

The package `witharrows`*

F. Pantigny
fpantigny@wanadoo.fr

March 11, 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.

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

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

```

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

```

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

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. 15.

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).

```

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

```

*This document corresponds to the version 1.16 of `witharrows`, at the date of 2019/03/11.

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.

```

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

```

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

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

```

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

```

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

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.

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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 `TikzCode` presented p. 18.

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. 17.

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`.

Almost all the options can be given directly between square brackets to the environment `{WithArrows}`. **There must be no space before the opening bracket ([) of the options of the environment.** The 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`, `CodeBefore` and `CodeAfter`).

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

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

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

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

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

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

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

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

Special commands are available in `CodeAfter`: 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. 11.

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 = {}}}`.

2 Precise positioning of the arrows

The environment `{WithArrows}` defines, during the composition of the array, two series of nodes materialized in red in the following example.⁶

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

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.

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

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

⁶The 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{WithArrows}
(a+b)(a+ib)(a-b)(a-ib)
& = (a+b)(a-b)\cdot(a+ib)(a-ib) \ \
& = (a^2-b^2)(a^2+b^2) \ \Arrow[i]{because $(x-y)(x+y)=x^2-y^2$}\ \
& = a^4-b^4
\end{WithArrows}$

```

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

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 2x K'y_0 = \sqrt{x} \ \Arrow[...]\ \
...
\end{WithArrows}$

```

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

The environment `{WithArrows}` gives also a `groups` option (with a *s* in the name). With this option, the arrows are divided into several “groups”. Each group is a set of connected⁷ arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

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

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).

⁷More 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 ~.

$$\begin{aligned}
\sum_{k=0}^n \frac{\cos kx}{\cos^k x} &= \sum_{k=0}^n \frac{\Re(e^{ikx})}{(\cos x)^k} \\
&= \sum_{k=0}^n \Re\left(\frac{e^{ikx}}{(\cos x)^k}\right) \\
&= \Re\left(\sum_{k=0}^n \left(\frac{e^{ix}}{\cos x}\right)^k\right) \\
&= \Re\left(\frac{1 - \left(\frac{e^{ix}}{\cos x}\right)^{n+1}}{1 - \frac{e^{ix}}{\cos x}}\right) \\
&= \Re\left(\frac{1 - \frac{e^{i(n+1)x}}{\cos^{n+1} x}}{1 - \frac{e^{ix}}{\cos x}}\right) \\
&= \Re\left(\frac{\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos^{n+1} x}}{\frac{\cos x - e^{ix}}{\cos x}}\right) \\
&= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos x - e^{ix}}\right) \\
&= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - (\cos(n+1)x + i \sin(n+1)x)}{\cos x - (\cos x + i \sin x)}\right) \\
&= \frac{1}{\cos^n x} \Re\left(\frac{(\cos^{n+1} x - \cos(n+1)x) - i \sin(n+1)x}{-i \sin x}\right) \\
&= \frac{1}{\cos^n x} \cdot \frac{\sin(n+1)x}{\sin x}
\end{aligned}$$

(cos x)^k is real
ℜ(z + z') = ℜ(z) + ℜ(z')
sum of terms of a geometric progression
algebraic calculation
reduction to common denominator
ℜ(kz) = k · ℜ(z) if k is real
algebraic form of the complexes

3 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:

```

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

```

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

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

4 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}`.⁸

⁸In fact, it's possible to use the package `witharrows` without the package `amsmath`.


```

 $\begin{WithArrows}[interline=2ex]
\varphi(x,y) = 0 \quad \& \quad \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0
\Arrow{$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:

```

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

```

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

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

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

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

The following is composed with `{WithArrows}[c,displaystyle]`. The results are strictly identical.¹⁰

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

5 Arrows in nested environments

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

```

 $\begin{WithArrows}$ 
\varphi(x,y)=0
& \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \Arrow{the numbers are real}
& \Leftrightarrow
\left\{ \begin{array}{l}
\begin{WithArrows}[c]
x+2y & = 0 \\
2x+4y & = 0
\end{WithArrows}
\end{array} \right. \right.
& \Leftrightarrow
\left\{ \begin{array}{l}
\begin{WithArrows}[c]
x+2y & = 0 \Arrow{tikz=-}{the same equation} \\
x+2y & = 0
\end{WithArrows}
\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 `CodeAfter` option. Indeed, in `CodeAfter`, a special version of `\Arrow` is available (we will call it “`\Arrow in CodeAfter`”).

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

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

¹⁰In 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 `CodeAfter` :

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

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

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

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

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

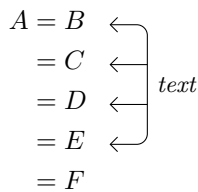
$$\begin{aligned} \varphi(x, y) = 0 &\Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0 \\ &\Leftrightarrow \begin{cases} x + 2y = 0 \\ 2x + 4y = 0 \end{cases} \\ &\Leftrightarrow \begin{cases} x + 2y = 0 \\ x + 2y = 0 \end{cases} \quad \left. \begin{array}{l} \textit{division by 2} \\ \swarrow \end{array}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

The package `witharrows` gives also another command available only in `CodeAfter`: 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,
                  CodeAfter = {\MultiArrow{1,...,4}{text}} ]
A & = B \\
  & = C \\
  & = D \\
  & = E \\
  & = F
\end{WithArrows}

```



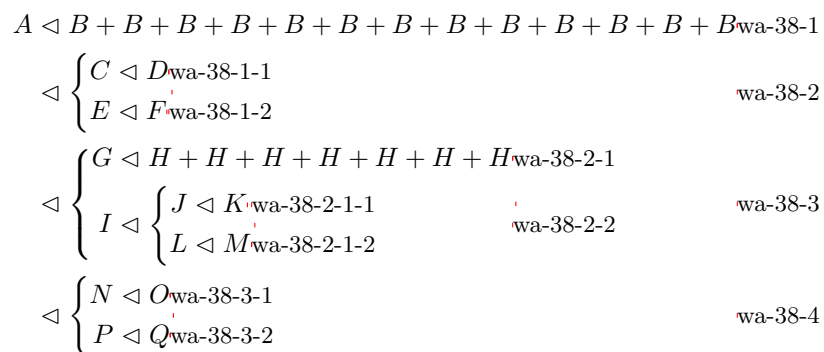
As of now, there is no option available for the command `\MultiArrow` (maybe in a future release).

6 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.¹¹



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;
- 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 arrow¹²;
- the Tikz style `WithArrows/arrow/tips` is the style for the tip of the arrow (loaded by `WithArrows/arrow`).

¹¹There is an option `show-node-names` to show the names of these nodes.

¹²More precisely, this style is given to the Tikz option “`every path`” before drawing the arrow with the code of the option `TikzCode`. This style is modified (in TeX scopes) by the option `tikz` of `witharrows`.

For example, we can draw an arrow from `wa-38-2-1-2-r.south` to `wa-38-3-2-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 `CodeAfter` 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}
```

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

7 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}
```

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

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

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.¹³

```
\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.¹⁴

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

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

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

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.

¹³Notice that the environment `{DispWithArrows}` is compatible with the command `\qedhere` of `amsthm`.

¹⁴Even in this case, it's possible to put a "manual tag" with the command `\tag`.

```

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

```

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

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}

```

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

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 package `amsmath` is loaded, it's possible to use the environment `{subequations}` and the command `\intertext` in the environments `{DispWithArrows}` and `{DispWithArrows*}` (and even the `\intertext` of `nccmath` if this package is loaded).

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*}`.¹⁵

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}

```

¹⁵The 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.

$$\begin{aligned}
S_{2(p+1)} &= \sum_{k=1}^{2(p+1)} (-1)^k k^2 & (6) \\
&= \sum_{k=1}^{2p} (-1)^k k^2 + (-1)^{2p+1} (2p+1)^2 + (-1)^{2p+2} (2p+2)^2 & (7) \\
&= S_{2p} - (2p+1)^2 + (2p+2)^2 & (8) \\
&= 2p^2 + p - 4p^2 - 4p - 1 + 4p^2 + 8p + 4 & (9) \\
&= 2p^2 + 5p + 3 & (10)
\end{aligned}$$

The environments `{DispWithArrows}` and `{DispWithArrows*}` provide an option `wrap-lines`. With this option, the lines of the label are automatically wrapped on the right.¹⁶

```

\begin{DispWithArrows*}[displaystyle,wrap-lines]
S_n
& = \frac{1}{n} \operatorname{Re} \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right)
\Arrow{sum of terms of a geometric progression of ratio $e^{i \frac{\pi}{2n}}$}
& = \frac{1}{n} \operatorname{Re} \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^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*}

```

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

sum of terms of a geometric progression of ratio $e^{i \frac{\pi}{2n}}$
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 CodeAfter 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.¹⁷

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} \operatorname{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 $e^{i \frac{\pi}{2n}}$}

```

¹⁶It's possible to avoid the hyphenations of the words with the option "`align = flush left`" of Tikz.

¹⁷It's possible to disable this feature with the option `standard-behaviour-with-items`.

```

& = \frac{1}{n} \Re \left( \frac{1 - \bigl(e^{i\frac{\pi}{2n}}\bigr)^n}{1 - e^{i\frac{\pi}{2n}}} \right)
\Arrow{$\bigl(e^{i\frac{\pi}{2n}}\bigr)^n = e^{i\frac{\pi}{2} = i}$} \
& = \frac{1}{n} \Re \left( \frac{1 - i}{1 - e^{i\frac{\pi}{2n}}} \right)
\end{DispWithArrows}
\end{enumerate}

```

$$\begin{aligned}
1. S_n &= \frac{1}{n} \Re \left(\sum_{k=0}^{n-1} \left(e^{i\frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - \left(e^{i\frac{\pi}{2n}} \right)^n}{1 - e^{i\frac{\pi}{2n}}} \right) \\
&= \frac{1}{n} \Re \left(\frac{1 - i}{1 - e^{i\frac{\pi}{2n}}} \right)
\end{aligned}
\left. \begin{array}{l} \text{we use the formula for a sum of terms of a geometric progression of ratio } e^{i\frac{2\pi}{n}} \\ \left(e^{i\frac{\pi}{2n}} \right)^n = e^{i\frac{\pi}{2}} = i \end{array} \right\} \tag{11}$$

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}` allows only two columns.
- The environment `{DispWithArrows}` can not be inserted in an environment `{gather}` of `amsmath`.
- An environment `{DispWithArrows}` is always unbreakable (even with `\allowdisplaybreaks` of `amsmath`).
- The commands `\label`, `\tag`, `\notag` and `\nonumber` are allowed only in the second column.
- After an `\item` of a LaTeX list, no vertical space is added.
- **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 `autonom`, `cleveref`, `fancyref`, `fncylab`, `hyperref`, `listlbls`, `prettyref`, `refcheck`, `refstyle`, `showlabels`, `smartref`, `typedref` and `varioref`, and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.¹⁸

It is not compatible with `showkeys` (not all the labels are shown).

8 Advanced features

8.1 The option `TikzCode` : how to change the shape of the arrows

The option `TikzCode` allows the user to change the shape of the arrows.¹⁹

For example, the options “up” and “down” described previously (cf. p. 8) are programmed internally with `TikzCode`.

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:

¹⁸We recall that `varioref`, `hyperref`, `cleveref` and `autonom` must be loaded in this order. The package `witharrows` can be loaded anywhere.

¹⁹If the option `wrap-lines` is used in an environment `{DispWithArrows}` or `{DispWithArrows*}`, the option `TikzCode` will have no effect for the arrows of this environment but only for the arrows in the nested environments `{WithArrows}`.

```
\draw (#1) to node [#3] (#2) ;
```

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

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

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

The environments `{DispWithArrows}` and its starred version `{DispWithArrows*}` provide a command `\WithArrowsRightX` which can be used in a definition of `TikzCode`. 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. 23.

8.2 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. 23.

8.3 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).

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

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).

