The default value of the option `format` is, in fact, `rl`.

For example, if we want only one column left-aligned, we use the option `format=l`.

```
\begin{WithArrows}[format = l]
f(x) \ge g(x) \Arrow{by squaring both sides} \\
f(x)^2 \ge g(x)^2 \Arrow{by moving to left side} \\
f(x)^2 - g(x)^2 \ge 0
\end{WithArrows}
```

$$\begin{array}{l} f(x) \geq g(x) \\ f(x)^2 \geq g(x)^2 \\ f(x)^2 - g(x)^2 \geq 0 \end{array} \begin{array}{l} \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{by squaring both sides} \\ \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{by moving to left side} \end{array}$$

In the following example, we use five columns all centered (the environment `{DispWithArrows*}` is presented p. 16).

```
\begin{DispWithArrows*}[format = ccccc,
                        wrap-lines,
                        tikz = {align = flush left},
                        interline=1mm]
k & \leq & t & \leq & k+1 \\
\frac{1}{k+1} & \leq & \frac{1}{t} & \leq & \frac{1}{k} \\
\Arrow{we can integrate the inequalities since $k \leq k+1$ } \\
\int\limits_k^{k+1} \frac{dt}{k+1} & \leq & \int\limits_k^{k+1} \frac{dt}{t} & \leq & \int\limits_k^{k+1} \frac{dt}{k} \\
& \leq & \ln(k+1) - \ln(k) & \leq & \frac{1}{k} \\
\end{DispWithArrows*}
```

$$\begin{array}{ccccc} k & \leq & t & \leq & k+1 \\ \frac{1}{k+1} & \leq & \frac{1}{t} & \leq & \frac{1}{k} \\ \int_k^{k+1} \frac{dt}{k+1} & \leq & \int_k^{k+1} \frac{dt}{t} & \leq & \int_k^{k+1} \frac{dt}{k} \\ \frac{1}{k+1} & \leq & \ln(k+1) - \ln(k) & \leq & \frac{1}{k} \end{array} \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \text{we can integrate the} \\ \text{inequalities since } k \leq k+1$$

3 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.⁸

⁸The option `show-nodes` 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.

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

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.

$$\begin{aligned}
\text{Therefore } I &= \int_{\frac{\pi}{4}}^0 \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right)(-du) \quad \text{This arrow uses the } lr \text{ option.} \\
&= \int_0^{\frac{\pi}{4}} \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right) du \\
&= \int_0^{\frac{\pi}{4}} \ln\left(1 + \frac{1 - \tan u}{1 + \tan u}\right) du \\
&= \int_0^{\frac{\pi}{4}} \ln\left(\frac{1 + \tan u + 1 - \tan u}{1 + \tan u}\right) du \\
&= \int_0^{\frac{\pi}{4}} \ln\left(\frac{2}{1 + \tan u}\right) du \quad \text{This arrow uses a } ll \text{ option and a} \\
&= \int_0^{\frac{\pi}{4}} (\ln 2 - \ln(1 + \tan u)) du \quad \text{jump equal to 2} \\
&= \frac{\pi}{4} \ln 2 - \int_0^{\frac{\pi}{4}} \ln(1 + \tan u) du \\
&= \frac{\pi}{4} \ln 2 - I
\end{aligned}$$

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

```

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

```

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

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.

```

 $\begin{WithArrows}[displaystyle,group]
2xy'-3y=\sqrt{x}
& \Leftrightarrow 2x(K'y_0+Ky'_0)-3Ky_0 = \sqrt{x} \\
& \Leftrightarrow 2xK'y_0 + K(2xy'_0-3y_0) = \sqrt{x} \\
& \Leftrightarrow 2xK'y_0 = \sqrt{x} \quad \downarrow \text{we replace } y_0 \text{ by its value} \\
& \Leftrightarrow 2xK'x^{\frac{3}{2}} = x^{\frac{1}{2}} \quad \downarrow \text{simplification of the } x \\
& \Leftrightarrow K' = \frac{1}{2x^2} \quad \downarrow \text{antiderivation} \\
& \Leftrightarrow K = -\frac{1}{2x}
\end{WithArrows}$ 

```

$$\begin{aligned}
2xy' - 3y = \sqrt{x} &\iff 2x(K'y_0 + Ky'_0) - 3Ky_0 = \sqrt{x} \\
&\iff 2xK'y_0 + K(2xy'_0 - 3y_0) = \sqrt{x} \\
&\iff 2xK'y_0 = \sqrt{x} \\
&\iff 2xK'x^{\frac{3}{2}} = x^{\frac{1}{2}} \\
&\iff K' = \frac{1}{2x^2} \\
&\iff K = -\frac{1}{2x}
\end{aligned}$$

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 connected⁹ arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

$$\begin{aligned}
A &= B \\
&= C + D \\
&= D' \\
&= E + F + G + H + I \\
&= K + L + M \\
&= N \\
&= O
\end{aligned}$$

In an environment which uses the option `group` or the option `groups`, it’s still possible to give an option of position (`ll`, `lr`, `rl`, `rr` or `i`) to an individual arrow. Such arrow will be drawn irrespective of the groups. It’s also possible to start a new group by applying the option `new-group` to an given arrow.

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]`

In the following example, we have used the option `group` for the environment and the option `rr` for the last arrow (that’s why the last arrow is not aligned with the others).

⁹More precisely: for each arrow *a*, we note *i(a)* the number of its initial row and *f(a)* the number of its final row; 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 ~.

$$\begin{aligned}
\sum_{k=0}^n \frac{\cos kx}{\cos^k x} &= \sum_{k=0}^n \frac{\Re(e^{ikx})}{(\cos x)^k} \\
&= \sum_{k=0}^n \Re\left(\frac{e^{ikx}}{(\cos x)^k}\right) && \left. \begin{array}{l} (\cos x)^k \text{ is real} \\ \Re(z+z') = \Re(z) + \Re(z') \end{array} \right\} \\
&= \Re\left(\sum_{k=0}^n \left(\frac{e^{ix}}{\cos x}\right)^k\right) && \left. \begin{array}{l} \text{sum of terms of a geometric progression} \\ \text{algebraic calculation} \end{array} \right\} \\
&= \Re\left(\frac{1 - \left(\frac{e^{ix}}{\cos x}\right)^{n+1}}{1 - \frac{e^{ix}}{\cos x}}\right) \\
&= \Re\left(\frac{1 - \frac{e^{i(n+1)x}}{\cos^{n+1} x}}{1 - \frac{e^{ix}}{\cos x}}\right) && \left. \begin{array}{l} \text{reduction to common denominator} \\ \Re(kz) = k \cdot \Re(z) \text{ if } k \text{ is real} \end{array} \right\} \\
&= \Re\left(\frac{\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos^{n+1} x}}{\frac{\cos x - e^{ix}}{\cos x}}\right) \\
&= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos x - e^{ix}}\right) \\
&= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - (\cos(n+1)x + i \sin(n+1)x)}{\cos x - (\cos x + i \sin x)}\right) && \left. \begin{array}{l} \text{algebraic form of the complexes} \end{array} \right\} \\
&= \frac{1}{\cos^n x} \Re\left(\frac{(\cos^{n+1} x - \cos(n+1)x) - i \sin(n+1)x}{-i \sin x}\right) \\
&= \frac{1}{\cos^n x} \cdot \frac{\sin(n+1)x}{\sin x}
\end{aligned}$$

4 The options “up” and “down” for individual arrows

At the local level, there are also two options for individual arrows, called “up” and “down”. The following example illustrates these types of arrows:

```

\(\begin{WithArrows}
A &= B
\Arrow[up]{an arrow of type \texttt{up}} \\\
&= C + C + C + C + C + C + C + C + C \\\
&= C + C + C + C + C + C + C + C
\Arrow[down]{an arrow of type \texttt{down}} \\\
&= E + E
\end{WithArrows}\)

```

$$\begin{array}{l}
A = B \quad \xrightarrow{\text{an arrow of type up}} \\
= C + C + C + C + C + C + C + C + C \\
= C + C + C + C + C + C + C + C \\
= E + E \quad \xleftarrow{\text{an arrow of type down}}
\end{array}$$

The options `up` and `down` require the package `varwidth` and the Tikz library `calc`. If they are not loaded, an error will be raised.

5 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}`.¹⁰

¹⁰In fact, it's possible to use the package `witharrows` without the package `amsmath`.

As in the environments of `amsmath`, it's possible to change the spacing between two given rows 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). This option is designed to be used with positive values only.

```

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

```

$$\begin{array}{l}
 A = (a + 1)^2 \\
 = a^2 + 2a + 1
 \end{array}
 \left. \vphantom{\begin{array}{l} A \\ \end{array}} \right) \textit{we expand}$$

In the environments of `amsmath` (or `mathtools`), the spacing between rows 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 a given environment.¹¹

```

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

```

$$\begin{array}{l}
 F = \frac{1}{2}G \\
 = H + \frac{1}{2}K \\
 = K
 \end{array}
 \left. \vphantom{\begin{array}{l} F \\ = \\ = \end{array}} \right) \textit{we expand} \\
 \left. \vphantom{\begin{array}{l} = \\ = \end{array}} \right) \textit{we go on}$$

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 & \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0 \\
\Arrow{\$x\$ and \$y\$ are real} \\
& \Leftrightarrow \left\{ \begin{array}{l} x+y=0 \\ x+2y=0 \end{array} \right. \\
\begin{aligned}
x+y & = 0 \\
x+2y & = 0
\end{aligned} \\
\right. \\
\end{WithArrows}

```

$$\varphi(x,y) = 0 \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0$$

$$\Leftrightarrow \left\{ \begin{array}{l} x+y=0 \\ x+2y=0 \end{array} \right. \left. \vphantom{\varphi(x,y) = 0} \right) \textit{x and y are real}$$

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.

¹¹It's also possible to change `\jot` with the environment `{spreadlines}` of `mathtools`.

```

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

```

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

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

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

```

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

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

```

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

```

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

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}`:

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

The following is composed with `{WithArrows}[c,displaystyle]`. The results are strictly identical.¹²

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

6 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`, `code-before` and `code-after`). 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{the same equation}
& \Leftrightarrow
\left. \begin{array}{l}
x+2y = 0
\end{array} \right.
& \Leftrightarrow x+2y=0
\end{WithArrows}
```

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

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

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

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

A command `\Arrow in code-after` takes three arguments :

- a specification of the start row of the arrow ;
- a specification of the end row of the arrow ;

¹²In versions of `amsmath` older than the 5 nov. 2016, a thin space was added on the left of an environment `{aligned}`. The new versions do not add this space and neither do `{WithArrows}`.

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

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

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

The arrow we want to draw starts in the row 2 of the sub-environment number 1 (and therefore, the specification is 1-2) and ends in the row 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 `code-after` :

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

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

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

With the option `v`, the arrow drawn is vertical to an abscissa computed with the start row and the end row 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 `code-after`. However, it's always possible to translate an arrow with `xoffset` (or `xshift` of Tikz).

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

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

The package `witharrows` gives also another command available only in `code-after`: the command `\MultiArrow`. This command draws a “rak”. The list of the rows 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 `pgffor`.

```
\begin{WithArrows}[tikz = rounded corners,
                    code-after = {\MultiArrow{1,...,4}{text}} ]
A & = B \\
  & = C \\
  & = D \\
  & = E \\
  & = F
\end{WithArrows}
```

As of now, there is no option available for the command `\MultiArrow` (maybe in a future release).

7 Arrows from outside environments `{WithArrows}`

If someone wants to draw arrows from outside the environments `{WithArrows}`, he can use the Tikz nodes created in the environments.

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 row number in the environment. At the end, we have the suffixe `l` for a “left node” and `r` for a “right node”.

For illustrative purposes, we give an example of nested environments `{WithArrows}`, and, for each “right node”, the name of that node.¹³

$$\begin{aligned}
 & A \triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B + B \text{wa-40-1-r} \\
 & \triangleleft \left\{ \begin{array}{l} C \triangleleft D \text{wa-40-1-1-r} \\ E \triangleleft F \text{wa-40-1-2-r} \end{array} \right. \text{wa-40-2-r} \\
 & \triangleleft \left\{ \begin{array}{l} G \triangleleft H + H + H + H + H + H + H \text{wa-40-2-1-r} \\ I \triangleleft \left\{ \begin{array}{l} J \triangleleft K \text{wa-40-2-1-1-r} \\ L \triangleleft M \text{wa-40-2-1-2-r} \end{array} \right. \text{wa-40-2-2-r} \end{array} \right. \text{wa-40-3-r} \\
 & \triangleleft \left\{ \begin{array}{l} N \triangleleft O \text{wa-40-3-1-r} \\ P \triangleleft Q \text{wa-40-3-2-r} \end{array} \right. \text{wa-40-4-r}
 \end{aligned}$$

The package `witharrows` provides some tools facilitating the use of these nodes:

- the command `\WithArrowsLastEnv` gives the number of the last environment of level 0 (*i.e.* which is not included in another environment of the package `witharrows`);
- a name can be given to a given environment with the option `name` and, in this case, the nodes created in the environment will have aliases constructed with this name;
- the Tikz style `WithArrows/arrow` is the style used by `witharrows` when drawing an arrow¹⁴;

¹³There is an option `show-node-names` to show the names of these nodes.

¹⁴More precisely, this style is given to the Tikz option “`every path`” before drawing the arrow with the code of the option `tikz-code`. This style is modified (in TeX scopes) by the option `tikz` of `witharrows`.

- the Tikz style `WithArrows/arrow/tips` is the style for the tip of the arrow (loaded by `WithArrows/arrow`).

For example, we can draw an arrow from `wa-40-2-1-2-r.south` to `wa-40-3-2-r.north` with the following Tikz command.

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

$$\begin{array}{l}
 A \triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B \\
 \triangleleft \left\{ \begin{array}{l} C \triangleleft D \\ E \triangleleft F \end{array} \right. \\
 \triangleleft \left\{ \begin{array}{l} G \triangleleft H + H + H + H + H + H + H \\ I \triangleleft \left\{ \begin{array}{l} J \triangleleft K \\ L \triangleleft M \end{array} \right. \end{array} \right. \\
 \triangleleft \left\{ \begin{array}{l} N \triangleleft O \\ P \triangleleft Q \end{array} \right. \leftarrow
 \end{array}$$

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

In the following example, we create two environments `{WithArrows}` named “`first`” and “`second`” and we draw a line between a node of the first and a node of the second.

```

\begin{WithArrows}[name=first]
A & = B \\
& = C
\end{WithArrows}

\bigskip
\begin{WithArrows}[name=second]
A' & = B' \\
& = C'
\end{WithArrows}

\begin{tikzpicture}[remember picture,overlay]
\draw [WithArrows/arrow]
      ([xshift=3mm]first-1-r.south)
      to ([xshift=3mm]second-1-r.north) ;
\end{tikzpicture}
```

$$\begin{array}{l}
 A = B \\
 = C \\
 A' = B' \\
 = C'
 \end{array}$$

8 The environment `{DispWithArrows}`

As previously said, the environment `{WithArrows}` bears similarities with the environment `{aligned}` of `amsmath` (and `mathtools`). This extension also provides an environment `{DispWithArrows}` which is similar to the environments `{align}` and `{flalign}` of `amsmath`.

The environment `{DispWithArrows}` must be used *outside* math mode. Like `{align}`, it should be used in horizontal mode.

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

$$A = (a + 1)^2 \tag{1}$$

$$= a^2 + 2a + 1 \tag{2}$$

It's possible to use the command `\notag` (or `\nonumber`) to suppress a tag.

It's possible to use the command `\tag` to put a special tag (e.g. `*`).

It's also possible to put a label to the line of an equation with the command `\label`.

These commands must be in the second column of the environment.

```
\begin{DispWithArrows}
A & = (a+1)^2 \Arrow{we expand} \notag \\
& = a^2 + 2a + 1 \tag{$\star$} \label{my-equation}
\end{DispWithArrows}
```

$$A = (a + 1)^2 \tag{*}$$

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

A link to the equation `(*)`. This link has been composed with `\eqref{my-equation}` (the command `\eqref` is a command of `amsmath`).

If `amsmath` (or `mathtools`) is loaded, it's also possible to use `\tag*` which, as in `amsmath`, typesets the tag without the parenthesis. For example, it's possible to use it to put the symbol `\square` of `amssymb`. This symbol is often used to mark the end of a proof.¹⁵

```
\begin{DispWithArrows}
A & = (a+1)^2 \Arrow{we expand} \notag \\
& = a^2 + 2a + 1 \tag*{$\square$}
\end{DispWithArrows}
```

$$A = (a + 1)^2$$

$$= a^2 + 2a + 1 \tag{\square}$$

It's also possible to suppress all the autogenerated numbers with the boolean option `notag` (or `nonumber`), at the global or environment level. There is also an environment `{DispWithArrows*}` which suppresses all these numbers.¹⁶

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

¹⁵Notice that the environment `{DispWithArrows}` is compatible with the command `\qedhere` of `amsthm`.

¹⁶Even in this case, it's possible to put a "manual tag" with the command `\tag`.

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

In fact, there is also another option `tagged-lines` which can be used to control the lines that will be tagged. The value of this option is a list of the numbers of the lines that must to be tagged. For example, with the option `tagged-lines = {first,3,last}`, only the first, the third and the last line of the environment will be tagged. There is also the special value `all` which means that all the lines will be tagged.

```

\begin{DispWithArrows}[tagged-lines = last]
A & = A_1 \Arrow{first stage} \\
& = A_2 \Arrow{second stage} \\
& = A_3 \\
\end{DispWithArrows}

```

$$\begin{aligned}
 A &= A_1 \\
 &= A_2 \\
 &= A_3 \quad \left. \vphantom{A} \right\} \textit{second stage}
 \end{aligned}
 \tag{3}$$

With the option `fleqn`, the environment is composed flush left (in a way similar to the option `fleqn` of the standard classes of LaTeX). In this case, the left margin can be controlled with the option `mathindent` (with a name inspired by the parameter `\mathindent` of standard LaTeX). The default value of this parameter is 25 pt.

```

\begin{DispWithArrows}[fleqn,mathindent = 1cm]
A & = (a+1)^2 \Arrow{we expand} \\
& = a^2 + 2a + 1 \\
\end{DispWithArrows}

```

$$\begin{aligned}
 A &= (a + 1)^2 \\
 &= a^2 + 2a + 1 \quad \left. \vphantom{A} \right\} \textit{we expand}
 \end{aligned}
 \tag{4}$$

Remark : By design, the option `fleqn` of `witharrows` is independent of the option `fleqn` of LaTeX. Indeed, since the environments of `witharrows` are meant to be used with arrows on the right side, the user may want to use `witharrows` with the option `fleqn` (in order to have more space on the right of the equations for the arrows) while still centering the classical equations.

If the option `leqno` is used as a class option, the labels will be composed on the left also for the environments `{DispWithArrows}` and `{DispWithArrows*}`.¹⁷

If the package `amsmath` is loaded, it's possible to use the command `\intertext` in the environments `{DispWithArrows}`. It's also possible to use the environment `{subequations}`. However, there is, for the environments `{DispWithArrows}` an option `subequations` to encapsulate the environment in an environment `{subequations}`.

In the following example, the key `{subequations}` is fixed by the command `\WithArrowsOptions`. Each environment `{DispWithArrows}` will subnumerated (in the scope of the `\WithArrowsOptions`)

```

\WithArrowsOptions{subequations}
First environment.
\begin{DispWithArrows}
A & = B \\
& = C \\
\end{DispWithArrows}
Second environment.

```

¹⁷The package `amsmath` has an option `leqno` but `witharrows`, of course, is not aware of that option: `witharrows` only checks the option `leqno` of the document class.

```
\begin{DispWithArrows}
D & = E \\
& = F
\end{DispWithArrows}
```

First environment.

$$A = B \tag{6a}$$

$$= C \tag{6b}$$

Second environment.

$$D = E \tag{7a}$$

$$= F \tag{7b}$$

If there is not enough space to put the tag at the end of a line, there is no automatic positioning of the label on the next line (as in the environments of `amsmath`). However, in `{DispWithArrows}`, the user can use the command `\tagnextline` to manually require the composition of the tag on the following line.

```
\begin{DispWithArrows}[displaystyle]
S_{2(p+1)}
& = \sum_{k=1}^{2(p+1)} (-1)^k k^2 \\
& \smash[b]{= \sum_{k=1}^{2p} (-1)^k k^2}
& \quad + (-1)^{2p+1} (2p+1)^2 + (-1)^{2p+2} (2p+2)^2 \tagnextline \\
& = S_{2p} - (2p+1)^2 + (2p+2)^2 \\
& = p(2p+1) - (2p+1)^2 + (2p+2)^2 \\
& = 2p^2 + 5p + 3
\end{DispWithArrows}
```

$$\left. \begin{aligned}
S_{2(p+1)} &= \sum_{k=1}^{2(p+1)} (-1)^k k^2 & (8) \\
&= \sum_{k=1}^{2p} (-1)^k k^2 + (-1)^{2p+1} (2p+1)^2 + (-1)^{2p+2} (2p+2)^2 & (9) \\
&= S_{2p} - (2p+1)^2 + (2p+2)^2 & (10) \\
&= 2p^2 + p - 4p^2 - 4p - 1 + 4p^2 + 8p + 4 & (11) \\
&= 2p^2 + 5p + 3 & (12)
\end{aligned} \right|$$

The environments `{DispWithArrows}` and `{DispWithArrows*}` provide an option `wrap-lines`. With this option, the lines of the label are automatically wrapped on the right.¹⁸

```
\begin{DispWithArrows*}[displaystyle,wrap-lines]
S_n
& = \frac{1}{n} \operatorname{Re} \left( \sum_{k=0}^{n-1} \operatorname{bigl}(e^{i \frac{\pi}{2n}} \operatorname{bigr})^k \right)
\Arrow{sum of terms of a geometric progression of ratio $e^{i \frac{2\pi}{n}}$} \\
& = \frac{1}{n} \operatorname{Re} \left( \frac{1 - \operatorname{bigl}(e^{i \frac{\pi}{2n}} \operatorname{bigr})^n}{1 - e^{i \frac{\pi}{2n}}} \right)
\Arrow{This line has been wrapped automatically.} \\
& = \frac{1}{n} \operatorname{Re} \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{DispWithArrows*}
```

¹⁸It's possible to avoid the hyphenations of the words with the option `"align = flush left"` of `Tikz`.

$$\begin{aligned}
S_n &= \frac{1}{n} \Re \left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - \left(e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{aligned}$$

sum of terms of a geometric progression of ratio $e^{i \frac{2\pi}{n}}$
This line has been wrapped automatically.

The option `wrap-lines` doesn't apply to the environments `{WithArrows}` nested in an environment `{DispWithArrows}` or `{DispWithArrows*}`. However, it applies to the instructions `\Arrow` and `\MultiArrow` of the code-after of the environments `{DispWithArrows}` or `{DispWithArrows*}`.

We have said that the environments `{DispWithArrows}` and `{DispWithArrows*}` should be used in horizontal mode and not in vertical mode. However, there is an exception. These environments can be used directly after a `\item` of a LaTeX list. In this case, no vertical space is added before the environment.¹⁹

Here is an example. The use of `{DispWithArrows}` gives the ability to tag an equation (and also to use `wrap-lines`).

```

\begin{enumerate}
\item
\begin{DispWithArrows}%
[displaystyle, wrap-lines, tagged-lines = last, fleqn, mathindent = 0 pt]
S_n
& = \frac{1}{n} \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right)
\Arrow{we use the formula for a sum of terms of a geometric progression of
ratio  $e^{i \frac{2\pi}{n}}$ }
& = \frac{1}{n} \Re \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right)
\Arrow{\$ \left( e^{i \frac{\pi}{2n}} \right)^n = e^{i \frac{\pi}{2} = i} \$}
& = \frac{1}{n} \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{DispWithArrows}
\end{enumerate}

```

$$\begin{aligned}
1. S_n &= \frac{1}{n} \Re \left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - \left(e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{aligned}$$

we use the formula for a sum of terms of a geometric progression of ratio $e^{i \frac{2\pi}{n}}$
 $\left(e^{i \frac{\pi}{2n}} \right)^n = e^{i \frac{\pi}{2}} = i$
(13)

The environment `{DispWithArrows}` is similar to the environment `{align}` of `amsmath`. However, `{DispWithArrows}` is not constructed upon `{align}` (in fact, it's possible to use `witharrows` without `amsmath`).

There are differences between `{DispWithArrows}` and `{align}`.

- The environment `{DispWithArrows}` cannot be inserted in an environment `{gather}` of `amsmath`.

¹⁹It's possible to disable this feature with the option `standard-behaviour-with-items`.

- An environment `{DispWithArrows}` is always unbreakable (even with `\allowdisplaybreaks` of `amsmath`).
- The commands `\label`, `\tag`, `\notag` and `\nonumber` are allowed only in the last column.
- After an `\item` of a LaTeX list, no vertical space is added (this can be changed with the option `standard-behaviour-with-items`).
- **Last but not least, by default, the elements of a `\{DispWithArrows\}` are composed in `textstyle` and not in `displaystyle` (it's possible to change this point with the option `displaystyle`).**

Concerning the references, the package `witharrows` is compatible with the extensions `autonum`, `cleveref`, `fancyref`, `fncylab`, `hyperref`, `listbls`, `prettyref`, `refcheck`, `refstyle`, `showlabels`, `smartref`, `typedref` and `varioref`, and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.²⁰ It is not compatible with `showkeys` (not all the labels are shown).

8.1 The option `<...>` of `DispWithArrows`

The environment `{DispWithArrows}` provides an option `left-brace`. When present, the value of this option is composed on the left, followed by a curly brace (hence the name) and the body of the environment.²¹

For lisibility, this option `left-brace` is also available with a special syntax: it's possible to give this option between angle brackets (`<` and `>`) just after `{DispWithArrows}` (before the optional arguments between square brackets).

The following code is an example of multi-case equations.²²

```
\begin{DispWithArrows}< \binom{n}{p} = >[format = ll,fleqn,displaystyle]
0 & \quad \text{if } p > n
\Arrow{if fact, it's a special case\\ of the following one} \\
\frac{n(n-1)\cdots(n-p+1)}{p!} & \quad \text{if } 0 \leq p \leq n \\
0 & \quad \text{if } p < 0
\end{DispWithArrows}
```

$$\binom{n}{p} = \begin{cases} 0 & \text{if } p > n \\ \frac{n(n-1)\cdots(n-p+1)}{p!} & \text{if } 0 \leq p \leq n \\ 0 & \text{if } p < 0 \end{cases} \begin{matrix} \\ \left. \vphantom{\begin{cases} \end{cases}} \right\} \text{if fact, it's a special case} \\ \left. \vphantom{\begin{cases} \end{cases}} \right\} \text{of the following one} \end{matrix} \quad (14)$$

$$\binom{n}{p} = \begin{cases} \frac{n(n-1)\cdots(n-p+1)}{p!} & \text{if } 0 \leq p \leq n \end{cases} \quad (15)$$

$$\binom{n}{p} = \begin{cases} 0 & \text{if } p < 0 \end{cases} \quad (16)$$

In the following example, we subnumerate the equations with the option `subnumerate` (available when the package `amsmath` is loaded).

