

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

F. Pantigny  
fpantigny@wanadoo.fr

October 19, 2024

## 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.

The package `witharrows` is entirely contained in the file `witharrows.sty`. This file may be put in the current directory or in a `texmf` tree. However, the best is to install `witharrows` with a TeX distribution such as MiKTeX, TeX Live or MacTeX.

In fact, `witharrows` may also be used with plain-TeX and, in that case, the only required file is `witharrows.tex`: see p. 23. In what follows, we describe the LaTeX package.

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). However, the file `witharrows.dtx` of the present documentation should be compiled with `LuaLaTeX`. This package loads the packages `l3keys2e`, `varwidth`, `tikz` and the Tikz libraries `arrows.meta` and `bending`. The final user only has to load the package with the classical instruction: `\usepackage{witharrows}`.

The arrows are drawn with Tikz and that's why **several compilations may be necessary**.<sup>1</sup>

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 % <----- don't put \\ here
\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 two environments `{DispWithArrows}` and `{DispWithArrows*}` which are similar to the environments `{align}` and `{align*}` of the package `amsmath`: cf. p. 17.

---

\*This document corresponds to the version 2.9a of `witharrows`, at the date of 2024/10/19.

<sup>1</sup>If you use Overleaf, Overleaf will do automatically a number compilations sufficient (by using `latexmk`).

# 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 number<sup>2</sup> of rows the arrow must jump (the default value is, of course, 1).

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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.

```

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

```

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

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

---

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

In fact, it's possible to change more drastically the shape or the arrows with the option `tikz-code` (presented p. 24).

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

```

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

```

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

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. 20.

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

```

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

```

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

It's possible to put commands `\` in the text to force new lines<sup>3</sup>. 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 provides 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.

```

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

```

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

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

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

```

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

```

$$A = ((a + b) + 1)^2 \\ = (a + b)^2 + 2(a + b) + 1 \quad \left. \vphantom{A} \right\} \textit{first expansion.} \\ = a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \left. \vphantom{A} \right\} \textit{second expansion.}$$

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}

```

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

<sup>3</sup>By 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.

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

The same example with the option `displaystyle`:

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

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}`.<sup>5</sup>

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

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

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

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

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

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

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

---

<sup>5</sup>It’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 = {}}}`.

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

Special commands are available in `code-after`: a command `\WithArrowsNbLines` which gives the number of lines (=rows) of the current environment (this is a command and not a counter), a special form of the command `\Arrow` and the command `\MultiArrow`: the latter two commands are described in the section concerning the nested environments, p. 14.

## 2 Numbers of columns

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

The letters `R`, `C` and `L` add empty groups `{}` which provide correct spaces when these columns contain symbols with the type `\mathrel` (such as `=`, `≤`, etc.) or `\mathbin` (such as `+`, `×`, etc.). This system is inspired by the environment `{IEEEeqnarray}` of the package `IEEEtrantools`.

The initial value of the parameter `format` is, in fact, `rL`.

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

```

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

```

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

In the following example, we use five columns all centered (the environment `{DispWithArrows*}` is presented p. 17).

```

 $\begin{DispWithArrows*}[format = cCcCc,
wrap-lines,
interline=1mm]
k & \leq & t & \leq & k+1 \\
\frac{1}{k+1} & \leq & \frac{1}{t} & \leq & \frac{1}{k} \\
\int_k^{k+1} \frac{dt}{k+1} & \leq & \int_k^{k+1} \frac{dt}{t} & \leq & \int_k^{k+1} \frac{dt}{k} \\
& \leq & \ln(k+1) - \ln(k) & \leq & \frac{1}{k}
\end{DispWithArrows*}$ 

```

$$\begin{aligned}
 k &\leq t \leq k+1 \\
 \frac{1}{k+1} &\leq \frac{1}{t} \leq \frac{1}{k} \\
 \int_k^{k+1} \frac{dt}{k+1} &\leq \int_k^{k+1} \frac{dt}{t} \leq \int_k^{k+1} \frac{dt}{k} \\
 \frac{1}{k+1} &\leq \ln(k+1) - \ln(k) \leq \frac{1}{k}
 \end{aligned}
 \left. \vphantom{\int} \right\} \textit{we can integrate the inequalities since } k \leq k+1$$

### 3 Precise positioning of the arrows

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

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

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.

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

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

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

---

<sup>6</sup>The 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) \ \backslash
& = (a^2-b^2)(a^2+b^2) \ \Arrowleft[i]{because \ $(x-y)(x+y)=x^2-y^2$}\ \backslash
& = a^4-b^4
\end{WithArrows}

```

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

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}
& \Longleftarrow 2x(K'y_0+Ky_0')-3Ky_0 = \sqrt{x} \ \backslash
& \Longleftarrow 2xK'y_0 + K(2xy_0'-3y_0) = \sqrt{x} \ \backslash
& \Longleftarrow 2x K'y_0 = \sqrt{x} \ \Arrowleft{...}\ \backslash
...
\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 connected<sup>7</sup> 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`, `r1`, `rr` or `i`) to an individual arrow<sup>8</sup>. 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]`

<sup>7</sup>More 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 ~.

<sup>8</sup>Such arrow will be called *independent* in the technical documentation





Arrows of type `o` may themselves be covered by other arrows of type `o`.

```

 $\begin{WithArrows}[groups]
A & = B \Arrow{one}\Arrow[o,jump=2]{two}\Arrow[o,jump=3]{three}\\
& = C \\
& = D \\
& = E + E + E + E + E + E + E
\end{WithArrows}$ 

```

The horizontal space between an arrow of type `o` and the arrows immediately covered is fixed by the dimension `xoffset-for-o-arrows` which can be set with the command `\WithArrowsOptions` (initial value: 2 mm).

## 5 The options “up” and “down” for individual arrows

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

```

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

```

The options `up` and `down` require the Tikz library `calc`. If it has not been previously loaded by the user, an error will be raised.

In fact, the options `up` and `down` may be used with a value which is a list of couples key-value.

- The key `radius` is the radius of the rounded corner of the arrow.<sup>10</sup>
- The key `width` is the width of the (horizontal part of) the arrow:

<sup>10</sup>The initial value of this parameter is 4 pt, which is the default value of the “rounded corners” of Tikz.