```

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

```

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

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:

```

 $\WithArrowsOptions{adjust = 0pt}$ 

```

8.4 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:

$$\begin{aligned}
 A &= (a + b)^2 \\
 &= a^2 + b^2 + 2ab \quad \downarrow \text{We expand}^{20}
 \end{aligned}$$

²⁰A footnote.

8.5 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.

8.6 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}
```

9 Examples

9.1 With only one column

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

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

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

9.2 MoveEqLeft

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

```
$$\begin{WithArrows}[interline=0.5ex]
\MoveEqLeft \arccos(x) = \arcsin \frac{45}{13} + \arcsin \frac{5}{13}
\Arrow{because both are in $[-\frac{\pi}{2}, \frac{\pi}{2}]$} \\
& \Leftrightarrow x = \sin \left( \arcsin \frac{45}{13} + \arcsin \frac{5}{13} \right) \\
& \Leftrightarrow x = \frac{45}{13} \cos \arcsin \frac{5}{13} + \frac{5}{13} \cos \arcsin \frac{45}{13}
\Arrow{\$ \forall x \in [-1, 1], \cos(\arcsin x) = \sqrt{1-x^2}$} \\
& \Leftrightarrow x = \frac{45}{13} \sqrt{1 - \bigl( \frac{5}{13} \bigr)^2} \\
& + \frac{5}{13} \sqrt{1 - \bigl( \frac{45}{13} \bigr)^2} \\
\end{WithArrows}$$
```

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


9.3 Modifying the shape of the nodes

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

```

\begin{WithArrows}%
  [interline = 4mm,
   tikz = {every node/.style = {circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt,
                                font = \tiny}}]
E & \Longleftarrow 3 (2x+4) = 6
\Arrow{$\div 3$}\
& \Longleftarrow 2x+4 = 2
\Arrow{$-4$}\
& \Longleftarrow 2x = -2
\Arrow{$\div 2$} \
& \Longleftarrow 2x = -1
\end{WithArrows}

```

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


9.4 Examples with the option TikzCode

We recall that the option `TikzCode` is the Tikz code used by `witharrows` to draw the arrows.²¹

The value by default of `TikzCode` 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.

9.4.1 Example 1

In the following example, we define the value of `TikzCode` 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.

²¹If an environment `{DispWithArrows}` or `{DispWithArrows*}` is used with the option `wrap-lines`, the value of the option `TikzCode` is not used for this environment (but is used for the environments nested inside).

```

\begin{DispWithArrows*}%
  [displaystyle,
   ygap = 2mm,
   ystart = 0mm,
   TikzCode = {\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{kn}{n}\bigr)
.....

```

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

9.4.2 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 `TikzCode`, 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,
   TikzCode = {\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{kn}{n}\bigr)
\Arrow{\$\cos x = \Re(e^{ix})\$}\
.....

```

$$S_n = \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) \quad (12)$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i \frac{k\pi}{2n}}\right) \quad (13)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right) \quad (14)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right) \quad (15)$$

$$= \frac{1}{n} \Re\left(\frac{1 - \left(e^{i \frac{\pi}{2n}}\right)^n}{1 - e^{i \frac{\pi}{2n}}}\right) \quad (16)$$

$$= \frac{1}{n} \Re\left(\frac{1 - i}{1 - e^{i \frac{\pi}{2n}}}\right) \quad (17)$$

cos $x = \Re(e^{ix})$

$\Re(z + z') = \Re(z) + \Re(z')$

exp is a morphism for \times et +

sum of terms of a geometric progression of ratio $e^{i \frac{2\pi}{n}}$

9.4.3 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,
  TikzCode = {\draw[rounded corners,
    every node/.style = {circle,
      draw,
      auto = false,
      inner sep = 1pt,
      fill = gray!50,
      font = \tiny }]}

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

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


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

9.5 Automatic numbered loop

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

```

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

```

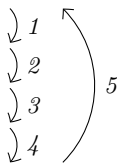
The command `\WithArrowsNbLines` is a command available in `CodeAfter` which gives the total number of lines (=rows) of the current environment (it's a command and not a counter).

```

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

```

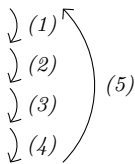
$a.$ f est continuous on E
 $b.$ f est continuous in 0
 $c.$ f is bounded on the unit sphere
 $d.$ $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K \|x\|$
 $e.$ f is lipschitzian



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

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

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



10 Implementation

10.1 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.²²

```
1 \RequirePackage{tikz}
2 \usetikzlibrary{arrows.meta,bending}
3 \RequirePackage{expl3}[2019/02/15]
```

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

```
4 \RequirePackage{l3keys2e}
5 \ProvidesExplPackage
6   {witharrows}
7   {\myfiledate}
8   {\myfileversion}
9   {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`.

```
10 \RequirePackage { xparse } [ 2018-10-17 ]
```

10.2 The packages `footnote` and `footnotehyper`

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 1.16), 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.

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

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

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

```
13 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
14 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { witharrows } }
15 \cs_new_protected:Npn \@@_msg_redirect_name:nn
16   { \msg_redirect_name:nnn { witharrows } }
```

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

```
17 \keys_define:nn { WithArrows / package }
18   {
19     footnote .bool_gset:N = \g_@@_footnote_bool ,
20     footnotehyper .bool_gset:N = \g_@@_footnotehyper_bool ,
21     unknown .code:n =
22       \msg_fatal:nn { witharrows } { Option-unknown-for~package }
23   }
24 \@@_msg_new:nn { Option-unknown~for~package }
25   {
26     You~can't~use~the~option~'\l_keys_key_tl'~when~loading~the~
27     package~witharrows.~Try~to~use~the~command~
28     \token_to_str:N\WithArrowsOptions.
29   }
```

²²cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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

```

30 \ProcessKeysOptions { WithArrows / package }

31 \@@_msg_new:nn { Option-incompatible-with-Beamer }
32   {
33     The-option-'\l_keys_key_tl'\ is-incompatible-
34     with-Beamer-because-Beamer-has-its-own-system-to-extract-footnotes.
35   }

36 \@@_msg_new:nn { footnote-with-footnotehyper-package }
37   {
38     You-can't-use-the-option-'footnote'~because-the-package~
39     footnotehyper-has-already-been-loaded.~
40     If-you-want,~you-can-use-the-option-'footnotehyper'~and-the-footnotes~
41     within-the-environments-of~witharrows~will-be-extracted-with-the-tools~
42     of-the-package-footnotehyper.
43   }

44 \@@_msg_new:nn { footnotehyper-with-footnote-package }
45   {
46     You-can't-use-the-option-'footnotehyper'~because-the-package~
47     footnote-has-already-been-loaded.~
48     If-you-want,~you-can-use-the-option-'footnote'~and-the-footnotes~
49     within-the-environments-of~witharrows~will-be-extracted-with-the-tools~
50     of-the-package-footnote.
51   }

52 \bool_if:NT \g_@@_footnote_bool
53   {
54     \@ifclassloaded { beamer }
55       { \msg_fatal:nn { witharrows } { Option-incompatible-with-Beamer } }
56       { }
57     \@ifpackageloaded { footnotehyper }
58       { \msg_fatal:nn { witharrows } { footnote-with-footnotehyper-package } }
59       { }
60     \usepackage { footnote }
61   }

62 \bool_if:NT \g_@@_footnotehyper_bool
63   {
64     \@ifclassloaded { beamer }
65       { \msg_fatal:nn { witharrows } { Option-incompatible-with-Beamer } }
66       { }
67     \@ifpackageloaded { footnote }
68       { \msg_fatal:nn { witharrows } { footnotehyper-with-footnote-package } }
69       { }
70     \usepackage { footnotehyper }
71     \bool_gset_true:N \g_@@_footnote_bool
72   }

```

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

10.3 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).

```

73 \bool_new:N \c_@@_leqno_bool
74 \DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
75 \DeclareOption* { }
76 \ProcessOptions*

```

10.4 Some technical definitions

```

77 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
78 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { witharrows } }
79 \cs_generate_variant:Nn \@@_error:nn { n 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`.²³

```

80 \AtBeginDocument
81 {
82   \clist_map_inline:nn
83     {
84       amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
85       typedref, varwidth
86     }
87     {
88       \bool_new:c { c_@@_#1_loaded_bool }
89       \ifpackageloaded { #1 }
90         { \bool_set_true:c { c_@@_#1_loaded_bool } }
91         { }
92     }
93 }

```

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

```

94 \sys_if_engine luatex:TF
95 {
96   \cs_new_protected:Npn \@@_strcmp:nn #1 #2
97     { \lua_now:e { l3kernel.strptime('#1','#2') } }
98 }
99 {
100   \cs_new_protected:Npn \@@_strcmp:nn #1 #2
101     { \pdfTeX_strcmp:D { #1 } { #2 } }
102 }

```

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

```

103 \cs_new_protected:Npn \@@_sort_seq:N #1
104 {
105   \seq_sort:Nn #1
106   {
107     \int_compare:nNnTF
108       {
109         \@@_strcmp:nn
110           { \str_lower_case:n { ##1 } }
111           { \str_lower_case:n { ##2 } }
112       }
113       > 0
114       \sort_return_swapped:
115       \sort_return_same:
116   }
117 }

```

The following variant will be used in the following command.

²³It's not possible to use `\ifpackageloaded` in the core of the functions because `\ifpackageloaded` is available only in the preamble.

```
118 \cs_generate_variant:Nn \seq_set_split:Nnn { N x x }
```

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.

```
119 \cs_new_protected:Npn \@@_save:N #1
120 {
121   \seq_set_split:Nxx \l_tmpa_seq
122     { \char_generate:nn { ` } { 12 } }
123     { \cs_to_str:N #1 }
124   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
```

The string `\l_tmpa_str` will contain the *type* of the variable.

```
125   \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
126   \use:c { \l_tmpa_str _if_exist:cF }
127     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
128     {
129       \use:c { \l_tmpa_str _new:c }
130       { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
131     }
132   \use:c { \l_tmpa_str _gset_eq:cN }
133     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
134 }
```

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

```
135 \cs_new_protected:Npn \@@_restore:N #1
136 {
137   \seq_set_split:Nxx \l_tmpa_seq
138     { \char_generate:nn { ` } { 12 } }
139     { \cs_to_str:N #1 }
140   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
141   \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
142   \use:c { \l_tmpa_str _set_eq:Nc }
143     #1 { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
144 }
```

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.

```
145 \tikzset
146 {
147   @@_node_style / .style =
148   {
149     above = \l_@@_ystart_dim ,
150     inner-sep = \c_zero_dim ,
151     minimum-width = \c_zero_dim ,
152     minimum-height = \l_@@_ygap_dim
153   }
154 }
```

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.²⁴

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.

```
155 \tikzset
156 {
157   @@_standard / .style =
```

²⁴The `v`-nodes, created near the end of line in `{DispWithArrows}` and `{DispWithArrows*}` are not shown with the option `show-nodes`.

```

158     {
159         remember~picture ,
160         overlay ,
161         name~prefix = wa - \l_@@_prefix_str -
162     }
163 }

```

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`).

```

164 \tikzset
165 {
166     WithArrows / arrow / tips / .style =
167     { > = { Straight~Barb [ scale = 1.2 , bend ] } }
168 }

```

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

```

169 \tikzset
170 {
171     WithArrows / arrow / .style =
172     {
173         align = left ,

```

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

```

174         auto = left ,
175         font = \small \itshape ,
176         WithArrows / arrow / tips ,
177         bend-left = 45 ,
178         ->
179     }
180 }

```

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`).

```

181 \AtBeginDocument
182 {
183     \bool_if:NF \c_@@_amsmath_loaded_bool
184     {
185         \cs_set_protected:Npn \spread@equation
186         {
187             \openup \jot
188             \cs_set_eq:NN \spread@equation \prg_do_nothing:
189         }
190     }
191 }

```

10.5 Variables

The boolean `\l_@@_in-WithArrows_bool` will be raised in an environment `{WithArrows}` and the boolean `\l_@@_in_dispwitharrows_bool` in an environment `{DispWithArrows}` or `{DispWithArrows*}`. The boolean `\l_@@_in_CodeAfter_bool` will be raised during the execution of the `CodeAfter` (option `CodeAfter`).