```
\begin{DispWithArrows}< \label{system} (\ref*{system}) \Leftrightarrow >[
format = 1, subequations ]
x+y+z = -3 \Arrow{tikz=-,jump=2}{3 equations} \\
xy+xz+yz=-2 \\
xyz = -15 \label{last-equation}
\end{DispWithArrows}
```

²⁰We recall that `varioref`, `hyperref`, `cleveref` and `autonum` must be loaded in this order. The package `witharrows` can be loaded anywhere.

²¹The option `left-brace` can also be used without value: in this case, only the brace is drawn...

²²The environment `{cases}` of `amsmath` is a way to compose such multi-cases equations. However, it's not possible to use the automatic numbering of equations with this environment. The environment `{numcases}` of the extension `cases` (written by Donald Arseneau) provides this possibility but, of course, it's not possible to draw arrows with this extension.

$$(17) \Leftrightarrow \left(\begin{array}{l} x + y + z = -3 \\ xy + xz + yz = -2 \\ xyz = -15 \end{array} \right) \begin{array}{l} (17a) \\ (17b) \\ (17c) \end{array} \text{ 3 equations}$$

The whole system is the equation (17) (this reference has been coded by `\eqref{system}`) whereas the last equation is the equation (17c) (this reference has been coded by `\eqref{last-equation}`). The command `\ref*` used in the code above is provided by `hyperref`. It's a variant of `\ref` which doesn't create interactive link.

With the option `replace-left-brace-by`, it's possible to replace the left curly brace by another extensible delimiter. For example, "`replace-left-brace-by = [\enskip`" will compose with a bracket and add also a `\enskip` after this bracket.

9 Advanced features

9.1 Utilisation with plain-TeX

The extension `witharrows` can be used with plain-TeX. In this case, the extension must be loaded with `\input`:

```
\input{witharrows.tex}
```

In plain-TeX, there is not environments as in LaTeX. Instead of using the environment `{Witharrows}`, with `\begin{WithArrows}` and `\end{WithArrows}`, one should use a pseudo-environment delimited by `\WithArrows` and `\endWithArrows` (idem for `{DispWithArrows}`).

```

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

```

The version of `witharrows` for plain-TeX doesn't provide all the functionalities of the LaTeX version. In particular, the functionalities which deal with the number of the equations are not available (since they rely upon the system of tags of LaTeX).

9.2 The option `tikz-code` : how to change the shape of the arrows

The option `tikz-code`²³ allows the user to change the shape of the arrows.²⁴

For example, the options "up" and "down" described previously (cf. p. 9) are programmed internally with `tikz-code`.

The value of this option must be a valid Tikz drawing instruction (with the final semicolon) 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).

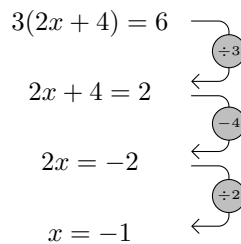
²³For historical reasons, the option `tikz-code` has an alias: `TikzCode`.

²⁴If the option `wrap-lines` is used in an environment `{DispWithArrows}` or `{DispWithArrows*}`, the option `tikz-code` will have no effect for the arrows of this environment but only for the arrows in the nested environments `{WithArrows}`.

```

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

```



The environments `{DispWithArrows}` and its starred version `{DispWithArrows*}` provide a command `\WithArrowsRightX` which can be used in a definition of `tikz-code`. This command gives the x -value of the right side of the composition box (taking into account the eventual tags of the equations). For an example of use, see p. 26.

9.3 The command `\WithArrowsNewStyle`

The extension `witharrows` provides a command `\WithArrowsNewStyle` to define styles in a way similar to the “styles” of Tikz.

The command `\WithArrowsNewStyle` takes two mandatory arguments. The first is the name of the style and the second is a list of key-value pairs. The scope of the definition done by `\WithArrowsNewStyle` is the current TeX scope.

The style can be used as a key at the document level (with `\WithArrowsOptions`) or at the environment level (in the optional arguments of `{WithArrows}` and `{DispWithArrows}`). The style can also be used in another command `\WithArrowsNewStyle`.

For an example of use, see p. 26.

9.4 Vertical positioning of the arrows

There are four parameters for fine tuning of the vertical positioning of the arrows : `ygap`, `ystart`, `start-adjust` and `end-adjust`.

We first explain the behaviour when the parameters `start-adjust` and `end-adjust` are equal to zero:

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

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

However, for aesthetic reasons, when it's possible, `witharrows` starts the arrow a bit higher (by an amount `start-adjust`) and ends the arrow a bit lower (by an amount `end-adjust`). By default, both parameters `start-adjust` and `end-adjust` are equal to 0.4 ex.

Here is for example the behaviour without the mechanism of `start-adjust` and `end-adjust` (this was the standard behaviour for versions prior to 1.13).

```

 $\begin{WithArrows}[start-adjust=0pt, end-adjust=0pt]$ 
 $A \& = (a+1)^2 \xrightarrow{\text{we expand}} \backslash$ 
 $\& = a^2 + 2a + 1$ 
 $\end{WithArrows}$ 

```

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

Here is the standard behaviour since version 1.13 (the parameters `start-adjust` and `end-adjust` are used with the default value 0.4 ex). The arrow is longer and the result is more aesthetic.

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

It's also possible to use the option `adjust` which sets both `start-adjust` and `end-adjust`.

Since the mechanism of `start-adjust` and `end-adjust` has been added in version 1.13 of `witharrows`, that version is not strictly compatible with older versions. However, it's possible to restore the previous behaviour simply by setting `start-adjust` and `end-adjust` to 0 pt:

```

 $\WithArrowsOptions{adjust = 0pt}$ 

```

9.5 Footnotes in the environments of `witharrows`

If you want to put footnotes in an environment `{WithArrows}` or `{DispWithArrows}`, 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:

$$\begin{aligned}
A &= (a + b)^2 \\
&= a^2 + b^2 + 2ab \quad \downarrow \text{We expand}^{25}
\end{aligned}$$

²⁵A footnote.

9.6 Option no-arrows

The option `no-arrows` is a convenience given to the user. With this option the arrows are not drawn. However, an analyse of the arrows is done and some errors can be raised, for example if an arrow would arrive after the last row of the environment.

9.7 Note for developers

If you want to construct an environment upon an environment of `witharrows`, we recommend to call the environment with the construction `\WithArrows-\endWithArrows` or `\DispWithArrows-\endDispWithArrows` (and not `\begin{WithArrows}-\end{WithArrows}`, etc.).

By doing so, the error messages generated by `witharrows` will (usually) mention the name of your environment and they will be easier to understand by the final user.

By example, you can define an environment `{DWA}` which is an alias of `{DispWithArrows}`:
`\NewDocumentEnvironment {DWA} {} {\DispWithArrows}{\endDispWithArrows}`

If you use this environment `{DWA}` in math mode, you will have the following error message:
 The environment `{DWA}` should be used only outside math mode.

Another example is the definition of the environment `{DispWithArrows*}` internally in the package `witharrows` by the following code:

```
\NewDocumentEnvironment {DispWithArrows*} {}
  {\WithArrowsOptions{notag}%
   \DispWithArrows}
  {\endDispWithArrows}
```

10 Examples

10.1 \MoveEqLeft

It's possible to use `\MoveEqLeft` of `mathtools`. Don't forget that `\MoveEqLeft` has also the value of an ampersand (&). That's important for the placement of an eventual command `\Arrow`.

```


$$\begin{aligned}
& \arccos(x) = \arcsin \frac{4}{5} + \arcsin \frac{5}{13} \\
& \Leftrightarrow x = \sin \left( \arcsin \frac{4}{5} + \arcsin \frac{5}{13} \right) \\
& \Leftrightarrow x = \frac{4}{5} \cos \arcsin \frac{5}{13} + \frac{5}{13} \cos \arcsin \frac{4}{5} \\
& \Leftrightarrow x = \frac{4}{5} \sqrt{1 - \left(\frac{5}{13}\right)^2} + \frac{5}{13} \sqrt{1 - \left(\frac{4}{5}\right)^2}
\end{aligned}$$

```

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

10.2 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}%
  [format = c,
   interline = 4mm,
   tikz = {every node/.style = {circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt,
                                font = \tiny}}]

  3 (2x+4) = 6 \Arrow{$\div 3$} \\  

  2x+4 = 2 \Arrow{$-4$} \\  

  2x = -2 \Arrow{$\div 2$} \\  

  2x = -1
\end{WithArrows}
```

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

10.3 Examples with the option tikz-code

We recall that the option `tikz-code` is the Tikz code used by `witharrows` to draw the arrows.²⁶

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

10.3.1 Example 1

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

```
\begin{DispWithArrows*}%
  [displaystyle,
   ygap = 2mm,
   ystart = 0mm,
   tikz-code = {\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(\tfrac{\pi}{2} \cdot \tfrac{k}{n}\bigr)
  \dots\dots\dots
```

²⁶If an environment `{DispWithArrows}` or `{DispWithArrows*}` is used with the option `wrap-lines`, the value of the option `tikz-code` is not used for this environment (but is used for the environments nested inside).

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

10.3.2 Example 2

It's possible to modify the previous example to have the "text width" automatically computed with the right margin (in a way similar as the `wrap-lines` option) in the environments `{DispWithArrows}` and `{DispWithArrows*}`. In the definition of `tikz-code`, we use the command `\WithArrowsRightX` which is the x -value of the right margin of the current composition box (it's a TeX command and not a dimension). For lisibility, we use a style. This example requires the Tikz library `calc`.

```

\WithArrowsNewStyle{MyStyle}
  {displaystyle,
  ygap = 2mm,
  xoffset = 0pt,
  ystart = 0mm,
  tikz-code = {\path let \p1 = (##1)
                in (##1)
                  -- node [anchor = west,
                           text width = {\WithArrowsRightX - \x1 - 0.5 em}]
                     {##3}
                  (##2) ;
  \draw let \p1 = (##1)
        in (##1) -- ++(\WithArrowsRightX - \x1,0) |- (##2) ; }}

begin{DispWithArrows}[MyStyle]
  S_n
  & = \frac{1}{n} \sum_{k=0}^{n-1} \cos\bigl(\tfrac{\pi}{2} \cdot \tfrac{k}{n}\bigr) \\
  \Arrow{\cos x = \Re(e^{ix})} \\
  .....

```

$$S_n = \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) \quad \overline{\hspace{10em}} \quad (18)$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i\frac{k\pi}{2n}}\right) \quad \overleftarrow{\hspace{10em}} \quad (19)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i\frac{k\pi}{2n}}\right) \quad \overleftarrow{\hspace{10em}} \quad (20)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i\frac{\pi}{2n}}\right)^k\right) \quad \overleftarrow{\hspace{10em}} \quad (21)$$

$$= \frac{1}{n} \Re\left(\frac{1 - \left(e^{i\frac{\pi}{2n}}\right)^n}{1 - e^{i\frac{\pi}{2n}}}\right) \quad \overleftarrow{\hspace{10em}} \quad (22)$$

$$= \frac{1}{n} \Re\left(\frac{1 - i}{1 - e^{i\frac{\pi}{2n}}}\right) \quad (23)$$

10.3.3 Example 3

In the following example, we change the shape of the arrow depending on whether the start row is longer than the end row or not. This example requires the Tikz library calc.

```
\begin{WithArrows}[ll,interline=5mm,xoffset=5mm,
  tikz-code = {\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}
```

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

10.4 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 `code-after`. 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,...,\WithArrowsNbLines}
    { \pgfmathtruncatemacro{\i}{\j-1}
      \Arrow[rr]{\i}{\j}{\i} }
    \Arrow[rr,xoffset=1cm,tikz=<-]{1}{\WithArrowsNbLines}{\WithArrowsNbLines}}

```

The command `\WithArrowsNbLines` is a command available in `code-after` which gives the total number of lines (=rows) of the current environment (it's a command and not a counter).

```

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

```

a. f est continuous on E
b. f est continuous in 0
c. f is bounded on the unit sphere
d. $\exists K > 0 \ \forall x \in E \ \|f(x)\| \leq K \|x\|$
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 to have, for example, numbers in parenthesis, the best way is to change the value of `tikz-code`:

```
tikz-code = {\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. $\exists K > 0 \ \forall x \in E \ \|f(x)\| \leq K \|x\|$
e. f is lipschitzian

11 Implementation

11.1 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.²⁷

```
1 <@@=wa>
2 <*LaTeX>
3 \RequirePackage{tikz}
4 \RequirePackage{expl3}[2019/02/15]
5 </LaTeX>
6 <*plain-TeX>
7 \input tikz.tex
8 \input expl3-generic.tex
9 </plain-TeX>
10 \usetikzlibrary{arrows.meta,bending}
```

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

```
11 <*LaTeX>
12 \RequirePackage{l3keys2e}
13 \ProvidesExplPackage
14   {witharrows}
15   {\myfiledate}
16   {\myfileversion}
17   {Draws arrows for explanations on the right}
```

The package `xparse` will be used to define the environments `{WithArrows}`, `{DispWithArrows}`, `{DispWithArrows*}` and the commands `\Arrow`, `\WithArrowsOptions` and `\WithArrowsNewStyle`.

```
18 \RequirePackage { xparse } [ 2018-10-17 ]
19 </LaTeX>
20 <*plain-TeX>
21 \ExplSyntaxOn
22 \catcode ` \@ = 11
23 </plain-TeX>
```

11.2 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 2.0), 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.

```
24 <*LaTeX>
25 \bool_new:N \g__wa_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.

```
26 \bool_new:N \g__wa_footnote_bool
27 </LaTeX>
```

²⁷cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

```

28 \cs_new_protected:Npn \__wa_msg_new:nn { \msg_new:nnn { witharrows } }
29 \cs_new_protected:Npn \__wa_msg_new:nnn { \msg_new:nnnn { witharrows } }
30 \cs_new_protected:Npn \__wa_msg_redirect_name:nn
31   { \msg_redirect_name:nnn { witharrows } }
32 \cs_new_protected:Npn \__wa_error:n { \msg_error:nn { witharrows } }
33 \cs_new_protected:Npn \__wa_warning:n { \msg_error:nn { witharrows } }
34 \cs_new_protected:Npn \__wa_fatal:n { \msg_fatal:nn { witharrows } }
35 \cs_new_protected:Npn \__wa_error:nn { \msg_error:nnn { witharrows } }
36 \cs_generate_variant:Nn \__wa_error:nn { n x }

```

We define a set of keys `WithArrows/package` for these options.

```

37 (*LaTeX)
38 \keys_define:nn { WithArrows / package }
39   {
40     footnote .bool_gset:N = \g__wa_footnote_bool ,
41     footnotehyper .bool_gset:N = \g__wa_footnotehyper_bool ,
42     unknown .code:n =
43       \__wa_fatal:n { Option-unknown-for-package }
44   }
45 \__wa_msg_new:nn { Option-unknown-for-package }
46   {
47     You-can't-use-the-option-'\l_keys_key_tl'~when~loading~the~
48     package~witharrows.~Try-to-use~the~command~
49     \token_to_str:N\WithArrowsOptions.
50   }

```

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

```

51 \ProcessKeysOptions { WithArrows / package }

52 \__wa_msg_new:nn { Option-incompatible-with-Beamer }
53   {
54     The-option-'\l_keys_key_tl'\ is~incompatible~
55     with-Beamer~because-Beamer-has~its~own~system~to~extract~footnotes.
56   }
57 \__wa_msg_new:nn { footnote-with-footnotehyper-package }
58   {
59     You-can't-use-the-option-'footnote'~because-the-package~
60     footnotehyper~has~already~been~loaded.~
61     If~you~want,~you~can~use~the~option-'footnotehyper'~and~the~footnotes~
62     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
63     of~the~package-footnotehyper.\\
64     If~you~go~on,~the~package-footnote~won't~be~loaded.
65   }
66 \__wa_msg_new:nn { footnotehyper-with-footnote-package }
67   {
68     You-can't-use-the-option-'footnotehyper'~because-the-package~
69     footnote~has~already~been~loaded.~
70     If~you~want,~you~can~use~the~option-'footnote'~and~the~footnotes~
71     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
72     of~the~package-footnote.\\
73     If~you~go~on,~the~package-footnotehyper~won't~be~loaded.
74   }

75 \bool_if:NT \g__wa_footnote_bool
76   {
77     \@ifclassloaded { beamer }
78     { \msg_info:nn { witharrows } { Option-incompatible-with-Beamer } }
79     {
80       \@ifpackageloaded { footnotehyper }
81       { \__wa_error:n { footnote-with-footnotehyper-package } }
82       { \usepackage { footnote } }

```

```

83     }
84 }
85 \bool_if:NT \g__wa_footnotehyper_bool
86 {
87   \@ifclassloaded { beamer }
88   { \__wa_info:n { Option~incompatible~with~Beamer } }
89   {
90     \@ifpackageloaded { footnote }
91     { \__wa_error:n { footnotehyper~with~footnote~package } }
92     { \usepackage { footnotehyper } }
93   }
94   \bool_gset_true:N \g__wa_footnote_bool
95 }

```

The flag `\g__wa_footnote_bool` is raised and so, we will only have to test `\g__wa_footnote_bool` in order to know if we have to insert an environment `{savenotes}` (the `\begin{savenotes}` is in `\@@_pre_halign:n` and `\end{savenotes}` at the end of the environments `{WithArrows}` and `{DispWithArrows}`).

11.3 The class option `leqno`

The boolean `\c__leqno_bool` will indicate if the class option `leqno` is used. When this option is used in LaTeX, the command `\@eqnnum` is redefined (as one can see in the file `leqno.clo`). That's enough to put the labels on the left in our environments `{DispWithArrows}` and `{DispWithArrows*}`. However, that's not enough when our option `wrap~lines` is used. That's why we have to know if this option is used as a class option. With the following programming, `leqno` *can't* be given as an option of `witharrows` (by design).

```

96 \bool_new:N \c__wa_leqno_bool
97 \DeclareOption { leqno } { \bool_set_true:N \c__wa_leqno_bool }
98 \DeclareOption* { }
99 \ProcessOptions*
100 </LaTeX>

```

11.4 Some technical definitions

```

101 \cs_generate_variant:Nn \tl_put_right:Nn { N v }
102 \cs_generate_variant:Nn \seq_set_split:Nnn { N x x }

```

We create booleans in order to know if some packages are loaded. For example, for the package `amsmath`, the boolean is called `\c__amsmath_loaded_bool`.²⁸

```

103 \AtBeginDocument
104 {
105   \clist_map_inline:nn
106     {
107       amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
108       typedref, unicode-math, varwidth
109     }
110     {
111       \bool_new:c { c__wa_#1_loaded_bool }
112     }
113     <LaTeX>
114     \ifpackageloaded { #1 }
115     { \bool_set_true:c { c__wa_#1_loaded_bool } }
116     { }
117     </LaTeX>
118     <plain-TeX>
119     \bool_set_false:c { c__wa_#1_loaded_bool }

```

²⁸It's not possible to use `\@ifpackageloaded` in the core of the preamble because `\@ifpackageloaded` is available only in the preamble.

```

119 </plain-TeX>
120   }
121 }

```

We define a command `\@@_strcmp:nn` to compare two token lists. It will be available whether the engine is pdfTeX, XeTeX or LuaTeX.

```

122 \sys_if_engine luatex:TF
123 {
124   \cs_new_protected:Npn \__wa_strcmp:nn #1 #2
125     { \lua_now:e { l3kernel.strcmp('#1','#2') } }
126 }
127 {
128   \cs_new_protected:Npn \__wa_strcmp:nn #1 #2
129     { \pdfTEX_strcmp:D { #1 } { #2 } }
130 }

```

We can now define a command `\@@_sort_seq:N` which will sort a sequence.

```

131 \cs_new_protected:Npn \__wa_sort_seq:N #1
132 {
133   \seq_sort:Nn #1
134   {
135     \int_compare:nNnTF
136     {
137       \__wa_strcmp:nn
138       { \str_lower_case:n { ##1 } }
139       { \str_lower_case:n { ##2 } }
140     }
141     > 0
142     \sort_return_swapped:
143     \sort_return_same:
144   }
145 }

```

The following command converts each item of a sequence from `tl` to `str`. It will be used when creating list of keys (a key name is always a `str`).

```

146 \cs_new_protected:Npn \__wa_convert_to_str_seq:N #1
147 {
148   \seq_clear:N \l_tmpa_seq
149   \seq_map_inline:Nn #1
150   {
151     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
152   }
153   \seq_set_eq:NN #1 \l_tmpa_seq
154 }
155 \cs_new_protected:Npn \__wa_set_seq_of_str_from_clist:Nn #1 #2
156 {
157   \seq_set_from_clist:Nn #1 { #2 }
158   \__wa_convert_to_str_seq:N #1
159 }

```

The command `\@@_save:N` saves a `expl3` variable by creating a global version of the variable. For a variable named `\l_name_type`, the corresponding global variable will be named `\g_name_type`. The type of the variable is determined by the suffix `type` and is used to apply the corresponding `expl3` commands.

```

160 \cs_new_protected:Npn \__wa_save:N #1
161 {
162   \seq_set_split:Nxx \l_tmpa_seq
163     { \char_generate:nn { ` } { 12 } }
164     { \cs_to_str:N #1 }
165   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl

```

The string `\l_tmpa_str` will contains the `type` of the variable.


```

166 \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
167 \use:c { \l_tmpa_str_if_exist:cF }
168 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ }
169 {
170 \use:c { \l_tmpa_str_new:c }
171 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ }
172 }
173 \use:c { \l_tmpa_str_gset_eq:cN }
174 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
175 }

```

The command `\@@_restore:N` affects to the `expl3` variable the value of the (previously) set value of the corresponding *global* variable.

```

176 \cs_new_protected:Npn \__wa_restore:N #1
177 {
178 \seq_set_split:Nxx \l_tmpa_seq
179 { \char_generate:nn { ` } { 12 } }
180 { \cs_to_str:N #1 }
181 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
182 \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
183 \use:c { \l_tmpa_str_set_eq:Nc }
184 #1 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ }
185 }

```

We define a Tikz style `@@_node_style` for the *l*-nodes and *r*-nodes that will be created in the `\halign`. These 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.

```

186 \tikzset
187 {
188 __wa_node_style / .style =
189 {
190 above = \l__wa_ystart_dim ,
191 inner-sep = \c_zero_dim ,
192 minimum-width = \c_zero_dim ,
193 minimum-height = \l__wa_ygap_dim
194 }
195 }

```

If the user uses the option `show-nodes` (it’s a `l3keys` option), the Tikz options `draw` and `red` will be appended to this style. This feature may be useful for debugging.²⁹

The style `@@_standard` is loaded 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.

```

196 \tikzset
197 {
198 __wa_standard / .style =
199 {
200 remember~picture ,
201 overlay ,
202 name~prefix = wa - \l__wa_prefix_str -
203 }
204 }

```

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`).

```

205 \tikzset
206 {

```

²⁹The *v*-nodes, created near the end of line in `{DispWithArrows}` and `{DispWithArrows*}` are not shown with the option `show-nodes`.

```

207   WithArrows / arrow / tips / .style =
208     { > = { Straight~Barb [ scale = 1.2 , bend ] } }
209   }

```

The style `WithArrows/arrow` will be used to draw the arrows (more precisely, it will be passed to `every~path`).

```

210 \tikzset
211 {
212   WithArrows / arrow / .style =
213     {
214       align = left ,

```

We have put the option `align = left` because we want to give the user the possibility of using `\` in the labels.

```

215       auto = left ,
216 <*LaTeX>
217       font = \small \itshape ,
218 </LaTeX>
219       WithArrows / arrow / tips ,
220       bend~left = 45 ,
221       ->
222     }
223 }

```

The option `subequations` is an option which uses the environment `{subequations}` of `amsmath`. That's why, if `amsmath` is loaded, we add the key `subequations` to the list of the keys available in `\WithArrowsOptions` and `{DispWithArrows}`.

```

224 <*LaTeX>
225 \AtBeginDocument
226 {
227   \bool_if:NTF \c__wa_amsmath_loaded_bool
228     {
229     \seq_put_right:Nn \l__wa_options-WithArrowsOptions_seq { subequations }
230     \seq_put_right:Nn \l__wa_options-DispWithArrows_seq { subequations }
231   }

```

In order to increase the interline in the environments `{WithArrows}`, `{DispWithArrows}`, etc., 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` if `amsmath` is not loaded (we put the code in a `\AtBeginDocument` because the flag `\c_@@_amsmath_loaded_bool` is itself set in a `\AtBeginDocument`).

```

232   {
233 </LaTeX>
234   \cs_new_protected:Npn \spread@equation
235     {
236       \openup \jot
237       \cs_set_eq:NN \spread@equation \prg_do_nothing:
238     }
239 <*LaTeX>
240   }
241 }
242 </LaTeX>

```

```

243 \tl_new:N \l__wa_left_brace_tl
244 \tl_set_eq:NN \l__wa_left_brace_tl \c_novalue_tl

```

11.5 Variables

The boolean `\l_@@_in-WithArrows_bool` will be raised in an environment `{WithArrows}` and the boolean `\l_@@_in-dispwitharrows_bool` will be raised in an environment `{DispWithArrows}` or

`{DispWithArrows*}`. The boolean `\l_@@_in_code_after_bool` will be raised during the execution of the code-after (option code-after).

```
245 \bool_new:N \l__wa_in_WithArrows_bool
246 \bool_new:N \l__wa_in_DispWithArrows_bool
247 \bool_new:N \l__wa_in_code_after_bool
```

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

```
248 \seq_new:N \g__wa_position_in_the_tree_seq
249 \seq_gput_right:Nn \g__wa_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`.

```
250 \int_new:N \g__wa_last_env_int
```

The following integer indicates the position of the box that will be created for an environment `{WithArrows}` (not an environment `{DispWithArrows}`) : 0 (=t=`\vtop`), 1 (=c=`\vcenter`) or 2 (=b=`\vbox`).

```
251 \int_new:N \l__wa_pos_env_int
```

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

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

The option `v` can be used only in `\Arrow` in code-after (see below).

```
252 \int_new:N \l__wa_pos_arrow_int
253 \int_set:Nn \l__wa_pos_arrow_int 3
```

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

- `\g_@@_arrow_int` to count the arrows created in the environment ;
- `\g_@@_line_int` to count the lines of the `\halign` ;
- `\g_@@_col_int` to count the column of the `\halign`.