- with the value `max`, the width of the arrow is adjusted with respect of the position of the nodes (that’s the behaviour by default of the arrows `up` and `down` as shown in the previous example);
- with a numerical value, the width of the arrow is directly fixed to that numerical value;
- with the value `min`, the width of the arrow is adjusted with respect to the contents of the label of the arrow.

```

$\begin{WithArrows}
A & = B
\Arrow[up={radius=0pt,width=2cm}]{we try} \\
& = C + C + C + C + C + C + C + C + C
\end{WithArrows}$

```

$$\begin{array}{l}
 A = B \\
 = C + C + C + C + C + C + C + C + C
 \end{array}
 \begin{array}{l}
 \xrightarrow{\text{we try}} \\
 \downarrow
 \end{array}$$

```

$\begin{WithArrows}
A & = B
\Arrow[up={width=min}]{we try} \\
& = C + C + C + C + C + C + C + C + C
\end{WithArrows}$

```

$$\begin{array}{l}
 A = B \\
 = C + C + C + C + C + C + C + C + C
 \end{array}
 \begin{array}{l}
 \xrightarrow{\text{we try}} \\
 \downarrow
 \end{array}$$

The options relative to the arrows `up` and `down` can be fixed at the global or environment level with the key `up-and-down`. This key may also be used as prefix as illustrated now.

```

\WithArrowsOptions{up-and-down/width=min}

```

## 6 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}`.<sup>11</sup>

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

```

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

```

---

<sup>11</sup>In fact, it’s possible to use the package `witharrows` without the package `amsmath`.

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

In the environments of `amsmath` (or `mathtools`), the spacing between rows is fixed by a parameter called `\jot` (it's a dimension and not a skip). That's also the case for the environment `{WithArrows}`. An option `\jot` has been given to the environment `{WithArrows}` in order to change the value of this parameter `\jot` for a given environment.<sup>12</sup>

```


$$\begin{aligned}
 F &= \frac{1}{2}G \\
 &= H + \frac{1}{2}K \\
 &= K
 \end{aligned}$$


```

$$\begin{aligned}
 F &= \frac{1}{2}G \\
 &= H + \frac{1}{2}K \\
 &= K
 \end{aligned}
 \left. \vphantom{\begin{aligned} F \\ = H + \frac{1}{2}K \\ = K \end{aligned}} \right) \textit{we expand}$$

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

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

```


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


```

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

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

```


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


```

---

<sup>12</sup>It's also possible to change `\jot` with the environment `{spreadlines}` of `mathtools`.

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

$$\Leftrightarrow \begin{cases} x + y = 0 \\ x + 2y = 0 \end{cases} \quad \left. \vphantom{\begin{cases} x + y = 0 \\ x + 2y = 0 \end{cases}} \right\} x \text{ and } y \text{ are real}$$

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

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

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

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

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

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

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

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

`\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.

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

## 7 Arrows in nested environments

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

```

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

```

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

```

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

The package `witharrows` provides also another command available only in `code-after`: the command `\MultiArrow`. This command draws a “rak”. The list of the rows of the environment concerned by this rak are given in the first argument of the command `\MultiArrow`. This list is given with the syntax of the list in a `\foreach` command of `pgffor`.

```

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





```

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

```

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

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

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

```

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

```

```

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

```

```

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

```

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

## 9 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}

```

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

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}

```

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

A link to the equation [\( \$\star\$ \)](#).<sup>15</sup>

If `amsmath` (or `mathtools`) is loaded, it's also possible to use `\tag*` which, as in `amsmath`, typesets the tag without the parentheses. 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.<sup>16</sup>

```

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

```

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

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.<sup>17</sup>

```

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

```

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

<sup>15</sup>In this document, the references have been customized with `\labelformat{equation}{(#1)}` in the preamble.

<sup>16</sup>Notice that the environment `{DispWithArrows}` is compatible with the command `\qedhere` of `amsthm`.

<sup>17</sup>Even in this case, it's possible to put a "manual tag" with the command `\tag`.

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

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

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

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 initial value of this parameter is 25 pt. It's possible to use as value for that key a *skip* (*=glue*).

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

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

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

If the option `leqno` is used as a class option, the labels will be composed on the left also for the environments `{DispWithArrows}` and `{DispWithArrows*}`.<sup>18</sup>

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

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

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

---

<sup>18</sup>The package `amsmath` has an option `leqno` but `witharrows`, of course, is not aware of that option: `witharrows` only checks the option `leqno` of the document class.

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

First environment.

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

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

Second environment.

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

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

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

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

$$S_{2(p+1)} = \sum_{k=1}^{2(p+1)} (-1)^k k^2 \tag{8}$$

$$= \sum_{k=1}^{2p} (-1)^k k^2 + (-1)^{2p+1} (2p+1)^2 + (-1)^{2p+2} (2p+2)^2 \tag{9}$$

$$= S_{2p} - (2p+1)^2 + (2p+2)^2 \tag{10}$$

$$= 2p^2 + p - 4p^2 - 4p - 1 + 4p^2 + 8p + 4 \tag{11}$$

$$= 2p^2 + 5p + 3 \tag{12}$$

The environments `{DispWithArrows}` and `{DispWithArrows*}` provide an option `wrap-lines`. With this option, the lines of the labels are automatically wrapped on the right.<sup>2</sup>

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

$$\begin{aligned}
S_n &= \frac{1}{n} \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac{1}{n} \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{aligned}
\left. \begin{array}{l} \text{sum of terms of a geometric} \\ \text{progression of ratio } e^{i \frac{2\pi}{n}} \\ \text{This line has been wrapped} \\ \text{automatically.} \end{array} \right\}$$

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

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

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

```

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

```

$$\begin{aligned}
1. S_n &= \frac{1}{n} \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac{1}{n} \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{aligned}
\left. \begin{array}{l} \text{we use the formula for a sum of terms of a geometric} \\ \text{progression of ratio } e^{i \frac{2\pi}{n}} \\ \left( e^{i \frac{\pi}{2n}} \right)^n = e^{i \frac{\pi}{2}} = i \end{array} \right\} \tag{13}$$

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

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

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

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

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

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

## 9.1 The option `<...>` of `DispWithArrows`

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

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

The following code is an example of multi-case equations.<sup>22</sup>

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

$$\binom{n}{p} = \begin{cases} 0 & \text{if } p > n \\ \frac{n(n-1)\cdots(n-p+1)}{p!} & \text{if } 0 \leq p \leq n \\ 0 & \text{if } p < 0 \end{cases} \begin{array}{l} \left. \begin{array}{l} \text{if fact, it's a special case} \\ \text{of the following one} \end{array} \right\} \end{array} \quad \begin{array}{l} (14) \\ (15) \\ (16) \end{array}$$

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

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

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

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

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

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

The whole system is the equation (17) (this reference has been coded by `\ref{system}`) whereas the last equation is the equation (17c) (this reference has been coded by `\ref{last-equation}`). The command `\ref*` used in the code above is a variant of the command `\ref` which does not create interactive link (even when `hyperref` is loaded).

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

## 10 Advanced features

### 10.1 Utilisation with Beamer

#### New 2.9

If `witharrows` is used with Beamer, the command `\Arrow` takes in as argument between angular brackets (after the optional argument in square brackets) to specify the *overlays* which are implied (internally, `witharrows` merely uses the command `\only` of Beamer).

```
\Arrow[jump=2]<3->{Example}
```

### 10.2 Use with plain-Tex

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

```
\input{witharrows}
```

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

```

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

```

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

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

The option `tikz-code` allows the user to change the shape of the arrows.<sup>23</sup>

For example, the options “up” and “down” described previously (cf. p. 10) are programmed internally with `tikz-code`.