```

192 \bool_new:N \l_@@_in-WithArrows_bool
193 \bool_new:N \l_@@_in-DispWithArrows_bool
194 \bool_new:N \l_@@_in-CodeAfter_bool

```

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

```
195 \seq_new:N \g_@@_position_in_the_tree_seq
196 \seq_gput_right:Nn \g_@@_position_in_the_tree_seq 1
```

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

```
197 \int_new:N \g_@@_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`).

```
198 \int_new:N \l_@@_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 `CodeAfter` 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 `CodeAfter` (see below).

```
199 \int_new:N \l_@@_pos_arrow_int
200 \int_set:Nn \l_@@_pos_arrow_int 3
```

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

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

```
201 \seq_new:N \g_@@_arrow_int_seq
202 \int_new:N \g_@@_arrow_int
203 \seq_new:N \g_@@_line_int_seq
204 \int_new:N \g_@@_line_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`.

```
205 \clist_new:N \l_@@_tags_clist
206 \clist_set:Nn \l_@@_tags_clist { all }
```

```
207 \str_set:Nn \l_@@_CommandName_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 `CommandName` is defined to be `Explanation`, this string will contain `"\Arrow alias \Explanation"`.

```
208 \str_set:Nx \l_@@_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`.

```
209 \seq_new:N \g_@@_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.

```
210 \bool_new:N \l_@@_sbwi_bool
```

10.6 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* level²⁵;
- with `\begin{WithArrows}[...]`: this level will be called *environment* level;
- with `\Arrow[...]` (included in `CodeAfter`): 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 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).

```
211 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
212 {
213   \str_if_empty:NTF \l_@@_previous_key_str
214     {
215       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
216       #1
217     }
218     { \@@_error:n { Incompatible~options } }
219 }

220 \cs_new_protected:Npn \@@_fix_pos_option:n #1
221 { \@@_eval_if_allowed:n { \int_set:Nn \l_@@_pos_arrow_int { #1 } } }
```

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

```
222 \keys_define:nn { WithArrows / Global }
223 {
224   ygap .dim_set:N = \l_@@_ygap_dim ,
225   ygap .value_required:n = true ,
226   ygap .initial:n = 0.4 ex ,
227   ystart .dim_set:N = \l_@@_ystart_dim ,
228   ystart .value_required:n = true ,
229   ystart .initial:n = 0.4 ex ,
230   more-columns .code:n =
231     \@@_msg_redirect_name:nn { Third-column~in~WithArrows } { none } ,
232   more-columns .value_forbidden:n = true,
233   CommandName .code:n =
234     \str_set:Nn \l_@@_CommandName_str { #1 }
235     \str_set:Nx \l_@@_string_Arrow_for_msg_str
236     { \c_backslash_str Arrow~alias~\c_backslash_str #1 } ,
237   CommandName .value_required:n = true ,
238   TikzCode .tl_set:N = \l_@@_tikz_code_tl,
239   TikzCode .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
240   TikzCode .value_required:n = true ,
241   displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
242   displaystyle .default:n = true ,
```

²⁵This 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.


```

243 show-nodes .code:n =
244   \tikzset { @@_node_style / .append~style = { draw , red } } ,
245 show-nodes .value_forbidden:n = true,
246 show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
247 show-node-names .default:n = true ,
248 group .code:n =
249   \str_if_empty:NTF \l_@@_previous_key_str
250   {
251     \str_set:Nn \l_@@_previous_key_str { group }
252     \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
253     \int_set:Nn \l_@@_pos_arrow_int 7
254   }
255   { \@@_error:n { Incompatible~options } } ,
256 group .value_forbidden:n = true ,
257 groups .code:n =
258   \str_if_empty:NTF \l_@@_previous_key_str
259   {
260     \str_set:Nn \l_@@_previous_key_str { groups }
261     \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
262     { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
263     \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
264     \int_set:Nn \l_@@_pos_arrow_int 6
265   }
266   { \@@_error:n { Incompatible~options } } ,
267 groups .value_forbidden:n = true ,
268 tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
269 tikz .initial:n = \c_empty_tl ,
270 tikz .value_required:n = true ,
271 rr .value_forbidden:n = true ,
272 rr .code:n = \@@_fix_pos_option:n 3 ,
273 ll .value_forbidden:n = true ,
274 ll .code:n = \@@_fix_pos_option:n 1 ,
275 rl .value_forbidden:n = true ,
276 rl .code:n = \@@_fix_pos_option:n 2 ,
277 lr .value_forbidden:n = true ,
278 lr .code:n = \@@_fix_pos_option:n 0 ,
279 i .value_forbidden:n = true ,
280 i .code:n = \@@_fix_pos_option:n 5 ,
281 xoffset .dim_set:N = \l_@@_xoffset_dim ,
282 xoffset .value_required:n = true ,
283 xoffset .initial:n = 3 mm ,
284 jot .dim_set:N = \jot ,
285 jot .value_required:n = true ,
286 interline .skip_set:N = \l_@@_interline_skip ,
287 interline .value_required:n = true ,
288 start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
289 start-adjust .value_required:n = true ,
290 start-adjust .initial:n = 0.4 ex ,
291 end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
292 end-adjust .value_required:n = true ,
293 end-adjust .initial:n = 0.4 ex ,
294 adjust .code:n =
295   \dim_zero_new:N \l_@@_start_adjust_dim
296   \dim_set:Nn \l_@@_start_adjust_dim { #1 }
297   \dim_zero_new:N \l_@@_end_adjust_dim
298   \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
299 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 CodeAfter”.

```

300 no-arrows .code:n =
301   \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn

```

```

302     \cs_set_eq:NN \@@_draw_arrow:nnn \use_none:nnn ,
303     no-arrows .value_forbidden:n = true ,
304 }

```

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`.

```

305 \keys_define:nn { WithArrows / WithArrowsSpecific }
306 {
307   t .code:n          = \int_set:Nn \l_@@_pos_env_int 0 ,
308   t .value_forbidden:n = true ,
309   c .code:n          = \int_set:Nn \l_@@_pos_env_int 1 ,
310   c .value_forbidden:n = true ,
311   b .code:n          = \int_set:Nn \l_@@_pos_env_int 2 ,
312   b .value_forbidden:n = true
313 }

```

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`.

```

314 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
315 {
316   fleqn .bool_set:N = \l_@@_fleqn_bool ,
317   fleqn .default:n = true ,
318   mathindent .dim_set:N = \l_@@_mathindent_dim ,
319   mathindent .value_required:n = true ,
320   mathindent .initial:n = 25 pt ,
321   notag .code:n =
322     \str_if_eq:nnTF { #1 } { true }
323     { \clist_clear:N \l_@@_tags_clist }
324     { \clist_set:Nn \l_@@_tags_clist { all } } ,
325   notag .default:n = true ,
326   nonumber .meta:n = notag ,
327   allow-multiple-labels .code:n =
328     \@@_msg_redirect_name:nn { Multiple~labels } { none } ,
329   allow-multiple-labels .value_forbidden:n = true ,
330   wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
331   wrap-lines .default:n = true ,
332   tagged-lines .code:n =
333     \clist_set:Nn \l_@@_tags_clist { #1 }
334     \clist_if_in:NnT \l_@@_tags_clist { first }
335     {
336       \clist_remove_all:Nn \l_@@_tags_clist { first }
337       \clist_put_left:Nn \l_@@_tags_clist \c_one_int
338     } ,
339   tagged-lines .value_required:n = true ,

```

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`).

```

340   standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
341   standard-behaviour-with-items .default:n = true
342 }

```

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

```

343 \keys_define:nn { WithArrows / Env }
344 {
345   name .code:n =
346     \seq_if_in:NnTF \g_@@_names_seq { #1 }
347     { \@@_error:n { Duplicate~name } }
348     { \seq_gput_left:Nn \g_@@_names_seq { #1 } }
349     \str_set:Nn \l_@@_name_str { #1 } ,

```

```

350 name .value_required:n = true ,
351 CodeBefore .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
352 CodeBefore .value_required:n = true,
353 CodeAfter .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
354 CodeAfter .value_required:n = true
355 }

```

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

```

356 \keys_define:nn { WithArrows }
357 {
358   WithArrows .inherit:n =
359   {
360     WithArrows / Global ,
361     WithArrows / WithArrowsSpecific ,
362     WithArrows / Env
363   } ,
364   DispWithArrows .inherit:n =
365   {
366     WithArrows / DispWithArrowsSpecific ,
367     WithArrows / Global ,
368     WithArrows / Env ,
369   } ,
370   WithArrowsOptions .inherit:n =
371   {
372     WithArrows / Global ,
373     WithArrows / WithArrowsSpecific ,
374     WithArrows / DispWithArrowsSpecific
375   }
376 }

```

```

377 \keys_define:nn { WithArrows / WithArrows }
378 {
379   unknown .code:n =
380     \@@_sort_seq:N \l_@@_options_WithArrows_seq
381     \@@_error:n { Unknown-option-WithArrows }
382 }

```

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

```

383 \seq_set_from_clist:Nn \l_@@_options_WithArrows_seq
384 {
385   adjust, b, c, CodeAfter, CodeBefore, CommandName, displaystyle, end-adjust,
386   group, groups, i, interline, jot, ll, lr, more-columns, name, no-arrows, rl,
387   rr, show-node-names, show-nodes, start-adjust, t, tikz, TikzCode, xoffset,
388   ygap, ystart
389 }

```

```

390 \keys_define:nn { WithArrows / DispWithArrows }
391 {
392   unknown .code:n =
393     \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
394     \@@_error:n { Unknown-option-DispWithArrows }
395 }

```

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

```

396 \seq_set_from_clist:Nn \l_@@_options_DispWithArrows_seq
397 {

```

```

398   allow-multiple-labels, CodeAfter, CodeBefore, CommandName, TikzCode, adjust,
399   displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
400   mathindent, name, no-arrows, nonumber, notag, rl, rr, show-node-names,
401   show-nodes, start-adjust, tagged-lines, tikz, wrap-lines, xoffset, ygap,
402   ystart
403 }
404 \keys_define:nn { WithArrows / WithArrowsOptions }
405 {
406   allow-duplicate-names .code:n =
407   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
408   allow-duplicate-names .value_forbidden:n = true ,
409   unknown .code:n =
410   \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
411   \@@_error:n { Unknown~option~WithArrowsOptions }
412 }

```

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

```

413 \seq_set_from_clist:Nn \l_@@_options_WithArrowsOptions_seq
414 {
415   allow-duplicate-names, allow-multiple-labels, b, c, CommandName,
416   more-columns, TikzCode, adjust, displaystyle, end-adjust, fleqn, group,
417   groups, i, interline, jot, ll, lr, mathindent, no-arrows, nonumber, notag,
418   rl, rr, show-node-names, show-nodes, standard-behaviour-with-items,
419   start-adjust, t, tagged-lines, tikz, wrap-lines, xoffset, ygap, ystart
420 }

```

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”.

```

421 \cs_new_protected:Npn \@@_set_independent:
422 {
423   \str_if_empty:NTF \l_@@_previous_key_str
424   {
425     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
426     \str_set:Nn \l_@@_status_arrow_str { independent }
427     \str_if_eq:VnF \l_keys_value_tl { NoValue }
428     { \@@_error:n { Value~for~a~key } }
429   }
430   { \@@_error:n { Incompatible~options~in~Arrow } }
431 }

```

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”.

```

432 \keys_define:nn { WithArrows / Arrow / FirstPass }
433 {
434   jump .code:n =
435   \int_compare:nTF { #1 > 0 }
436   { \int_set:Nn \l_@@_jump_int { #1 } }
437   { \@@_error:n { Negative-jump } } ,
438   jump .value_required:n = true,
439   rr .code:n = \@@_set_independent: ,
440   ll .code:n = \@@_set_independent: ,
441   rl .code:n = \@@_set_independent: ,
442   lr .code:n = \@@_set_independent: ,
443   i .code:n = \@@_set_independent: ,

```

```

444 rr .default:n = NoValue ,
445 ll .default:n = NoValue ,
446 rl .default:n = NoValue ,
447 lr .default:n = NoValue ,
448 i .default:n = NoValue ,
449 new-group .value_forbidden:n = true,
450 new-group .code:n =
451   \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
452   { \str_set:Nn \l_@@_status_arrow_str { new-group } }
453   { \@@_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.