These 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.

```
254 \seq_new:N \g__wa_arrow_int_seq
255 \int_new:N \g__wa_arrow_int
256 \seq_new:N \g__wa_line_int_seq
257 \int_new:N \g__wa_line_int
258 \seq_new:N \g__wa_col_int_seq
259 \int_new:N \g__wa_col_int
```

For the environment `{DispWithArrows}`, the comma list `\l_@@_tags_clist` will be the list of the numbers of lines to be tagged (with the counter `equation` of LaTeX). In fact, `\l_@@_tags_clist` may contain non negative integers but also three special values, `first`, `last` and `all`.

```
260 {*LaTeX}
261 \clist_new:N \l__wa_tags_clist
262 \clist_set:Nn \l__wa_tags_clist { all }
```

During the execution of an environment `{DispWithArrows}`, if a row must be tagged, the (local) value of `\l_@@_tags_clist` will be put (by convention) to `all`.

```

263 \cs_new_protected:Npn \__wa_test_if_to_tag:
264   {
265     \clist_if_in:NVT \l__wa_tags_clist \g__wa_line_int
266     { \clist_set:Nn \l__wa_tags_clist { all } }
267   }
268 </LaTeX>

```

If the user has given a value for the option `command-name` (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 `command-name` is “`Arrow`” and thus, by default, the name of the command will be `\Arrow`.

```

269 \str_new:N \l__wa_command_name_str
270 \str_set:Nn \l__wa_command_name_str { Arrow }

```

The string `\l_@@_string_Arrow_for_msg_str` is only a string that will be displayed in some error messages. For example, if `command-name` is defined to be `Explanation`, this string will contain “`\Arrow alias \Explanation`”.

```

271 \str_new:N \l__wa_string_Arrow_for_msg_str
272 \str_set:Nx \l__wa_string_Arrow_for_msg_str { \token_to_str:N \Arrow }

```

The sequence `\g_@@_names_seq` will be the list of all the names of environments used (via the option `name`) in the document: two environments must not have the same name. However, it’s possible to use the option `allow-duplicate-names`.

```

273 \seq_new:N \g__wa_names_seq

```

The boolean `\l_@@_sbwi_bool` corresponds to the option `standard-behaviour-with-items`. Since the version 1.16 of `witharrows`, no vertical space is added between an `\item` of a LaTeX list and an environment `{DispWithArrows}`. With the option `standard-behaviour-with-items`, it’s possible to restore the previous behaviour (which corresponds to the standard behaviour of `{align}` of `amsmath`). `\l_@@_sbwi_bool` is the boolean corresponding to this option.

```

274 <*LaTeX>
275 \bool_new:N \l__wa_sbwi_bool
276 </LaTeX>

277 <*LaTeX>
278 \bool_new:N \l__wa_tag_star_bool
279 \bool_new:N \l__wa_tag_next_line_bool
280 \bool_new:N \l__wa_qedhere_bool
281 </LaTeX>
282 \bool_new:N \l__wa_in_first_columns_bool
283 \bool_new:N \l__wa_new_group_bool
284 \bool_new:N \l__wa_initial_r_bool
285 \bool_new:N \l__wa_final_r_bool
286 \tl_new:N \l__wa_initial_tl
287 \tl_new:N \l__wa_final_tl
288 \int_new:N \l__wa_nb_cols_int

```

The string `\l_@@_format_str` will contain the *format* of the array which is a succession of letters `r`, `c` and `l` specifying the type of the columns of the `\halign` (except the column for the labels of the equations in the environment `{DispWithArrows}`).

```

289 \str_new:N \l__wa_format_str

```

The option boolean corresponds to the option `subequations`.

```

290 <*LaTeX>
291 \bool_new:N \l__wa_subequations_bool
292 </LaTeX>

```

11.6 The definition of the options

There are four levels where options can be set:

- with `\usepackage[...]{witharrows}`: this level will be called *package* level;
- with `\WithArrowsOptions{...}`: this level will be called *global* level³⁰;
- with `\begin{WithArrows}[...]`: this level will be called *environment* level;
- with `\Arrow[...]` (included in `code-after`): this level will be called *local* level.

When we scan a list of options, we want to be able to raise an error if two options of position (11, r1, i, etc.) of the arrows are present. That's why we keep the first option of position in a variable called `\l_@@_previous_key_str`. The following function `\@@_eval_if_allowed:n` will execute its argument only if a first key of position has not been set (and raise an error elsewhere).

```
293 \cs_new_protected:Npn \__wa_eval_if_allowed:n #1
294 {
295   \str_if_empty:NTF \l__wa_previous_key_str
296     {
297       \str_set_eq:NN \l__wa_previous_key_str \l_keys_key_tl
298       #1
299     }
300     { \__wa_error:n { Incompatible~options } }
301 }
302 \cs_new_protected:Npn \__wa_fix_pos_option:n #1
303 { \__wa_eval_if_allowed:n { \int_set:Nn \l__wa_pos_arrow_int { #1 } } }
```

First a set of keys that will be used at the global or environment level of options.

```
304 \keys_define:nn { WithArrows / Global }
305 {
306   ygap .dim_set:N = \l__wa_ygap_dim ,
307   ygap .value_required:n = true ,
308   ygap .initial:n = 0.4 ex ,
309   ystart .dim_set:N = \l__wa_ystart_dim ,
310   ystart .value_required:n = true ,
311   ystart .initial:n = 0.4 ex ,
312   more-columns .code:n =
313     \__wa_msg_redirect_name:nn { Too-much-columns-in-WithArrows } { none } ,
314   more-columns .value_forbidden:n = true,
315   command-name .code:n =
316     \str_set:Nn \l__wa_command_name_str { #1 }
317     \str_set:Nx \l__wa_string_Arrow_for_msg_str
318     { \c_backslash_str Arrow-alias-\c_backslash_str #1 } ,
319   command-name .value_required:n = true ,
320   tikz-code .tl_set:N = \l__wa_tikz_code_tl,
321   tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
322   tikz-code .value_required:n = true ,
323   TikzCode .meta:n = { tikz-code = #1 } ,
324   displaystyle .bool_set:N = \l__wa_displaystyle_bool ,
325   displaystyle .default:n = true ,
326   show-nodes .code:n =
327     \tikzset { __wa_node_style / .append-style = { draw , red } } ,
328   show-nodes .value_forbidden:n = true,
329   show-node-names .bool_set:N = \l__wa_show_node_names_bool ,
330   show-node-names .default:n = true ,
331   group .code:n =
332     \str_if_empty:NTF \l__wa_previous_key_str
```

³⁰This level is called *global level* but the settings done by `\WithArrowsOptions` are local in the TeX sense: their scope corresponds to the current TeX group.

```

333     {
334       \str_set:Nn \l__wa_previous_key_str { group }
335       \seq_remove_all:Nn \l__wa_options_Arrow_seq { xoffset }
336       \int_set:Nn \l__wa_pos_arrow_int 7
337     }
338     { \__wa_error:n { Incompatible-options } } ,
339 group .value_forbidden:n = true ,
340 groups .code:n =
341   \str_if_empty:NTF \l__wa_previous_key_str
342   {
343     \str_set:Nn \l__wa_previous_key_str { groups }
344     \seq_if_in:NnF \l__wa_options_Arrow_seq { new-group }
345     { \seq_put_right:Nn \l__wa_options_Arrow_seq { new-group } }
346     \seq_remove_all:Nn \l__wa_options_Arrow_seq { xoffset }
347     \int_set:Nn \l__wa_pos_arrow_int 6
348   }
349   { \__wa_error:n { Incompatible-options } } ,
350 groups .value_forbidden:n = true ,
351 tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
352 tikz .initial:n = \c_empty_tl ,
353 tikz .value_required:n = true ,
354 rr .value_forbidden:n = true ,
355 rr .code:n = \__wa_fix_pos_option:n 3 ,
356 ll .value_forbidden:n = true ,
357 ll .code:n = \__wa_fix_pos_option:n 1 ,
358 rl .value_forbidden:n = true ,
359 rl .code:n = \__wa_fix_pos_option:n 2 ,
360 lr .value_forbidden:n = true ,
361 lr .code:n = \__wa_fix_pos_option:n 0 ,
362 i .value_forbidden:n = true ,
363 i .code:n = \__wa_fix_pos_option:n 5 ,
364 xoffset .dim_set:N = \l__wa_xoffset_dim ,
365 xoffset .value_required:n = true ,
366 xoffset .initial:n = 3 mm ,
367 jot .dim_set:N = \jot ,
368 jot .value_required:n = true ,
369 interline .skip_set:N = \l__wa_interline_skip ,
370 interline .value_required:n = true ,
371 start-adjust .dim_set:N = \l__wa_start_adjust_dim ,
372 start-adjust .value_required:n = true ,
373 start-adjust .initial:n = 0.4 ex ,
374 end-adjust .dim_set:N = \l__wa_end_adjust_dim ,
375 end-adjust .value_required:n = true ,
376 end-adjust .initial:n = 0.4 ex ,
377 adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
378 adjust .value_required:n = true ,

```

With the option `no-arrows`, the arrows won't be drawn. However, the “first pass” of the arrows is done and some errors may be detected. The nullification of `\@@_draw_arrows:nn` is for the standard arrows and the nullification of `\@@_draw_arrow:nnn` is for “Arrow in code-after”.

```

379 no-arrows .code:n =
380   \cs_set_eq:NN \__wa_draw_arrows:nn \use_none:nn
381   \cs_set_eq:NN \__wa_draw_arrow:nnn \use_none:nnn ,
382 no-arrows .value_forbidden:n = true ,
383 }

```

Now a set of keys specific to the environments `{WithArrows}` (and not `{DispWithArrow}`). Despite its name, this set of keys will also be used in `\WithArrowsOptions`.

```

384 \keys_define:nn { WithArrows / WithArrowsSpecific }
385 {
386   t .code:n = \int_set:Nn \l__wa_pos_env_int 0 ,
387   t .value_forbidden:n = true ,
388   c .code:n = \int_set:Nn \l__wa_pos_env_int 1 ,

```

```

389   c .value_forbidden:n = true ,
390   b .code:n = \int_set:Nn \l__wa_pos_env_int 2 ,
391   b .value_forbidden:n = true
392 }

```

The following list of the (left) extensible delimiters of LaTeX is only for the validation of the key `replace-left-brace-by`.

```

393 \clist_new:N \c__wa_extensible_delimiters_clist
394 \clist_set:Nn \c__wa_extensible_delimiters_clist
395 {
396   ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
397 }
398 <*LaTeX>
399 \AtBeginDocument
400 {
401   \bool_if:nT
402     { \c__wa_amsmath_loaded_bool || \use:c { c__wa_unicode-math_loaded_bool } }
403     {
404       \clist_put_right:Nn \c__wa_extensible_delimiters_clist { \lvert, \lVert }
405     }
406 }
407 </LaTeX>

```

Now a set of keys specific to the environments `{DispWithArrows}` and `{DispWithArrows*}` (and not `{WithArrows}`). Despite its name, this set of keys will also be used in `\WithArrowsOptions`.

```

408 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
409 {
410   fleqn .bool_set:N = \l__wa_fleqn_bool ,
411   fleqn .default:n = true ,
412   mathindent .dim_set:N = \l__wa_mathindent_dim ,
413   mathindent .value_required:n = true ,
414   mathindent .initial:n = 25 pt ,
415 <*LaTeX>
416   notag .code:n =
417     \str_if_eq:nnTF { #1 } { true }
418     { \clist_clear:N \l__wa_tags_clist
419       { \clist_set:Nn \l__wa_tags_clist { all } } ,
420     notag .default:n = true ,

```

Since the option `subequations` is an option which insert the environment `{DispWithArrows}` in an environment `{subequations}` of `amsmath`, we must test whether the package `amsmath` is loaded.

```

421   subequations .code:n =
422     \bool_if:NTF \c__wa_amsmath_loaded_bool
423     { \bool_set_true:N \l__wa_subequations_bool }
424     {
425       \__wa_error:n { amsmath-not-loaded }
426       \group_begin:
427         \globaldefs = 1
428         \__wa_msg_redirect_name:nn { amsmath-not-loaded } { info }
429       \group_end:
430     } ,
431   subequations .default:n = true ,
432   subequations .value_forbidden:n = true ,
433   nonumber .meta:n = notag ,
434   allow-multiple-labels .code:n =
435     \__wa_msg_redirect_name:nn { Multiple-labels } { none } ,
436   allow-multiple-labels .value_forbidden:n = true ,
437   tagged-lines .code:n =
438     \clist_set:Nn \l__wa_tags_clist { #1 }
439     \clist_if_in:NnT \l__wa_tags_clist { first }
440     {
441       \clist_remove_all:Nn \l__wa_tags_clist { first }

```

```

442     \clist_put_left:Nn \l__wa_tags_clist \c_one_int
443   } ,
444   tagged-lines .value_required:n = true ,
445 </LaTeX>
446   wrap-lines .bool_set:N = \l__wa_wrap_lines_bool ,
447   wrap-lines .default:n = true ,
448   replace-left-brace-by .code:n =
449   {
450     \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
451     \clist_if_in:NVTF
452       \c__wa_extensible_delimiters_clist
453       \l_tmpa_tl
454       { \tl_set:Nn \l__wa_replace_left_brace_by_tl { #1 } }
455       { \__wa_error:n { Bad~value~for~replace~brace~by } }
456   } ,
457   replace-left-brace-by .initial:n = \lbrace ,

```

Since the version 1.16 of `witharrows`, no vertical space is added between an `\item` of a LaTeX list and an environment `{DispWithArrows}`. With the option `standard-behaviour-with-items`, it's possible to restore the previous behaviour (which corresponds to the standard behaviour of `{align}` of `amsmath`).

```

458 <*LaTeX>
459   standard-behaviour-with-items .bool_set:N = \l__wa_sbwi_bool ,
460   standard-behaviour-with-items .default:n = true
461 </LaTeX>
462 }

```

Now a set of keys which will be used in all the environments (but not in `\WithArrowsOptions`).

```

463 \keys_define:nn { WithArrows / Env }
464 {
465   name .code:n =

```

First, we convert the value in a `str` because the list of the names will be a list of `str`.

```

466     \str_set:Nn \l_tmpa_str { #1 }
467     \seq_if_in:NVTF \g__wa_names_seq \l_tmpa_str
468       { \__wa_error:n { Duplicate~name } }
469       { \seq_gput_left:NV \g__wa_names_seq \l_tmpa_str }
470     \str_set_eq:NN \l__wa_name_str \l_tmpa_str ,
471   name .value_required:n = true ,
472   code-before .code:n = \tl_put_right:Nn \l__wa_code_before_tl { #1 } ,
473   code-before .value_required:n = true ,
474   CodeBefore .meta:n = { code-before = #1 } ,
475   code-after .code:n = \tl_put_right:Nn \l__wa_code_after_tl { #1 } ,
476   code-after .value_required:n = true ,
477   CodeAfter .meta:n = { code-after = #1 } ,
478   format .code:n =
479     \tl_if_empty:nTF { #1 }
480       { \__wa_error:n { Invalid~option~format } }
481       {
482         \regex_match:nnTF { \A[rcl]*\Z } { #1 }
483         { \tl_set:Nn \l__wa_format_str { #1 } }
484         { \__wa_error:n { Invalid~option~format } }
485       } ,
486   format .value_required:n = true ,
487 }

```

Now, we begin the construction of the major sets of keys, named “`WithArrows / WithArrows`”, “`WithArrows / DispWithArrows`” and “`WithArrows / WithArrowsOptions`”. Each of these sets of keys will be completed after.

```

488 \keys_define:nn { WithArrows }
489 {
490   WithArrows .inherit:n =

```



```

491     {
492       WithArrows / Global ,
493       WithArrows / WithArrowsSpecific ,
494       WithArrows / Env
495     } ,
496     DispWithArrows .inherit:n =
497     {
498       WithArrows / DispWithArrowsSpecific ,
499       WithArrows / Global ,
500       WithArrows / Env ,
501     } ,
502     WithArrowsOptions .inherit:n =
503     {
504       WithArrows / Global ,
505       WithArrows / WithArrowsSpecific ,
506       WithArrows / DispWithArrowsSpecific
507     }
508 }

```

A sequence of `str` for the options available in `{WithArrows}`. This sequence will be used in the error messages and can be modified dynamically.

```

509 \seq_new:N \l__wa_options_WithArrows_seq
510 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_WithArrows_seq
511 {
512   adjust, b, c, code-after, code-before, command-name,
513   displaystyle, end-adjust,
514   format, group, groups, i,
515   interline, jot, ll,
516   lr, more-columns, name,
517   no-arrows, rl, rr,
518   show-node-names, show-nodes, start-adjust,
519   t, tikz, tikz-code,
520   xoffset, ygap, ystart
521 }
522 \__wa_convert_to_str_seq:N \l__wa_options_WithArrows_seq

523 \keys_define:nn { WithArrows / WithArrows }
524 {
525   unknown .code:n =
526     \__wa_sort_seq:N \l__wa_options_WithArrows_seq
527     \__wa_error:n { Unknown-option~WithArrows }
528 }

529 \keys_define:nn { WithArrows / DispWithArrows }
530 {
531   left-brace .tl_set:N = \l__wa_left_brace_tl ,
532   unknown .code:n =
533     \__wa_sort_seq:N \l__wa_options_DispWithArrows_seq
534     \__wa_error:n { Unknown-option~DispWithArrows }
535 }

```

A sequence of the options available in `{DispWithArrows}`. This sequence will be used in the error messages and can be modified dynamically.

```

536 \seq_new:N \l__wa_options_DispWithArrows_seq
537 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_DispWithArrows_seq
538 {
539   code-after, code-before, command-name, tikz-code, adjust,
540   displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
541   left-brace, ll, lr, mathindent, name, no-arrows, replace-left-brace-by, rl,
542   rr, show-node-names, show-nodes, start-adjust, tikz, wrap-lines, xoffset,
543   ygap, ystart,

```

```

544 <*LaTeX>
545     allow-multiple-labels, tagged-lines, nonumber, notag
546 </LaTeX>
547   }

548 \keys_define:nn { WithArrows / WithArrowsOptions }
549   {
550     allow-duplicate-names .code:n =
551       \__wa_msg_redirect_name:nn { Duplicate~name } { none } ,
552     allow-duplicate-names .value_forbidden:n = true ,
553     unknown .code:n =
554       \__wa_sort_seq:N \l__wa_options_WithArrowsOptions_seq
555       \__wa_error:n { Unknown-option-WithArrowsOptions }
556   }

```

A sequence of the options available in `\WithArrowsOptions`. This sequence will be used in the error messages and can be modified dynamically.

```

557 \seq_new:N \l__wa_options_WithArrowsOptions_seq
558 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_WithArrowsOptions_seq
559   {
560     allow-duplicate-names, b, c, command-name, more-columns, tikz-code, adjust,
561     displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
562     mathindent, no-arrows, rl, rr, show-node-names, show-nodes, start-adjust, t,
563     tikz, wrap-lines, xoffset, ygap, ystart,
564 <*LaTeX>
565     allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
566     tagged-lines
567 </LaTeX>
568   }

```

The command `\@@_set_independent:` is a command without argument that will be used to specify that the arrow will be “independent” (of the potential groups of the option `group` or `groups`). This information will be stored in the field “status” of the arrow. Another value of the field “status” is “new-group”.

```

569 \cs_new_protected:Npn \__wa_set_independent:
570   {
571     \str_if_empty:NTF \l__wa_previous_key_str
572       {
573         \str_set_eq:NN \l__wa_previous_key_str \l_keys_key_tl
574         \str_set:Nn \l__wa_status_arrow_str { independent }
575         \str_if_eq:VnF \l_keys_value_tl { NoValue }
576         { \__wa_error:n { Value-for-a-key } }
577       }
578     { \__wa_error:n { Incompatible-options-in-Arrow } }
579   }

```

The options of an individual arrow are parsed twice. The first pass is when the command `\Arrow` is read. The second pass is when the arrows are drawn (after the end of the environment `{WithArrows}` or `{DispWithArrows}`). Now, we present the keys set for the first pass. The main goal is to extract informations which will be necessary during the scan of the arrows. For instance, we have to know if some arrows are “independent” or use the option “new-group”.

```

580 \keys_define:nn { WithArrows / Arrow / FirstPass }
581   {
582     jump .code:n =
583       \int_compare:nTF { #1 > 0 }
584         { \int_set:Nn \l__wa_jump_int { #1 } }
585         { \__wa_error:n { Negative-jump } } ,
586     jump .value_required:n = true,
587     rr .code:n = \__wa_set_independent: ,
588     ll .code:n = \__wa_set_independent: ,
589     rl .code:n = \__wa_set_independent: ,

```

```

590 lr .code:n = \__wa_set_independent: ,
591 i .code:n = \__wa_set_independent: ,
592 rr .default:n = NoValue ,
593 ll .default:n = NoValue ,
594 rl .default:n = NoValue ,
595 lr .default:n = NoValue ,
596 i .default:n = NoValue ,
597 new-group .value_forbidden:n = true,
598 new-group .code:n =
599   \int_compare:nTF { \l__wa_pos_arrow_int = 6 }
600     { \str_set:Nn \l__wa_status_arrow_str { new-group } }
601     { \__wa_error:n { new-group-without-groups } } ,

```

The other keys don't give any information necessary during the scan of the arrows. However, you try to detect errors and that's why all the keys are listed in this keys set. An unknown key will be detected at the point of the command `\Arrow` and not at the end of the environment.

```

602 tikz-code .code:n = \prg_do_nothing: ,
603 tikz-code .value_required:n = true ,
604 tikz .code:n = \prg_do_nothing: ,
605 tikz .value_required:n = true ,

```

The option `xoffset` is not allowed when the option `group` or the option `groups` is used (since it would be meaningless).

```

606 xoffset .code:n =
607   \int_compare:nNnT \l__wa_pos_arrow_int > 5
608     { \__wa_error:n { Option-xoffset-forbidden } } ,
609 xoffset .value_required:n = true ,
610 start-adjust .code:n = \prg_do_nothing: ,
611 start-adjust .value_required:n = true ,
612 end-adjust .code:n = \prg_do_nothing: ,
613 end-adjust .value_required:n = true ,
614 adjust .code:n = \prg_do_nothing: ,
615 adjust .value_required:n = true ,
616 unknown .code:n =
617   \__wa_sort_seq:N \l__wa_options_Arrow_seq
618   \seq_if_in:NVTF \l__wa_options_WithArrows_seq \l_keys_key_tl
619     {
620       \str_set:Nn \l_tmpa_str
621         { ~However,~this~key~can~be~used~in~the~options~of~\{WithArrows\}. }
622     }
623     { \str_clear:N \l_tmpa_str }
624   \__wa_error:n { Unknown-option-in-Arrow }
625 }

```

A sequence of the options available in `\Arrow`. This sequence will be used in the error messages and can be modified dynamically.

```

626 \seq_new:N \l__wa_options_Arrow_seq
627 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_Arrow_seq
628 {
629   adjust, end-adjust, i, jump, ll, lr, rl, rr, start-adjust, tikz, tikz-code,
630   xoffset
631 }

```

```

632 \cs_new_protected:Npn \__wa_fix_pos_arrow:n #1
633 {
634   \str_if_empty:NT \l__wa_previous_key_str
635     {
636       \str_set_eq:NN \l__wa_previous_key_str \l_keys_key_tl
637       \int_set:Nn \l__wa_pos_arrow_int { #1 }
638     }
639 }

```

The options of the individual commands `\Arrows` are scanned twice. The second pass is just before the drawing of the arrow. In this set of keys, we don't put an item for the unknown keys because an unknown key would have been already detected during the first pass.

```

640 \keys_define:nn {WithArrows / Arrow / SecondPass }
641   {
642     tikz-code .tl_set:N = \l__wa_tikz_code_tl ,
643     tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
644     tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
645     tikz .initial:n = \c_empty_tl ,
646     rr .code:n = \__wa_fix_pos_arrow:n 3 ,
647     ll .code:n = \__wa_fix_pos_arrow:n 1 ,
648     rl .code:n = \__wa_fix_pos_arrow:n 2 ,
649     lr .code:n = \__wa_fix_pos_arrow:n 0 ,
650     i .code:n = \__wa_fix_pos_arrow:n 5 ,

```

The option `xoffset` is not allowed when the option `group` or the option `groups` is used (since it would be meaningless). An error has been raised during the first pass. Here, we manage to avoid a second error which would be redundant.

```

651   xoffset .code:n =
652     \int_compare:nNnF \l__wa_pos_arrow_int > 5
653     { \dim_set:Nn \l__wa_xoffset_dim { #1 } } ,
654   start-adjust .dim_set:N = \l__wa_start_adjust_dim,
655   end-adjust .dim_set:N = \l__wa_end_adjust_dim,
656   adjust .code:n =
657     \dim_set:Nn \l__wa_start_adjust_dim { #1 }
658     \dim_set:Nn \l__wa_end_adjust_dim { #1 } ,
659 }

```

`\WithArrowsOptions` is the command of the `witharrows` package to fix options at the document level. It's possible to fix in `\WithArrowsOptions` some options specific to `{WithArrows}` (in contrast with `{DispWithArrows}`) or specific to `{DispWithArrows}` (in contrast with `{WithArrows}`). That's why we have constructed a set of keys specific to `\WithArrowsOptions`.

```

660 {*LaTeX}
661 \NewDocumentCommand \WithArrowsOptions { m }
662 {/LaTeX}
663 {*plain-TeX}
664 \cs_set_protected:Npn \WithArrowsOptions #1
665 {/plain-TeX}
666 {
667   \str_clear_new:N \l__wa_previous_key_str
668   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
669 }

```

11.7 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 `command-name` and the user command for `\@@_Arrow` will be different. This mechanism can be useful when the user has already a command named `\Arrow` he still wants to use in the environments `{WithArrows}` or `{DispWithArrows}`.

```

670 {*LaTeX}
671 \NewDocumentCommand \__wa_Arrow { 0 { } m ! 0 { } }
672 {/LaTeX}
673 {*plain-TeX}
674 \cs_new_protected:Npn \__wa_Arrow
675 {
676   \peek_meaning:NTF [
677     { \__wa_Arrow_i }
678     { \__wa_Arrow_i [ ] }

```

```

679 }
680 \cs_new_protected:Npn \__wa_Arrow_i [ #1 ] #2
681 {
682   \peek_meaning:NTF [
683     { \__wa_Arrow_ii [ #1 ] { #2 } }
684     { \__wa_Arrow_ii [ #1 ] { #2 } [ ] }
685   }
686 \cs_new_protected:Npn \__wa_Arrow_ii [ #1 ] #2 [ #3 ]
687 </plain-TeX>
688 {

```

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.

```

689   \int_gincr:N \g__wa_arrow_int

```

We will construct a global property list to store the informations of the considered arrow. The six fields of this property list are “initial”, “final”, “status”, “options”, “label” and “input-line”. In order to compute the value of “final” (the destination row of the arrow), we have to take into account a potential option jump. In order to compute the value of the field “status”, we have to take into account options as `ll`, `rl`, `rr`, `lr`, etc. or `new-group`.

We will do that job with a first analyze of the options of the command `\Arrow` with a dedicated set of keys called `WithArrows/Arrow/FirstPass`.

```

690   \str_clear_new:N \l__wa_previous_key_str
691   \keys_set:nn { WithArrows / Arrow / FirstPass } { #1 , #3 }

```

We construct now a global property list to store the informations of the considered arrow with the six fields “initial”, “final”, “status”, “options”, “label” and “input-line”.