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

The initial value is the following:

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

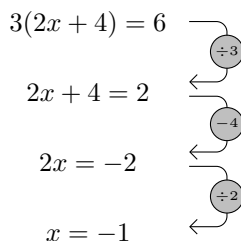
In the following example, we replace this default path by a path with three segments (and the node overwriting the second segment).

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

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

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

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



The environments `{DispWithArrows}` and its starred version `{DispWithArrows*}` provide a command `\WithArrowsRightX` which can be used in a definition of `tikz-code`. This command gives the  $x$ -value of the right side of the composition box (taking into account the eventual tags of the equations). For an example of use, see p. 30.

### 10.4 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.<sup>24</sup>

<sup>23</sup>If the option `wrap-lines` is used in an environment `{DispWithArrows}` or `{DispWithArrows*}`, the option `tikz-code` will have no effect for the arrows of this environment but only for the arrows in the nested environments `{WithArrows}`.

<sup>24</sup>We recall that, in particular, every LaTeX environment is a TeX group.



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. 30.

At this time, there is no style for individual arrows. However, it's, of course, possible to define new commands based upon the command `\Arrow`. For example :

```
\newcommand{\ThickArrow}{\Arrow[tikz=thick]}
```

This new command `\ThickArrow` still accepts options between square brackets. It's possible to write `\ThickArrow[jump=2]` because, in fact, `\Arrow[tikz=thick][jump=2]` is an allowed syntax for the command `\Arrow` (it's possible to put an arbitrary number of optional arguments between square brackets after `\Arrow`).

## 10.5 The key right-overlap

The key `right-overlap` is a boolean key whose initial value is `true`. It deals with the environments `{WithArrows}` only.

When the key `right-overlap` is in force, the arrows (and their labels) are drawn in an overlapping position and are not relevant for the computation of the dimensions of the TeX box containing the environment `{WithArrows}`.

When the key `right-overlap` is set to `false` (with `\WithArrowsOptions` or within an individual environment `{WithArrows}`), the overlapping on the right is taken into account in the dimensions of the encompassing box.

```

$ \left\{ \begin{array}{l} 2x + 3y = 5 \\ -2x - 5y = 2 \end{array} \right. \begin{array}{l} \text{we add } L_1 \text{ to } L_2 \\ \end{array} \left\{ \begin{array}{l} 2x + 3y = 5 \\ -2y = 7 \end{array} \right. \\
\end{array}

```

$$\left\{ \begin{array}{l} 2x + 3y = 5 \\ -2x - 5y = 2 \end{array} \right. \begin{array}{l} \text{we add } L_1 \text{ to } L_2 \\ \end{array} \left\{ \begin{array}{l} 2x + 3y = 5 \\ -2y = 7 \end{array} \right.$$

The tuning `right-overlap = false` may also be useful in conjunction with the class `standalone`.

## 10.6 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 (initial value: 0.4 ex);
- the option `ygap` sets the vertical distance between two consecutive arrows (initial 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`:

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

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

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

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

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

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

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

An arrow of jump equal to 1 has a maximal length<sup>25</sup> equal to the parameter `max-length-of-arrow`. The initial value of this parameter is 2 cm.

In the following example, the value of `max-length-of-arrow` has been fixed to 1.5 cm.

```
\[\begin{WithArrows}[max-length-of-arrow = 1.5cm]
A
& =
\begin{vmatrix}
1 & a & a^2 & a^3 & a^4 \\
1 & b & b^2 & b^3 & b^4 \\
1 & c & c^2 & c^3 & c^4 \\
1 & d & d^2 & d^3 & d^4 \\
1 & e & e^2 & e^3 & e^4
\end{vmatrix}
\Arrow{
$L_2 \ \gets L_2-L_1$ \\
$L_3 \ \gets L_3-L_1$ \\
$L_4 \ \gets L_4-L_1$ \\
$L_5 \ \gets L_5-L_1$ % don't put \\ here
} \\
& =
\begin{vmatrix}
1 & a & a^2 & a^3 & a^4 \\
0 & b-a & b^2-a^2 & b^3-a^3 & b^4-a^4 \\
0 & c-a & c^2-a^2 & c^3-a^3 & c^4-a^4 \\
0 & d-a & d^2-a^2 & d^3-a^3 & d^4-a^4 \\
0 & e-a & e^2-a^2 & e^3-a^3 & e^4-a^4
\end{vmatrix}
\end{WithArrows}\]
```

---

<sup>25</sup>We call *length* of an arrow the difference between the *y*-value of its start point and the *y* value of its end point.

$$\begin{aligned}
A &= \begin{vmatrix} 1 & a & a^2 & a^3 & a^4 \\ 1 & b & b^2 & b^3 & b^4 \\ 1 & c & c^2 & c^3 & c^4 \\ 1 & d & d^2 & d^3 & d^4 \\ 1 & e & e^2 & e^3 & e^4 \end{vmatrix} \\
&= \begin{vmatrix} 1 & a & a^2 & a^3 & a^4 \\ 0 & b-a & b^2-a^2 & b^3-a^3 & b^4-a^4 \\ 0 & c-a & c^2-a^2 & c^3-a^3 & c^4-a^4 \\ 0 & d-a & d^2-a^2 & d^3-a^3 & d^4-a^4 \\ 0 & e-a & e^2-a^2 & e^3-a^3 & e^4-a^4 \end{vmatrix} \begin{array}{l} \left. \begin{array}{l} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - L_1 \\ L_4 \leftarrow L_4 - L_1 \\ L_5 \leftarrow L_5 - L_1 \end{array} \right\} \end{array}
\end{aligned}$$

## 10.7 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:

$$\begin{aligned}
A &= (a + b)^2 \\
&= a^2 + b^2 + 2ab \quad \left. \vphantom{A} \right\} \textit{We expand}^{26}
\end{aligned}$$

## 10.8 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.

## 10.9 Note for the users of AUCTeX

In a editor of text with a LaTeX-oriented mode, the environments `{DispWithArrows}` and `{DispWithArrows*}` should be formatted like the environment `equation` of LaTeX, that is to say with a formatting adapted to the math mode of TeX.

In Emacs with the AUCTeX mode, it's possible to achieve such a customization by adding the strings "DispWithArrows" and "DispWithArrows\*" to the variable `font-latex-math-environments`. It's possible to do that with the "easy customization" interface of Emacs:

```
M-x customize > [Text] > [TeX] > [Font LaTeX]
```

---

<sup>26</sup>A footnote.

## 10.10 Note for the developers

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

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}
```

## 11 Examples

### 11.1 \MoveEqLeft

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

```

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

```

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

### 11.2 A command \DoubleArrow

By using the key `o` (cf. p. 9) available at the local level, it's easy to write a command `\DoubleArrow` for two arrows going in opposite directions.