```

454 TikzCode .code:n = \prg_do_nothing: ,
455 TikzCode .value_required:n = true ,
456 tikz .code:n = \prg_do_nothing: ,
457 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).

```

458 xoffset .code:n =
459   \int_compare:nNnT \l_@@_pos_arrow_int > 5
460   { \@@_error:n { Option~xoffset~forbidden } } ,
461 xoffset .value_required:n = true ,
462 start-adjust .code:n = \prg_do_nothing: ,
463 start-adjust .value_required:n = true ,
464 end-adjust .code:n = \prg_do_nothing: ,
465 end-adjust .value_required:n = true ,
466 adjust .code:n = \prg_do_nothing: ,
467 adjust .value_required:n = true ,
468 unknown .code:n =
469   \@@_sort_seq:N \l_@@_options_Arrow_seq
470   \@@_error:n { Unknown~option~in~Arrow }
471 }

```

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

```

472 \seq_set_from_clist:Nn \l_@@_options_Arrow_seq
473 {
474   adjust, end-adjust, i, jump, ll, lr, rl, rr, start-adjust, tikz, TikzCode,
475   xoffset
476 }

```

```

477 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1
478 {
479   \str_if_empty:NT \l_@@_previous_key_str
480   {
481     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
482     \int_set:Nn \l_@@_pos_arrow_int { #1 }
483   }
484 }

```

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.

```

485 \keys_define:nn {WithArrows / Arrow / SecondPass }
486 {
487   TikzCode .tl_set:N = \l_@@_tikz_code_tl ,
488   TikzCode .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,

```

```

489   tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
490   tikz .initial:n = \c_empty_tl ,
491   rr .code:n = \@@_fix_pos_arrow:n 3 ,
492   ll .code:n = \@@_fix_pos_arrow:n 1 ,
493   rl .code:n = \@@_fix_pos_arrow:n 2 ,
494   lr .code:n = \@@_fix_pos_arrow:n 0 ,
495   i .code:n = \@@_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.

```

496   xoffset .code:n =
497     \int_compare:nNnF \l_@@_pos_arrow_int > 5
498     { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
499   start-adjust .dim_set:N = \l_@@_start_adjust_dim,
500   end-adjust .dim_set:N = \l_@@_end_adjust_dim,
501   adjust .code:n =
502     \dim_set:Nn \l_@@_start_adjust_dim { #1 }
503     \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
504 }

```

`\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 construct with `{WithArrows}`). That's why we have constructed a set of keys specific to `\WithArrowsOptions`.

```

505 \NewDocumentCommand \WithArrowsOptions { m }
506 {
507   \str_clear_new:N \l_@@_previous_key_str
508   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
509 }

```

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

```

510 \NewDocumentCommand \@@_Arrow { 0 { } m ! 0 { } }
511 {

```

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.

```

512   \int_gincr:N \g_@@_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`.

```

513   \str_clear_new:N \l_@@_previous_key_str
514   \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:

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

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

```
516 \int_set:Nn \l_tmpa_int { \g_@@_line_int + \l_@@_jump_int }
517 \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`.

```
518 \prop_put:NnV \l_tmpa_prop { status } \l_@@_status_arrow_str
```

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

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

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

```
520 \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).

```
521 \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.

```
522 \prop_gclear_new:c
523 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
524 \prop_gset_eq:cN
525 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
526 \l_tmpa_prop
527 }
```

```
528 \cs_new_protected:Npn \@@_Arrow_first_column:
529 { \@@_error:n { Arrow-in~first~column } \@@_Arrow }
```

10.8 The environment `{WithArrows}`

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

```
530 \cs_new_protected:Npn \@@_pre_environment: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.

```
531 {
532 \str_clear_new:N \l_@@_type_env_str
533 \str_set:NV \l_@@_type_env_str \@currenenvir
```

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`.

```
534 \cs_if_exist:NT \tikz@library@external@loaded
535 { \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.

```
536 \str_clear_new:N \l_@@_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.

```
537 \str_clear_new:N \l_@@_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.

```
538 \dim_zero_new:N \l_@@_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).

```
539 \str_clear_new:N \l_@@_input_line_str
```

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

```
540 \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
541 \int_gzero:N \g_@@_arrow_int
542 \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
543 \int_gzero:N \g_@@_line_int
```

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

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

The nesting tree is used to create a prefix which will be used in the names of the Tikz nodes and in the names of the arrows (each arrow is a property list of 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).

```
545 \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
546 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
547 \str_clear_new:N \l_@@_prefix_str
548 \str_set:Nx \l_@@_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

```
549 \cs_set_eq:NN \ \@@_cr:
550 \dim_zero:N \mathsurround
```

These counters will be used later as variables.

```
551 \int_zero_new:N \l_@@_initial_int
552 \int_zero_new:N \l_@@_final_int
553 \int_zero_new:N \l_@@_arrow_int
554 \int_zero_new:N \l_@@_pos_of_arrow_int
555 \int_zero_new:N \l_@@_jump_int
556 \int_set:Nn \l_@@_jump_int \c_one_int
```

The boolean `\l_@@_in_first_column_bool` will be used to know wether we are in the first column of the environment `{WithArrows}` or `{DispWithArrows}`.

```
557 \bool_set_false:N \l_@@_in_first_column_bool
```

In (the second 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_@@_labels_seq`.

```
558 \seq_clear_new:N \l_@@_labels_seq
559 \bool_set_false:N \l_@@_tag_next_line_bool
```


The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.²⁶

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

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

```
561 \tl_clear_new:N \l_@@_code_before_tl
562 \tl_clear_new:N \l_@@_code_after_tl
```

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

```
563 \str_clear_new:N \l_@@_previous_key_str
564 \bool_if:NT \l_@@_in_WithArrows_bool
565   { \keys_set:nn { WithArrows / WithArrows } { #1 } }
566 \bool_if:NT \l_@@_in_DisWithArrows_bool
567   { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

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

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

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

```
569 \l_@@_code_before_tl
```

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

```
570 \cs_set_eq:cN \l_@@_CommandName_str \@@_Arrow
571 \cs_set_eq:NN \notag \@@_notag:
572 \cs_set_eq:NN \nonumber \@@_notag:
573 \cs_set_eq:NN \tag \@@_tag
574 \cs_set_eq:NN \@@_old_label \label
575 \cs_set_eq:NN \label \@@_label:n
576 \cs_set_eq:NN \tagnextline \@@_tagnextline:
577 }
```

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

Now, we begin the environment `{WithArrows}`.

```
578 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
579 {
580   \bool_set_true:N \l_@@_in_WithArrows_bool
581   \bool_set_false:N \l_@@_in_DisWithArrows_bool
582   \@@_pre_environment:n { #1 }
583   \if_mode_math: \else:
584     \@@_error:n { WithArrows~outside~math~mode }
585   \fi:
```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`²⁷ 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 mode²⁸ and therefore, we can use `\vcenter`.

```
586 \int_case:nn \l_@@_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
587 \bgroup
```

²⁶It's recalled that, by design, the option `interline` of an environment doesn't apply in the nested environments.

²⁷Notice that the use of `\vtop` seems color-safe here...

²⁸An error is raised if the environment is used outside math mode.

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).

```
588 \spread@equation
```

We begin the `\halign` and the preamble.

```
589 \ialign \bgroup
```

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

```
590 \int_gincr:N \g_@@_line_int
591 \cs_set_eq:cN \l_@@_CommandName_str \@@_Arrow_first_column:
592 \bool_set_true:N \l_@@_in_first_column_bool
593 \strut \hfil
594 $
595 \bool_if:NT \l_@@_displaystyle_bool \displaystyle
596 { ## }
597 $
598 &
```

```
599 $
600 \bool_if:NT \l_@@_displaystyle_bool \displaystyle
601 { { } ## }
602 $
```

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).

```
603 \tikz [ remember~picture , overlay ]
604 \node
605 [
606   node~contents = { } ,
607   @@_node_style ,
608   name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
609   alias =
610   {
611     \str_if_empty:NF \l_@@_name_str
612     { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
613   }
614 ]
615 ;
616 \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).

```
617 \tikz [ remember~picture , overlay ]
618 \node
619 [
620   node~contents = { } ,
621   @@_node_style ,
622   name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
623   alias =
624   {
625     \str_if_empty:NF \l_@@_name_str
626     { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
627   }
628 ]
629 ;
630 \bool_if:NT \l_@@_show_node_names_bool
```

```

631     {
632         \hbox_overlap_right:n
633         { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int }
634     }

```

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

```

635     &&
636     \@@_error:n { Third-column-in-WithArrows }
637     $
638     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
639     { ## }
640     $
641     \cr
642 }

```

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`).

```

643 {
644     \
645     \egroup
646     \egroup
647     \@@_post_environment:

```

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`).

```

648     \bool_if:NT \g_@@_footnote_bool { \end { savenotes } }
649 }

```

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

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

```

650 \cs_new_protected:Npn \@@_post_environment:

```

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

```

651 {
652     \cs_set:Npn \WithArrowsRightX { \g_@@_right_x_dim }

```

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

```

653     \int_compare:nNnT \g_@@_arrow_int > 0 \@@_scan_arrows:

```

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

```

654     \group_begin:
655     \tikzset { every~picture / .style = @@_standard }

```

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

```

656     \cs_set:Npn \WithArrowsNbLines { \int_use:N \g_@@_line_int }

```

The command `\MultiArrow` is available in `CodeAfter`, and we have a special version of `\Arrow`, called “`\Arrow` in `CodeAfter`” in the documentation.²⁹

```

657     \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
658     \cs_set_eq:cN \l_@@_CommandName_str \@@_Arrow_code_after
659     \bool_set_true:N \l_@@_in_CodeAfter_bool
660     \l_@@_code_after_tl
661     \group_end:

```

²⁹As for now, `\MultiArrow` has no option, and that's why its internal name is a name of `expl3` with the signature `:nn` whereas `\Arrow` in `CodeAfter` provides options and has the name of a function defined with `\NewDocumentCommand`.

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

```

662 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
663 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
664 \seq_gput_right:Nx \g_@@_position_in_the_tree_seq
665   { \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`.

```

666 \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
667   { \int_gincr:N \g_@@_last_env_int }

```

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

```

668 \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
669 \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
670 \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
671 \int_gset:Nn \g_@@_line_int \l_tmpa_tl
672 }

```

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

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).

```

673 \cs_new_protected:Npn \@@_cr:
674   {
675     \scan_stop:
676     \bool_if:NT \l_@@_in_first_column_bool { & { } }
677     \group_align_safe_begin:
678     \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
679   }

```

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.

```

680 \cs_new_protected:Npn \@@_cr_i:
681   { \peek_meaning:NTF [ \@@_cr_ii: { \@@_cr_ii: [ \c_zero_dim ] } }

```

```

682 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
683   {
684     \group_align_safe_end:

```

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

```

685 \bool_if:NT \l_@@_in_DispatchWithArrows_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`.

```

686   {
687     \clist_if_in:NnTF \l_@@_tags_clist { all }
688     {

```

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).

```
689 \tl_if_empty:NT \l_@@_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 `&`).