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

```

692   \prop_put:NnV \l_tmpa_prop { initial } \g__wa_line_int

```

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

```

693   \int_set:Nn \l_tmpa_int { \g__wa_line_int + \l__wa_jump_int }
694   \prop_put:NnV \l_tmpa_prop { final } \l_tmpa_int

```

3. The “status” of the arrow, with 3 possible values: empty, `independent`, or `new-group`.

```

695   \prop_put:NnV \l_tmpa_prop { status } \l__wa_status_arrow_str

```

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

```

696   \prop_put:Nnn \l_tmpa_prop { options } { #1 , #3 }

```

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

```

697   \prop_put:Nnn \l_tmpa_prop { label } { #2 }

```

6. The number of the line where the command `\Arrow` is issued in the TeX source (as of now, this is only useful for an error message).

```

698   \prop_put:Nnx \l_tmpa_prop { input-line } \msg_line_number:

```

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.

```

699   \prop_gclear_new:c
700   { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \g__wa_arrow_int _ prop }
701   \prop_gset_eq:cN
702   { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \g__wa_arrow_int _ prop }
703   \l_tmpa_prop
704 }

```

The command `\Arrow` (or the corresponding command with a name given by the user with the option `command-name`) will be available only in the last column of the environments `{WithArrows}` and `{DispWithArrows}`. In the other columns, the command will be linked to the following command `\@@_Arrow_first_columns`: which will raise an error.

```
705 \cs_new_protected:Npn \__wa_Arrow_first_columns:
706   { \__wa_error:n { Arrow~not~in~last~column } \__wa_Arrow }
```

11.8 The environments `{WithArrows}` and `{DispWithArrows}`

11.8.1 Code before the `\halign`

The command `\@@_pre_halign:n` is a code common to the environments `{WithArrows}` and `{DispWithArrows}`. The argument is the list of options given to the environment.

```
707 \cs_new_protected:Npn \__wa_pre_halign:n #1
```

First, the initialisation of `\l_@@_type_env_str` which is the name of the encompassing environment. In fact, this token list is used only in the error messages.

```
708   {
709   <*LaTeX>
710     \str_clear_new:N \l__wa_type_env_str
711     \str_set:NV \l__wa_type_env_str \@currenvir
712   </LaTeX>
```

We deactivate the potential externalization of Tikz. The Tikz elements created by `witharrows` can't be externalized since they are created in Tikz pictures with `overlay` and `remember picture`.

```
713   \cs_if_exist:NT \tikz@library@external@loaded
714     { \tikzset { external / export = false } }
```

The token list `\l_@@_name_str` will contain the potential name of the environment (given with the option `name`). This name will be used to create aliases for the names of the nodes.

```
715   \str_clear_new:N \l__wa_name_str
```

The parameter `\l_@@_status_arrow_str` will be used to store the “status” of an individual arrow. It will be used to fill the field “status” in the property list describing an arrow.

```
716   \str_clear_new:N \l__wa_status_arrow_str
```

The dimension `\l_@@_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_arrow_int`) is used.

```
717   \dim_zero_new:N \l__wa_x_dim
```

The variable `\l_@@_input_line_str` will be used only to store, for each command `\Arrow` the line (in the TeX file) where the command is issued. This information will be stored in the field “input-line” of the arrow. As of now, this information is used only in the error message of a arrow impossible to draw (because it arrives after the last row of the environment).

```
718   \str_clear_new:N \l__wa_input_line_str
```

The initialisation of the counters `\g_@@_arrow_int`, `\g_@@_line_int` and `\g_@@_col_int`. However, we have to save their previous values with the stacks created for this end.

```
719   \seq_gput_right:NV \g__wa_arrow_int_seq \g__wa_arrow_int
720   \int_gzero:N \g__wa_arrow_int
721   \seq_gput_right:NV \g__wa_line_int_seq \g__wa_line_int
722   \int_gzero:N \g__wa_line_int
723   \seq_gput_right:NV \g__wa_col_int_seq \g__wa_col_int
724   \int_gzero:N \g__wa_col_int
```

In the preamble of the `\halign`, there will be *two* counters of the columns. The aim of this programming is to detect the utilisation of a command `\omit` in a cell of the `\halign` (it should be forbidden). For example, in the part of the preamble concerning the third column (if there is a third column in the environment), we will have the following instructions :

```

\int_gincr:N \g__col_int
\int_gset:Nn \g__static_col_int 3

```

The counter `\g__col_int` is incremented dynamically and the second is static. If the user has used a command `\omit`, the dynamic incrementation is not done in the cell and, at this end of the row, the difference between the counters may infer the presence of `\omit` at least once.

```

725 \int_gzero_new:N \g__wa_static_col_int

```

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

```

726 \seq_gput_right:Nn \g__wa_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 six 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).

```

727 \seq_set_eq:NN \l_tmpa_seq \g__wa_position_in_the_tree_seq
728 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
729 \str_clear_new:N \l__wa_prefix_str
730 \str_set:Nx \l__wa_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }

```

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

```

731 \cs_set_eq:NN \@_cr \@@_cr:
732 \dim_zero:N \mathsurround

```

These counters will be used later as variables.

```

733 \int_zero_new:N \l__wa_initial_int
734 \int_zero_new:N \l__wa_final_int
735 \int_zero_new:N \l__wa_arrow_int
736 \int_zero_new:N \l__wa_pos_of_arrow_int
737 \int_zero_new:N \l__wa_jump_int

```

The counter `\l__wa_jump_int` corresponds to the option `jump`. Now, we set the default value for this option.

```

738 \int_set:Nn \l__wa_jump_int \c_one_int

```

The string `\l__wa_format_str` corresponds to the option `format`. Now, we set the default value for this option.

```

739 \str_set:Nn \l__wa_format_str { rl }

```

In (the last column of) `{DispWithArrows}`, it's possible to put several labels (for the same number of equation). That's why these labels will be stored in a sequence `\l__wa_labels_seq`.

```

740 \LaTeX
741 \seq_clear_new:N \l__wa_labels_seq
742 \bool_set_false:N \l__wa_tag_next_line_bool
743 \LaTeX

```

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.³¹

```

744 \skip_zero:N \l__wa_interline_skip

```

³¹It's recalled that, by design, the option `interline` of an environment doesn't apply in the nested environments.

The value corresponding to the key `code-before` 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 `code-after`.

```
745 \tl_clear_new:N \l__wa_code_before_tl
746 \tl_clear_new:N \l__wa_code_after_tl
```

We process the options given to the environment `{WithArrows}` or `{DispWithArrows}`.

```
747 \str_clear_new:N \l__wa_previous_key_str
748 \bool_if:NT \l__wa_in-WithArrows_bool
749   { \keys_set:nn { WithArrows / WithArrows } { #1 } }
750 \bool_if:NT \l__wa_in-DispWithArrows_bool
751   { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

Now we link the command `\Arrow` (or the corresponding command with a name given by the user with the option `command-name`: that's why the following line must be after the loading of the options) to the command `\@@_Arrow_first_columns`: which will raise an error.

```
752 \cs_set_eq:cN \l__wa_command_name_str \__wa_Arrow_first_columns:
```

It's only in the last column of the environment that it will be linked to the command `\@@_Arrow:`.

The counter `\l_@@_nb_cols_int` is the number of columns in the `\halign` (excepted the column for the labels of equations in `{DispWithArrows}` and excepted eventuals other columns in `{WithArrows}` allowed by the option `more-columns`).

```
753 \int_set:Nn \l__wa_nb_cols_int { \str_count:N \l__wa_format_str }
```

Be careful! The following counter `\g_@@_col_int` will be used for two usages:

- during, the construction of the preamble of the `\halign`, it will be used as counter for the number of the column under construction in the preamble (since the preamble is constructed backwards, `\g_@@_col_int` will go decreasing from `\l_@@_nb_cols_int` to 1) ;
- once the preamble constructed, the primitive `\halign` is executed, and, in each row of the `\halign`, the counter `\g_@@_col_int` will be increased from column to column.

```
754 \int_gset_eq:NN \g__wa_col_int \l__wa_nb_cols_int
```

We convert the format in a sequence because we use it as a stack (with the top of the stack at the end of the sequence) in the construction of the preamble.

```
755 \seq_clear_new:N \l__wa_format_seq
756 \seq_set_split:NnV \l__wa_format_seq { } \l__wa_format_str
```

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`).

```
757 \*LaTeX
758   \bool_if:NT \g__wa_footnote_bool { \begin { savenotes } }
759 \*LaTeX
```

We execute the code `\l_@@_code_before_tl` of the option `code-before` of the environment after the eventual `\begin{savenotes}` and, symmetrically, 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 `code-after`).

```
760 \l__wa_code_before_tl
```

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).

```
761 \spread@equation
```



```

762 <*LaTeX>
763   \cs_set_eq:NN \notag \__wa_notag:
764   \cs_set_eq:NN \nonumber \__wa_nonumber:
765   \cs_set_eq:NN \tag \__wa_tag
766   \cs_set_eq:NN \__wa_old_label \label
767   \cs_set_eq:NN \label \__wa_label:n
768   \cs_set_eq:NN \tagnextline \__wa_tagnextline:
769 </LaTeX>
770 }

```

This is the end of `\@@_pre_halign:n`.

11.8.2 The construction of the preamble of the `\halign`

The control sequence `\@@_construct_halign:` will “start” the `\halign` and the preamble. In fact, it constructs all the preamble excepted the end of the last column (more precisely: except the part concerning the construction of the left node and the right node).

The same function `\@@_construct_halign:` will be used both for the environment `{WithArrows}` and the environment `{DispWithArrows}`.

Several important points must be noted concerning that construction of the preamble.

- The construction of the preamble is done by reading backwards the format `\l_@@_format_str` and adding the corresponding tokens in the input stream of TeX. That means that the part of the preamble concerning the last cell will be constructed first.
- The function `\@@_construct_halign:` is recursive in order to treat succesively all the letters of the preamble.
- Each part of the preamble is created with a `\use:x` function. This expansion of the preamble gives the ability of controlling which parts of the code will be expanded during the construction of the preamble (other parts will be expanded and executed only during the execution of the `\halign`).
- The counter `\g_@@_col_int` is used during the loop of the construction of the preamble but, it will also appears in the preamble (we could have chosen two differents counters but this way saves a counter).

```

771 \cs_new_protected:Npn \__wa_construct_halign:
772 {
773   \seq_pop_right:NNTF \l__wa_format_seq \l__wa_type_col_str
774   {

```

Here is the `\use:x` which is fundamental: it will really construct the part of the preamble corresponding to a column by expanding only some parts of the following code.

```

775   \use:x
776   {

```

Before the recursive call of `\@@_construct_halign:`, we decrease the integer `\g_@@_col_int`. But, during the construction of the column which is constructed first (that is to say which is the last column of the `\halign`), it is *not* lowered because `\int_decr:N`, which is protected, won't be expanded by the `\use:x`.

We begin the construction of a generic column.

```

777       \int_gdecr:N \g__wa_col_int
778       \__wa_construct_halign:
779       \int_compare:nNnT \g__wa_col_int = \l__wa_nb_cols_int
780       {

```

We redefine the command `\Arrow` (or the name given to the corresponding command by the option `command-name`) in each cell of the last column. The braces around `\l_@@_command_name_str` are mandatory because `\l_@@_command_name_str` will be expanded by the `\use:x` and the command `\cs_set_eq:cN` must still be efficient during the execution of the `\halign`.

```

781         \cs_set_eq:cN { \l__wa_command_name_str } \__wa_Arrow
782 <*LaTeX>
783         \bool_if:NT \l__wa_in_DispWithArrows_bool
784         {

```

The command `\@@_test_if_to_tag:` (which is protected and, thus, will not be expanded during the construction of the preamble) will test, at each row, whether the current row must be tagged (and the tag will be put in the very last column).

```
785         \_wa_test_if_to_tag:
```

The command `\@@_set_qedhere:` will do a redefinition of `\qedhere` in each cell of the last column.

```
786         \bool_if:NT \c_wa_amsthm_loaded_bool \_wa_set_qedhere:
787     }
788 </LaTeX>
789 }
790 \str_if_eq:VnT \l_wa_type_col_str { c } \hfil
791 \str_if_eq:VnT \l_wa_type_col_str { r } \hfill
792 \int_gincr:N \g_wa_col_int
793 \int_gset:Nn \g_wa_static_col_int { \int_use:N \g_wa_col_int }
794 \c_math_toggle_token
795 {
796     { }
797     \bool_if:NT \l_wa_displaystyle_bool \displaystyle
798     ####
799 }
800 \c_math_toggle_token
801 \int_compare:nNnTF \g_wa_col_int = \l_wa_nb_cols_int
802 { \_wa_construct_nodes: }
803 {
```

The following glue (`\hfil`) will be added only if we are not in the last cell because, in the last cell, a glue (`=skip`) is added between the nodes (in `\@@_construct_nodes:`).

```
804     \str_if_eq:VnT \l_wa_type_col_str { l } \hfil
805     \str_if_eq:VnT \l_wa_type_col_str { c } \hfil
806     \bool_if:NT \l_wa_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
807     &
808 }
809 }
810 }
```

Now the tokens that will be inserted after the analyze of all the tokens of the format: here is the token `\halign`.

```
811 {
812     \bool_if:NTF \l_wa_in_WithArrows_bool
813     {
814         \ialign
815         \bgroup
816     }
817     {
818         \halign to \l_wa_linewidth_dim
819         \bgroup
820         \bool_if:NT \l_wa_fleqn_bool
821         { \skip_horizontal:N \l_wa_mathindent_dim }
822     }
823     \int_gincr:N \g_wa_line_int
824     \int_gzero:N \g_wa_col_int
825     \tl_if_eq:NNF \l_wa_left_brace_tl \c_novalue_tl
826     {
827         \skip_horizontal:n
828         { \box_wd:N \l_wa_left_brace_box + \l_wa_delim_wd_dim }
829     }
830     \strut
831 }
832 }
```

The command `\@@_construct_nodes:` is only for the lisibility of the code because, in fact, it is used only once. It constructs the “left node” and the “right node” at the end of each row of the arrow.

```
833 \cs_new_protected:Npn \_wa_construct_nodes:
834 {
```

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

```

835 \tikz [ remember~picture , overlay ]
836 \node
837 [
838   node~contents = { } ,
839   __wa_node_style ,
840   name = wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - l ,
841   alias =
842   {
843     \str_if_empty:NF \l__wa_name_str
844     { \l__wa_name_str - \int_use:N \g__wa_line_int - l }
845   }
846 ]
847 ;
848 \hfil

```

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

```

849 \tikz [ remember~picture , overlay ]
850 \node
851 [
852   node~contents = { } ,
853   __wa_node_style ,
854   name = wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - r ,
855   alias =
856   {
857     \str_if_empty:NF \l__wa_name_str
858     { \l__wa_name_str - \int_use:N \g__wa_line_int - r }
859   }
860 ]
861 ;
862 \bool_if:NT \l__wa_show_node_names_bool
863 {
864   \hbox_overlap_right:n
865   { \small wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - r }
866 }
867 }

```

11.8.3 The environment `{WithArrows}`

```

868 <*LaTeX>
869 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
870 </LaTeX>
871 <*plain-TeX>
872 \cs_new_protected:Npn \WithArrows
873 {
874   \group_begin:
875   \peek_meaning:NTF [
876     { \WithArrows_i }
877     { \WithArrows_i [ ] }
878   ]
879   \cs_new_protected:Npn \WithArrows_i [ #1 ]
880 </plain-TeX>
881 {
882   \bool_set_true:N \l__wa_in-WithArrows_bool
883   \bool_set_false:N \l__wa_in-DispWithArrows_bool
884 <*plain-TeX>
885   \str_clear_new:N \l__wa_type_env_str
886   \str_set:Nn \l__wa_type_env_str { WithArrows }
887 </plain-TeX>

```

```

888   \__wa_pre_halign:n { #1 }
889   \if_mode_math: \else:
890     \__wa_error:n { WithArrows-outside~math~mode }
891   \fi:

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`³² 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 mode³³ and therefore, we can use `\vcenter`.

```

892   \int_case:nn \l__wa_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
893   \bgroup

```

We begin the `\halign` and the preamble. During the construction of the preamble, `\l_tmpa_int` will be incremented during each column constructed.

```

894   \__wa_construct_halign:

```

In fact, the construction of the preamble is not finished. We add a little more.

An environment `{WithArrows}` should have a number of columns equal to the length of its format (by default, 2 since the default format is `r1`). Nevertheless, if the user wants to use more columns (without arrows) it's possible with the option `more-columns`.

```

895   &&
896   \__wa_error:n { Too-much-columns-in-WithArrows }
897   \c_math_toggle_token
898   \bool_if:NT \l__wa_displaystyle_bool \displaystyle
899   { ## }
900   \c_math_toggle_token
901   \cr
902   }

```

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`).

```

903 {*plain-TeX}
904 \cs_new_protected:Npn \endWithArrows
905 </plain-TeX>
906 {
907   \
908   \egroup
909   \egroup
910   \__wa_post_halign:

```

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`).

```

911 {*LaTeX}
912   \bool_if:NT \g__wa_footnote_bool { \end { savenotes } }
913 </LaTeX>
914 {*plain-TeX}
915   \group_end:
916 </plain-TeX>
917 }

```

This is the end of the environment `{WithArrows}`.

11.8.4 After the construction of the `\halign`

The command `\@@_post_halign:` is a code common to the second part of the environment `{WithArrows}` and the environment `{DispWithArrows}`.

```

918 \cs_new_protected:Npn \__wa_post_halign:

```

³²Notice that the use of `\vtop` seems color-safe here...

³³An error is raised if the environment is used outside math mode.

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

```
919 {
920   \cs_set:Npn \WithArrowsRightX { \g__wa_right_x_dim }
```

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

```
921   \int_compare:nNnT \g__wa_arrow_int > 0 \__wa_scan_arrows:
```

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

```
922   \group_begin:
923     \tikzset { every~picture / .style = __wa_standard }
```

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

```
924     \cs_set:Npn \WithArrowsNbLines { \int_use:N \g__wa_line_int }
```

The command `\MultiArrow` is available in `code-after`, and we have a special version of `\Arrow`, called “`\Arrow` in `code-after`” in the documentation.³⁴

```
925     \cs_set_eq:NN \MultiArrow \__wa_MultiArrow:nn
926     \cs_set_eq:cN \l__wa_command_name_str \__wa_Arrow_code_after
927     \bool_set_true:N \l__wa_in_code_after_bool
928     \l__wa_code_after_tl
929   \group_end:
```

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

```
930   \seq_gpop_right:NN \g__wa_position_in_the_tree_seq \l_tmpa_tl
931   \seq_gpop_right:NN \g__wa_position_in_the_tree_seq \l_tmpa_tl
932   \seq_gput_right:Nx \g__wa_position_in_the_tree_seq
933     { \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`.

```
934   \int_compare:nNnT { \seq_count:N \g__wa_position_in_the_tree_seq } = 1
935     { \int_gincr:N \g__wa_last_env_int }
```

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

```
936   \seq_gpop_right:NN \g__wa_arrow_int_seq \l_tmpa_tl
937   \int_gset:Nn \g__wa_arrow_int \l_tmpa_tl
938   \seq_gpop_right:NN \g__wa_line_int_seq \l_tmpa_tl
939   \int_gset:Nn \g__wa_line_int \l_tmpa_tl
940   \seq_gpop_right:NN \g__wa_col_int_seq \l_tmpa_tl
941   \int_gset:Nn \g__wa_col_int \l_tmpa_tl
942 }
```

That's the end of the command `\@@_post_halign:`.

³⁴As 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 `code-after` provides options and has the name of a function defined with `\NewDocumentCommand`.

11.8.5 The command of end of row

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 an `\halign` structure.

First, we remove an eventual token `*` (just after the `\@`: there should not be space between the two) since the commands `\@` and `\@*` are equivalent in an environment `{WithArrows}` (an environment `{WithArrows}`, like an environment `{aligned}` of `amsmath`, is always unbreakable).

```
943 \cs_new_protected:Npn \@@_cr:
944   {
945     \scan_stop:
```

We try to detect some `\omit` (as of now, an `\omit` in the last column is not detected).

```
946     \int_compare:nNnF \g__wa_col_int = \g__wa_static_col_int
947       { \@@_error:n { omit-probably-used } }
948     \prg_replicate:nn { \l__wa_nb_cols_int - \g__wa_static_col_int } { & { } }
949     \group_align_safe_begin:
950     \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
951   }
```

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.

```
952 \cs_new_protected:Npn \@@_cr_i:
953   { \peek_meaning:NTF [ \@@_cr_ii: { \@@_cr_ii: [ \c_zero_dim ] } }
```

Now, we test if the next token is the token `\end`. Indeed, we want to test if the following tokens are `\end{WithArrows}` (or `\end{DispWithArrows}`, etc). In this case, we raise an error because the user must not put `\@` at the end of its alignment.

```
954 \<LaTeX>
955 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
956   {
957     \peek_meaning_ignore_spaces:NTF \end
958     {
959       \@@_cr_iii:n { #1 }
```

The analyse of the argument of the token `\end` must be after the `\group_align_safe_end:` which is the beginning of `\@@_cr_iii:n`.

```
960       \@@_wa_analyze_end:Nn
961     }
962     { \@@_cr_iii:n { #1 } }
963   }
964 \cs_new_protected:Npn \@@_cr_iii:n #1
965 \</LaTeX>
966 \<*plain-TeX>
967 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
968 \</plain-TeX>
969   {
970     \group_align_safe_end:
```

For the environment `{DispWithArrows}`, the behaviour of `\@` is different because we add the last column which is the column for the tag (number of the equation). Even if there is no tag, this column is used for the `v`-nodes.³⁵

```
971     \bool_if:NT \l__wa_in_DispWithArrows_bool
```

At this stage, we know that we have a tag to put if (and only if) the value of `\l_@@_tags_clist` is the comma list `all` (only one element). Maybe, previously, the value of `\l_@@_tags_clist` was, for example, `1,last` (which means that only the first line and the last line must be tagged). However, in this case, the comparison with the number of line has been done before and, now, if we are in a line to tag, the value of `\l_@@_tags_clist` is `all`.

³⁵The `v`-nodes are used to compute the abscissa of the right margin, used by the option `wrap-lines`.

```

972     {
973     (*LaTeX)
974     \clist_if_in:NnTF \l__wa_tags_clist { all }
975     {

```

Here, we can't use `\refstepcounter{equation}` because if the user has issued a `\tag` command, we have to use `\l_@@_tag_tl` and not `\theequation`. That's why we have to do the job done by `\refstepcounter` manually.

First, the incrementation of the counter (potentially).

```

976     \tl_if_empty:NT \l__wa_tag_tl { \int_gincr:N \c@equation }

```

We store in `\g_tmpa_tl` the tag we will have to compose at the end of the line. We use a global variable because we will use it in the *next* cell (after the `&`).

```

977     \cs_gset:Npx \g_tmpa_tl
978     { \tl_if_empty:NTF \l__wa_tag_tl \theequation \l__wa_tag_tl }

```

It's possible to put several labels for the same line (it's not possible in the environments of `amsmath`). That's why the different labels of a same line are stored in a sequence `\l_@@_labels_seq`.

```

979     \seq_if_empty:NF \l__wa_labels_seq
980     {

```

Now, we do the job done by `\refstepcounter` and by the redefinitions of `\refstepcounter` done by some packages (the incrementation of the counter has been done yet).

First an action which is in the definition of `\refstepcounter`. The command `\p@equation` is redefined by some extensions like `fnclab`.

```

981     \cs_set:Npx \@currentlabel { \p@equation \g_tmpa_tl }

```

Then, an action done by `hyperref` in its redefinition of `\refstepcounter`.

```

982     \bool_if:NT \c__wa_hyperref_loaded_bool
983     {
984     \str_set:Nn \This@name { equation }
985     \hyper@refstepcounter { equation }
986     }

```

Then, an action done by `cleveref` in its redefinition of `\refstepcounter`. The package `cleveref` creates in the aux file a command `\cref@currentlabel` similar to `\@currentlabel` but with more informations.

```

987     \bool_if:NT \c__wa_cleveref_loaded_bool
988     {
989     \cref@constructprefix { equation } \cref@result
990     \protected@edef \cref@currentlabel
991     {
992     [
993     \cs_if_exist:NTF \cref@equation@alias
994     \cref@equation@alias
995     { equation }
996     ]
997     [ \arabic { equation } ] [ \cref@result ]
998     \p@equation \g_tmpa_tl
999     }
1000    }

```

Now, we can issue the command `\label` (some packages may have redefined `\label`, for example `typedref`) for each item in the sequence of the labels (it's possible with `witharrows` to put several labels to the same line and that's why the labels are in the sequence `\l_@@_labels_seq`).

```

1001     \seq_map_function:NN \l__wa_labels_seq \__wa_old_label
1002     }

```

We save the booleans `\l_@@_tag_star_bool` and `\l_@@_qedhere_bool` because they will be used in the *next* cell (after the `&`). We recall that the cells of a `\halign` are TeX groups.

```

1003     \__wa_save:N \l__wa_tag_star_bool
1004     \__wa_save:N \l__wa_qedhere_bool
1005     \bool_if:NT \l__wa_tag_next_line_bool
1006     {
1007     \openup -\jot

```

```

1008         \bool_set_false:N \l__wa_tag_next_line_bool
1009         \notag \&
1010     }
1011     &
1012     \__wa_restore:N \l__wa_tag_star_bool
1013     \__wa_restore:N \l__wa_qedhere_bool
1014     \bool_if:NT \l__wa_qedhere_bool
1015     { \hbox_overlap_left:n \__wa_qedhere_i: }
1016     \cs_set_eq:NN \theequation \g_tmpa_tl
1017     \bool_if:NT \l__wa_tag_star_bool
1018     { \cs_set_eq:NN \tagform@ \prg_do_nothing: }

```

We use `\@eqnnum` (we recall that there are two definitions of `\@eqnnum`, a standard definition and another, loaded if the class option `leqno` is used). However, of course, the position of the v-node is not the same whether the option `leqno` is used or not. That's here that we use the flag `\c_@@_leqno_bool`.

```

1019     \hbox_overlap_left:n
1020     {
1021         \bool_if:NF \c__wa_leqno_bool
1022         {
1023             \tikz [ __wa_standard ]
1024                 \coordinate ( \int_use:N \g__wa_line_int - v ) ;
1025         }
1026         \quad
1027         \@eqnnum
1028     }
1029     \bool_if:NT \c__wa_leqno_bool
1030     {
1031         \tikz [ __wa_standard ]
1032             \coordinate ( \int_use:N \g__wa_line_int - v ) ;
1033     }
1034 }
1035 {
1036     \__wa_save:N \l__wa_qedhere_bool
1037 </LaTeX>
1038     &
1039 <*LaTeX>
1040     \__wa_restore:N \l__wa_qedhere_bool
1041     \bool_if:NT \l__wa_qedhere_bool
1042     { \hbox_overlap_left:n \__wa_qedhere_i: }
1043 </LaTeX>
1044     \tikz [ __wa_standard ]
1045         \coordinate ( \int_use:N \g__wa_line_int - v ) ;
1046 <*LaTeX>
1047 }
1048 </LaTeX>
1049 }
1050 \dim_compare:nNnT { #1 } < \c_zero_dim
1051 { \__wa_error:n { option~of~cr~negative } }
1052
1053 \cr
1054 \noalign
1055 {
1056     \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1057     \skip_vertical:n { \l_tmpa_dim + \l__wa_interline_skip }
1058     \scan_stop:
1059 }
1060 }

```

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

The following command will be used when, after a `\&` (and its optional arguments) there is a `\end`. You want to know if this is the end of the environment `{WithArrows}` (or `{DispWithArrows}`, etc.) because, in this case, we will explain that the environment must not be ended by `\&`. If it is not the

case, that means it's a classical situation of LaTeX environments not correctly imbricated and there will be a LaTeX error.

```

1061 <*LaTeX>
1062 \cs_new_protected:Npn \__wa_analyze_end:Nn #1 #2
1063   {
1064     \exp_args:NV \str_if_eq:nnT \l__wa_type_env_str { #2 }
1065     { \__wa_warning:n { newline-at-the-end-of-env } }

```

We repeat in the stream the `\end{...}` we have extracted.

```

1066     \end { #2 }
1067   }
1068 </LaTeX>

```

11.8.6 The environment `{DispWithArrows}`

For the environment `{DispWithArrows}`, the general form of the construction is of the type:

```
\[\vtop{\halign to \displaywidth {...}}\]
```

The purpose of the `\vtop` is to have an environment unbreakable.