```
\NewDocumentCommand \DoubleArrow { 0 {} m m }
{
  \Arrow[tikz=>,#1]{#2}%
  \Arrow[o,tikz=<-,#1]{#3}
}
```

Example of use:

```


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


```

$$\begin{aligned}
 A &= (a+b)^2 \\
 &= a^2 + 2ab + b^2
 \end{aligned}
 \begin{array}{l}
 \downarrow \text{expansion} \\
 \uparrow \text{factorization}
 \end{array}$$

### 11.3 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}%
  [format = c,
   interline = 4mm,
   tikz = {every node/.style = {circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt,
                                font = \tiny}}]
3 (2x+4) = 6 \Arrow{\div 3} \\
2x+4 = 2 \Arrow{\$-4\$} \\
2x = -2 \Arrow{\div 2} \\
2x = -1
\end{WithArrows}

```

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

### 11.4 Examples with the option tikz-code

We recall that the option `tikz-code` is the Tikz code used by `witharrows` to draw the arrows.<sup>27</sup>

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

---

<sup>27</sup>If an environment `{DispWithArrows}` or `{DispWithArrows*}` is used with the option `wrap-lines`, the value of the option `tikz-code` is not used for this environment (but is used for the environments nested inside).

### 11.4.1 Example 1

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

```
\begin{DispWithArrows*}%
  [displaystyle,
   ygap = 2mm,
   ystart = 0mm,
   tikz-code = {\draw (#1) -- ++(4.5cm,0) |- (#2) ;
                \path (#1) -- (#2)
                  node[text width = 4.2cm, right, midway] {#3} ;}]
S_n
& = \frac{1}{n} \sum_{k=0}^{n-1} \cos\bigl(\tfrac{\pi}{2}\cdot\tfrac{k}{n}\bigr)
.....
```

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

### 11.4.2 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 `tikz-code`, we use the command `\WithArrowsRightX` which is the  $x$ -value of the right margin of the current composition box (it's a TeX command and not a dimension). For lisibility, we use a style. This example requires the Tikz library `calc`.

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

```

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

```

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

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

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

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

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

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

### 11.4.3 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}[11,interline=5mm,xoffset=5mm,
  tikz-code = {\draw[rounded corners,
    every node/.style = {circle,
      draw,
      auto = false,
      inner sep = 1pt,
      fill = gray!50,
      font = \tiny }]}
  let \p1 = (~#1),
      \p2 = (~#2)
  in \ifdim \x1 > \x2
    (\p1) -- node {~#3} (\x1,\y2) -- (\p2)
  \else
    (\p1) -- (\x2,\y1) -- node {~#3} (\p2)
  \fi ;}]
E & \Longleftarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\Arrow{$\times 15$}\
& \Longleftarrow 5(x+4) + 3(5x+3) = 105 \
& \Longleftarrow 5x+20 + 15x+9 = 105 \
& \Longleftarrow 20x+29 = 105
\Arrow{$-29$}\
& \Longleftarrow 20x = 76
\Arrow{$\div 20$}\
& \Longleftarrow x = \frac{38}{10}
\end{WithArrows}

```

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

## 11.5 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 `code-after`. In the following example, we write this command with `\NewDocumentCommand` (of L3) and `\foreach` of `pgffor` (which is loaded when `witharrows` is loaded).

```

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

```

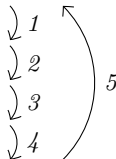
The command `\WithArrowsNbLines` is a command available in `code-after` which gives the total number of lines (=rows) of the current environment (it's a command and not a counter).

```

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

```

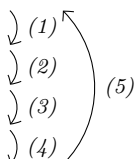
*a. f* est continuous on *E*  
*b. f* est continuous in 0  
*c. f* is bounded on the unit sphere  
*d.*  $\exists K > 0 \ \forall x \in E \ \|f(x)\| \leq K\|x\|$   
*e. f* is lipschitzian



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

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

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





## 12 Implementation

### 12.1 Declaration of the package and extensions loaded

The prefix `witharrows` has been registered for this extension.

See: <http://mirrors.ctan.org/macros/latex/contrib/l3kernel/l3prefixes.pdf>

<@@=witharrows>

First, `tikz` and some Tikz libraries are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in L3 code fails.<sup>28</sup>

```
1 <*LaTeX>
2 \RequirePackage{tikz}
3 </LaTeX>
4 <*plain-TeX>
5 \input tikz.tex
6 \input expl3-generic.tex
7 </plain-TeX>
8 \usetikzlibrary{arrows.meta}
9 \usepgfmodule{bending} % https://texnique.fr/osqa/questions/12199
```

Then, we can give the traditional declaration of a package written with L3:

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

17 \RequirePackage { varwidth }
18 </LaTeX>

19 <*plain-TeX>
20 \ExplSyntaxOn
21 \catcode ` \@ = 11
22 </plain-TeX>

23 <*LaTeX>
24 \ProvideDocumentCommand{\IfPackageLoadedT}{mm}
25   {\IfPackageLoadedTF{#1}{#2}{}}
26 \ProvideDocumentCommand{\IfPackageLoadedF}{mm}
27   {\IfPackageLoadedTF{#1}{-}{#2}}
28 </LaTeX>
```

### 12.2 The packages `footnote` and `footnotehyper`

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 2.9a), 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 `\c_@@_footnotehyper_bool` will indicate if the option `footnotehyper` is used.

```
29 <*LaTeX>
30 \bool_new:N \c_@@_footnotehyper_bool
```

---

<sup>28</sup>cf. [tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails](http://tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails)

The boolean `\c_@@_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.

```

31 \bool_new:N \c_@@_footnote_bool
32 </LaTeX>

33 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
34 \cs_new_protected:Npn \@@_msg_new:nnn #1 #2 #3
35 {
36   \bool_if:NTF \c_@@_messages_for_Overleaf_bool
37     { \msg_new:nnn { witharrows } { #1 } { #2 \ \ #3 } }
38     { \msg_new:nnnn { witharrows } { #1 } { #2 } { #3 } }
39 }
40 \cs_new_protected:Npn \@@_msg_redirect_name:nn
41 { \msg_redirect_name:nnn { witharrows } }
42 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
43 \cs_new_protected:Npn \@@_warning:n { \msg_warning:nn { witharrows } }
44 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { witharrows } }
45 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { witharrows } }
46 \cs_generate_variant:Nn \@@_error:nn { n e }

```

We also create a command which will generate usually an error but only a warning on Overleaf. The argument is given by currying.

```

47 \cs_new_protected:Npn \@@_error_or_warning:n
48 { \bool_if:NTF \c_@@_messages_for_Overleaf_bool \@@_warning:n \@@_error:n }