```
690 \cs_gset:Npx \g_tmpa_tl
691 { \tl_if_empty:NTF \l_@@_tag_tl \theequation \l_@@_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`.

```
692 \seq_if_empty:NF \l_@@_labels_seq
693 {
```

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`.

```
694 \cs_set:Npx \@currentlabel { \p@equation \g_tmpa_tl }
```

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

```
695 \bool_if:NT \c_@@_hyperref_loaded_bool
696 {
697   \str_set:Nn \This@name { equation }
698   \hyper@refstepcounter { equation }
699 }
```

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.

```
700 \bool_if:NT \c_@@_cleveref_loaded_bool
701 {
702   \cref@constructprefix { equation } \cref@result
703   \protected@edef \cref@currentlabel
704   {
705     [
706       \cs_if_exist:NTF \cref@equation@alias
707       \cref@equation@alias
708       { equation }
709     ]
710     [ \arabic { equation } ] [ \cref@result ]
711     \p@equation \g_tmpa_tl
712   }
713 }
```

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`).

```
714 \seq_map_function:NN \l_@@_labels_seq \@_old_label
715 }
```

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.

```
716 \@_save:N \l_@@_tag_star_bool
717 \@_save:N \l_@@_qedhere_bool
718 \bool_if:NT \l_@@_tag_next_line_bool
719 {
720   \openup -\jot
721   \bool_set_false:N \l_@@_tag_next_line_bool
722   \notag \&
723 }
724 &
725 \@_restore:N \l_@@_tag_star_bool
```

```

726     \@@_restore:N \l_@@_qedhere_bool
727     \bool_if:NT \l_@@_qedhere_bool
728       { \hbox_overlap_left:n \@@_qedhere_i: }
729     \cs_set_eq:NN \theequation \g_tmpa_tl
730     \bool_if:NT \l_@@_tag_star_bool
731       { \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`.

```

732     \hbox_overlap_left:n
733     {
734       \bool_if:NF \c_@@_leqno_bool
735       {
736         \tikz [ @@@_standard ]
737           \coordinate ( \int_use:N \g_@@_line_int - v ) ;
738       }
739       \quad
740       \@eqnnum
741     }
742     \bool_if:NT \c_@@_leqno_bool
743     {
744       \tikz [ @@@_standard ]
745         \coordinate ( \int_use:N \g_@@_line_int - v ) ;
746     }
747   }
748   {
749     \@@_save:N \l_@@_qedhere_bool
750     &
751     \@@_restore:N \l_@@_qedhere_bool
752     \bool_if:NT \l_@@_qedhere_bool
753       { \hbox_overlap_left:n \@@_qedhere_i: }
754     \tikz [ @@@_standard ]
755       \coordinate ( \int_use:N \g_@@_line_int - v ) ;
756   }
757 }
758 \cr \noalign { \skip_vertical:n { #1 + \l_@@_interline_skip } \scan_stop: }
759 }

```

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

10.9 The commands `tag`, `notag`, `label`, `tagnextline` and `qedhere` for `DispWithArrows`

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

```

760 \cs_new_protected:Npn \@@_if_in_second_col_of_disp:Nn #1 #2
761 {
762   \bool_if:NTF \l_@@_in-WithArrows_bool
763     { \@@_error:nn { Not~allowed~in~WithArrows } { #1 } }
764     {
765       \bool_if:NTF \l_@@_in_first_column_bool
766         { \@@_error:nn { Not~allowed~in~DispWithArrows } { #1 } }
767         { #2 }
768     }
769 }

```

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

```
770 \cs_new_protected:Npn \@@_notag:
771   { \@@_if_in_second_col_of_disp:Nn \notag { \clist_clear:N \l_@@_tags_clist } }
```

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

```
772 \NewDocumentCommand \@@_tag { s m }
773   {
774     \@@_if_in_second_col_of_disp:Nn \tag
775     {
776       \tl_if_empty:NF \l_@@_tag_tl
777       { \@@_error:nn { Multiple-tags } { #2 } }
778       \clist_set:Nn \l_@@_tags_clist { all }
779       \bool_if:nT \c_@@_mathtools_loaded_bool
780       {
781         \MH_if_boolean:nT { show_only_refs }
782         {
783           \MH_if_boolean:nF { show_manual_tags }
784           { \clist_clear:N \l_@@_tags_clist }
785         }
786       }
787       \tl_set:Nn \l_@@_tag_tl { #2 }
788       \bool_set:Nn \l_@@_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`.³⁰

```
789     \bool_if:nT { #1 && ! \bool_if_p:N \c_@@_amsmath_loaded_bool }
790     { \@@_error:n { tag*~without~amsmath } }
791   }
792 }
```

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`.

```
793 \cs_new_protected:Npn \@@_label:n #1
794   {
795     \@@_if_in_second_col_of_disp:Nn \label
796     {
797       \seq_if_empty:NF \l_@@_labels_seq
798       {
799         \bool_if:NTF \c_@@_cleveref_loaded_bool
800         { \@@_error:n { Multiple-labels~with~cleveref } }
801         { \@@_error:n { Multiple-labels } }
802       }
803       \seq_put_right:Nn \l_@@_labels_seq { #1 }
804       \bool_if:nT \c_@@_mathtools_loaded_bool
805       {
806         \MH_if_boolean:nT { show_only_refs }
807         {
808           \cs_if_exist:cTF { MT_r_#1 }
809           { \clist_set:Nn \l_@@_tags_clist { all } }
810           { \clist_clear:N \l_@@_tags_clist }
811         }
812       }
813       \bool_if:nT \c_@@_autonum_loaded_bool
814       {
815         \cs_if_exist:cTF { autonum@#1Referenced }
```

³⁰There are two versions of `@eqnnum`, a standard version and a version for the option `leqno`.

```

816         { \clist_set:Nn \l_@@_tags_clist { all } }
817         { \clist_clear:N \l_@@_tags_clist }
818     }
819 }
820 }

```

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

```

821 \cs_new_protected:Npn \@@_tagnextline:
822 {
823     \@@_if_in_second_col_of_disp:Nn \tagnextline
824     { \bool_set_true:N \l_@@_tag_next_line_bool }
825 }

```

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 second column of the environment `{DispWithArrows}` (only if the package `amsthm` has been loaded). `\@@_qedhere:` raises the boolean `\l_@@_qedhere_bool`.

```

826 \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }

```

In the third column of the `\halign` of `{DispWithArrows}`, 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`.

```

827 \cs_new_protected:Npn \@@_qedhere_i:
828 {
829     \group_begin:
830     \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.

```

831     \cs_set_eq:NN \qed@elt \setQED@elt
832     \QED@stack \relax \relax
833     \group_end:
834 }

```

10.10 The environment `{DispWithArrows}`

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

```

\[\vcenter{\halign to \displaywidth {...}}\]

```

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

```

835 \bool_new:N \l_@@_inlabel_bool
836 \NewDocumentEnvironment { DispWithArrows } { ! 0 { } }
837 {

```

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).

```

838     \bool_if:nT \c_@@_mathtools_loaded_bool
839     {
840         \MH_if_boolean:nT { show_only_refs }
841         {
842             \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`.

```

843         \MH_set_boolean_T:n { show_only_refs }
844     }
845 }

```


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`.

```
846 \bool_if:NT \c_@@_typedref_loaded_bool { \str_set:Nn \sr@name { equation } }
```

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

```
847 \bool_if:NT \c_@@_amsmath_loaded_bool \intertext@
848 \bool_set_true:N \l_@@_in_DispWithArrows_bool
849 \@@_pre_environment:n { #1 }
```

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_@@_inlabel_bool`.

```
850 \bool_if:NF \l_@@_sbwi_bool
851 {
852   \if@inlabel
853   \bool_set_true:N \l_@@_inlabel_bool
854   \fi
855 }
```

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

```
856 \tl_clear_new:N \l_@@_tag_tl
```

```
857 \bool_set_false:N \l_@@_qedhere_bool
```

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

```
858 \bool_set_false:N \l_@@_tag_star_bool
859 \if_mode_math:
860   \@@_error:n { DispWithArrows~in~math~mode }
861 \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_@@_inlabel_bool`.

```
862 \bool_if:NTF \l_@@_inlabel_bool
863   { $ } % $
864   {
```

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

```
865   \if_mode_vertical:
866   \nointerlineskip
867   \hbox_to_wd:nn { .6 \linewidth } { }
868   \fi:
869   $$
870 }
```

We use a `\vcenter` in order to prevent page breaks in the environment.

```
871 \bool_if:NTF \l_@@_inlabel_bool \vtop \vcenter \bgroup
872 \spread@equation
873 \bool_if:NTF \l_@@_fleqn_bool
874   { \tabskip = \c_zero_skip }
875   { \tabskip = 0 pt plus 1000 pt minus 1000 pt }
876 \halign to \bool_if:NTF \l_@@_inlabel_bool \linewidth \displaywidth
877 \bgroup
878 \int_gincr:N \g_@@_line_int
879 \cs_set_eq:cN \l_@@_CommandName_str \@@_Arrow_first_column:
880 \bool_set_true:N \l_@@_in_first_column_bool
881 \strut
882 \bool_if:NT \l_@@_fleqn_bool { \skip_horizontal:n \l_@@_mathindent_dim }
883 \hfil
884 $
```

```

885 \bool_if:NT \l_@@_displaystyle_bool \displaystyle
886 { ## }
887 $
888 \tabskip = \c_zero_skip
889 &
890 \clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int
891 { \clist_set:Nn \l_@@_tags_clist { all } }

```

The command `\qedhere` of `amsthm` is redefined here.

```

892 \bool_if:NT \c_@@_amsthm_loaded_bool
893 { \cs_set_eq:NN \qedhere \@@_qedhere: }
894 $
895 \bool_if:NT \l_@@_displaystyle_bool \displaystyle
896 { { } ## }
897 $
898 \tabskip = 0 pt plus 1000 pt minus 1000 pt
899 \tikz [ remember~picture , overlay ]
900 \node
901 [
902   node~contents = { } ,
903   @@_node_style ,
904   name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
905   alias =
906   {
907     \str_if_empty:NF \l_@@_name_str
908     { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
909   }
910 ]
911 ;
912 \hfil
913 \tikz [ remember~picture , overlay ]
914 \node
915 [
916   node~contents = { } ,
917   @@_node_style ,
918   name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
919   alias =
920   {
921     \str_if_empty:NF \l_@@_name_str
922     { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
923   }
924 ]
925 ;
926 \bool_if:NT \l_@@_show_node_names_bool
927 {
928   \hbox_overlap_right:n
929   { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int }
930 }
931 &

```

We put now the contents of the third column in math mode because because we don't want an error if the user uses a third column with contents in math mode: the error will be in the next cell.

```

932 $ ## $
933 \tabskip = \c_zero_skip
934 &&
935 \@@_error:n { Third~column~in~DispWithArrows }
936 \bool_if:nT \c_false_bool { ## }
937 \cr
938 }

```

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

```

939 {
940   \clist_if_in:NnT \l_@@_tags_clist { last }
941   { \clist_set:Nn \l_@@_tags_clist { all } }
942   \

```

The following `\egroup` is for the `\halign`.

```
943 \egroup
```

The following `\egroup` is for the `\vcenter` (aimed to prevent page breaks).

```
944 \egroup
```

If we are in an environment `{DispWithArrows}` or `{DispWithArrows*}`, 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.

```
945 \bool_if:NT \l_@@_in_DispWithArrows_bool
946 {
947   \dim_gzero_new:N \g_@@_right_x_dim
948   \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
949   \begin { tikzpicture } [ @@_standard ]
950     \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
951     {
952       \cs_if_free:cTF
953         { pgf@sh@ns@wa - \l_@@_prefix_str - \l_tmpa_int - v }
954         { \@@_error:n { Inexistent-v-node } }
955         {
956           \tikz@parse@node\pgfutil@firstofone ( \l_tmpa_int - v )
957           \dim_set:Nn \l_tmpa_dim \pgf@x
958           \dim_compare:nNnT \l_tmpa_dim < \g_@@_right_x_dim
959             { \dim_gset:Nn \g_@@_right_x_dim \l_tmpa_dim }
960         }
961     }
962   \end { tikzpicture }
963 }
```

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

```
964 \@@_post_environment:
```

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).

```
965 \bool_if:nT \c_@@_mathtools_loaded_bool
966 { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
967 \bool_if:NTF \l_@@_inlabel_bool
968 { $ % $
969   \skip_vertical:N \belowdisplayskip
970 }
971 { $$ }
```

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`).