However, if we are just after an item of a LaTeX list or at the beginning of a `{minipage}`, the construction is slightly different:

```
\[\vtop{\halign to \linewidth {...}}\]
```

The boolean `\l_@@_in_label_or_minipage_bool` will be raised if we are just after a `\item` of a list of LaTeX or at the beginning of a `{minipage}`.

```

1069 <*LaTeX>
1070 \bool_new:N \l__wa_in_label_or_minipage_bool
1071 </LaTeX>
1072 <*LaTeX>
1073 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1074 </LaTeX>
1075 <*plain-TeX>
1076 \cs_new_protected:Npn \DispWithArrows
1077   {
1078     \group_begin:
1079     \peek_meaning:NTF <
1080       { \DispWithArrows_i }
1081       { \DispWithArrows_i < \c_novalue_tl > }
1082   }
1083 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1084   {
1085     \peek_meaning:NTF [
1086       { \DispWithArrows_ii < #1 > }
1087       { \DispWithArrows_ii < #1 > [ ] }
1088   }
1089 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1090 </plain-TeX>
1091   {
1092     \bool_set_true:N \l__wa_in_DispWithArrows_bool
1093 <*plain-TeX>
1094     \str_clear_new:N \l__wa_type_env_str
1095     \str_set:Nn \l__wa_type_env_str { DispWithArrows }
1096 </plain-TeX>

```

If `mathtools` has been loaded with the option `showonlyrefs`, we disable the code of `mathtools` for the option `showonlyrefs` with the command `\MT_showonlyrefs_false`: (it will be reactivated at the end of the environment).

```

1097 <*LaTeX>
1098   \bool_if:nT \c__wa_mathtools_loaded_bool
1099   {
1100     \MH_if_boolean:nT { show_only_refs }
1101     {
1102       \MT_showonlyrefs_false:

```

However, we have to re-raise the flag `{show_only_refs}` of `mhsetup` because it has been switched off by `\MT_showonlyrefs_false`: and we will use it in the code of the new version of `\label`.

```
1103     \MH_set_boolean_T:n { show_only_refs }
1104   }
1105 }
```

An action done by `typedref` in its redefinition of `\refstepcounter`. The command `\sr@name` is a prefix added to the name of the label by the redefinition of `\label` done by `typedref`.

```
1106   \bool_if:NT \c_wa_typedref_loaded_bool { \str_set:Nn \sr@name { equation } }
```

The command `\intertext@` is a command of `amsmath` which loads the definition of `\intertext`.

```
1107   \bool_if:NT \c_wa_amsmath_loaded_bool \intertext@
1108 /LaTeX
1109   \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l__wa_left_brace_tl { #1 } }
1110   \__wa_pre_halign:n { #2 }
```

If `subequations` is used, we encapsulate the environment in an environment `{subequations}` of `amsmath`.

```
1111 (*LaTeX)
1112   \bool_if:NT \l__wa_subequations_bool { \begin { subequations } }
1113 /LaTeX
```

Since the version 1.16 of `witharrows`, no space is added between an `\item` of a LaTeX list and an environment `{DispWithArrows}` except with the option `standard-behaviour-with-items` stored in the boolean `\l_@@_sbwi_bool`. We have to know if we are just after an `\item` and this information will be stored in `\l_@@_in_label_or_minipage_bool`.

```
1114 (*LaTeX)
1115   \bool_if:NF \l__wa_sbwi_bool
1116   {
1117     \if@inlabel
1118     \bool_set_true:N \l__wa_in_label_or_minipage_bool
1119     \fi
1120     \if@minipage
1121     \bool_set_true:N \l__wa_in_label_or_minipage_bool
1122     \fi
1123   }
1124 /LaTeX
1125   \tl_if_eq:NNF \l__wa_left_brace_tl \c_novalue_tl
1126   {
```

We compute the value of the width of the left delimiter.

```
1127   \hbox_set:Nn \l_tmpa_box
1128   {
```

Even if the default value of `\nulldelimiterspace` is 1.2 pt, we take it into account.

```
1129     \group_begin:
1130     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1131     \c_math_toggle_token
1132     \left \l__wa_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1133     \c_math_toggle_token
1134     \group_end:
1135   }
1136   \dim_zero_new:N \l__wa_delim_wd_dim
1137   \dim_set:Nn \l__wa_delim_wd_dim { \box_wd:N \l_tmpa_box }
1138   \box_clear_new:N \l__wa_left_brace_box
1139   \hbox_set:Nn \l__wa_left_brace_box
1140   {
1141     \group_begin:
1142     \cs_set_eq:NN \label \__wa_old_label
1143     \c_math_toggle_token
1144     \bool_if:NT \l__wa_displaystyle_bool \displaystyle
1145     \l__wa_left_brace_tl
1146     { }
1147     \c_math_toggle_token
```

```

1148         \group_end:
1149     }
1150 }

```

The token list `\l_@@_tag_tl` will contain the argument of the command `\tag`.

```

1151 (*LaTeX)
1152     \tl_clear_new:N \l__wa_tag_tl

1153     \bool_set_false:N \l__wa_qedhere_bool

```

The boolean `\l_@@_tag_star_bool` will be raised if the user uses the command `\tag` with a star.

```

1154     \bool_set_false:N \l__wa_tag_star_bool
1155 </LaTeX>

1156     \if_mode_math:
1157         \__wa_fatal:n { DispWithArrows~in~math~mode }
1158     \fi:

```

The construction is not exactly the same whether we are just after an `\item` of a LaTeX list or not. We know if we are after an `\item` thanks to the boolean `\l_@@_in_label_or_minipage_bool`.

```

1159 (*plain-TeX)
1160     \dim_zero_new:N \linewidth
1161     \dim_set_eq:NN \linewidth \displaywidth
1162 </plain-TeX>
1163 (*LaTeX)
1164     \bool_if:NTF \l__wa_in_label_or_minipage_bool
1165         { \c_math_toggle_token }
1166         {
1167 </LaTeX>

```

We don't use `\[` of LaTeX because some extensions, like `autonom`, do a redefinition of `\[`. However, we put the following lines which are in the definition of `\[` even though they are in case of misuse.

```

1168     \if_mode_vertical:
1169         \nointerlineskip
1170         \hbox_to_wd:nn { .6 \linewidth } { }
1171     \fi:
1172     \c_math_toggle_token \c_math_toggle_token
1173 (*LaTeX)
1174 }
1175 </LaTeX>

1176     \dim_zero_new:N \l__wa_linewidth_dim
1177 (*LaTeX)
1178     \bool_if:NTF \l__wa_in_label_or_minipage_bool
1179         { \dim_set_eq:NN \l__wa_linewidth_dim \linewidth }
1180         { \dim_set_eq:NN \l__wa_linewidth_dim \displaywidth }
1181 </LaTeX>
1182 (*plain-TeX)
1183     \dim_set_eq:NN \l__wa_linewidth_dim \displaywidth
1184 </plain-TeX>

1185     \box_clear_new:N \l__wa_halign_box
1186     \setbox \l__wa_halign_box \vtop \bgroup
1187     \tabskip =
1188     \bool_if:NTF \l__wa_fleqn_bool
1189         \c_zero_skip
1190         { 0 pt plus 1000 pt minus 1000 pt }

1191     \__wa_construct_halign:
1192     \tabskip = 0 pt plus 1000 pt minus 1000 pt
1193     &

```

If the user tries to use more columns than the length of the format, we have to raise an error. However, the error won't be in the next column which is the columns for the labels of the equations. The error will be after... and it must be after. That means that we must not have an error in the next column simply because we are not in math mode. That's why this column, even if it is for the labels, is in math mode.

```

1194     $ ## $
1195     \tabskip = \c_zero_skip
1196     &&
1197     \_wa_fatal:n { Too-much-columns~in~DispWithArrows }
1198     \bool_if:nT \c_false_bool { ## }
1199     \cr
1200   }

```

We begin the second part of the environment {DispWithArrows}.

```

1201 <*plain-TeX>
1202 \cs_new_protected:Npn \endDispWithArrows
1203 </plain-TeX>
1204 {
1205 <*LaTeX>
1206   \clist_if_in:NnT \l__wa_tags_clist { last }
1207   { \clist_set:Nn \l__wa_tags_clist { all } }
1208 </LaTeX>
1209   \

```

The following \egroup is for the \halign.

```

1210   \egroup
1211   \unskip \unpenalty \unskip \unpenalty
1212   \box_set_to_last:N \l_tmpa_box
1213   \nointerlineskip
1214   \box_use:N \l_tmpa_box
1215   \dim_gzero_new:N \g__wa_alignment_dim
1216   \dim_gset:Nn \g__wa_alignment_dim { \box_wd:N \l_tmpa_box }
1217   \box_clear_new:N \l__wa_new_box
1218   \hbox_set:Nn \l__wa_new_box { \hbox_unpack_clear:N \l_tmpa_box }
1219   \dim_compare:nNnT
1220     { \box_wd:N \l__wa_new_box } < \g__wa_alignment_dim
1221     { \dim_gset:Nn \g__wa_alignment_dim { \box_wd:N \l__wa_new_box } }

```

The \egroup is for the box \l_@@_halign_box.

```

1222   \egroup
1223   \tl_if_eq:NNTF \l__wa_left_brace_tl \c_novaluel_tl
1224     { \box_use_drop:N \l__wa_halign_box }
1225     {
1226       \hbox_to_wd:nn \l__wa_linewidth_dim
1227       {
1228         \bool_if:NNTF \l__wa_fleqn_bool
1229           { \skip_horizontal:n \l__wa_mathindent_dim }
1230         \hfil
1231         \hbox_to_wd:nn \g__wa_alignment_dim
1232         {
1233           \box_use_drop:N \l__wa_left_brace_box
1234           \dim_set:Nn \l_tmpa_dim
1235             {
1236               \box_ht:N \l__wa_halign_box
1237               + \box_dp:N \l__wa_halign_box
1238             }
1239           \group_begin:
1240           \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1241           \c_math_toggle_token
1242           \left \l__wa_replace_left_brace_by_tl
1243           \vcenter to \l_tmpa_dim { \vfil }
1244           \right.
1245           \c_math_toggle_token

```

```

1246         \group_end:
1247         \hfil
1248     }
1249     \hfil
1250 }
1251 \skip_horizontal:n { - \l__wa_linewidth_dim }
1252 \vcenter { \box_use_drop:N \l__wa_halign_box }
1253 }

```

We compute the dimension `\g_@@_right_x_dim`. As a first approximation, `\g_@@_right_x_dim` is the x -value of the right side of the current composition box. In fact, we must take into account the potential labels of the equations. That's why we compute `\g_@@_right_x_dim` with the v -nodes of each row specifically built in this goal. `\g_@@_right_x_dim` is the minimal value of the x -value of these nodes.

```

1254     \dim_gzero_new:N \g__wa_right_x_dim
1255     \dim_gset_eq:NN \g__wa_right_x_dim \c_max_dim
1256 (*LaTeX)
1257     \begin { tikzpicture } [ __wa_standard ]
1258 
```

The code in `\@@_post_halign:` is common to `{WithArrows}` and `{DispWithArrows}`.

```

1280     \__wa_post_halign:

```

If `mathtools` has been loaded with the option `showonlyrefs`, we reactivate the code of `mathtools` for the option `showonlyrefs` with the command `\MT_showonlyrefs_true:` (it has been deactivated in the beginning of the environment).

```

1281 (*LaTeX)
1282     \bool_if:nT \c__wa_mathtools_loaded_bool
1283     { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1284     \bool_if:NTF \l__wa_in_label_or_minipage_bool
1285     {
1286         \c_math_toggle_token
1287         \skip_vertical:N \belowdisplayskip
1288     }
1289     { \c_math_toggle_token \c_math_toggle_token }
1290 
```

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`).

```

1296     \bool_if:NT \g__wa_footnote_bool { \end { savenotes } }
1297 
```

$$\left. \begin{array}{l} \text{plain-TeX} \\ \text{group_end:} \\ \text{plain-TeX} \\ \text{LaTeX} \end{array} \right\} \text{ignorespacesafterend}$$

```

1303 
```

With the environment `{DispWithArrows*}`, the equations are not numbered. We don't put `\begin{DispWithArrows}` and `\end{DispWithArrows}` because there is a `\@currenenvir` in some error messages.

```

1305 
```

$$\text{NewDocumentEnvironment } \{ \text{DispWithArrows* } \} \{ \}$$

```

1307   {
1308     \WithArrowsOptions { notag }
1309     \DispWithArrows
1310   }
1311   \endDispWithArrows
1312 
```

11.9 The commands `\tag`, `\notag`, `\label`, `\tagnextline` and `\qedhere` for `{DispWithArrows}`

Some commands are allowed only in the last column of the environment `{DispWithArrows}`. We write a command `\@@_if_in_last_col_of_disp:Nn` to execute this command only if we are in the last column. If we are in another column, an error is raised. The first argument of `\@@_if_in_last_col_of_disp:Nn` is the name of the command used in the error message and the second is the code to execute.

```

1313 \cs_new_protected:Npn \__wa_if_in_last_col_of_disp:Nn #1 #2
1314   {
1315     \bool_if:NTF \l__wa_in_WithArrows_bool
1316       { \__wa_error:nm { Not-allowed-in-WithArrows } { #1 } }
1317     {
1318       \int_compare:nNnTF \g__wa_col_int < \l__wa_nb_cols_int
1319         { \__wa_error:nm { Not-allowed-in-DispWithArrows } { #1 } }
1320         { #2 }
1321     }
1322   }
```

The command `\@@_notag:` will be linked to the command `\notag` and also to in the environments `{WithArrows}` and `{DispWithArrows}`.

```

1323 
```

$$\text{cs_new_protected:Npn } _ _ \text{wa_notag:}$$

```

1325   { \__wa_if_in_last_col_of_disp:Nn \notag { \clist_clear:N \l__wa_tags_clist } }
```

The command `\@@_nonumber:` will be linked to the command `\nonumber` and also to in the environments `{WithArrows}` and `{DispWithArrows}`.

```

1326 \cs_new_protected:Npn \__wa_nonumber:
1327   { \__wa_if_in_last_col_of_disp:Nn \nonumber { \clist_clear:N \l__wa_tags_clist } }
```

The command `\@@_tag` will be linked to `\tag` in `{WithArrows}` and `{DispWithArrows}`. We do the definition with `\NewDocumentCommand` because this command has a starred version.

```

1328 \NewDocumentCommand \__wa_tag { s m }
1329   {
```

```

1330 \__wa_if_in_last_col_of_disp:Nn \tag
1331 {
1332   \tl_if_empty:NF \l__wa_tag_tl
1333   { \__wa_error:nn { Multiple~tags } { #2 } }
1334   \clist_set:Nn \l__wa_tags_clist { all }
1335   \bool_if:nT \c__wa_mathtools_loaded_bool
1336   {
1337     \MH_if_boolean:nT { show_only_refs }
1338     {
1339       \MH_if_boolean:NF { show_manual_tags }
1340       { \clist_clear:N \l__wa_tags_clist }
1341     }
1342   }
1343   \tl_set:Nn \l__wa_tag_tl { #2 }
1344   \bool_set:Nn \l__wa_tag_star_bool { #1 }

```

The starred version `\tag*` can't be used if `amsmath` has not been loaded because this version does the job by deactivating the command `\tagform@` inserted by `amsmath` in the (two versions of the) command `\@eqnnum`.³⁶

```

1345   \bool_if:nT { #1 && ! \bool_if_p:N \c__wa_amsmath_loaded_bool }
1346   { \__wa_error:n { tag*~without~amsmath } }
1347 }
1348 }

```

The command `\@@_label:n` will be linked to `\label` in the environments `{WithArrows}` and `{DispWithArrows}`. In these environments, it's possible to put several labels for the same line (it's not possible in the environments of `amsmath`). That's why we store the different labels of a same line in a sequence `\l_@@_labels_seq`.

```

1349 \cs_new_protected:Npn \__wa_label:n #1
1350 {
1351   \__wa_if_in_last_col_of_disp:Nn \label
1352   {
1353     \seq_if_empty:NF \l__wa_labels_seq
1354     {
1355       \bool_if:NTF \c__wa_cleveref_loaded_bool
1356       { \__wa_error:n { Multiple~labels~with~cleveref } }
1357       { \__wa_error:n { Multiple~labels } }
1358     }
1359     \seq_put_right:Nn \l__wa_labels_seq { #1 }
1360     \bool_if:nT \c__wa_mathtools_loaded_bool
1361     {
1362       \MH_if_boolean:nT { show_only_refs }
1363       {
1364         \cs_if_exist:cTF { MT_r_#1 }
1365         { \clist_set:Nn \l__wa_tags_clist { all } }
1366         { \clist_clear:N \l__wa_tags_clist }
1367       }
1368     }
1369     \bool_if:nT \c__wa_autonum_loaded_bool
1370     {
1371       \cs_if_exist:cTF { autonum@#1Referenced }
1372       { \clist_set:Nn \l__wa_tags_clist { all } }
1373       { \clist_clear:N \l__wa_tags_clist }
1374     }
1375   }
1376 }

```

The command `\@@_tagnextline:` will be linked to `\tagnextline` in `{DispWithArrows}`.

```

1377 \cs_new_protected:Npn \__wa_tagnextline:

```

³⁶There are two versions of `@eqnnum`, a standard version and a version for the option `leqno`.

```

1378 {
1379   \__wa_if_in_last_col_of_disp:Nn \tagnextline
1380   { \bool_set_true:N \l__wa_tag_next_line_bool }
1381 }

```

The environments `{DispWithArrows}` and `{DispWithArrows*}` are compliant with the command `\qedhere` of `amsthm`. However, this compatibility requires a special version of `\qedhere`.

This special version is called `\@@_qedhere:` and will be linked with `\qedhere` in the last column of the environment `{DispWithArrows}` (only if the package `amsthm` has been loaded). `\@@_qedhere:` raises the boolean `\l_@@_qedhere_bool`.

```

1382 \cs_new_protected:Npn \__wa_qedhere: { \bool_set_true:N \l__wa_qedhere_bool }
1383 \cs_new_protected:Npn \__wa_set_qedhere: { \cs_set_eq:NN \qedhere \__wa_qedhere: }

```

In the last column of the `\halign` of `{DispWithArrows}` (column of the labels, that is to say the numbers of the equations), a command `\@@_qedhere_i:` will be issued if the flag `\l_@@_qedhere_bool` has been raised. The code of this command is an adaptation of the code of `\qedhere` in `amsthm`.

```

1384 \cs_new_protected:Npn \__wa_qedhere_i:
1385 {
1386   \group_begin:
1387   \cs_set_eq:NN \qed \qedsymbol

```

The line `\cs_set_eq:NN \qed@elt \setQED@elt` is a preparation for an action on the QED stack. Despite its form, the instruction `\QED@stack` executes an operation on the stack. This operation prints the QED symbol and nullify the top of the stack.

```

1388   \cs_set_eq:NN \qed@elt \setQED@elt
1389   \QED@stack \relax \relax
1390   \group_end:
1391 }
1392 </LaTeX>

```

11.10 We draw the arrows

The arrows are divided in groups. There is two reasons for this division.

- If the option `group` or the option `groups` is used, all the arrows of a group are drawn on a same vertical at an abscissa of `\l_@@_x_dim`.
- For aesthetic reasons, the starting point of all the starting arrows of a group is raised upwards by the value `\l_@@_start_adjust_dim`. Idem for the ending arrows.

If the option `group` is used (`\l_@@_pos_arrow_int = 7`), we scan the arrows twice: in the first step we only compute the value of `\l_@@_x_dim` for the whole group, and, in the second step (`\l_@@_pos_arrow_int` is set to 8), we divide the arrows in groups (for the vertical ajustement) and we actually draw the arrows.

```

1393 \cs_new_protected:Npn \__wa_scan_arrows:
1394 {
1395   \group_begin:
1396   \int_compare:nNnT \l__wa_pos_arrow_int = 7
1397   {
1398     \__wa_scan_arrows_i:
1399     \int_set:Nn \l__wa_pos_arrow_int 8
1400   }
1401   \__wa_scan_arrows_i:
1402   \group_end:
1403 }

1404 \cs_new_protected:Npn \__wa_scan_arrows_i:
1405 {

```


`\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_@@_first_arrows_seq` will be the list the arrows of the group starting at the first line of the group (we may have several arrows starting from the same line). We have to know all these arrows because of the adjustment by `\l_@@_start_adjust_dim`.
`\l_@@_last_line_of_group_int` will be the last line involved in the group (impossible to guess in advance).
`\l_@@_last_arrows_seq` will be the list of all the arrows of the group ending at the last line of the group (impossible to guess in advance).

```

1406   \int_zero_new:N \l__wa_first_arrow_of_group_int
1407   \int_zero_new:N \l__wa_first_line_of_group_int
1408   \int_zero_new:N \l__wa_last_line_of_group_int
1409   \seq_clear_new:N \l__wa_first_arrows_seq
1410   \seq_clear_new:N \l__wa_last_arrows_seq

```

The boolean `\l_@@_new_group_bool` is a switch that we will use to indicate that a group is finished (and the lines of that group have to be drawn). This boolean is not directly connected to the option `new-group` of an individual arrow.

```

1411   \bool_set_true:N \l__wa_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 arrows of that group.

```

1412   \int_set:Nn \l__wa_arrow_int \c_one_int
1413   \int_until_do:nNnn \l__wa_arrow_int > \g__wa_arrow_int
1414   {

```

We extract from the property list of the current arrow the fields “initial”, “final”, “status” and “input-line”. For the two former, we have to do conversions to integers.

```

1415     \prop_get:cnN
1416     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1417     { initial } \l_tmpa_tl
1418     \int_set:Nn \l__wa_initial_int \l_tmpa_tl
1419     \prop_get:cnN
1420     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1421     { final } \l_tmpa_tl
1422     \int_set:Nn \l__wa_final_int \l_tmpa_tl
1423     \prop_get:cnN
1424     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1425     { status } \l__wa_status_arrow_str
1426     \prop_get:cnN
1427     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1428     { input-line } \l__wa_input_line_str

```

We recall that, after the construction of the `\halign`, `\g_@@_line_int` is the total number of lines of the environment. Therefore, the conditionnal `\l_@@_final_int > \g_@@_line_int` tests whether an arrow arrives after the last line of the environment. In this case, we raise an error (except in the second step of treatment for the option `group`). The arrow will be completely ignored, even for the computation of `\l_@@_x_dim`.

```

1429     \int_compare:nNnTF \l__wa_final_int > \g__wa_line_int
1430     {
1431         \int_compare:nNnF \l__wa_pos_arrow_int = 8
1432         { \__wa_error:n { Too~few~lines~for~an~arrow } }
1433     }
1434     \__wa_code_for_possible_arrow:

```

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

```

1435     \int_incr:N \l__wa_arrow_int
1436 }

```

After the last arrow of the environment, we have to draw the last group of arrows. If we are in option `group` and in the first step of treatment ($\backslash l_@@_pos_arrow_int = 7$), we don't draw because, in the first step, we don't draw anything. If there is no arrow in the group, we don't draw (this situation occurs when all the arrows of the potential group arrive after the last line of the environment).

```

1437   \bool_if:nT
1438     {
1439       \int_compare_p:n { \l__wa_pos_arrow_int != 7 }
1440       &&
1441       \int_compare_p:nNn \l__wa_first_arrow_of_group_int > 0
1442     }
1443     { \__wa_draw_arrows:nn \l__wa_first_arrow_of_group_int \g__wa_arrow_int }
1444   }

1445 \cs_new_protected:Npn \__wa_code_for_possible_arrow:
1446   {

```

We test whether 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, except if we are with the option `group` and in the first step of treatment ($\backslash l_@@_pos_arrow_int = 7$).

```

1447   \bool_if:nT
1448     {
1449       \int_compare_p:nNn \l__wa_arrow_int > \c_one_int
1450       &&
1451       ( \int_compare_p:n { \l__wa_initial_int > \l__wa_last_line_of_group_int }
1452         &&
1453         \int_compare_p:n { \l__wa_pos_arrow_int != 7 }
1454         ||
1455         \str_if_eq_p:Vn \l__wa_status_arrow_str { new-group }
1456       )
1457     }
1458     {
1459       \int_compare:nNnF \l__wa_first_arrow_of_group_int = \c_zero_int
1460       {
1461         \__wa_draw_arrows:nn
1462           \l__wa_first_arrow_of_group_int
1463           { \l__wa_arrow_int - 1 }
1464       }
1465       \bool_set_true:N \l__wa_new_group_bool
1466     }

```

The flag $\backslash l_@@_new_group_bool$ indicates if we have to begin a new group of arrows. In fact, we have to begin a new group in three circumstances: if we are at the first arrow of the environment (that's why the flag is raised before the beginning of the loop), if we have just finished a group (that's why the flag is raised in the previous conditionnal, for topological reasons or if the previous arrows had the status "new-group"). At the beginning of a group, we have to initialize the following variables: $\backslash l_@@_first_arrow_int$, $\backslash l_@@_first_line_of_group_int$, $\backslash l_@@_last_line_of_group$, $\backslash l_@@_first_arrows_seq$, $\backslash l_@@_last_arrows_seq$.

```

1467   \bool_if:nTF \l__wa_new_group_bool
1468     {
1469       \bool_set_false:N \l__wa_new_group_bool
1470       \int_set_eq:NN \l__wa_first_arrow_of_group_int \l__wa_arrow_int
1471       \int_set_eq:NN \l__wa_first_line_of_group_int \l__wa_initial_int
1472       \int_set_eq:NN \l__wa_last_line_of_group_int \l__wa_final_int
1473       \seq_clear:N \l__wa_first_arrows_seq
1474       \seq_put_left:NV \l__wa_first_arrows_seq \l__wa_arrow_int
1475       \seq_clear:N \l__wa_last_arrows_seq
1476       \seq_put_left:NV \l__wa_last_arrows_seq \l__wa_arrow_int

```

If we are in option `group` and in the second step of treatment ($\backslash l_@@_pos_arrow_int = 8$), we don't initialize $\backslash l_@@_x_dim$ because we want to use the same value of $\backslash l_@@_x_dim$ (computed during the first step) for all the groups.

```

1477     \int_compare:nT { \l__wa_pos_arrow_int != 8 }
1478     { \dim_set:Nn \l__wa_x_dim { - \c_max_dim } }
1479 }

```

If we are not at the beginning of a new group.

```

1480 {

```

If the arrow is independent, we don't take into account this arrow for the detection of the end of the group.

```

1481     \bool_if:nF
1482     { \str_if_eq_p:Vn \l__wa_status_arrow_str { independent } }
1483     {

```

If the arrow is not independent, the arrow belongs to the current group and we have to take it into account in some variables.

```

1484         \int_compare:nT
1485         { \l__wa_initial_int = \l__wa_first_line_of_group_int }
1486         { \seq_put_left:NV \l__wa_first_arrows_seq \l__wa_arrow_int }
1487     \int_compare:nNnTF \l__wa_final_int > \l__wa_last_line_of_group_int
1488     {
1489         \int_set_eq:NN \l__wa_last_line_of_group_int \l__wa_final_int
1490         \seq_clear:N \l__wa_last_arrows_seq
1491         \seq_put_left:NV \l__wa_last_arrows_seq \l__wa_arrow_int
1492     }
1493     {
1494         \int_compare:nNnT \l__wa_final_int = \l__wa_last_line_of_group_int
1495         { \seq_put_left:NV \l__wa_last_arrows_seq \l__wa_arrow_int }
1496     }
1497 }
1498 }

```

If the arrow is not independent, we update the current x -value (in $\l_@@@_x_dim$) with the dedicated command $\l_@@@_update_x:nn$. If we are in option group and in the second step of treatment ($\l_@@_pos_arrow_int = 8$), we don't initialize $\l_@@_x_dim$ because we want to use the same value of $\l_@@_x_dim$ (computed during the first step) for all the groups.