```

We try to detect whether the compilation is done on Overleaf. We use `\c_sys_jobname_str` because, with Overleaf, the value of `\c_sys_jobname_str` is always “output”.

```

49 \bool_set:Nn \c_@@_messages_for_Overleaf_bool
50 {
51   \str_if_eq_p:on \c_sys_jobname_str { _region_ } % for Emacs
52   || \str_if_eq_p:on \c_sys_jobname_str { output } % for Overleaf
53 }

54 \bool_new:N \g_@@_beamer_bool

```

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

```

55 <*LaTeX>
56 \keys_define:nn { WithArrows / package }
57 {
58   footnote .bool_set:N = \c_@@_footnote_bool ,
59   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
60   footnote .usage:n = load ,
61   footnotehyper .usage:n = load ,
62   beamer .bool_gset:N = \g_@@_beamer_bool ,
63   beamer .default:n = true ,
64   beamer .usage:n = load ,
65   unknown .code:n = \@@_fatal:n { Option-unknown~for~package }
66 }

67 \@@_msg_new:nn { Option-unknown~for~package }
68 {
69   You~can't~use~the~option~'\l_keys_key_str'~when~loading~the~
70   package~witharrows.~Try~to~use~the~command~
71   \token_to_str:N\WithArrowsOptions.
72 }

```

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

```

73 \ProcessKeysOptions { WithArrows / package }

74 \IfClassLoadedT { beamer } { \bool_gset_true:N \g_@@_beamer_bool }
75 \IfPackageLoadedT { beamerarticle } { \bool_gset_true:N \g_@@_beamer_bool }

```

```

76 \@@_msg_new:n { footnote-with-footnotehyper-package }
77 {
78   Footnote-forbidden.\
79   You-can't-use-the-option-'footnote'~because-the-package~
80   footnotehyper-has-already-been-loaded.~
81   If-you-want,~you-can-use-the-option-'footnotehyper'~and-the-footnotes~
82   within-the-environments-of-witharrows-will-be-extracted-with-the-tools~
83   of-the-package-footnotehyper.\
84   If-you-go-on,~the-package-footnote-won't-be-loaded.
85 }

86 \@@_msg_new:n { footnotehyper-with-footnote-package }
87 {
88   You-can't-use-the-option-'footnotehyper'~because-the-package~
89   footnote-has-already-been-loaded.~
90   If-you-want,~you-can-use-the-option-'footnote'~and-the-footnotes~
91   within-the-environments-of-witharrows-will-be-extracted-with-the-tools~
92   of-the-package-footnote.\
93   If-you-go-on,~the-package-footnotehyper-won't-be-loaded.
94 }

95 \bool_if:NT \c_@@_footnote_bool
96 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

97   \@ifclassloaded { beamer }
98     { \bool_set_false:N \c_@@_footnote_bool }
99     {
100       \@ifpackageloaded { footnotehyper }
101         { \@_error:n { footnote-with-footnotehyper-package } }
102         { \usepackage { footnote } }
103     }
104 }

105 \bool_if:NT \c_@@_footnotehyper_bool
106 {

```

The class beamer has its own system to extract footnotes and that's why we have nothing to do if beamer is used.

```

107   \@ifclassloaded { beamer }
108     { \bool_set_false:N \c_@@_footnote_bool }
109     {
110       \@ifpackageloaded { footnote }
111         { \@_error:n { footnotehyper-with-footnote-package } }
112         { \usepackage { footnotehyper } }
113       \bool_set_true:N \c_@@_footnote_bool
114     }
115 }

```

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

### 12.3 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).

```

116 \bool_new:N \c_@@_leqno_bool
117 \DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
118 \DeclareOption* { }
119 \ProcessOptions*
120 </LaTeX>

```

## 12.4 Collecting options

The following technic allows to create user commands with the ability to put an arbitrary number of *[list of (key=val)]* after the name of the command.

*Exemple :*

```

\@@_collect_options:n { \F } [x=a,y=b] [z=c,t=d] { arg }

```

will be transformed in :  $\F\{x=a,y=b,z=c,t=d\}\{arg\}$

Therefore, by writing :  $\def\G{\@@_collect_options:n{\F}}$ ,  
the command  $\G$  takes in an arbitrary number of optional arguments between square brackets.

```

121 <*LaTeX>
122 \cs_new_protected:Npn \@@_collect_options:n #1
123 {
124   \peek_meaning:NTF [
125     { \@@_collect_options:nw { #1 } }
126     { #1 { } }
127   }

```

We use  $\NewDocumentCommand$  in order to be able to allow nested brackets within the argument between  $[$  and  $]$ .

```

128 \NewDocumentCommand \@@_collect_options:nw { m r[] }
129 { \@@_collect_options:nn { #1 } { #2 } }
130
131 \cs_new_protected:Npn \@@_collect_options:nn #1 #2
132 {
133   \peek_meaning:NTF [
134     { \@@_collect_options:nw { #1 } { #2 } }
135     { #1 { #2 } }
136   }
137
138 \cs_new_protected:Npn \@@_collect_options:nw #1#2[#3]
139 { \@@_collect_options:nn { #1 } { #2 , #3 } }
140 </LaTeX>

```

## 12.5 Some technical definitions

```

141 \cs_generate_variant:Nn \seq_set_split:Nnn { N e e }
142 \cs_generate_variant:Nn \keys_precompile:nnN { n n c }
143 \prg_generate_conditional_variant:Nnn \tl_if_novalue:n { o } { F }
144 \exp_args_generate:n { N N N n o }

```

We define a command  $\@@_sort\_seq:N$  which will sort a sequence.

```

145 \cs_new_protected:Npn \@@_sort_seq:N #1
146 {
147   \seq_sort:Nn #1
148   {
149     \str_compare:eNeTF
150     { \str_lowercase:n { ##1 } } < { \str_lowercase:n { ##2 } }
151     \sort_return_same:
152     \sort_return_swapped:
153   }

```

```
154 }
```

The following command creates a sequence of strings (`str`) from a `clist`.

```
155 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
156 {
157   \seq_set_from_clist:Nn #1 { #2 }
158   \seq_set_map_e:Nn #1 #1 { \tl_to_str:n { ##1 } }
159 }
```

The command `\@@_save:N` saves a L3 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 L3 commands.