```
972 \bool_if:NT \g_@@_footnote_bool { \end {savenotes} }
973 \ignorespacesafterend
974 }
```

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.

```
975 \NewDocumentEnvironment { DispWithArrows* } { }
976 {
977   \WithArrowsOptions { notag }
978   \DispWithArrows
979 }
980 \endDispWithArrows
```

10.11 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.

```
981 \cs_new_protected:Npn \@@_scan_arrows:
982   {
983     \group_begin:
984     \int_compare:nNnT \l_@@_pos_arrow_int = 7
985       {
986         \@@_scan_arrows_i:
987         \int_set:Nn \l_@@_pos_arrow_int 8
988       }
989     \@@_scan_arrows_i:
990     \group_end:
991   }

992 \cs_new_protected:Npn \@@_scan_arrows_i:
993   {
```

`\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 ajustement 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 the list of all the arrows of the group ending at the last line of the group (impossible to guess in advance).

```
994   \int_zero_new:N \l_@@_first_arrow_of_group_int
995   \int_zero_new:N \l_@@_first_line_of_group_int
996   \int_zero_new:N \l_@@_last_line_of_group_int
997   \seq_clear_new:N \l_@@_first_arrows_seq
998   \seq_clear_new:N \l_@@_last_arrows_seq
```

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

```
999   \bool_set_true:N \l_@@_new_group_bool
```

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

```
1000   \int_set:Nn \l_@@_arrow_int \c_one_int
1001   \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1002   {
```

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.

```

1003     \prop_get:cnN
1004     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1005     { initial } \l_tmpa_tl
1006     \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1007     \prop_get:cnN
1008     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1009     { final } \l_tmpa_tl
1010     \int_set:Nn \l_@@_final_int \l_tmpa_tl
1011     \prop_get:cnN
1012     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1013     { status } \l_@@_status_arrow_str
1014     \prop_get:cnN
1015     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1016     { input-line } \l_@@_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`.

```

1017     \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1018     {
1019         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1020         { \_@@_error:n { Too~few~lines~for~an~arrow } }
1021     }
1022     \@@_code_for_possible_arrow:

```

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

```

1023     \int_incr:N \l_@@_arrow_int
1024 }

```

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 (`\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 arrives after the last line of the environment).

```

1025     \bool_if:nT
1026     {
1027         \int_compare_p:n { \l_@@_pos_arrow_int != 7 }
1028         &&
1029         \int_compare_p:nNn \l_@@_first_arrow_of_group_int > 0
1030     }
1031     { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1032 }

```

```

1033 \cs_new_protected:Npn \@@_code_for_possible_arrow:
1034 {

```

We test if the previous arrow was in fact the last arrow of a group. In this case, we have to draw all the arrows of that group, except if we are with the option `group` and in the first step of treatment (`\l_@@_pos_arrow_int = 7`).

```

1035     \bool_if:nT
1036     {
1037         \int_compare_p:nNn \l_@@_arrow_int > \c_one_int
1038         &&
1039         ( \int_compare_p:n { \l_@@_initial_int > \l_@@_last_line_of_group_int }
1040         &&
1041         \int_compare_p:n { \l_@@_pos_arrow_int != 7 }
1042         ||

```

```

1043     \str_if_eq_p:Vn \l_@@_status_arrow_str { new-group }
1044   )
1045 }
1046 {
1047   \int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
1048   {
1049     \@@_draw_arrows:nn
1050     \l_@@_first_arrow_of_group_int
1051     { \l_@@_arrow_int - 1 }
1052   }
1053   \bool_set_true:N \l_@@_new_group_bool
1054 }

```

The flag `\l_@@_new_group_bool` indicates if we have to begin a new group of arrows. In fact, we have to begin a new group in 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: `\l_@@_first_arrow_int`, `\l_@@_first_line_of_group_int`, `\l_@@_last_line_of_group`, `\l_@@_first_arrows_seq`, `\l_@@_last_arrows_seq`.

```

1055   \bool_if:nTF \l_@@_new_group_bool
1056   {
1057     \bool_set_false:N \l_@@_new_group_bool
1058     \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1059     \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1060     \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1061     \seq_clear:N \l_@@_first_arrows_seq
1062     \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
1063     \seq_clear:N \l_@@_last_arrows_seq
1064     \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int

```

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.

```

1065     \int_compare:nT { \l_@@_pos_arrow_int != 8 }
1066     { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1067   }

```

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

```

1068   {

```

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

```

1069     \bool_if:nF
1070     { \str_if_eq_p:Vn \l_@@_status_arrow_str { independent } }
1071     {

```

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

```

1072     \int_compare:nT
1073     { \l_@@_initial_int = \l_@@_first_line_of_group_int }
1074     { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1075     \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1076     {
1077       \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1078       \seq_clear:N \l_@@_last_arrows_seq
1079       \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
1080     }
1081     {
1082       \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1083       { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1084     }

```

```

1085     }
1086 }

```

If the arrow is not independent, we update the current x -value (in `\l_@@_x_dim`) with the dedicated command `\@@_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.

```

1087   \bool_if:nF { \str_if_eq_p:Vn \l_@@_status_arrow_str { independent } }
1088   {
1089     \int_compare:nT { \l_@@_pos_arrow_int != 8 }
1090     { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1091   }
1092 }

```

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

```

1093 \cs_generate_variant:Nn \keys_set:nn { n o }
1094 \cs_new_protected:Npn \@@_keys_set:
1095   { \keys_set_known:no { WithArrows / Arrow / SecondPass } }

```

The macro `\@@_draw_arrows:nn` draws all the arrows whose numbers are between `#1` and `#2`. `#1` and `#2` must be expressions that expands to an integer (they are expanded in the beginning of the macro). This macro is nullified by the option `no-arrows`.

```

1096 \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1097   {
1098     \group_begin:
1099     \int_zero_new:N \l_@@_first_arrow_int
1100     \int_set:Nn \l_@@_first_arrow_int { #1 }
1101     \int_zero_new:N \l_@@_last_arrow_int
1102     \int_set:Nn \l_@@_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.

```

1103   \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1104   \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1105   {

```

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.

```

1106     \prop_get:cnN
1107     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1108     { initial } \l_tmpa_tl
1109     \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1110     \prop_get:cnN
1111     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1112     { final } \l_tmpa_tl
1113     \int_set:Nn \l_@@_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).

```

1114     \int_compare:nT { \l_@@_final_int <= \g_@@_line_int } \@@_draw_arrows_i:
1115     \int_incr:N \l_@@_arrow_int
1116   }
1117   \group_end:
1118 }

```

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).

```

1119 \cs_new_protected:Npn \@@_draw_arrows_i:
1120   {
1121     \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.

```

1122   \prop_get:cnN
1123     { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1124     { options } \l_tmpa_tl
1125   \str_clear_new:N \l_@@_previous_key_str
1126   \exp_args:NNo \exp_args:No
1127   \@@_keys_set: { \l_tmpa_tl , tikz = { xshift = \l_@@_xoffset_dim } }

```

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

```

1128   \bool_set_false:N \l_@@_initial_r_bool
1129   \bool_set_false:N \l_@@_final_r_bool
1130   \int_case:nn \l_@@_pos_arrow_int
1131     {
1132       0 { \bool_set_true:N \l_@@_final_r_bool }
1133       2 { \bool_set_true:N \l_@@_initial_r_bool }
1134       3
1135         {
1136           \bool_set_true:N \l_@@_initial_r_bool
1137           \bool_set_true:N \l_@@_final_r_bool
1138         }
1139     }

```

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 `CodeAfter` (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).

```

1140   \int_compare:nNnT \l_@@_pos_arrow_int = 5
1141     {
1142       \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1143       \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1144     }

```

`\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`.

```

1145   \tl_set:Nx \l_@@_initial_tl
1146     {
1147       \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl
1148       .south
1149     }
1150   \tl_set:Nx \l_@@_final_tl
1151     { \int_use:N \l_@@_final_int - \bool_if:NTF \l_@@_final_r_bool rl .north }

```


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

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

```

1152   \prop_get:cnN
1153     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1154     { label }
1155     \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:NxTF` which is *not* expandable.

```

1156   \seq_if_in:NxTF \l_@@_first_arrows_seq
1157     { \int_use:N \l_@@_arrow_int }
1158     { \bool_set_true:N \l_tmpa_bool }
1159     { \bool_set_false:N \l_tmpa_bool }
1160   \seq_if_in:NxTF \l_@@_last_arrows_seq
1161     { \int_use:N \l_@@_arrow_int }
1162     { \bool_set_true:N \l_tmpb_bool }
1163     { \bool_set_false:N \l_tmpb_bool }
1164   \int_compare:nNnT \l_@@_pos_arrow_int = 5
1165     {
1166       \bool_set_true:N \l_tmpa_bool
1167       \bool_set_true:N \l_tmpb_bool
1168     }

```

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`).

```

1169   \begin { tikzpicture } [ @@_standard ]
1170     \tikz@scan@one@point \pgfutil@firstofone ( \l_@@_initial_tl )
1171     \tl_gset:Nx \g_tmpa_tl
1172     {
1173       \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1174         { \dim_use:N \pgf@x }
1175         { \dim_use:N \l_@@_x_dim } ,
1176       \bool_if:NTF \l_tmpa_bool
1177         { \dim_eval:n { \pgf@y + \l_@@_start_adjust_dim } }
1178         { \dim_use:N \pgf@y }
1179     }
1180     \tikz@scan@one@point \pgfutil@firstofone ( \l_@@_final_tl )
1181     \tl_gset:Nx \g_tmpb_tl
1182     {
1183       \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1184         { \dim_use:N \pgf@x }
1185         { \dim_use:N \l_@@_x_dim } ,
1186       \bool_if:NTF \l_tmpb_bool
1187         { \dim_eval:n { \pgf@y - \l_@@_end_adjust_dim } }
1188         { \dim_use:N \pgf@y }
1189     }
1190   \end { tikzpicture }

```

Eventually, we can draw the arrow with the code in `\l_@@_tikz_code_tl`. We recall that the value by default for this token list is: “`\draw (#1) to node {#3} (#2) ;`”. This value can be modified with the option `TikzCode`. 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*.³¹

```
1191 \@@_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).

```
1192 \group_end:
1193 }
```

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`.

```
1194 \cs_new_protected:Npn \@@_def_function_tmpa:n #1
1195 {
1196   \cs_set:Npn \@@_tmpa:nnn ##1 ##2 ##3
1197   {
1198     \begin{tikzpicture}
1199     [
1200       @@_standard ,
1201       every~path / .style = WithArrows / arrow
1202     ]
1203     #1
1204     \end{tikzpicture}
1205   }
1206 }
```

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

```
1207 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1208 {
```

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

```
1209 \bool_if:nT { \l_@@_wrap_lines_bool && \l_@@_in_DispWithArrows_bool }
1210 { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }
```

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

```
1211 \exp_args:NV \@@_def_function_tmpa:n \l_@@_tikz_code_tl
1212 \@@_tmpa:nnn { #1 } { #2 } { #3 }
1213 }
1214 \cs_generate_variant:Nn \@@_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 `TikzCode`).

```
1215 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1216 {
```

First, we draw the arrow without the label.

```
1217 \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;
```

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

```
1218 \tikz@parse@node \pgfutil@firstofone ( @@_label.west )
```

³¹There 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).