```

1499     \bool_if:nF { \str_if_eq_p:Vn \l__wa_status_arrow_str { independent } }
1500     {
1501         \int_compare:nT { \l__wa_pos_arrow_int != 8 }
1502         { \__wa_update_x:nn \l__wa_initial_int \l__wa_final_int }
1503     }
1504 }

```

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

```

1505 \cs_generate_variant:Nn \keys_set:nn { n o }
1506 \cs_new_protected:Npn \__wa_keys_set:
1507 { \keys_set_known:no { WithArrows / Arrow / SecondPass } }

```

The macro $\l_@@_draw_arrows:nn$ 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). This macro is nullified by the option `no-arrows`.

```

1508 \cs_new_protected:Npn \__wa_draw_arrows:nn #1 #2
1509 {
1510     \group_begin:
1511     \int_zero_new:N \l__wa_first_arrow_int
1512     \int_set:Nn \l__wa_first_arrow_int { #1 }
1513     \int_zero_new:N \l__wa_last_arrow_int
1514     \int_set:Nn \l__wa_last_arrow_int { #2 }

```

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

```

1515   \int_set:Nn \l__wa_arrow_int \l__wa_first_arrow_int
1516   \int_until_do:nNnn \l__wa_arrow_int > \l__wa_last_arrow_int
1517   {

```

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.

```

1518       \prop_get:cnN
1519         { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1520         { initial } \l_tmpa_tl
1521       \int_set:Nn \l__wa_initial_int \l_tmpa_tl
1522       \prop_get:cnN
1523         { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1524         { final } \l_tmpa_tl
1525       \int_set:Nn \l__wa_final_int \l_tmpa_tl

```

If the arrow ends after the last line of the environment, we don’t draw the arrow (an error has already been raised in `\@@_scan_arrows:`). We recall that, after the construction of the `\halign`, `\g_@@_line_int` is the total number of lines of the environment).

```

1526         \int_compare:nT { \l__wa_final_int <= \g__wa_line_int } \__wa_draw_arrows_i:
1527         \int_incr:N \l__wa_arrow_int
1528     }
1529   \group_end:
1530 }

```

The macro `\@@_draw_arrows_i:` is only for the lisibility of the code. The first `\group_begin:` is for the options of the arrows (but we remind that the options `ll`, `rr`, `rl`, `lr`, `i` and `jump` have already been extracted and are not present in the field `options` of the property list of the arrow).

```

1531 \cs_new_protected:Npn \__wa_draw_arrows_i:
1532 {
1533   \group_begin:

```

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.

```

1534   \prop_get:cnN
1535     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1536     { options } \l_tmpa_tl
1537   \str_clear_new:N \l__wa_previous_key_str
1538   \exp_args:NNo \exp_args:No
1539   \__wa_keys_set: { \l_tmpa_tl , tikz = { xshift = \l__wa_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`:

```

1540   \bool_set_false:N \l__wa_initial_r_bool
1541   \bool_set_false:N \l__wa_final_r_bool
1542   \int_case:nn \l__wa_pos_arrow_int
1543     {
1544       0 { \bool_set_true:N \l__wa_final_r_bool }
1545       2 { \bool_set_true:N \l__wa_initial_r_bool }
1546       3
1547         {
1548           \bool_set_true:N \l__wa_initial_r_bool
1549           \bool_set_true:N \l__wa_final_r_bool
1550         }
1551     }

```

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

The option `v` can be used only in `\Arrow` in `code-after` (see below).

In case of option `i` at a local or global level (`\l_@@_pos_arrow_int = 5`), we have to compute the x -value of the arrow (which is vertical). The computed x -value is stored in `\l_@@_x_dim` (the same variable used when the option `group` or the option `groups` is used).

```

1552   \int_compare:nNnT \l__wa_pos_arrow_int = 5
1553     {
1554       \dim_set:Nn \l__wa_x_dim { - \c_max_dim }
1555       \__wa_update_x:nn \l__wa_initial_int \l__wa_final_int
1556     }

```

`\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`.

```

1557   \tl_set:Nx \l__wa_initial_tl
1558     {
1559       \int_use:N \l__wa_initial_int - \bool_if:NTF \l__wa_initial_r_bool rl
1560       .south
1561     }
1562   \tl_set:Nx \l__wa_final_tl
1563     { \int_use:N \l__wa_final_int - \bool_if:NTF \l__wa_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 `show-nodes` to visualize the nodes).

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

```

1564   \prop_get:cnN
1565     { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1566     { label }
1567     \l_tmpa_tl

```

Now, we have to know if the arrow starts at the first line of the group and/or ends at the last line of the group. That’s the reason why we have stored in `\l_@@_first_arrows_seq` the list of all the arrows starting at the first line of the group and in `\l_@@_last_arrows_seq` the list of all the arrows ending at the last line of the group. We compute these values in the booleans `\l_tmpa_bool` and `\l_tmpb_bool`. These computations can’t be done in the following `{tikzpicture}` because the command `\seq_if_in:NnTF` which is *not* expandable.

```

1568   \seq_if_in:NxTF \l__wa_first_arrows_seq
1569     { \int_use:N \l__wa_arrow_int }
1570     { \bool_set_true:N \l_tmpa_bool }
1571     { \bool_set_false:N \l_tmpa_bool }
1572   \seq_if_in:NxTF \l__wa_last_arrows_seq
1573     { \int_use:N \l__wa_arrow_int }
1574     { \bool_set_true:N \l_tmpb_bool }
1575     { \bool_set_false:N \l_tmpb_bool }
1576   \int_compare:nNnT \l__wa_pos_arrow_int = 5
1577     {
1578       \bool_set_true:N \l_tmpa_bool
1579       \bool_set_true:N \l_tmpb_bool
1580     }

```

We compute and store in `\g_tmpa_tl` and `\g_tmpb_tl` the exact coordinates of the extremities of the arrow.

- Concerning the x -values, the abscissa computed in `\l_@@_x_dim` will be used if the option of position is `i`, `group` or `groups`.

- Concerning the y -values, an adjustment is done for each arrow starting at the first line of the group and each arrow ending at the last line of the group (with the values of `\l_@@_start_adjust_dim` and `\l_@@_end_adjust_dim`).

```

1581 <*LaTeX>
1582   \begin { tikzpicture } [ __wa_standard ]
1583 </LaTeX>
1584 <*plain-TeX>
1585   \tikzpicture [ __wa_standard ]
1586 </plain-TeX>
1587   \tikz@scan@one@point \pgfutil@firstofone ( \l__wa_initial_tl )
1588   \tl_gset:Nx \g_tmpa_tl
1589   {
1590     \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1591     { \dim_use:N \pgf@x }
1592     { \dim_use:N \l__wa_x_dim } ,
1593     \bool_if:NTF \l_tmpa_bool
1594     { \dim_eval:n { \pgf@y + \l__wa_start_adjust_dim } }
1595     { \dim_use:N \pgf@y }
1596   }
1597   \tikz@scan@one@point \pgfutil@firstofone ( \l__wa_final_tl )
1598   \tl_gset:Nx \g_tmpb_tl
1599   {
1600     \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1601     { \dim_use:N \pgf@x }
1602     { \dim_use:N \l__wa_x_dim } ,
1603     \bool_if:NTF \l_tmpb_bool
1604     { \dim_eval:n { \pgf@y - \l__wa_end_adjust_dim } }
1605     { \dim_use:N \pgf@y }
1606   }
1607 <*LaTeX>
1608   \end { tikzpicture }
1609 </LaTeX>
1610 <*plain-TeX>
1611   \endtikzpicture
1612 </plain-TeX>

```

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 `tikz-code`. We use the variant `\@@_draw_arrow:nno` of the macro `\@@_draw_arrow:nnn` because of the characters *underscore* in the name `\l_tmpa_tl`: if the user uses the Tikz library `babel`, the third argument of the command `\@@_draw_arrow:nno` will be rescanned because this third argument will be in the argument of a command `node` of an instruction `\draw` of Tikz... and we will have an error because of the characters *underscore*.³⁷

```

1613   \__wa_draw_arrow:nno \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).

```

1614   \group_end:
1615 }

```

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`.

```

1616 \cs_new_protected:Npn \__wa_def_function_tmpa:n #1
1617 {
1618   \cs_set:Npn \__wa_tmpa:nnn ##1 ##2 ##3
1619   {

```

³⁷There were other solutions: use another name without *underscore* (like `\ltmpat1`) or use the package `underscore` (with this package, the characters *underscore* will be rescanned without errors, even in text mode).

```

1620 <*LaTeX>
1621     \begin{tikzpicture}
1622 </LaTeX>
1623 <*plain-TeX>
1624     \tikzpicture
1625 </plain-TeX>
1626     [
1627         __wa_standard ,
1628         every~path / .style = WithArrows / arrow
1629     ]
1630     #1
1631 <*LaTeX>
1632     \end{tikzpicture}
1633 </LaTeX>
1634 <*plain-TeX>
1635     \endtikzpicture
1636 </plain-TeX>
1637     }
1638 }

```

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

```

1639 \cs_new_protected:Npn \__wa_draw_arrow:nnn #1 #2 #3
1640 {

```

If the option `wrap-lines` is used, we have to use a special version of `\l_@@_tikz_code_tl` (which corresponds to the option `tikz-code`).

```

1641     \bool_if:nT { \l__wa_wrap_lines_bool && \l__wa_in_DispWithArrows_bool }
1642     { \tl_set_eq:NN \l__wa_tikz_code_tl \c__wa_tikz_code_wrap_lines_tl }

```

Now, the main lines of this function `\@@_draw_arrow:nnn`.

```

1643     \exp_args:NV \__wa_def_function_tmpa:n \l__wa_tikz_code_tl
1644     \__wa_tmpa:nnn { #1 } { #2 } { #3 }
1645 }
1646 \cs_generate_variant:Nn \__wa_draw_arrow:nnn { n n o }

```

If the option `wrap-lines` is used, we have to use a special version of `\l_@@_tikz_code_tl` (which corresponds to the option `tikz-code`).

```

1647 \tl_const:Nn \c__wa_tikz_code_wrap_lines_tl
1648 {

```

First, we draw the arrow without the label.

```

1649     \draw ( #1 ) to node ( __wa_label ) { } ( #2 ) ;

```

We retrieve in `\pgf@x` the abscissa of the left-side of the label we will put.

```

1650     \tikz@parse@node \pgfutil@firstofone ( __wa_label.west )

```

We compute in `\l_tmpa_dim` the maximal width possible for the label. Here is the use of `\g_@@_right_x_dim` which has been computed previously with the `v-nodes`.

```

1651     \dim_set:Nn \l_tmpa_dim
1652     { \g__wa_right_x_dim - \pgf@x - \pgfkeysvalueof { / pgf / inner~xsep } }

```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “text width”.³⁸

```

1653     \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;

```

Maybe the current value of the parameter “text width” is shorter than `\l_tmpa_dim`. In this case, we must use “text width” (we update `\l_tmpa_dim`).

```

1654     \tl_if_empty:NF \g_tmpa_tl
1655     {
1656         \dim_set:Nn \l_tmpb_dim \g_tmpa_tl

```

³⁸In fact, it’s not the current value of “text width”: it’s the value of “text width” set in the option `tikz` provided by `witharrows`. These options are given to Tikz in a “every path”. That’s why we have to retrieve it in a path.

```

1657     \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1658     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1659   }

```

Now, we can put the label with the right value for “text width”.

```

1660   \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1661   {
1662     \path ( __wa_label.west )
1663     node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1664     { #3 } ;
1665   }
1666 }

```

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

```

1667 \cs_new_protected:Npn \__wa_update_x:nn #1 #2
1668 {
1669   \int_step_inline:nnn { #1 } { #2 }
1670   {
1671     <*LaTeX>
1672     \begin { tikzpicture } [ __wa_standard ]
1673     </LaTeX>
1674     <*plain-TeX>
1675     \tikzpicture [ __wa_standard ]
1676     </plain-TeX>
1677     \tikz@scan@one@point \pgfutil@firstofone ( ##1 - 1 )
1678     \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \l__wa_x_dim \pgf@x }
1679     <*LaTeX>
1680     \end { tikzpicture }
1681     </LaTeX>
1682     <*plain-TeX>
1683     \endtikzpicture
1684     </plain-TeX>
1685     \dim_set_eq:NN \l__wa_x_dim \g_tmpa_dim
1686   }
1687 }

```

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 sense of the nested environments). This macro is fully expandable and, thus, can be used directly in the name of a Tikz node.

```

1688 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g__wa_last_env_int }

```

11.11 The command `\Arrow` in code-after

The option `code-after` is an option of the environment `{WithArrows}` (this option is only available at the environment level). In the option `code-after`, 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/Arrow/code-after`.

```

1689 \keys_define:nn { WithArrows / Arrow / code-after }
1690 {
1691   tikz      .code:n =
1692   \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
1693   tikz      .value_required:n = true ,
1694   rr        .value_forbidden:n = true ,
1695   rr        .code:n      = \__wa_fix_pos_option:n 0 ,
1696   ll        .value_forbidden:n = true,
1697   ll        .code:n      = \__wa_fix_pos_option:n 1 ,

```



```

1698   rl      .value_forbidden:n = true ,
1699   rl      .code:n             = \__wa_fix_pos_option:n 2 ,
1700   lr      .value_forbidden:n = true ,
1701   lr      .code:n             = \__wa_fix_pos_option:n 3 ,
1702   v       .value_forbidden:n = true ,
1703   v       .code:n             = \__wa_fix_pos_option:n 4 ,
1704   tikz-code .tl_set:N          = \l__wa_tikz_code_tl ,
1705   tikz-code .value_required:n = true ,
1706   xoffset  .dim_set:N          = \l__wa_xoffset_dim ,
1707   xoffset  .value_required:n = true ,
1708   unknown  .code:n            =
1709     \__wa_sort_seq:N \l__wa_options_Arrow_code_after_seq
1710     \__wa_error:n { Unknown-option~Arrow-in~code-after }
1711 }

```

A sequence of the options available in `\Arrow` in `code-after`. This sequence will be used in the error messages and can be modified dynamically.

```

1712 \seq_new:N \l__wa_options_Arrow_code_after_seq
1713 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_Arrow_code_after_seq
1714 { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }

```

```

1715 (*LaTeX)
1716 \NewDocumentCommand \__wa_Arrow_code_after { 0 { } m m m ! 0 { } }
1717 
1718 (*plain-TeX)
1719 \cs_new_protected:Npn \__wa_Arrow_code_after
1720 {
1721   \peek_meaning:NTF [
1722     { \__wa_Arrow_code_after_i }
1723     { \__wa_Arrow_code_after_i [ ] }
1724   ]
1725   \cs_new_protected:Npn \__wa_Arrow_code_after_i [ #1 ] #2 #3 #4
1726   {
1727     \peek_meaning:NTF [
1728       { \__wa_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1729       { \__wa_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
1730     ]
1731     \cs_new_protected:Npn \__wa_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
1732     
1733     {
1734       \int_set:Nn \l__wa_pos_arrow_int 1
1735       \str_clear_new:N \l__wa_previous_key_str
1736       \group_begin:
1737         \keys_set:nn { WithArrows / Arrow / code-after }
1738           { #1, #5, tikz = { xshift = \l__wa_xoffset_dim } }
1739         \bool_set_false:N \l__wa_initial_r_bool
1740         \bool_set_false:N \l__wa_final_r_bool
1741         \int_case:nn \l__wa_pos_arrow_int
1742           {
1743             0
1744             {
1745               \bool_set_true:N \l__wa_initial_r_bool
1746               \bool_set_true:N \l__wa_final_r_bool
1747             }
1748             2 { \bool_set_true:N \l__wa_initial_r_bool }
1749             3 { \bool_set_true:N \l__wa_final_r_bool }
1750           }

```

We prevent drawing a arrow from a line to itself.

```

1751   \tl_if_eq:nnTF { #2 } { #3 }
1752   { \__wa_error:nn { Both~lines~are~equal } { #2 } }

```

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

```

1753     {
1754         \cs_if_free:cTF { pgf@sh@ns@wa - \l__wa_prefix_str - #2 - 1 }
1755         { \__wa_error:nx { Wrong~line~in~Arrow } { #2 } }
1756         {
1757             \cs_if_free:cTF { pgf@sh@ns@wa - \l__wa_prefix_str - #3 - 1 }
1758             { \__wa_error:nx { Wrong~line~in~Arrow } { #3 } }
1759             {
1760                 \int_compare:nNnTF \l__wa_pos_arrow_int = 4
1761                 {
1762                     (*LaTeX)
1763                     \begin { tikzpicture } [ __wa_standard ]
1764                     (/LaTeX)
1765                     (*plain-TeX)
1766                     \tikzpicture [ __wa_standard ]
1767                     (/plain-TeX)
1768                     \tikz@scan@one@point \pgfutil@firstofone (#2-1.south)
1769                     \dim_set_eq:NN \l_tmpa_dim \pgf@x
1770                     \dim_set_eq:NN \l_tmpb_dim \pgf@y
1771                     \tikz@scan@one@point \pgfutil@firstofone (#3-1.north)
1772                     \dim_set:Nn \l_tmpa_dim
1773                     { \dim_max:nn \l_tmpa_dim \pgf@x }
1774                     \tl_gset:Nx \g_tmpa_tl
1775                     { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1776                     \tl_gset:Nx \g_tmpb_tl
1777                     { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1778                     (*LaTeX)
1779                     \end { tikzpicture }
1780                     (/LaTeX)
1781                     (*plain-TeX)
1782                     \endtikzpicture
1783                     (/plain-TeX)
1784                 }
1785                 {
1786                     (*LaTeX)
1787                     \begin { tikzpicture } [ __wa_standard ]
1788                     (/LaTeX)
1789                     (*plain-TeX)
1790                     \tikzpicture [ __wa_standard ]
1791                     (/plain-TeX)
1792                     \tikz@scan@one@point \pgfutil@firstofone
1793                     ( #2-\bool_if:NTF\l__wa_initial_r_bool rl .south )
1794                     \tl_gset:Nx \g_tmpa_tl
1795                     { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1796                     \tikz@scan@one@point \pgfutil@firstofone
1797                     ( #3-\bool_if:NTF\l__wa_final_r_bool rl .north )
1798                     \tl_gset:Nx \g_tmpb_tl
1799                     { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1800                     (*LaTeX)
1801                     \end { tikzpicture }
1802                     (/LaTeX)
1803                     (*plain-TeX)
1804                     \endtikzpicture
1805                     (/plain-TeX)
1806                 }
1807             }
1808         }
1809     }
1810 }
1811 \group_end:
1812 }

```

11.12 The command `\MultiArrow` in `code-after`

The command `\@@_MultiArrow:n` will be linked to `\MultiArrow` when the `code-after` is executed.

```
1813 \cs_new_protected:Npn \__wa_MultiArrow:n #1 #2
1814 {
```

The user of the command `\MultiArrow` (in `code-after`) 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`.

```
1815   \foreach \x in { #1 }
1816   {
1817     \cs_if_free:ctF { pgf@sh@ns@wa - \l__wa_prefix_str - \x - l }
1818     { \__wa_error:nx { Wrong~line~specification~in~MultiArrow } \x }
1819     { \clist_gput_right:Nx \g_tmpa_clist \x }
1820   }
```

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

```
1821   \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
1822   { \__wa_error:n { Too~small~specification~for~MultiArrow } }
1823   {
1824     \clist_sort:Nn \g_tmpa_clist
1825     {
1826       \int_compare:nTF { ##1 > ##2 }
1827       \sort_return_swapped:
1828       \sort_return_same:
1829     }
```

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

```
1830     \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...

```
1831     \clist_reverse:N \g_tmpa_clist
1832     \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).

```
1833     \exp_args:NV \__wa_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
1834 <*LaTeX>
1835   \begin { tikzpicture }
1836 </LaTeX>
1837 <*plain-TeX>
1838   \tikzpicture
1839 </plain-TeX>
1840   [
1841     __wa_standard ,
1842     every~path / .style = { WithArrows / arrow }
1843   ]
1844   \draw [<->] ([xshift = \l__wa_xoffset_dim]\l_tmpa_tl-r.south)
1845     -- ++(5mm,0)
1846     -- node (__wa_label) {}
1847     ([xshift = \l__wa_xoffset_dim+5mm]\l_tmpb_tl-r.south)
1848     -- ([xshift = \l__wa_xoffset_dim]\l_tmpb_tl-r.south) ;
1849   \tikz@parse@node \pgfutil@firstofone (__wa_label.west)
1850   \dim_set:Nn \l_tmpa_dim { 20 cm }
1851   \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
1852   \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
1853   \bool_if:nT { \l__wa_wrap_lines_bool && \l__wa_in_DispWithArrows_bool }
1854   {
```

```

1855         \dim_set:Nn \l_tmpb_dim
1856         { \g__wa_right_x_dim - \pgf@x - 0.3333 em }
1857         \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1858         { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1859     }
1860     \path (__wa_label.west)
1861     node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
1862 <*LaTeX>
1863     \end{tikzpicture}
1864 </LaTeX>
1865 <*plain-TeX>
1866     \endtikzpicture
1867 </plain-TeX>
1868 }
1869 }

1870 \cs_new_protected:Npn \__wa_MultiArrow_i:n #1
1871 {
1872 <*LaTeX>
1873     \begin { tikzpicture }
1874 </LaTeX>
1875 <*plain-TeX>
1876     \tikzpicture
1877 </plain-TeX>
1878     [
1879     __wa_standard ,
1880     every-path / .style = { WithArrows / arrow }
1881     ]
1882     \foreach \k in { #1 }
1883     {
1884         \draw [ <- ]
1885         ( [xshift = \l__wa_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
1886     } ;
1887 <*LaTeX>
1888     \end{tikzpicture}
1889 </LaTeX>
1890 <*plain-TeX>
1891     \endtikzpicture
1892 </plain-TeX>
1893 }

```

11.13 The error messages of the package

```

1894 \str_const:Nn \c__wa_option_ignored_str
1895 { If-you-go-on,-this-option-will-be-ignored. }
1896 \str_const:Nn \c__wa_command_ignored_str
1897 { If-you-go-on,-this-command-will-be-ignored. }
1898 <*LaTeX>
1899 \__wa_msg_new:nn { amsmath-not-loaded }
1900 {
1901     You-can't-use-the-option-' \l_keys_key_tl '~because-the-
1902     package-' amsmath '~has-not-been-loaded.\
1903     If-you-go-on,-this-option-will-be-ignored-in-the-rest-
1904     of-the-document.
1905 }
1906 </LaTeX>
1907 \__wa_msg_new:nn { Bad-value-for-replace-brace-by }
1908 {
1909     Bad-value-for-the-option-' \l_keys_key_tl '~The-value-must-begin-
1910     with-an-extensible-left-delimiter.-The-possible-values-are:~,
1911     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
1912     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~

```

```

1913 \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
1914 \token_to_str:N \lfloor\ and~\token_to_str:N \lceil\
1915 (and~\token_to_str:N \lvert\ and~\token_to_str:N \lVert\
1916 if~amsmath-or-unicode-math-is-loaded-in-LaTeX).\
1917 \c_wa_option_ignored_str
1918 }
1919 \__wa_msg_new:nn { option-of-cr-negative }
1920 {
1921   The~argument~of~the~command~\token_to_str:N\~
1922   should~be~positive~in~the~row~\int_use:N \g_wa_line_int\
1923   of~your~environment~\{\l__wa_type_env_str\}.\
1924   \c_wa_option_ignored_str
1925 }
1926 \__wa_msg_new:nn { omit-probably-used }
1927 {
1928   There~is~a~problem.~Maybe~you~have~used~a~command~
1929   \token_to_str:N\omit\ in~the~line~\int_use:N \g_wa_line_int\
1930   (or~another~line)~of~your~environment~\{\l__wa_type_env_str\}.\
1931   You~can~go~on~but~you~may~have~others~errors.
1932 }
1933 (*LaTeX)
1934 \__wa_msg_new:nn { newline-at-the-end-of-env }
1935 {
1936   Your~environment~\{\l__wa_type_env_str\}~should~not~end~with~
1937   a~\token_to_str:N \\.
1938   This~warning~might~become~an~error~in~a~future~version.
1939 }
1940 (/LaTeX)
1941 \__wa_msg_new:nn { Invalid-option-format }
1942 {
1943   The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
1944   must~not~be~empty.\
1945   \c_wa_option_ignored_str
1946 }
1947 \__wa_msg_new:nn { Value-for-a-key }
1948 {
1949   The~key~'\l_keys_key_tl'~should~be~used~without~value. \
1950   However,~you~can~go~on~for~this~time.
1951 }
1952 \__wa_msg_new:nnn { Unknown-option-in-Arrow }
1953 {
1954   The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
1955   \l_wa_string_Arrow_for_msg_str\ in~the~row~
1956   \int_use:N \g_wa_line_int\ of~your~environment~
1957   \{\l__wa_type_env_str\}. \l_tmpa_str \
1958   \c_wa_option_ignored_str \
1959   For~a~list~of~the~available~keys,~type~H~<return>.
1960 }
1961 {
1962   The~available~keys~are~(in~alphabetic~order):~
1963   \seq_use:Nnnn \l__wa_options_Arrow_seq {-and-} {,} {-and-}.
1964 }
1965 \__wa_msg_new:nnn { Unknown-option-WithArrows }
1966 {
1967   The~key~'\l_keys_key_tl'~is~unknown~in~\{\l__wa_type_env_str\}. \
1968   \c_wa_option_ignored_str \
1969   For~a~list~of~the~available~keys,~type~H~<return>.
1970 }
1971 {
1972   The~available~keys~are~(in~alphabetic~order):~
1973   \seq_use:Nnnn \l__wa_options-WithArrows_seq {-and-} {,} {-and-}.