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

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

```
166   \str_set:Ne \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
167   \use:c { \l_tmpa_str _if_exist:cF }
168   { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
169   {
170     \use:c { \l_tmpa_str _new:c }
171     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
172   }
173   \use:c { \l_tmpa_str _gset_eq:cN }
174   { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
175 }
```

The command `\@@_restore:N` affects to the L3 variable the value of the (previously) set value of the corresponding *global* variable.

```
176 \cs_new_protected:Npn \@@_restore:N #1
177 {
178   \seq_set_split:Nee \l_tmpa_seq
179     { \char_generate:nn { ` } { 12 } }
180     { \cs_to_str:N #1 }
181   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
182   \str_set:Ne \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
183   \use:c { \l_tmpa_str _set_eq:Nc }
184   #1 { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
185 }
```

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

```
186 \tikzset
187 {
188   \@@_node_style / .style =
189   {
190     above = \l_@@_ystart_dim ,
191     inner-sep = \c_zero_dim ,
192     minimum-width = \c_zero_dim ,
193     minimum-height = \l_@@_ygap_dim
194   }
195 }
```

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

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

```

196 \tikzset
197 {
198   @@_standard / .style =
199   {
200     remember~picture ,
201     overlay ,
202     name~prefix = wa - \l_@@_prefix_str -
203   } ,
204   @@_standard_arrow / .style =
205   {
206     @@_standard ,
207     every~path / .style = WithArrows / arrow
208   }
209 }

```

The following line is a security when using `xelatex` and RTL language (cf. question 683570 on TeX StackExchange).

```

210 \sys_if_engine_xetex:T
211 {
212   \tikzset
213   {
214     @@_standard_arrow / .append~style =
215     { every~node / .append~style = { text = . } }
216   }
217 }

```

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`). This style is documented in the documentation of `witharrows`.

```

218 \tikzset
219 {
220   WithArrows / arrow / tips / .style =
221   { > = { Straight~Barb [ scale = 1.2 , bend ] } }
222 }

```

The style `WithArrows/arrow` will be used to draw the arrows (more precisely, it will be passed to `every~path`). This style is documented in the documentation of `witharrows`.

```

223 \tikzset
224 {
225   WithArrows / arrow / .style =
226   {
227     align = flush~left ,

```

Before the version 2.7, it was `align = left`.

```

228     auto = left ,
229     \<LaTeX
230     font = \small \itshape ,
231     \>/LaTeX
232     WithArrows / arrow / tips ,
233     bend~left = 45 ,
234     ->
235   }

```

---

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

236 }

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

```

237 <*LaTeX>
238 \AtBeginDocument
239 {
240   \IfPackageLoadedTF { amsmath }
241   {
242     \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
243     \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
244   }

```

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.

```

245 {
246 </LaTeX>
247   \cs_new_protected:Npn \spread@equation
248   {
249     \openup \jot
250     \cs_set_eq:NN \spread@equation \prg_do_nothing:
251   }
252 <*LaTeX>
253 }
254 }
255 </LaTeX>

```

```

256 \tl_new:N \l_@@_left_brace_tl
257 \tl_set_eq:NN \l_@@_left_brace_tl \c_novalue_tl

```

## 12.6 Variables

The boolean `\l_@@_in_WithArrows_bool` will be raised in an environment `{WithArrows}` and the boolean `\l_@@_in_DispWithArrows_bool` will be raised in an environment `{DispWithArrows}` or `{DispWithArrows*}`. The boolean `\l_@@_in_code_after_bool` will be raised during the execution of the code-after (option `code-after`).

```

258 \bool_new:N \l_@@_in_WithArrows_bool
259 \bool_new:N \l_@@_in_DispWithArrows_bool
260 \bool_new:N \l_@@_in_code_after_bool

```

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

```

261 \seq_new:N \g_@@_position_in_the_tree_seq
262 \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`.

```

263 \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`).

```

264 \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 `code-after` where the options `i`, `group` and `groups` are not available).

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

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

```
265 \int_new:N \l_@@_pos_arrow_int
266 \int_set:Nn \l_@@_pos_arrow_int 3
```

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

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

```
267 \seq_new:N \g_@@_arrow_int_seq
268 \int_new:N \g_@@_arrow_int
269 \seq_new:N \g_@@_line_int_seq
270 \int_new:N \g_@@_line_int
271 \seq_new:N \g_@@_col_int_seq
272 \int_new:N \g_@@_col_int
```

We will also use a “static” version of the counter of columns, called `\g_@@_static_col_int`. The value will be set directly in each cell of the array by an instruction in the template of the `\halign`. The aim of this programming is to try to detect some use of `\omit` (which should be forbidden) in the cells of the `\halign`.

```
273 \seq_new:N \g_@@_static_col_int_seq
274 \int_new:N \g_@@_static_col_int
```

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

```
275 \LaTeX
276 \clist_new:N \l_@@_tags_clist
277 \clist_set:Nn \l_@@_tags_clist { all }
```

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

```
278 \cs_new_protected:Npn \@@_test_if_to_tag:
279 {
280   \clist_if_in:NoT \l_@@_tags_clist \g_@@_line_int
281   { \clist_set:Nn \l_@@_tags_clist { all } }
282 }
283 \LaTeX
```

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

```
284 \str_new:N \l_@@_command_name_str
285 \str_set:Nn \l_@@_command_name_str { Arrow }
```



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

```
286 \str_new:N \l_@@_string_Arrow_for_msg_str
287 \str_set:Ne \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`.

```
288 \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.

```
289 <*LaTeX>
290 \bool_new:N \l_@@_sbwi_bool
291 </LaTeX>

292 <*LaTeX>
293 \bool_new:N \l_@@_tag_star_bool
294 \bool_new:N \l_@@_tag_next_line_bool
295 \bool_new:N \l_@@_qedhere_bool
296 </LaTeX>
297 \bool_new:N \l_@@_in_first_columns_bool
298 \bool_new:N \l_@@_new_group_bool
299 \bool_new:N \l_@@_initial_r_bool
300 \bool_new:N \l_@@_final_r_bool
301 \tl_new:N \l_@@_initial_tl
302 \tl_new:N \l_@@_final_tl
303 \int_new:N \l_@@_nb_cols_int
```

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

```
304 \str_new:N \l_@@_format_str
```

The option `\l_@@_subequations_bool` corresponds to the option `subequations`.

```
305 <*LaTeX>
306 \bool_new:N \l_@@_subequations_bool
307 </LaTeX>
```

The dimension `\l_@@_arrow_width_dim` is only for the arrows of type `up` and `down`. A value of `\c_max_dim` means that the arrow has the maximal possible width. A value of `0 pt` means that the the arrow has a width adjusted to the content of the node.

```
308 \dim_new:N \l_@@_arrow_width_dim
309 \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim
```

The parameter `\l_@@_up_and_down_radius_dim` corresponds to option `radius_for_up_and_down`.

```
310 \dim_new:N \l_@@_up_and_down_radius_dim
311 \dim_set:Nn \l_@@_up_and_down_radius_dim { 4 pt }
```

The sequence `\l_@@_o_arrows_seq` will be used to store the numbers of the arrows which are of type `o` (for *over*) (they are drawn *after* the other arrows).

```
312 \seq_new:N \l_@@_o_arrows_seq
```

The dimension `\l_@@_xoffset_for_o_arrows_dim` is the xoffset added when drawing an arrow of type `o` (for *over*).

```
313 \dim_new:N \l_@@_xoffset_for_o_arrows_dim
314 \dim_set:Nn \l_@@_xoffset_for_o_arrows_dim { 2 mm }
```

The following boolean corresponds to the key `right-overlap`. When that key is `false`, the overlap on the right of the arrows (and their labels) is computed and it is used to change the width of the environment `{WithArrows}` in order to include the arrows on the right (and, hence, there is no overlap).