We compute in `\l_tmpa_dim` the maximal width possible for the label. `0.3333 em` is the default value of `inner sep` in the nodes of Tikz. Maybe we should put the exact Tikz parameter. Here is the use of `\g_@@_right_x_dim` which has been computed previously with the `v`-nodes.

```
1219 \dim_set:Nn \l_tmpa_dim { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “text width”.³²

```
1220 \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`).

```
1221 \tl_if_empty:NF \g_tmpa_tl
1222 {
1223   \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1224   \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1225     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1226 }
```

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

```
1227 \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1228 {
1229   \path ( @@_label.west )
1230     node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1231       { #3 } ;
1232 }
1233 }
```

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`).

```
1234 \cs_new_protected:Npn \@@_update_x:nn #1 #2
1235 {
1236   \int_step_inline:nnn { #1 } { #2 }
1237   {
1238     \begin { tikzpicture } [ @@_standard ]
1239       \tikz@scan@one@point \pgfutil@firstofone ( ##1 - 1 )
1240       \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \l_@@_x_dim \pgf@x }
1241     \end { tikzpicture }
1242     \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1243   }
1244 }
```

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.

```
1245 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
```

10.12 The command `Arrow` in `CodeAfter`

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

```
1246 \keys_define:nn { WithArrows / Arrow / CodeAfter }
1247 {
1248   tikz .code:n =
1249     \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
```

³²In 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.

```

1250 tikz .value_required:n = true ,
1251 rr .value_forbidden:n = true ,
1252 rr .code:n = \@@_fix_pos_option:n 0 ,
1253 ll .value_forbidden:n = true,
1254 ll .code:n = \@@_fix_pos_option:n 1 ,
1255 rl .value_forbidden:n = true ,
1256 rl .code:n = \@@_fix_pos_option:n 2 ,
1257 lr .value_forbidden:n = true ,
1258 lr .code:n = \@@_fix_pos_option:n 3 ,
1259 v .value_forbidden:n = true ,
1260 v .code:n = \@@_fix_pos_option:n 4 ,
1261 TikzCode .tl_set:N = \l_@@_tikz_code_tl ,
1262 TikzCode .value_required:n = true ,
1263 xoffset .dim_set:N = \l_@@_xoffset_dim ,
1264 xoffset .value_required:n = true ,
1265 unknown .code:n =
1266 \@@_sort_seq:N \l_@@_options_Arrow_CodeAfter_seq
1267 \@@_error:n { Unknown-option-Arrow-in-CodeAfter }
1268 }

```

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

```

1269 \seq_set_from_clist:Nn \l_@@_options_Arrow_CodeAfter_seq
1270 { ll, lr, rl, rr, tikz, TikzCode, v, x, offset }

1271 \NewDocumentCommand \@@_Arrow_code_after { 0 { } m m m ! 0 { } }
1272 {
1273   \int_set:Nn \l_@@_pos_arrow_int 1
1274   \str_clear_new:N \l_@@_previous_key_str
1275   \group_begin:
1276     \keys_set:nn { WithArrows / Arrow / CodeAfter }
1277     { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
1278     \bool_set_false:N \l_@@_initial_r_bool
1279     \bool_set_false:N \l_@@_final_r_bool
1280     \int_case:nn \l_@@_pos_arrow_int
1281     { 0
1282       {
1283         \bool_set_true:N \l_@@_initial_r_bool
1284         \bool_set_true:N \l_@@_final_r_bool
1285       }
1286     2 { \bool_set_true:N \l_@@_initial_r_bool }
1287     3 { \bool_set_true:N \l_@@_final_r_bool }
1288   }

```

We prevent drawing a arrow from a line to itself.

```

1289 \tl_if_eq:nnTF { #2 } { #3 }
1290 { \@@_error:nn { Both-lines-are-equal } { #2 } }

```

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

```

1291 {
1292   \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - 1 }
1293   { \@@_error:nx { Wrong-line-in-Arrow } { #2 } }
1294   {
1295     \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - 1 }
1296     { \@@_error:nx { Wrong-line-in-Arrow } { #3 } }
1297     {
1298       \int_compare:nNnTF \l_@@_pos_arrow_int = 4
1299       {
1300         \begin { tikzpicture } [ @@_standard ]
1301           \tikz@scan@one@point \pgfutil@firstofone (#2-1.south)
1302           \dim_set_eq:NN \l_tmpa_dim \pgf@x
1303           \dim_set_eq:NN \l_tmpb_dim \pgf@y
1304           \tikz@scan@one@point \pgfutil@firstofone (#3-1.north)

```

```

1305         \dim_set:Nn \l_tmpa_dim
1306         { \dim_max:nn \l_tmpa_dim \pgf@x }
1307         \tl_gset:Nx \g_tmpa_tl
1308         { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1309         \tl_gset:Nx \g_tmpb_tl
1310         { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1311     \end { tikzpicture }
1312 }
1313 {
1314     \begin { tikzpicture } [ @@_standard ]
1315         \tikz@scan@one@point \pgfutil@firstofone
1316         ( #2-\bool_if:NTF\l_@@_initial_r_bool rl .south )
1317         \tl_gset:Nx \g_tmpa_tl
1318         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1319         \tikz@scan@one@point \pgfutil@firstofone
1320         ( #3-\bool_if:NTF\l_@@_final_r_bool rl .north )
1321         \tl_gset:Nx \g_tmpb_tl
1322         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1323     \end { tikzpicture }
1324 }
1325 \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
1326 }
1327 }
1328 }
1329 \group_end:
1330 }

```

10.13 MultiArrow

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

```

1331 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
1332 {

```

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

```

1333     \foreach \x in { #1 }
1334     {
1335         \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
1336         { \@@_error:nx { Wrong-line-specification-in-MultiArrow } \x }
1337         { \clist_gput_right:Nx \g_tmpa_clist \x }
1338     }

```

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

```

1339     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
1340     { \@@_error:n { Too-small-specification-for-MultiArrow } }
1341     {
1342         \clist_sort:Nn \g_tmpa_clist
1343         {
1344             \int_compare:nTF { ##1 > ##2 }
1345             \sort_return_swapped:
1346             \sort_return_same:
1347         }

```

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

```

1348         \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...

```

1349         \clist_reverse:N \g_tmpa_clist
1350         \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).

```
1351 \exp_args:NV \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
1352 \begin { tikzpicture }
1353 [
1354   @@_standard ,
1355   every~path / .style = { WithArrows / arrow }
1356 ]
1357 \draw [ <-> ] ( [xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
1358   -- ++(5mm,0)
1359   -- node ( @@_label ) {
1360     ( [xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
1361     -- ( [xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south) ;
1362 \tikz@parse@node \pgfutil@firstofone ( @@_label.west)
1363 \dim_set:Nn \l_tmpa_dim { 20 cm }
1364 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
1365 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
1366 \bool_if:nT { \l_@@_wrap_lines_bool && \l_@@_in_DispWithArrows_bool }
1367 {
1368   \dim_set:Nn \l_tmpb_dim
1369     { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
1370   \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1371     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1372 }
1373 \path ( @@_label.west)
1374   node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
1375 \end{tikzpicture}
1376 }
1377 }
1378 \cs_new_protected:Npn \@@_MultiArrow_i:n #1
1379 {
1380   \begin {tikzpicture }
1381     [
1382       @@_standard ,
1383       every~path / .style = { WithArrows / arrow }
1384     ]
1385     \foreach \k in { #1 }
1386     {
1387       \draw [ <- ]
1388         ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
1389     } ;
1390   \end { tikzpicture }
1391 }
```

10.14 The error messages of the package

```
1392 \str_const:Nn \c_@@_option_ignored_str
1393 { If~you~go~on,~this~option~will~be~ignored. }
1394 \@@_msg_new:nn { Value~for~a~key }
1395 {
1396   The~key~'\l_keys_key_tl'~should~be~used~without~value. \\
1397   However,~you~can~go~on~for~this~time.
1398 }
1399 \@@_msg_new:nnn { Unknown~option~in~Arrow }
1400 {
1401   The~option~'\l_keys_key_tl'~
1402   is~unknown~for~the~command~\l_@@_string_Arrow_for_msg_str\
```

```

1403   in~the~row~\int_use:N \g_@@_line_int\
1404   of~your~environment~\{\l_@@_type_env_str\}. \\\
1405   \c_@@_option_ignored_str \\\
1406   For~a~list~of~the~available~keys,~type-H~<return>.
1407   }
1408   {
1409   The~available~keys~are~(in~alphabetic~order):~
1410   \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
1411   }
1412 \@@_msg_new:nnn { Unknown~option~WithArrows }
1413   {
1414   The~option~'\l_keys_key_tl'~is~unknown~in~\{\l_@@_type_env_str\}. \\\
1415   \c_@@_option_ignored_str \\\
1416   For~a~list~of~the~available~keys,~type-H~<return>.
1417   }
1418   {
1419   The~available~keys~are~(in~alphabetic~order):~
1420   \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.
1421   }
1422 \@@_msg_new:nnn { Unknown~option~DispWithArrows }
1423   {
1424   The~option~'\l_keys_key_tl'~is~unknown~in~\{\l_@@_type_env_str\}. \\\
1425   \c_@@_option_ignored_str \\\
1426   For~a~list~of~the~available~keys,~type-H~<return>.
1427   }
1428   {
1429   The~available~keys~are~(in~alphabetic~order):~
1430   \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.
1431   }
1432 \@@_msg_new:nnn { Unknown~option~WithArrowsOptions }
1433   {
1434   The~option~'\l_keys_key_tl'~is~unknown~in~
1435   \token_to_str:N \WithArrowsOptions. \\\
1436   \c_@@_option_ignored_str \\\
1437   For~a~list~of~the~available~keys,~type-H~<return>.
1438   }
1439   {
1440   The~available~keys~are~(in~alphabetic~order):~
1441   \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
1442   }
1443 \@@_msg_new:nnn { Unknown~option~Arrow~in~CodeAfter }
1444   {
1445   The~option~'\l_keys_key_tl'~is~unknown~in~
1446   \token_to_str:N \Arrow\
1447   in~\token_to_str:N \CodeAfter. \\\
1448   \c_@@_option_ignored_str \\\
1449   For~a~list~of~the~available~keys,~type-H~<return>.
1450   }
1451   {
1452   The~available~keys~are~(in~alphabetic~order):~
1453   \seq_use:Nnnn \l_@@_options_Arrow_CodeAfter_seq {~and~} {,~} {~and~}.
1454   }
1455 \@@_msg_new:nn { Third~column~in~WithArrows }
1456   {
1457   By~default,~an~environment~\{\l_@@_type_env_str\}~can~only~have~
1458   two~columns.~Maybe~you~have~forgotten~a~
1459   \c_backslash_str\c_backslash_str.~If~you~really~want~more~than~
1460   two~columns,~you~should~use~the~option~'more~columns'~at~a~global~
1461   level~or~for~an~environment. \\\
1462   However,~you~can~go~one~for~this~time.
1463   }

```

```

1464 \@@_msg_new:nn { Third-column-in-DispWithArrows }
1465 {
1466   An-environment-\{\l_@@_type_env_str\}-can-only-have-two-columns.~
1467   Maybe-you-have-forgotten-a-\c_backslash_str\c_backslash_str\
1468   at-the-end-of-row~\int_use:N \g_@@_line_int. \\\
1469   If-you-go-on,~you-may-have-other-errors.
1470 }
1471 \@@_msg_new:nn { Negative-jump }
1472 {
1473   You-can't-use-a-negative-value-for-the-option-'jump'-of-command-
1474   \l_@@_string_Arrow_for_msg_str\
1475   in-the-row~\int_use:N \g_@@_line_int\
1476   of-your-environment-\{\l_@@_type_env_str\}.~
1477   You-can-create-an-arrow-going-backwards-with-the-option-'<'-of-Tikz. \\\
1478   \c_@@_option_ignored_str
1479 }
1480 \@@_msg_new:nn { new-group-without-groups }
1481 {
1482   You-can't-use-the-option-'new-group'~for-the-command-
1483   \l_@@_string_Arrow_for_msg_str\
1484   because-you-are-not-in-'groups'~mode.~Try-to-use-the-option-
1485   'groups'~in-your-environment-\{\l_@@_type_env_str\}. \\\
1486   \c_@@_option_ignored_str
1487 }
1488 \@@_msg_new:nn
1489 { Too-few-lines-for-an-arrow }
1490 { Line~\l_@@_input_line_str\
1491   :~an-arrow-specified-in-the-row~\int_use:N \l_@@_initial_int\
1492   of-your-environment-\{\l_@@_type_env_str\}-can't-be-drawn~
1493   because-it-arrives-after-the-last-row-of-the-environment. \\\
1494   If-you-go-on,~this-arrow-will-be-ignored.
1495 }
1496 \@@_msg_new:nn { WithArrows-outside-math-mode }
1497 {
1498   The-environment-\{\l_@@_type_env_str\}-should-be-used-only-in-math-mode~
1499   like-the-environment-\{aligned\}-of-amsmath. \\\
1500   Nevertheless,~you-can-go-on.
1501 }
1502 \@@_msg_new:nn { DispWithArrows-in-math-mode }
1503 {
1504   The-environment-\{\l_@@_type_env_str\}-should-be-used-only-outside-math-
1505   mode-like-the-environment-\{align\}-of-amsmath. \\\
1506   If-you-go-on,~you-will-have-other-errors.
1507 }
1508 \@@_msg_new:nn { Incompatible-options-in-Arrow }
1509 {
1510   You-try-to-use-the-option~'\l_keys_key_tl'-but~
1511   this-option-is-incompatible-or-redundant-with-the-option~
1512   '\l_@@_previous_key_str'-set-in-the-same-command-
1513   \l_@@_string_Arrow_for_msg_str. \\\
1514   \c_@@_option_ignored_str
1515 }
1516 \@@_msg_new:nn { Incompatible-options }
1517 { You-try-to-use-the-option~'\l_keys_key_tl'-but~
1518   this-option-is-incompatible-or-redundant-with-the-option~
1519   '\l_@@_previous_key_str'-set-in-the-same-command-
1520   \bool_if:NT \l_@@_in_CodeAfter_bool
1521   {
1522     \l_@@_string_Arrow_for_msg_str\
1523     in-the-CodeAfter-of-your-environment-\{\l_@@_type_env_str\}
1524   }. \\\