```

```

1974 }
1975 \_wa_msg_new:nnn { Unknown~option~DispWithArrows }
1976 {
1977   The~key~'\l_keys_key_tl'~is~unknown~in~\{\l_wa_type_env_str\}. \\  

1978   \c_wa_option_ignored_str \\  

1979   For~a~list~of~the~available~keys,~type-H<return>.
1980 }
1981 {
1982   The~available~keys~are~(in~alphabetic~order):~
1983   \seq_use:Nnnn \l_wa_options_DispWithArrows_seq {~and~} {,~} {~and~}.
1984 }
1985 \_wa_msg_new:nnn { Unknown~option~WithArrowsOptions }
1986 {
1987   The~key~'\l_keys_key_tl'~is~unknown~in~
1988   \token_to_str:N \WithArrowsOptions. \\  

1989   \c_wa_option_ignored_str \\  

1990   For~a~list~of~the~available~keys,~type-H<return>.
1991 }
1992 {
1993   The~available~keys~are~(in~alphabetic~order):~
1994   \seq_use:Nnnn \l_wa_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
1995 }
1996 \_wa_msg_new:nnn { Unknown~option~Arrow~in~code~after }
1997 {
1998   The~key~'\l_keys_key_tl'~is~unknown~in~
1999   \token_to_str:N \Arrow\ in~code~after. \\  

2000   \c_wa_option_ignored_str \\  

2001   For~a~list~of~the~available~keys,~type-H<return>.
2002 }
2003 {
2004   The~available~keys~are~(in~alphabetic~order):~
2005   \seq_use:Nnnn \l_wa_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2006 }
2007 \_wa_msg_new:nn { Too~much~columns~in~WithArrows }
2008 {
2009   Your~environment~\{\l_wa_type_env_str\}~has~\int_use:N  

2010   \l_wa_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2011   Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2012   If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2013   the~option~'more~columns'~at~a~global~level~or~for~an~environment. \\  

2014   However,~you~can~go~one~for~this~time.
2015 }
2016 \_wa_msg_new:nn { Too~much~columns~in~DispWithArrows }
2017 {
2018   Your~environment~\{\l_wa_type_env_str\}~has~\int_use:N  

2019   \l_wa_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2020   Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\  

2021   at~the~end~of~row~\int_use:N \g_wa_line_int. \\  

2022   This~error~is~fatal.
2023 }
2024 \_wa_msg_new:nn { Negative~jump }
2025 {
2026   You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
2027   \l_wa_string_Arrow_for_msg_str\  

2028   in~the~row~\int_use:N \g_wa_line_int\  

2029   of~your~environment~\{\l_wa_type_env_str\}.~
2030   You~can~create~an~arrow~going~backwards~with~the~option~'<'~of~Tikz. \\  

2031   \c_wa_option_ignored_str
2032 }
2033 \_wa_msg_new:nn { new~group~without~groups }
2034 {

```

```

2035 You~can't~use~the~option~'new-group'~for~the~command~
2036 \l_wa_string_Arrow_for_msg_str\
2037 because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
2038 'groups'~in~your~environment~\{\l_wa_type_env_str\}. \\\
2039 \c_wa_option_ignored_str
2040 }
2041 \__wa_msg_new:nn
2042 { Too~few~lines~for~an~arrow }
2043 {
2044 Line~\l_wa_input_line_str\
2045 :~an~arrow~specified~in~the~row~\int_use:N \l_wa_initial_int\
2046 of~your~environment~\{\l_wa_type_env_str\}~can't~be~drawn~
2047 because~it~arrives~after~the~last~row~of~the~environment. \\\
2048 If~you~go~on,~this~arrow~will~be~ignored.
2049 }
2050 \__wa_msg_new:nn { WithArrows~outside~math~mode }
2051 {
2052 The~environment~\{\l_wa_type_env_str\}~should~be~used~only~in~math~mode~
2053 like~the~environment~\{aligned\}~of~amsmath. \\\
2054 Nevertheless,~you~can~go~on.
2055 }
2056 \__wa_msg_new:nn { DispWithArrows~in~math~mode }
2057 {
2058 The~environment~\{\l_wa_type_env_str\}~should~be~used~only~outside~math~
2059 mode~like~the~environment~\{align\}~of~amsmath. \\\
2060 This~error~is~fatal.
2061 }
2062 \__wa_msg_new:nn { Incompatible~options~in~Arrow }
2063 {
2064 You~try~to~use~the~option~'\l_keys_key_tl'~but~
2065 this~option~is~incompatible~or~redundant~with~the~option~
2066 '\l_wa_previous_key_str'~set~in~the~same~command~
2067 \l_wa_string_Arrow_for_msg_str. \\\
2068 \c_wa_option_ignored_str
2069 }
2070 \__wa_msg_new:nn { Incompatible~options }
2071 { You~try~to~use~the~option~'\l_keys_key_tl'~but~
2072 this~option~is~incompatible~or~redundant~with~the~option~
2073 '\l_wa_previous_key_str'~set~in~the~same~command~
2074 \bool_if:NT \l_wa_in_code_after_bool
2075 {
2076 \l_wa_string_Arrow_for_msg_str\
2077 in~the~code~after~of~your~environment~\{\l_wa_type_env_str\}
2078 }. \\\
2079 \c_wa_option_ignored_str
2080 }
2081 \__wa_msg_new:nn { Arrow~not~in~last~column }
2082 {
2083 You~should~use~the~command~\l_wa_string_Arrow_for_msg_str\
2084 only~in~the~last~column~(column~\int_use:N\l_wa_nb_cols_int)~
2085 of~your~environment~\{\l_wa_type_env_str\}. \\\
2086 However~you~can~go~on~for~this~time.
2087 }
2088 \__wa_msg_new:nn { Wrong~line~in~Arrow }
2089 {
2090 The~specification~of~line~'#1'~you~use~in~the~command~
2091 \l_wa_string_Arrow_for_msg_str\
2092 in~the~'code~after'~of~\{\l_wa_type_env_str\}~doesn't~exist. \\\
2093 \c_wa_option_ignored_str
2094 }

```

```

2095 \_wa_msg_new:nn { Both-lines-are-equal }
2096 {
2097   In-the-'code-after'-of~\{\l_wa_type_env_str\}~you~try~to~
2098   draw-an-arrow-going-to-itself-from-the-line-#1'~This-is-not-possible. \\\
2099   \c_wa_option_ignored_str
2100 }
2101 \_wa_msg_new:nn { Wrong-line-specification-in-MultiArrow }
2102 {
2103   The-specification-of-line-#1'~doesn't-exist. \\\
2104   If-you-go-on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
2105 }
2106 \_wa_msg_new:nn { Too-small-specification-for-MultiArrow }
2107 {
2108   The-specification-of-lines-you-gave-to~\token_to_str:N \MultiArrow\
2109   is-too-small:~you~need~at~least~two~lines. \\\
2110   \c_wa_command_ignored_str
2111 }
2112 \_wa_msg_new:nn { Not-allowed-in-DispWithArrows }
2113 {
2114   The-command~\token_to_str:N #1
2115   is-allowed-only-in-the-last-column~
2116   (column~\int_use:N\l_wa_nb_cols_int)-of~\{\l_wa_type_env_str\}. \\\
2117   \c_wa_option_ignored_str
2118 }
2119 \_wa_msg_new:nn { Not-allowed-in-WithArrows }
2120 {
2121   The-command~\token_to_str:N #1 is-not-allowed-in~\{\l_wa_type_env_str\}~
2122   (it's-allowed-in-the-last-column-of~\{DispWithArrows\}). \\\
2123   \c_wa_option_ignored_str
2124 }
2125 (*LaTeX)
2126 \_wa_msg_new:nn { tag*-without-amsmath }
2127 {
2128   We-can't-use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2129   (or~mathtools). \\\
2130   If-you-go-on,~the-command~\token_to_str:N\tag\
2131   will-be-used~instead.
2132 }
2133 \_wa_msg_new:nn { Multiple-tags }
2134 {
2135   You-can't-use~twice~the-command~\token_to_str:N\tag\
2136   in-a-line-of~the-environment~\{\l_wa_type_env_str\}. \\\
2137   If-you-go-on,~the-tag-#1'~will-be-used.
2138 }
2139 \_wa_msg_new:nn { Multiple-labels }
2140 {
2141   Normally,~we-can't-use~the-command~\token_to_str:N\label\
2142   twice-in-a-line-of~the-environment~\{\l_wa_type_env_str\}. \\\
2143   However,~you-can-go-on.~
2144   \bool_if:NT \c_wa_showlabels_loaded_bool
2145   { However,~only-the-last-label-will-be-shown-by-showlabels.~ }
2146   If-you-don't-want-to-see-this-message-again,~you-can-use-the-option~
2147   'allow-multiple-labels'~at~the-global-or-environment-level.
2148 }
2149 \_wa_msg_new:nn { Multiple-labels-with-cleveref }
2150 {
2151   Since-you-use-cleveref,~you-can't-use~the-command~\token_to_str:N\label\
2152   twice-in-a-line-of~the-environment~\{\l_wa_type_env_str\}. \\\
2153   If-you-go-on,~you-may~have~undefined~references.
2154 }

```



```

2155 </LaTeX>
2156 \__wa_msg_new:nn { Inexistent~v-node }
2157 {
2158   There-is-a-problem.-Maybe-you-have-put-a-command-\token_to_str:N\cr\
2159   instead-of-a-command-\token_to_str:N\\-at-the-end-of~
2160   the~row~\l_tmpa_int\
2161   of-your-environment-\{\l__wa_type_env_str\}. \\\
2162   This-error-is-fatal.
2163 }
2164 \__wa_msg_new:nn { Option~xoffset~forbidden }
2165 {
2166   You-can't-use-the-option-'xoffset'-in-the-command~
2167   \l__wa_string_Arrow_for_msg_str\
2168   while-you-are-using-the-option~
2169   ' \int_compare:nNnTF \l__wa_pos_arrow_int = 7
2170     { group }
2171     { groups } '. \\\
2172   \c__wa_option_ignored_str
2173 }
2174 \__wa_msg_new:nnn { Duplicate~name }
2175 {
2176   The~name~'\l_keys_value_tl'~is-already-used-and-you-shouldn't-use~
2177   the~same~environment~name~twice.~You-can-go-on,~but,~
2178   maybe,~you-will-have-incorrect~results. \\\
2179   For~a~list~of~the~names~already~used,~type-H~<return>. \\\
2180   If-you-don't-want-to-see-this-message-again,~use-the-option~
2181   'allow-duplicate-names'.
2182 }
2183 {
2184   The~names~already~defined~in~this~document~are:~
2185   \seq_use:Nnnn \g__wa_names_seq { ,~ } { ,~ } { ~and~ }.
2186 }

```

11.14 The command `\WithArrowsNewStyle`

A new key defined with `\WithArrowsNewStyle` will not be available at the local level.

```

2187 <*LaTeX>
2188 \NewDocumentCommand \WithArrowsNewStyle { m m }
2189 </LaTeX>
2190 <*plain-TeX>
2191 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2192 </plain-TeX>
2193 {
2194   \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2195   { \__wa_error:nn { Key-already-defined } { #1 } }
2196   {
2197     \keys_define:nn { WithArrows / Global }
2198     {
2199       #1 .code:n =
2200       { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
2201     }
2202     \seq_put_right:Nx \l__wa_options_WithArrows_seq { \tl_to_str:n { #1 } }
2203     \seq_put_right:Nx \l__wa_options_DispatchWithArrows_seq
2204     { \tl_to_str:n { #1 } }
2205     \seq_put_right:Nx \l__wa_options_WithArrowsOptions_seq
2206     { \tl_to_str:N { #1 } }

```

We now set the options in a TeX group in order to detect if some keys in #2 are unknown. If a key is unknown, an error will be raised. However, the key will, even so, be stored in the definition of key #1.

```

2207   \group_begin:
2208     \msg_set:nnn { witharrows } { Unknown-option~WithArrowsOptions }
2209     {
2210       The-key~'\l_keys_key_tl'~can't-be-set-in-the~

```

```

2211         definition-of-a-style.~You-can-go-on-for~this~time~
2212         but~you~should~suppress~this~key.
2213     }
2214     \WithArrowsOptions { #2 }
2215     \group_end:
2216 }
2217 }
2218 \__wa_msg_new:nn { Key~already~defined }
2219 {
2220     The~key~'#1'~is~already~defined. \
2221     If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2222     will~be~ignored.
2223 }

```

11.15 The options up and down

The options `up` and `down` are available for individual arrows. The corresponding code is given here. It is independent of the main code of the extension `witharrows`.

This code is the only part of the code of `witharrows` which uses the package `varwidth` and also the Tikz library `calc`. That's why we have decided not to load this package and this library. If they are not loaded, the user will have an error only when using the option `up` or the option `down`.

The token list `\c_@@_tikz_code_up_tl` is the value of `tikz-code` which will be used for an option `up`.

```

2224 \tl_const:Nn \c__wa_tikz_code_up_tl
2225 {
2226     \draw [ rounded-corners ]
2227         let \p1 = (#1) ,
2228             \p2 = (#2)
2229         in (\p1) -- node {
2230 <LaTeX>
2231             \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2232             \begin { varwidth } \l_tmpa_dim

```

`\narrowragged` is a command of the package `varwidth`.

```

2233             \narrowragged
2234             #3
2235         \end { varwidth }
2236 </LaTeX>
2237 <*plain-TeX>
2238             #3
2239 </plain-TeX>
2240         }
2241         (\x2,\y1) -- (\p2) ;
2242     }

```

Idem for the option `down`.

```

2243 \tl_const:Nn \c__wa_tikz_code_down_tl
2244 {
2245     \draw [ rounded-corners ]
2246         let \p1 = (#1) ,
2247             \p2 = (#2)
2248         in (\p1) -- (\x1,\y2) --
2249         node {
2250 <LaTeX>
2251             \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2252             \begin { varwidth } \l_tmpa_dim
2253                 \narrowragged
2254                 #3
2255             \end { varwidth }
2256 </LaTeX>
2257 <*plain-TeX>
2258                 #3

```

```

2259 </plain-TeX>
2260         }
2261         (\p2) ;
2262     }
2263 \keys_define:nn { WithArrows / Arrow / FirstPass }
2264 {
2265     up .code:n = \__wa_set_independent: ,
2266     down .code:n = \__wa_set_independent: ,
2267     up .default:n = NoValue ,
2268     down .default:n = NoValue
2269 }
2270 \keys_define:nn { WithArrows / Arrow / SecondPass }
2271 {
2272     up .code:n =
2273         \str_if_empty:NT \l__wa_previous_key_str
2274         {
2275             \str_set:Nn \l__wa_previous_key_str { up }
2276 (*LaTeX)
2277             \bool_if:NTF \c__wa_varwidth_loaded_bool
2278             {
2279 </LaTeX>
2280                 \cs_if_exist:cTF { tikz@library@calc@loaded }
2281                 {
2282                     \int_set:Nn \l__wa_pos_arrow_int \c_one_int

```

We have to set `\l__wa_wrap_lines_bool` to `false` because, otherwise, if the option `wrap_lines` is used at a higher level (global or environment), we will have a special affectation to `tikz-code` that will overwrite our affectation.

```

2283             \bool_set_false:N \l__wa_wrap_lines_bool
2284             \tl_set_eq:NN \l__wa_tikz_code_tl
2285             \c__wa_tikz_code_up_tl
2286         }
2287         { \__wa_error:n { calc-not-loaded } }
2288 (*LaTeX)
2289     }
2290     { \__wa_error:n { varwidth-not-loaded } }
2291 </LaTeX>
2292     } ,
2293     down .code:n =
2294         \str_if_empty:NT \l__wa_previous_key_str
2295         {
2296             \str_set:Nn \l__wa_previous_key_str { down }
2297 (*LaTeX)
2298             \bool_if:NTF \c__wa_varwidth_loaded_bool
2299             {
2300 </LaTeX>
2301                 \cs_if_exist:cTF { tikz@library@calc@loaded }
2302                 {
2303                     \int_set:Nn \l__wa_pos_arrow_int \c_one_int
2304                     \bool_set_false:N \l__wa_wrap_lines_bool
2305                     \tl_set_eq:NN \l__wa_tikz_code_tl
2306                     \c__wa_tikz_code_down_tl
2307                 }
2308                 { \__wa_error:n { calc-not-loaded } }
2309 (*LaTeX)
2310             }
2311             { \__wa_error:n { varwidth-not-loaded } }
2312 </LaTeX>
2313         }
2314     }

```

```

2315 \seq_put_right:Nn \l__wa_options_Arrow_seq { down }
2316 \seq_put_right:Nn \l__wa_options_Arrow_seq { up }
2317 \__wa_msg_new:nn { varwidth-not-loaded }
2318 {
2319   You-can't-use-the-option-\l_keys_key_tl'-because-
2320   you-don't-have-loaded-the-package-'varwidth'. \\\
2321   \c__wa_option_ignored_str
2322 }
2323 \__wa_msg_new:nn { calc-not-loaded }
2324 {
2325   You-can't-use-the-option-\l_keys_key_tl'-because-you-don't-have-loaded-the-
2326   Tikz-library-'calc'.You-should-add-\token_to_str:N\usetikzlibrary{calc}'-
2327   ~in-the-preamble-of-your-document. \\\
2328   \c__wa_option_ignored_str
2329 }
2330 <*plain-TeX>
2331 \catcode \@ = 12
2332 \ExplSyntaxOff
2333 </plain-TeX>

```

12 History

Changes between versions 1.0 and 1.1

Option for the command `\` and option `interline`
 Compatibility with `\usetikzlibrary{babel}`
 Possibility of nested environments `{WithArrows}`

Changes between versions 1.1 and 1.2

The package `witharrows` can now be loaded without having loaded previously `tikz` and the libraries `arrow.meta` and `bending` (this extension and these libraries are loaded silently by `witharrows`).
 New option groups (with a *s*)

Changes between versions 1.2 and 1.3

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

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`.

Changes between versions 1.4 and 1.5

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

Changes between versions 1.5 and 1.6

The code has been improved to be faster and the Tikz library `calc` is no longer required.
 A new option `name` is available for the environments `{WithArrows}`.

Changes between 1.6 and 1.7

New environments `{DispWithArrows}` and `{DispWithArrows*}`.

Changes between 1.7 and 1.8

The numbers and tags of the environment `{DispWithArrows}` are now compatible with all the major LaTeX packages concerning references (`autonum`, `cleveref`, `fancyref`, `hyperref`, `prettyref`, `refstyle`, `typedref` and `varioref`) and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.

Changes between 1.8 and 1.9

New option `wrap-lines` for the environments `{DispWithArrows}` and `{DispWithArrows*}`.

Changes between 1.9 and 1.10

If the option `wrap-lines` is used, the option “`text width`” of Tikz is still active: if the value given to “`text width`” is lower than the width computed by `wrap-lines`, this value is used to wrap the lines.

The option `wrap-lines` is now fully compatible with the class option `leqno`.

Correction of a bug: `\nointerlineskip` and `\makebox[.6\linewidth]{}` should be inserted in `{DispWithArrows}` only in vertical mode.

Changes between 1.10 and 1.11

New commands `\WithArrowsNewStyle` and `\WithArrowsRightX`.

Changes between 1.11 and 1.12

New command `\tagnextline`.

New option `tagged-lines`.

An option of position (`ll`, `lr`, `rl`, `rr` or `i`) is now allowed at the local level even if the option `group` or the option `groups` is used at the global or environment level.

Compatibility of `{DispWithArrows}` with `\qedhere` of `amsthm`.

Compatibility with the packages `refcheck`, `showlabels` and `listbls`.

The option `\AllowLineWithoutAmpersand` is deprecated because lines without ampersands are now always allowed.

Changes between 1.12 and 1.13

Options `start-adjust`, `end-adjust` and `adjust`.

This version is not strictly compatible with previous ones. To restore the behaviour of the previous versions, one has to use the option `adjust` with the value `0 pt`:

```
\WithArrowsOptions{adjust = 0pt}
```

Changes between 1.13 and 1.14

New options `up` and `down` for the arrows.

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments: [//www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)).

Modification of the code of `\WithArrowsNewStyle` following a correction of a bug in `l3keys` in the version of `l3kernel` of 2019/01/28.

New error message `Inexistent~v-node` to avoid a `pgf` error.

The error `Option incompatible with 'group(s)'` was suppressed in the version 1.12 but this was a mistake since this error is used with the option `xoffset` at the local level. The error is put back.

Changes between 1.14 and 1.15

Option `new-group` to start a new group of arrows (only available when the environment is composed with the option `groups`).

Tikz externalization is now deactivated in the environments of the extension `witharrows`.³⁹

Changes between 1.15 and 1.16

Option `no-arrows`

The behaviour of `{DispWithArrows}` after an `\item` of a LaTeX list has been changed : no vertical is added. The previous behaviour can be restored with the option `standard-behaviour-with-items`. A given name can no longer be used for two distinct environments. However, it's possible to deactivate this control with the option `allow-duplicate-names`.

Changes between 1.16 and 1.17

Option `format`

Changes between 1.17 and 1.18

New option `<...>` for `{DispWithArrows}`.

Option `subequations`.

Warning when `{WithArrows}` or `{DispWithArrows}` ends by `\\`.

No space before an environment `{DispWithArrows}` if we are at the beginning of a `{minipage}`.

Changes between 1.18 and 2.0

A version of `witharrows` is available for plain-Tex.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	A
<code>\\</code> 63, 72, 731, 907, 1009, 1209, 1902, 1916, 1921, 1923, 1930, 1937, 1944, 1949, 1957, 1958, 1967, 1968, 1977, 1978, 1988, 1989, 1999, 2000, 2013, 2021, 2030, 2038, 2047, 2053, 2059, 2067, 2078, 2085, 2092, 2098, 2103, 2109, 2116, 2122, 2129, 2136, 2142, 2152, 2159, 2161, 2171, 2178, 2179, 2220, 2320, 2327	<code>\A</code> 482
<code>\{</code> 396, 621, 1911, 1923, 1930, 1936, 1957, 1967, 1977, 2009, 2018, 2029, 2038, 2046, 2052, 2053, 2058, 2059, 2077, 2085, 2092, 2097, 2116, 2121, 2122, 2136, 2142, 2152, 2161	<code>\arabic</code> 997
<code>\}</code> 621, 1923, 1930, 1936, 1957, 1967, 1977, 2009, 2018, 2029, 2038, 2046, 2052, 2053, 2058, 2059, 2077, 2085, 2092, 2097, 2116, 2121, 2122, 2136, 2142, 2152, 2161	<code>\Arrow</code> 272, 1999
<code>\</code> 54, 1914, 1915, 1922, 1929, 1955, 1956, 1999, 2010, 2019, 2020, 2027, 2028, 2036, 2044, 2045, 2076, 2083, 2091, 2108, 2130, 2135, 2141, 2151, 2158, 2160, 2167, 2221	Arrow internal commands: <code>_wa_Arrow</code> 671, 674, 706, 781
	<code>\AtBeginDocument</code> 103, 225, 399
	B
	<code>\begin</code> 758, 1112, 1257, 1582, 1621, 1672, 1763, 1787, 1835, 1873, 2232, 2252
	<code>\belowdisplayskip</code> 1287
	<code>\bgroup</code> 815, 819, 893, 1186
	bool commands: <code>\bool_gset_true:N</code> 94
	<code>\bool_if:NTF</code> 75, 85, 227, 422, 748, 750, 758, 783, 786, 797, 806, 812, 820, 862, 898, 912, 971, 982, 987, 1005, 1014, 1017, 1021, 1029, 1041, 1106, 1107, 1112, 1115, 1144, 1164, 1178, 1188, 1228, 1284, 1295, 1296, 1315, 1355, 1559, 1563, 1593, 1603, 1793, 1797, 2074, 2144, 2277, 2298

³⁹Before this version, there was an error when using `witharrows` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `witharrows` because they use the options `overlay` and `remember picture`.

<code>\bool_if:nTF</code>	872, 879, 904, 918, 943, 952, 955, 964, 401, 1098, 1198, 1282, 1335, 1345, 1360, 1369, 1437, 1447, 1467, 1481, 1499, 1641, 1853
<code>\bool_if_p:N</code>	1345, 1393, 1404, 1445, 1506, 1508, 1531, 1616, 1639, 1667, 1719, 1725, 1731, 1813, 1870, 2191
<code>\bool_new:N</code>	25, 26, 96, 111, 245, 246, 247, 275, 278, 279, 280, 282, 283, 284, 285, 291, 1070
<code>\bool_set:Nn</code>	1344
<code>\bool_set_false:N</code>	118, 742, 883, 1008, 1153, 1154, 1469, 1540, 1541, 1571, 1575, 1739, 1740, 2283, 2304
<code>\bool_set_true:N</code>	97, 114, 423, 882, 927, 1092, 1118, 1121, 1380, 1382, 1411, 1465, 1544, 1545, 1548, 1549, 1570, 1574, 1578, 1579, 1745, 1746, 1748, 1749
<code>\c_false_bool</code>	1198
<code>\l_tmpa_bool</code>	1570, 1571, 1578, 1593
<code>\l_tmpb_bool</code>	1574, 1575, 1579, 1603
box commands:	
<code>\box_clear_new:N</code>	1138, 1185, 1217
<code>\box_dp:N</code>	1237
<code>\box_ht:N</code>	1236
<code>\box_set_to_last:N</code>	1212
<code>\box_use:N</code>	1214
<code>\box_use_drop:N</code>	1224, 1233, 1252
<code>\box_wd:N</code>	828, 1137, 1216, 1220, 1221
<code>\l_tmpa_box</code>	1127, 1137, 1212, 1214, 1216, 1218
C	
<code>\catcode</code>	22, 2331
char commands:	
<code>\char_generate:nn</code>	163, 179
clist commands:	
<code>\clist_clear:N</code>	418, 1325, 1327, 1340, 1366, 1373
<code>\clist_count:N</code>	1821
<code>\clist_gput_right:Nn</code>	1819
<code>\clist_if_in:NnTF</code> ..	265, 439, 451, 974, 1206
<code>\clist_map_inline:nn</code>	105
<code>\clist_new:N</code>	261, 393
<code>\clist_pop:NN</code>	1830, 1832
<code>\clist_put_left:Nn</code>	442
<code>\clist_put_right:Nn</code>	404
<code>\clist_remove_all:Nn</code>	441
<code>\clist_reverse:N</code>	1831
<code>\clist_set:Nn</code>	262, 266, 394, 419, 438, 1207, 1334, 1365, 1372
<code>\clist_sort:Nn</code>	1824
<code>\g_tmpa_clist</code>	1819, 1821, 1824, 1830, 1831, 1832, 1833
<code>\coordinate</code>	1024, 1032, 1045
<code>\cr</code>	901, 1053, 1199, 2158
cs commands:	
<code>\cs_generate_variant:Nn</code>	36, 101, 102, 1505, 1646
<code>\cs_gset:Npx</code>	977
<code>\cs_if_exist:NTF</code>	713, 993, 1364, 1371, 2280, 2301
<code>\cs_if_free:NTF</code>	1264, 1754, 1757, 1817
<code>\cs_new:Npn</code>	1688
<code>\cs_new_protected:Npn</code>	28, 29, 30, 32, 33, 34, 35, 124, 128, 131, 146, 155, 160, 176, 234, 263, 293, 302, 569, 632, 674, 680, 686, 705, 707, 771, 833,
<code>\cs_set:Npn</code>	920, 924, 1618
<code>\cs_set:Npx</code>	981
<code>\cs_set_eq:NN</code>	237, 380, 381, 731, 752, 763, 764, 765, 766, 767, 768, 781, 925, 926, 1016, 1018, 1142, 1383, 1387, 1388
<code>\cs_set_protected:Npn</code>	664
<code>\cs_to_str:N</code>	164, 180
D	
<code>\DeclareOption</code>	97, 98
dim commands:	
<code>\dim_compare:nNnTF</code>	1050, 1219, 1270, 1657, 1660, 1857
<code>\dim_eval:n</code>	1594, 1604
<code>\dim_gset:Nn</code>	1216, 1221, 1271, 1678
<code>\dim_gset_eq:NN</code>	1255
<code>\dim_gzero_new:N</code>	1215, 1254
<code>\dim_max:nn</code>	1056, 1678, 1773
<code>\dim_set:Nn</code>	653, 657, 658, 1056, 1137, 1234, 1269, 1478, 1554, 1651, 1656, 1772, 1850, 1852, 1855, 2231, 2251
<code>\dim_set_eq:NN</code>	1130, 1161, 1179, 1180, 1183, 1240, 1658, 1685, 1769, 1770, 1858
<code>\dim_use:N</code>	1591, 1592, 1595, 1601, 1602, 1605, 1663, 1775, 1777, 1795, 1799, 1861
<code>\dim_zero:N</code>	732
<code>\dim_zero_new:N</code>	717, 1136, 1160, 1176
<code>\c_max_dim</code>	1255, 1478, 1554
<code>\g_tmpa_dim</code>	1678, 1685
<code>\l_tmpa_dim</code>	1056, 1057, 1234, 1243, 1269, 1270, 1271, 1651, 1657, 1658, 1660, 1663, 1769, 1772, 1773, 1775, 1777, 1850, 1852, 1857, 1858, 1861, 2231, 2232, 2251, 2252
<code>\l_tmpb_dim</code>	1656, 1657, 1658, 1770, 1775, 1855, 1857, 1858
<code>\c_zero_dim</code>	191, 192, 953, 1050, 1056, 1130, 1240, 1660
<code>\displaystyle</code>	797, 898, 1144
<code>\displaywidth</code>	1161, 1180, 1183
<code>\DispWithArrows</code>	1076, 1309
DispWithArrows commands:	
<code>\DispWithArrows_i</code>	1080, 1081, 1083
<code>\DispWithArrows_ii</code>	1086, 1087, 1089
<code>\draw</code>	321, 643, 1649, 1844, 1884, 2226, 2245
E	
<code>\egroup</code>	908, 909, 1210, 1222
else commands:	
<code>\else:</code>	889
<code>\end</code> ..	912, 957, 1066, 1275, 1295, 1296, 1608, 1632, 1680, 1779, 1801, 1863, 1888, 2235, 2255
<code>\endDispWithArrows</code>	1202, 1311
<code>\endtikzpicture</code>	1278, 1611, 1635, 1683, 1782, 1804, 1866, 1891
<code>\endWithArrows</code>	904
exp commands:	
<code>\exp_args:NNo</code>	1538
<code>\exp_args:No</code>	1109, 1538
<code>\exp_args:NV</code>	1064, 1643, 1833