```
315 \bool_new:N \l_@@_right_overlap_bool
316 \bool_set_true:N \l_@@_right_overlap_bool
```

## 12.7 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* level<sup>30</sup>;
- with `\begin{WithArrows}[...]`: this level will be called *environment* level;
- with `\Arrow[...]` (included in `code-after`): this level will be called *local* level.

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

```
317 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
318 {
319   \str_if_empty:NTF \l_@@_previous_key_str
320     {
321       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
322       #1
323     }
324     { \@@_error:n { Incompatible-options } }
325 }
326 \cs_new_protected:Npn \@@_fix_pos_option:n #1
327 { \@@_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.

```
328 \keys_define:nn { WithArrows / Global }
329 {
330   max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
331   max-length-of-arrow .value_required:n = true ,
332   max-length-of-arrow .initial:n = 2 cm ,
333   ygap .dim_set:N = \l_@@_ygap_dim ,
334   ygap .initial:n = 0.4 ex ,
335   ygap .value_required:n = true ,
336   ystart .dim_set:N = \l_@@_ystart_dim ,
337   ystart .value_required:n = true ,
338   ystart .initial:n = 0.4 ex ,
339   more-columns .code:n =
340     \@@_msg_redirect_name:nn { Too-much-columns-in-WithArrows } { none } ,
341   more-columns .value_forbidden:n = true ,
```

---

<sup>30</sup>This 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.

```

342 command-name .code:n =
343   \str_set:Nn \l_@@_command_name_str { #1 }
344   \str_set:Ne \l_@@_string_Arrow_for_msg_str
345     { \c_backslash_str Arrow-alias~\c_backslash_str #1 } ,
346 command-name .value_required:n = true ,
347 tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
348 tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
349 tikz-code .value_required:n = true ,
350 displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
351 displaystyle .default:n = true ,
352 show-nodes .code:n =
353   \tikzset { @@_node_style / .append~style = { draw , red } } ,
354 show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
355 show-node-names .default:n = true ,
356 group .code:n =
357   \str_if_empty:NTF \l_@@_previous_key_str
358     {
359       \str_set:Nn \l_@@_previous_key_str { group }
360       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
361       \int_set:Nn \l_@@_pos_arrow_int 7
362     }
363     { \@@_error:n { Incompatible~options } } ,
364 group .value_forbidden:n = true ,
365 groups .code:n =
366   \str_if_empty:NTF \l_@@_previous_key_str
367     {
368       \str_set:Nn \l_@@_previous_key_str { groups }
369       \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
370         { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
371       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
372       \int_set:Nn \l_@@_pos_arrow_int 6
373     }
374     { \@@_error:n { Incompatible~options } } ,
375 groups .value_forbidden:n = true ,
376 tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
377 tikz .value_required:n = true ,
378 rr .code:n = \@@_fix_pos_option:n 3 ,
379 rr .value_forbidden:n = true ,
380 ll .code:n = \@@_fix_pos_option:n 1 ,
381 ll .value_forbidden:n = true ,
382 rl .code:n = \@@_fix_pos_option:n 2 ,
383 rl .value_forbidden:n = true ,
384 lr .code:n = \@@_fix_pos_option:n 0 ,
385 lr .value_forbidden:n = true ,
386 i .code:n = \@@_fix_pos_option:n 5 ,
387 i .value_forbidden:n = true ,
388 xoffset .dim_set:N = \l_@@_xoffset_dim ,
389 xoffset .value_required:n = true ,
390 xoffset .initial:n = 3 mm ,
391 jot .dim_set:N = \jot ,
392 jot .value_required:n = true ,
393 interline .skip_set:N = \l_@@_interline_skip ,
394 start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
395 start-adjust .initial:n = 0.4 ex ,
396 start-adjust .value_required:n = true ,
397 end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
398 end-adjust .initial:n = 0.4 ex ,
399 end-adjust .value_required:n = true ,
400 adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
401 adjust .value_required:n = true ,
402 up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,
403 up-and-down .value_required:n = true ,

```

With the option `no-arrows`, the arrows won't be drawn. However, the "first pass" of the arrows is

done and some errors may be detected. The nullification of `\@@_draw_arrows:nn` is for the standard arrows and the nullification of `\@@_draw_arrow:nnn` is for “Arrow in code-after”.

```

404   no-arrows .code:n =
405     \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn
406     \cs_set_eq:NN \@@_draw_arrow:nnn \use_none:nnn ,
407   no-arrows .value_forbidden:n = true
408 }

```

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`.

```

409 \keys_define:nn { WithArrows / WithArrowsSpecific }
410 {
411   t .code:n          = \int_set:Nn \l_@@_pos_env_int 0 ,
412   t .value_forbidden:n = true ,
413   c .code:n          = \int_set:Nn \l_@@_pos_env_int 1 ,
414   c .value_forbidden:n = true ,
415   b .code:n          = \int_set:Nn \l_@@_pos_env_int 2 ,
416   b .value_forbidden:n = true ,
417   right-overlap .bool_set:N      = \l_@@_right_overlap_bool ,
418   right-overlap .value_required:n = true
419 }

```

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

```

420 \clist_new:N \c_@@_ext_delimiters_clist
421 \clist_set:Nn \c_@@_ext_delimiters_clist
422 {
423   ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
424 }
425 <*LaTeX>
426 \AtBeginDocument
427 {
428   \bool_set_false:N \l_tmpa_bool
429   \IfPackageLoadedT { amsmath } { \bool_set_true:N \l_tmpa_bool }
430   \IfPackageLoadedT { unicode-math } { \bool_set_true:N \l_tmpa_bool }
431   \bool_if:NT \l_tmpa_bool
432     { \clist_put_right:Nn \c_@@_ext_delimiters_clist { \lvert, \lVert } }
433 }
434 </LaTeX>

```

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

```

435 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
436 {
437   fleqn .bool_set:N = \l_@@_fleqn_bool ,
438   fleqn .default:n = true ,
439   mathindent .skip_set:N = \l_@@_mathindent_skip ,
440   mathindent .initial:n = 25 pt ,
441   mathindent .value_required:n = true ,
442 <*LaTeX>
443   notag .code:n =
444     \str_if_eq:nnTF { #1 } { true }
445     { \clist_clear:N \l_@@_tags_clist }
446     { \clist_set:Nn \l_@@_tags_clist { all } } ,
447   notag .default:n = true ,

```

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

```

448   subequations .code:n =
449     \IfPackageLoadedTF { amsmath }
450     { \bool_set_true:N \l_@@_subequations_bool }