```



```

1525     \c_@@_option_ignored_str
1526   }
1527 \@@_msg_new:nn { Arrow-in~first~column }
1528   {
1529     You~should~not~use~the~command~\l_@@_string_Arrow_for_msg_str\
1530     in~the~first~column~of~your~environment~\{\l_@@_type_env_str\}~
1531     but~only~in~the~second~column.  \\\
1532     However~you~can~go~on~for~this~time.
1533   }
1534 \@@_msg_new:nn { Wrong~line~in~Arrow }
1535   {
1536     The~specification~of~line~'#1'~you~use~in~the~command~
1537     \l_@@_string_Arrow_for_msg_str\
1538     in~the~'CodeAfter'~of~\{\l_@@_type_env_str\}~doesn't~exist.  \\\
1539     If~you~go~on,~this~command~will~be~ignored.
1540   }
1541 \@@_msg_new:nn { Both~lines~are~equal }
1542   {
1543     In~the~'CodeAfter'~of~\{\l_@@_type_env_str\}~you~try~to~
1544     draw~an~arrow~going~to~it~self~from~the~line~'#1'.~This~is~not~possible.  \\\
1545     If~you~go~on,~this~command~will~be~ignored.
1546   }
1547 \@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }
1548   {
1549     The~specification~of~line~'#1'~doesn't~exist.  \\\
1550     If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
1551   }
1552 \@@_msg_new:nn { Too~small~specification~for~MultiArrow }
1553   {
1554     The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
1555     is~too~small:~you~need~at~least~two~lines.  \\\
1556     If~you~go~on,~this~command~will~be~ignored.
1557   }
1558 \@@_msg_new:nn { tag*~without~amsmath }
1559   {
1560     We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
1561     (or~mathtools).  \\\
1562     If~you~go~on,~the~command~\token_to_str:N\tag\
1563     will~be~used~instead.
1564   }
1565 \@@_msg_new:nn { Not~allowed~in~DispWithArrows }
1566   {
1567     The~command~\token_to_str:N #1
1568     is~not~allowed~in~the~first~column~of~\{\l_@@_type_env_str\}~but~
1569     only~in~the~second~column.  \\\
1570     If~you~go~on,~this~command~will~be~ignored.
1571   }
1572 \@@_msg_new:nn { Not~allowed~in~WithArrows }
1573   {
1574     The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
1575     (it's~allowed~in~the~second~column~of~\{DispWithArrows\}).  \\\
1576     If~you~go~on,~this~command~will~be~ignored.
1577   }
1578 \@@_msg_new:nn { Multiple~tags }
1579   {
1580     You~can't~use~twice~the~command~\token_to_str:N\tag\
1581     in~a~line~of~the~environment~\{\l_@@_type_env_str\}.  \\\
1582     If~you~go~on,~the~tag~'#1'~will~be~used.
1583   }

```

```

1584 \@@_msg_new:nn { Multiple~labels }
1585   {
1586     Normally,~we~can't~use~the~command~\token_to_str:N\label\
1587     twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\\
1588     However,~you~can~go~on.~
1589     \bool_if:NT \c_@@_showlabels_loaded_bool
1590     { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
1591     If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
1592     'allow-multiple-labels'~at~the~global~or~environment~level.
1593   }

1594 \@@_msg_new:nn { Multiple~labels~with~cleveref }
1595   {
1596     Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
1597     twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\\
1598     If~you~go~on,~you~may~have~undefined~references.
1599   }

1600 \@@_msg_new:nn { Inexistent~v~node }
1601   {
1602     There~is~a~problem.~Maybe~you~have~put~a~command~\token_to_str:N\cr\
1603     instead~of~a~command~\token_to_str:N\\~at~the~end~of~
1604     the~row~\int_use:N \l_tmpa_int\
1605     of~your~environment~\{\l_@@_type_env_str\}. \\\
1606     If~you~go~on,~you~may~have~an~incorrect~output.
1607   }

1608 \@@_msg_new:nn { Option~xoffset~forbidden }
1609   {
1610     You~can't~use~the~option~'xoffset'~in~the~command~
1611     \l_@@_string_Arrow_for_msg_str\
1612     while~you~are~using~the~option~
1613     ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
1614     { group }
1615     { groups } '. \\\
1616     \c_@@_option_ignored_str
1617   }

```

10.15 The command `WithArrowsNewStyle`

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

```

1618 \NewDocumentCommand \WithArrowsNewStyle { m m }
1619   {
1620     \keys_if_exist:nnTF { WithArrows / Global } { #1 }
1621     { \@@_error:nn { Key~already~defined } { #1 } }
1622     {
1623       \keys_define:nn { WithArrows / Global }
1624       {
1625         #1 .code:n =
1626         { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
1627       }
1628       \seq_put_right:Nn \l_@@_options_WithArrows_seq {#1}
1629       \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { #1 }
1630       \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { #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.

```

1631     \group_begin:
1632     \msg_set:nnn { witharrows } { Unknown~option~WithArrowsOptions }
1633     {
1634       The~option~'\l_keys_key_tl'~can't~be~set~in~the~
1635       definition~of~a~style.~You~can~go~on~for~this~time~
1636       but~you~should~suppress~this~key.
1637     }
1638     \WithArrowsOptions { #2 }
1639     \group_end:

```

```

1640     }
1641   }
1642 \@@_msg_new:nn { Key~already~defined }
1643   {
1644     The~key~'#1'~is~already~defined. \\
1645     If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
1646     will~be~ignored.
1647   }

```

10.16 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 if he uses the option `up` or the option `down`.

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

```

1648 \tl_const:Nn \c_@@_tikz_code_up_tl
1649   {
1650     \draw [ rounded~corners ]
1651       let \p1 = (#1) ,
1652         \p2 = (#2)
1653       in (\p1) -- node {
1654           \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
1655           \begin { varwidth } \l_tmpa_dim
1656             \raggedright
1657             #3
1658           \end { varwidth }
1659         }
1660       (\x2,\y1) -- (\p2) ;
1661   }

```

Idem for the option `down`.

```

1662 \tl_const:Nn \c_@@_tikz_code_down_tl
1663   {
1664     \draw [ rounded~corners ]
1665       let \p1 = (#1) ,
1666         \p2 = (#2)
1667       in (\p1) -- (\x1,\y2) --
1668         node {
1669           \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
1670           \begin { varwidth } \l_tmpa_dim
1671             \raggedright
1672             #3
1673           \end { varwidth }
1674         }
1675       (\p2) ;
1676   }

```

```

1677 \keys_define:nn { WithArrows / Arrow / FirstPass }
1678   {
1679     up .code:n = \@@_set_independent: ,
1680     down .code:n = \@@_set_independent: ,
1681     up .default:n = NoValue ,
1682     down .default:n = NoValue
1683   }

```

```

1684 \keys_define:nn { WithArrows / Arrow / SecondPass }
1685   {
1686     up .code:n =
1687       \str_if_empty:NT \l_@@_previous_key_str

```

```

1688     {
1689         \str_set:Nn \l_@@_previous_key_str { up }
1690         \bool_if:NTF \c_@@_varwidth_loaded_bool
1691         {
1692             \cs_if_exist:cTF { tikz@library@calc@loaded }
1693             {
1694                 \int_set:Nn \l_@@_pos_arrow_int \c_one_int

```

We have to set `\l_@@_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 `TikzCode` that will overwrite our affectation.

```

1695                 \bool_set_false:N \l_@@_wrap_lines_bool
1696                 \tl_set_eq:NN \l_@@_tikz_code_tl
1697                 \c_@@_tikz_code_up_tl
1698             }
1699             { \@@_error:n { calc~not~loaded } }
1700         }
1701         { \@@_error:n { varwidth~not~loaded } }
1702     } ,
1703     down .code:n =
1704     \str_if_empty:NT \l_@@_previous_key_str
1705     {
1706         \str_set:Nn \l_@@_previous_key_str { down }
1707         \bool_if:NTF \c_@@_varwidth_loaded_bool
1708         {
1709             \cs_if_exist:cTF { tikz@library@calc@loaded }
1710             {
1711                 \int_set:Nn \l_@@_pos_arrow_int \c_one_int
1712                 \bool_set_false:N \l_@@_wrap_lines_bool
1713                 \tl_set_eq:NN \l_@@_tikz_code_tl
1714                 \c_@@_tikz_code_down_tl
1715             }
1716             { \@@_error:n { calc~not~loaded } }
1717         }
1718         { \@@_error:n { varwidth~not~loaded } }
1719     }
1720 }

1721 \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
1722 \seq_put_right:Nn \l_@@_options_Arrow_seq { up }

1723 \@@_msg_new:nn { varwidth~not~loaded }
1724 {
1725     You~can't~use~the~option~'\l_keys_key_tl'~because~
1726     you~don't~have~loaded~the~package~'varwidth'. \\\
1727     \c_@@_option_ignored_str
1728 }

1729 \@@_msg_new:nn { calc~not~loaded }
1730 {
1731     You~can't~use~the~option~'\l_keys_key_tl'~because~you~don't~have~loaded~the~
1732     Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'
1733     ~in~your~preamble. \\\
1734     \c_@@_option_ignored_str
1735 }

1736 \@@_msg_new:nnn { Duplicate~name }
1737 {
1738     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
1739     the~same~environment~name~twice.~You~can~go~on,~but,~
1740     maybe,~you~will~have~incorrect~results. \\\
1741     For~a~list~of~the~names~already~used,~type~H~<return>. \\\
1742     If~you~don't~want~to~see~this~message~again,~use~the~option~
1743     'allow~duplicate~names'.
1744 }
1745 {

```

```

1746   The~names~already~defined~in~this~document~are:~
1747   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
1748   }

```

11 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`)
 Better error messages

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 `TikzCode`.
 Two new options `CodeBefore` and `CodeAfter` have been added at the environment level.
 A special version of `\Arrow` is available in `CodeAfter` in order to draw arrows in nested environments.
 A command `\MultiArrow` is available in `CodeAfter` to draw arrows of other shapes.

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}`.
 In the version 1.6.1, correction of a bug that leads to incompatibility with `\usetikzlibrary{babel}`.

Changes between 1.6.1 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 et 1.14

New options `up` and `down` for the arrows.

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because 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 et 1.15

Option `new-group` to start a new group of arrows (only available when the environment is composed with the option the option `groups`).

Tikz externalization is now deactivated in the environments of the extension `witharrows`.³³

Changes between 1.15 et 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`.

³³Before 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`.