<code>\ExplSyntaxOff</code>	2332	<code>\int_use:N</code>	700, 702, 793, 840, 844, 854, 858, 865, 924, 1024, 1032, 1045, 1416, 1420, 1424, 1427, 1519, 1523, 1535, 1559, 1563, 1565, 1569, 1573, 1688, 1922, 1929, 1956, 2009, 2018, 2021, 2028, 2045, 2084, 2116
<code>\ExplSyntaxOn</code>	21	<code>\int_zero_new:N</code>	733, 734, 735, 736, 737, 1406, 1407, 1408, 1511, 1513
F			
<code>\fi</code>	1119, 1122	<code>\c_one_int</code> ..	442, 738, 1412, 1449, 2282, 2303
fi commands:		<code>\l_tmpa_int</code> ..	693, 694, 1262, 1265, 1268, 2160
<code>\fi:</code>	891, 1158, 1171	<code>\c_zero_int</code>	1459
<code>\foreach</code>	1815, 1882	<code>\itshape</code>	217
G			
<code>\globaldefs</code>	427	J	
group commands:		<code>\jot</code>	236, 367, 1007
<code>\group_align_safe_begin:</code>	949	K	
<code>\group_align_safe_end:</code>	970	<code>\k</code>	1882, 1885
<code>\group_begin:</code> ..	426, 874, 922, 1078, 1129, 1141, 1239, 1386, 1395, 1510, 1533, 1736, 2207	keys commands:	
<code>\group_end:</code>	429, 915, 929, 1134, 1148, 1246, 1299, 1390, 1402, 1529, 1614, 1811, 2215	<code>\keys_define:nn</code>	38, 304, 384, 408, 463, 488, 523, 529, 548, 580, 640, 1689, 2197, 2263, 2270
H			
<code>\halign</code>	818	<code>\keys_if_exist:nnTF</code>	2194
hbox commands:		<code>\l_keys_key_tl</code>	47, 54, 297, 573, 618, 636, 1901, 1909, 1949, 1954, 1967, 1977, 1987, 1998, 2064, 2071, 2210, 2319, 2325
<code>\hbox_overlap_left:n</code>	1015, 1019, 1042	<code>\keys_set:nn</code> ..	668, 691, 749, 751, 1505, 1737
<code>\hbox_overlap_right:n</code>	864	<code>\keys_set_known:nn</code>	1507, 2200
<code>\hbox_set:Nn</code>	1127, 1139, 1218	<code>\l_keys_value_tl</code>	575, 2176
<code>\hbox_to_wd:nn</code>	1170, 1226, 1231	L	
<code>\hbox_unpack_clear:N</code>	1218	<code>\label</code>	766, 767, 1142, 1351, 2141, 2151
<code>\hfil</code>	790, 804, 805, 848, 1230, 1247, 1249	<code>\langle</code>	396, 1913
<code>\hfill</code>	791	<code>\lbrace</code>	396, 457, 1911
I			
<code>\ialign</code>	814	<code>\lbrack</code>	396, 1912
if commands:		<code>\lceil</code>	396, 1914
<code>\if_mode_math:</code>	889, 1156	<code>\left</code>	1132, 1242
<code>\if_mode_vertical:</code>	1168	<code>\lfloor</code>	396, 1914
<code>\ignorespacesafterend</code>	1302	<code>\lgroup</code>	396, 1912
<code>\input</code>	7, 8	<code>\linewidth</code>	1160, 1161, 1170, 1179
int commands:		<code>\lmoustache</code>	396, 1913
<code>\int_case:nn</code>	892, 1542, 1741	lua commands:	
<code>\int_compare:nNnTF</code>	135, 607, 652, 779, 801, 921, 934, 946, 1318, 1396, 1429, 1431, 1459, 1487, 1494, 1552, 1576, 1590, 1600, 1760, 2169	<code>\lua_now:e</code>	125
<code>\int_compare:nTF</code>	583, 599, 1477, 1484, 1501, 1526, 1821, 1826	<code>\lVert</code>	404, 1915
<code>\int_compare_p:n</code>	1439, 1451, 1453	<code>\lvert</code>	404, 1915
<code>\int_compare_p:nNn</code>	1441, 1449	M	
<code>\int_eval:n</code>	933	math commands:	
<code>\int_gdecr:N</code>	777	<code>\c_math_toggle_token</code>	794, 800, 897, 900, 1131, 1133, 1143, 1147, 1165, 1172, 1241, 1245, 1286, 1289, 1292
<code>\int_gincr:N</code>	689, 792, 823, 935, 976	<code>\mathsurround</code>	732
<code>\int_gset:Nn</code>	793, 937, 939, 941	MH commands:	
<code>\int_gset_eq:NN</code>	754	<code>\MH_if_boolean:nTF</code>	1100, 1283, 1337, 1339, 1362
<code>\int_gzero:N</code>	720, 722, 724, 824	<code>\MH_set_boolean_T:n</code>	1103
<code>\int_gzero_new:N</code>	725	msg commands:	
<code>\int_incr:N</code>	1435, 1527	<code>\msg_error:nn</code>	32, 33
<code>\int_new:N</code> ..	250, 251, 252, 255, 257, 259, 288	<code>\msg_error:nnn</code>	35
<code>\int_set:Nn</code>	253, 303, 336, 347, 386, 388, 390, 584, 637, 693, 738, 753, 1399, 1412, 1418, 1422, 1512, 1514, 1515, 1521, 1525, 1734, 2282, 2303	<code>\msg_fatal:nn</code>	34
<code>\int_set_eq:NN</code>	1470, 1471, 1472, 1489	<code>\msg_info:nn</code>	78
<code>\int_step_inline:nnn</code>	1669	<code>\msg_line_number:</code>	698
<code>\int_step_variable:nNn</code>	1262	<code>\msg_new:nnn</code>	28
<code>\int_until_do:nNnn</code>	1413, 1516	<code>\msg_new:nnnn</code>	29
		<code>\msg_redirect_name:nnn</code>	31
		<code>\msg_set:nnn</code>	2208

sys commands:	
\sys_if_engine luatex:TF	122
T	
\tabskip	806, 1187, 1192, 1195
\tag	765, 1330, 2128, 2130, 2135
tag internal commands:	
_wa_tag	765, 1328
\tagnextline	768, 1379
TeX and L ^A T _ε commands:	
\@	22, 2331
\@currentlabel	981
\@currenenv	711
\@eqnnum	1027
\@ifclassloaded	77, 87
\@ifpackageloaded	80, 90, 113
\@cequation	976
\@cref@constructprefix	989
\@cref@currentlabel	990
\@cref@equation@alias	993, 994
\@cref@result	989, 997
\@hyper@refstepcounter	985
\@if@inlabel	1117
\@if@minipage	1120
\@intertext@	1107
\@pequation	981, 998
\pgf@x	1269, 1591, 1601, 1652, 1678, 1769, 1773, 1795, 1799, 1856
\pgf@y	1594, 1595, 1604, 1605, 1770, 1777, 1795, 1799
\pgfutil@firstofone	1268, 1587, 1597, 1650, 1677, 1768, 1771, 1792, 1796, 1849
\protected@edef	990
\qed@elt	1388
\QED@stack	1389
\setQED@elt	1388
\spread@equation	234, 237, 761
\sr@name	1106
\tagform@	1018
\This@name	984
\tikz@library@external@loaded	713
\tikz@parse@node	1268, 1650, 1849
\tikz@scan@one@point	1587, 1597, 1677, 1768, 1771, 1792, 1796
\tikz@text@width	1653, 1851
\theequation	978, 1016
\tikz	835, 849, 1023, 1031, 1044
\tikzpicture	1260, 1585, 1624, 1675, 1766, 1790, 1838, 1876
\tikzset	186, 196, 205, 210, 327, 351, 644, 714, 923, 1692
tl commands:	
\c_empty_tl	352, 645
\c_novalue_tl	244, 825, 1081, 1125, 1223
\tl_clear_new:N	745, 746, 1152
\tl_const:Nn	1647, 2224, 2243
\tl_gset:Nn	1588, 1598, 1653, 1774, 1776, 1794, 1798, 1851
\tl_head:n	450
\tl_if_empty:NTF	976, 978, 1332, 1654, 1852
\tl_if_empty:nTF	479
\tl_if_eq:NNTF	825, 1125, 1223
\tl_if_eq:nnTF	1751
\tl_if_novalue:nTF	1109
\tl_new:N	243, 286, 287
\tl_put_right:Nn	101, 472, 475
\tl_set:Nn	450, 454, 483, 1109, 1343, 1557, 1562
\tl_set_eq:NN	244, 1642, 2284, 2305
\tl_to_str:N	2206
\tl_to_str:n	151, 2202, 2204
\g_tmpa_tl	977, 981, 998, 1016, 1588, 1613, 1653, 1654, 1656, 1774, 1794, 1807, 1851, 1852
\l_tmpa_tl	165, 181, 450, 453, 728, 930, 931, 933, 936, 937, 938, 939, 940, 941, 1417, 1418, 1421, 1422, 1520, 1521, 1524, 1525, 1536, 1539, 1567, 1613, 1830, 1844
\g_tmpb_tl	1598, 1613, 1776, 1798, 1807
\l_tmpb_tl	1832, 1847, 1848
token commands:	
\token_to_str:N	49, 272, 1911, 1912, 1913, 1914, 1915, 1921, 1929, 1937, 1988, 1999, 2104, 2108, 2114, 2121, 2128, 2130, 2135, 2141, 2151, 2158, 2159, 2221, 2326
U	
\unpenalty	1211
\unskip	1211
use commands:	
\use:N	167, 170, 173, 183, 402
\use:n	775
\use_none:nn	380
\use_none:nnn	381
\usepackage	82, 92
\usetikzlibrary	10, 2326
V	
\vbox	892
\vcenter	892, 1132, 1243, 1252
\vfil	1243
\vtop	892, 1186
W	
wa internal commands:	
\g_wa_alignment_dim	1215, 1216, 1220, 1221, 1231
\c_wa_amsmath_loaded_bool	227, 402, 422, 1107, 1345
\c_wa_amsthm_loaded_bool	786
_wa_analyze_end:Nn	960, 1062
_wa_Arrow_code_after	926, 1716, 1719
_wa_Arrow_code_after_i	1722, 1723, 1725
_wa_Arrow_code_after_ii	1728, 1729, 1731
_wa_Arrow_first_columns:	705, 752
_wa_Arrow_i	677, 678, 680
_wa_Arrow_ii	683, 684, 686
\g_wa_arrow_int	255, 689, 700, 702, 719, 720, 921, 937, 1413, 1443
\l_wa_arrow_int	735, 1412, 1413, 1416, 1420, 1424, 1427, 1435, 1449, 1463, 1470, 1474, 1476, 1486, 1491, 1495, 1515, 1516, 1519, 1523, 1527, 1535, 1565, 1569, 1573
\g_wa_arrow_int_seq	254, 719, 936
\c_wa_autonum_loaded_bool	1369
\c_wa_cleveref_loaded_bool	987, 1355
\l_wa_code_after_tl	475, 746, 928
\l_wa_code_before_tl	472, 745, 760
_wa_code_for_possible_arrow:	1434, 1445

`\g_wa_col_int` 259, 723, 724, 754,
777, 779, 792, 793, 801, 824, 941, 946, 1318
`\g_wa_col_int_seq` 258, 723, 940
`\c_wa_command_ignored_str` 1896, 2110
`\l_wa_command_name_str`
. 269, 270, 316, 752, 781, 926
`__wa_construct_halign:` 771, 778, 894, 1191
`__wa_construct_nodes:` 802, 833
`__wa_convert_to_str_seq:N` . . 146, 158, 522
`__wa_cr:` 731, 943
`__wa_cr_i:` 950, 952
`__wa_cr_ii:` 953, 955, 967
`__wa_cr_iii:n` 959, 962, 964
`__wa_def_function_tmpa:n` 1616, 1643
`\l_wa_delim_wd_dim` 828, 1136, 1137
`\l_wa_displaystyle_bool` 324, 797, 898, 1144
`__wa_draw_arrow:nnn`
. 381, 1613, 1639, 1646, 1807
`__wa_draw_arrows:nn` . . 380, 1443, 1461, 1508
`__wa_draw_arrows_i:` 1526, 1531
`\l_wa_end_adjust_dim` . . 374, 655, 658, 1604
`__wa_error:n` 32,
81, 91, 300, 338, 349, 425, 455, 468, 480,
484, 527, 534, 555, 576, 578, 585, 601, 608,
624, 706, 890, 896, 947, 1051, 1346, 1356,
1357, 1432, 1710, 1822, 2287, 2290, 2308, 2311
`__wa_error:nn` 35, 36,
1316, 1319, 1333, 1752, 1755, 1758, 1818, 2195
`__wa_eval_if_allowed:n` 293, 303
`\c_wa_extensible_delimiters_clist` . .
. 393, 394, 404, 452
`__wa_fatal:n` 34, 43, 1157, 1197, 1266
`\l_wa_final_int` . 734, 1422, 1429, 1472,
1487, 1489, 1494, 1502, 1525, 1526, 1555, 1563
`\l_wa_final_r_bool` 285,
1541, 1544, 1549, 1563, 1740, 1746, 1749, 1797
`\l_wa_final_tl` 287, 1562, 1597
`\l_wa_first_arrow_int` . . . 1511, 1512, 1515
`\l_wa_first_arrow_of_group_int`
. 1406, 1441, 1443, 1459, 1462, 1470
`\l_wa_first_arrows_seq`
. 1409, 1473, 1474, 1486, 1568
`\l_wa_first_line_of_group_int`
. 1407, 1471, 1485
`__wa_fix_pos_arrow:n`
. 632, 646, 647, 648, 649, 650
`__wa_fix_pos_option:n` . . . 302, 355, 357,
359, 361, 363, 1695, 1697, 1699, 1701, 1703
`\l_wa_fleqn_bool` 410, 820, 1188, 1228
`\g_wa_footnote_bool`
. 26, 40, 75, 94, 758, 912, 1296
`\g_wa_footnotehyper_bool` 25, 41, 85
`\l_wa_format_seq` 755, 756, 773
`\l_wa_format_str` . . . 289, 483, 739, 753, 756
`\l_wa_halign_box`
. 1185, 1186, 1224, 1236, 1237, 1252
`\c_wa_hyperref_loaded_bool` 982
`__wa_if_in_last_col_of_disp:Nn`
. 1313, 1325, 1327, 1330, 1351, 1379
`\l_wa_in_code_after_bool` . . 247, 927, 2074
`\l_wa_in_DispWithArrows_bool` 246,
750, 783, 806, 883, 971, 1092, 1641, 1853
`\l_wa_in_first_columns_bool` 282
`\l_wa_in_label_or_minipage_bool`
. 1070, 1118, 1121, 1164, 1178, 1284
`\l_wa_in_WithArrows_bool`
. 245, 748, 812, 882, 1315
`__wa_info:n` 88
`\l_wa_initial_int` 733, 1418,
1451, 1471, 1485, 1502, 1521, 1555, 1559, 2045
`\l_wa_initial_r_bool` 284,
1540, 1545, 1548, 1559, 1739, 1745, 1748, 1793
`\l_wa_initial_tl` 286, 1557, 1587
`\l_wa_input_line_str` 718, 1428, 2044
`\l_wa_interline_skip` 369, 744, 1057
`\l_wa_jump_int` 584, 693, 737, 738
`__wa_keys_set:` 1506, 1539
`__wa_label:n` 767, 1349
`\l_wa_labels_seq` 741, 979, 1001, 1353, 1359
`\l_wa_last_arrow_int` 1513, 1514, 1516
`\l_wa_last_arrows_seq`
. . . 1410, 1475, 1476, 1490, 1491, 1495, 1572
`\g_wa_last_env_int` 250, 935, 1688
`\l_wa_last_line_of_group_int`
. 1408, 1451, 1472, 1487, 1489, 1494
`\l_wa_left_brace_box` . 828, 1138, 1139, 1233
`\l_wa_left_brace_tl`
. . 243, 244, 531, 825, 1109, 1125, 1145, 1223
`\c_wa_leqno_bool` 96, 97, 1021, 1029
`\g_wa_line_int` 257,
265, 692, 693, 721, 722, 823, 840, 844,
854, 858, 865, 924, 939, 1024, 1032, 1045,
1262, 1429, 1526, 1922, 1929, 1956, 2021, 2028
`\g_wa_line_int_seq` 256, 721, 938
`\l_wa_linewidth_dim`
. . . . 818, 1176, 1179, 1180, 1183, 1226, 1251
`\l_wa_mathindent_dim` 412, 821, 1229
`\c_wa_mathtools_loaded_bool`
. 1098, 1282, 1335, 1360
`__wa_msg_new:nn`
. . . . 28, 45, 52, 57, 66, 1899, 1907, 1919,
1926, 1934, 1941, 1947, 2007, 2016, 2024,
2033, 2041, 2050, 2056, 2062, 2070, 2081,
2088, 2095, 2101, 2106, 2112, 2119, 2126,
2133, 2139, 2149, 2156, 2164, 2218, 2317, 2323
`__wa_msg_new:nnn`
. . . . 29, 1952, 1965, 1975, 1985, 1996, 2174
`__wa_msg_redirect_name:nn`
. 30, 313, 428, 435, 551
`__wa_MultiArrow:nn` 925, 1813
`__wa_MultiArrow_i:n` 1833, 1870
`\l_wa_name_str` . 470, 715, 843, 844, 857, 858
`\g_wa_names_seq` 273, 467, 469, 2185
`\l_wa_nb_cols_int` 288, 753, 754,
779, 801, 948, 1318, 2010, 2019, 2084, 2116
`\l_wa_new_box` 1217, 1218, 1220, 1221
`\l_wa_new_group_bool`
. 283, 1411, 1465, 1467, 1469
`__wa_nonumber:` 764, 1326
`__wa_notag:` 763, 1324
`__wa_old_label` 766, 1001, 1142
`\c_wa_option_ignored_str`
. 1894, 1917, 1924, 1945, 1958,
1968, 1978, 1989, 2000, 2031, 2039, 2068,
2079, 2093, 2099, 2117, 2123, 2172, 2321, 2328

<code>\l__wa_options_Arrow_code_after_seq</code> ..	1709, 1712, 1713, 2005
<code>\l__wa_options_Arrow_seq</code>	335, 344, 345, 346, 617, 626, 627, 1963, 2315, 2316
<code>\l__wa_options_DispWithArrows_seq</code> ..	230, 533, 536, 537, 1983, 2203
<code>\l__wa_options_WithArrows_seq</code>	509, 510, 522, 526, 618, 1973, 2202
<code>\l__wa_options_WithArrowsOptions_seq</code> ..	229, 554, 557, 558, 1994, 2205
<code>\l__wa_pos_arrow_int</code> .	252, 253, 303, 336, 347, 599, 607, 637, 652, 1396, 1399, 1431, 1439, 1453, 1477, 1501, 1542, 1552, 1576, 1590, 1600, 1734, 1741, 1760, 2169, 2282, 2303
<code>\l__wa_pos_env_int</code> ..	251, 386, 388, 390, 892
<code>\l__wa_pos_of_arrow_int</code>	736
<code>\g__wa_position_in_the_tree_seq</code>	248, 249, 726, 727, 930, 931, 932, 934
<code>__wa_post_halign:</code>	910, 918, 1280
<code>__wa_pre_halign:n</code>	707, 888, 1110
<code>\l__wa_prefix_str</code> ...	202, 700, 702, 729, 730, 840, 854, 865, 1265, 1416, 1420, 1424, 1427, 1519, 1523, 1535, 1565, 1754, 1757, 1817
<code>\l__wa_previous_key_str</code>	295, 297, 332, 334, 341, 343, 571, 573, 634, 636, 667, 690, 747, 1537, 1735, 2066, 2073, 2273, 2275, 2294, 2296
<code>__wa_qedhere:</code>	1382, 1383
<code>\l__wa_qedhere_bool</code>	280, 1004, 1013, 1014, 1036, 1040, 1041, 1153, 1382
<code>__wa_qedhere_i:</code>	1015, 1042, 1384
<code>\l__wa_replace_left_brace_by_tl</code>	454, 1132, 1242
<code>__wa_restore:N</code>	176, 1012, 1013, 1040
<code>\g__wa_right_x_dim</code>	920, 1254, 1255, 1270, 1271, 1652, 1856
<code>__wa_save:N</code>	160, 1003, 1004, 1036
<code>\l__wa_sbwi_bool</code>	275, 459, 1115
<code>__wa_scan_arrows:</code>	921, 1393
<code>__wa_scan_arrows_i:</code>	1398, 1401, 1404
<code>__wa_set_independent:</code>	569, 587, 588, 589, 590, 591, 2265, 2266
<code>__wa_set_qedhere:</code>	786, 1383
<code>__wa_set_seq_of_str_from_clist:Nn</code> ..	155, 510, 537, 558, 627, 1713
<code>\l__wa_show_node_names_bool</code>	329, 862
<code>\c__wa_showlabels_loaded_bool</code>	2144
<code>__wa_sort_seq:N</code> 131, 526, 533, 554, 617, 1709	
<code>\l__wa_start_adjust_dim</code> 371, 654, 657, 1594	
<code>\g__wa_static_col_int</code> ... 725, 793, 946, 948	
<code>\l__wa_status_arrow_str</code>	574, 600, 695, 716, 1425, 1455, 1482, 1499
<code>__wa_strcmp:n</code>	124, 128, 137
<code>\l__wa_string_Arrow_for_msg_str</code>	271, 272, 317, 1955, 2027, 2036, 2067, 2076, 2083, 2091, 2167
<code>\l__wa_subequations_bool</code> 291, 423, 1112, 1295	
<code>\l__wa_tag_next_line_bool</code>	279, 742, 1005, 1008, 1380
<code>\l__wa_tag_star_bool</code>	278, 1003, 1012, 1017, 1154, 1344
<code>\l__wa_tag_tl</code> 976, 978, 1152, 1332, 1343	
<code>__wa_tagnextline:</code>	768, 1377
<code>\l__wa_tags_clist</code> 261, 262, 265, 266, 418, 419, 438, 439, 441, 442, 974, 1206, 1207, 1325, 1327, 1334, 1340, 1365, 1366, 1372, 1373	
<code>__wa_test_if_to_tag:</code>	263, 785
<code>\c__wa_tikz_code_down_tl</code>	2243, 2306
<code>\l__wa_tikz_code_tl</code>	320, 642, 1642, 1643, 1704, 2284, 2305
<code>\c__wa_tikz_code_up_tl</code>	2224, 2285
<code>\c__wa_tikz_code_wrap_lines_tl</code> .	1642, 1647
<code>__wa_tmpa:n</code>	1618, 1644
<code>\l__wa_type_col_str</code> ..	773, 790, 791, 804, 805
<code>\l__wa_type_env_str</code>	710, 711, 885, 886, 1064, 1094, 1095, 1923, 1930, 1936, 1957, 1967, 1977, 2009, 2018, 2029, 2038, 2046, 2052, 2058, 2077, 2085, 2092, 2097, 2116, 2121, 2136, 2142, 2152, 2161
<code>\c__wa_typedref_loaded_bool</code>	1106
<code>__wa_update_x:n</code>	1502, 1555, 1667
<code>\c__wa_varwidth_loaded_bool</code>	2277, 2298
<code>__wa_warning:n</code>	33, 1065
<code>\l__wa_wrap_lines_bool</code>	446, 1641, 1853, 2283, 2304
<code>\l__wa_x_dim</code>	717, 1478, 1554, 1592, 1602, 1678, 1685
<code>\l__wa_xoffset_dim</code>	364, 653, 1539, 1706, 1738, 1844, 1847, 1848, 1885
<code>\l__wa_ygap_dim</code>	193, 306
<code>\l__wa_ystart_dim</code>	190, 309
<code>\WithArrows</code>	872
<code>\WithArrows commands:</code>	
<code>\WithArrows_i</code>	876, 877, 879
<code>\WithArrowsLastEnv</code>	1688
<code>\WithArrowsNbLines</code>	924
<code>\WithArrowsNewStyle</code>	2188, 2191, 2221
<code>\WithArrowsOptions</code> 49, 661, 664, 1308, 1988, 2214	
<code>\WithArrowsRightX</code>	920
	X
<code>\x</code> .	1815, 1817, 1818, 1819, 2231, 2241, 2248, 2251
	Y
<code>\y</code>	2241, 2248
	Z
<code>\Z</code>	482

Contents

1	Options for the shape of the arrows	1
2	Numbers of columns	6

3	Precise positioning of the arrows	6
4	The options 'up' and 'down' for individual arrows	9
5	Comparison with the environment {aligned}	9
6	Arrows in nested environments	12
7	Arrows from outside environments {WithArrows}	14
8	The environment {DispWithArrows}	16
	8.1 The option <...> of DispWithArrows	20
9	Advanced features	21
	9.1 Utilisation with plain-Tex	21
	9.2 The option tikz-code : how to change the shape of the arrows	21
	9.3 The command \WithArrowsNewStyle	22
	9.4 Vertical positioning of the arrows	22
	9.5 Footnotes in the environments of witharrows	23
	9.6 Option no-arrows	24
	9.7 Note for developers	24
10	Examples	24
	10.1 \MoveEqLeft	24
	10.2 Modifying the shape of the nodes	25
	10.3 Examples with the option tikz-code	25
	10.3.1 Example 1	25
	10.3.2 Example 2	26
	10.3.3 Example 3	27
	10.4 Automatic numbered loop	28
11	Implementation	29
	11.1 Declaration of the package and extensions loaded	29
	11.2 The packages footnote and footnotehyper	29
	11.3 The class option legno	31
	11.4 Some technical definitions	31
	11.5 Variables	34
	11.6 The definition of the options	37
	11.7 The command \Arrow	44
	11.8 The environments {WithArrows} and {DispWithArrows}	46
	11.8.1 Code before the \halign	46
	11.8.2 The construction of the preamble of the \halign	49
	11.8.3 The environment {WithArrows}	51
	11.8.4 After the construction of the \halign	52
	11.8.5 The command of end of row	54
	11.8.6 The environment {DispWithArrows}	57
	11.9 The commands \tag, \notag, \label, \tagnextline and \qedhere for {DispWithArrows}	62
	11.10 We draw the arrows	64
	11.11 The command \Arrow in code-after	72
	11.12 The command \MultiArrow in code-after	75
	11.13 The error messages of the package	76
	11.14 The command \WithArrowsNewStyle	81
	11.15 The options up and down	82
12	History	84
	Index	86