```

```

451     {
452       \@@_error:n { amsmath~not~loaded }
453       \group_begin:
454       \globaldefs = 1
455       \@@_msg_redirect_name:nn { amsmath~not~loaded } { info }
456       \group_end:
457     } ,
458     subequations .default:n = true ,
459     subequations .value_forbidden:n = true ,
460     nonumber .meta:n = notag ,
461     allow-multiple-labels .code:n =
462       \@@_msg_redirect_name:nn { Multiple~labels } { none } ,
463     allow-multiple-labels .value_forbidden:n = true ,
464     tagged-lines .code:n =
465       \clist_set:Nn \l_@@_tags_clist { #1 }
466       \clist_if_in:NnT \l_@@_tags_clist { first }
467       {
468         \clist_remove_all:Nn \l_@@_tags_clist { first }
469         \clist_put_left:Nn \l_@@_tags_clist 1
470       } ,
471     tagged-lines .value_required:n = true ,
472 \LaTeX
473     wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
474     wrap-lines .default:n = true ,
475     replace-left-brace-by .code:n =
476     {
477       \tl_set:Ne \l_tmpa_tl { \tl_head:n { #1 } }
478       \clist_if_in:NoTF
479         \c_@@_ext_delimiters_clist
480         \l_tmpa_tl
481         { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }
482         { \@@_error:n { Bad~value~for~replace~brace~by } }
483     } ,
484     replace-left-brace-by .initial:n = \lbrace ,

```

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

```

485 \LaTeX
486     standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
487     standard-behaviour-with-items .default:n = true
488 \LaTeX
489 }

```

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

```

490 \keys_define:nn { WithArrows / Env }
491 {
492     name .code:n =

```

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

```

493     \str_set:Nn \l_tmpa_str { #1 }
494     \seq_if_in:NoTF \g_@@_names_seq \l_tmpa_str
495       { \@@_error:n { Duplicate~name } }
496       { \seq_gput_left:No \g_@@_names_seq \l_tmpa_str }
497     \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
498     name .value_required:n = true ,
499     code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
500     code-before .value_required:n = true ,
501     CodeBefore .meta:n = { code-before = #1 } ,
502     code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
503     code-after .value_required:n = true ,
504     CodeAfter .meta:n = { code-after = #1 } ,

```

```

505 format .code:n =
506   \tl_if_empty:nTF { #1 }
507     { \@@_error:n { Invalid~option~format } }
508     {
509       \regex_match:nnTF { \A[rclRCL]*\Z } { #1 }
510       { \tl_set:Nn \l_@@_format_str { #1 } }
511       { \@@_error:n { Invalid~option~format } }
512     } ,
513 format .value_required:n = true
514 }

```

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

```

515 \keys_define:nn { WithArrows }
516 {
517   WithArrows .inherit:n =
518     {
519       WithArrows / Global ,
520       WithArrows / WithArrowsSpecific ,
521       WithArrows / Env
522     } ,
523   WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
524   DispWithArrows .inherit:n =
525     {
526       WithArrows / DispWithArrowsSpecific ,
527       WithArrows / Global ,
528       WithArrows / Env ,
529     } ,
530   DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
531   WithArrowsOptions .inherit:n =
532     {
533       WithArrows / Global ,
534       WithArrows / WithArrowsSpecific ,
535       WithArrows / DispWithArrowsSpecific ,
536     } ,
537   WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
538 }

```

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

```

539 \seq_new:N \l_@@_options_WithArrows_seq
540 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrows_seq
541 {
542   adjust, b, c, code-after, code-before, command-name,
543   right-overlap, displaystyle, end-adjust,
544   format, group, groups, i,
545   interline, jot, ll,
546   lr, max-length-of-arrow, more-columns, name,
547   no-arrows, rl, rr, up-and-down,
548   show-node-names, show-nodes, start-adjust,
549   t, tikz, tikz-code,
550   xoffset, ygap, ystart
551 }

552 \keys_define:nn { WithArrows / WithArrows }
553 {
554   unknown .code:n =
555     \@@_sort_seq:N \l_@@_options_WithArrows_seq
556     \@@_error:n { Unknown~option~WithArrows }
557 }

```

```

558 \keys_define:nn { WithArrows / DispWithArrows }
559 {
560   left-brace .tl_set:N = \l_@@_left_brace_tl ,
561   unknown .code:n =
562     \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
563     \@@_error:n { Unknown-option-DispWithArrows } ,
564 }

```

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

```

565 \seq_new:N \l_@@_options_DispWithArrows_seq
566 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
567 {
568   code-after, code-before, command-name, tikz-code, adjust,
569   displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
570   left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
571   up-and-down, replace-left-brace-by, rl, rr, show-node-names,
572   show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
573 <*LaTeX>
574   allow-multiple-labels, tagged-lines, nonumber, notag
575 </LaTeX>
576 }
577 \keys_define:nn { WithArrows / WithArrowsOptions }
578 {
579   allow-duplicate-names .code:n =
580     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
581   allow-duplicate-names .value_forbidden:n = true ,
582   xoffset-for-o-arrows .dim_set:N = \l_@@_xoffset_for_o_arrows_dim ,
583   xoffset-for-o-arrows .value_required:n = true ,
584   unknown .code:n =
585     \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
586     \@@_error:n { Unknown-option-WithArrowsOptions }
587 }

```

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

```

588 \seq_new:N \l_@@_options_WithArrowsOptions_seq
589 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrowsOptions_seq
590 {
591   allow-duplicate-names, b, c, command-name, right_overlap,
592   more-columns, tikz-code, adjust,
593   displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
594   mathindent, max-length-of-arrow, no-arrows, up-and-down, rl, rr,
595   show-node-names, show-nodes, start-adjust, t, tikz, wrap-lines, xoffset,
596   xoffset-for-o-arrows, ygap, ystart,
597 <*LaTeX>
598   allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
599   tagged-lines
600 </LaTeX>
601 }

```

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 possible value of the field “status” is “new-group”.

```

602 \cs_new_protected:Npn \@@_set_independent:
603 {
604   \str_if_eq:onF \l_keys_value_tl { NoValue }
605     { \@@_error:n { Value-for-a-key } }
606   \@@_set_independent_bis:
607 }

```

The command `\@@_set_independent_bis:` is the same as `\@@_set_independent:` except that the key may be used with a value.

```

608 \cs_new_protected:Npn \@@_set_independent_bis:
609   {
610     \str_if_empty:NTF \l_@@_previous_key_str
611     {
612       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
613       \str_set:Nn \l_@@_status_arrow_str { independent }
614     }
615     { \@@_error:n { Incompatible~options-in~Arrow } }
616   }

```

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 set of keys 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”.

```

617 \keys_define:nn { WithArrows / Arrow / FirstPass }
618   {
619     jump .code:n =
620       \int_compare:nNnTF { #1 } > \c_zero_int
621       { \int_set:Nn \l_@@_jump_int { #1 } }
622       { \@@_error:n { Negative~jump } } ,
623     jump .value_required:n = true,
624     rr .code:n = \@@_set_independent: ,
625     ll .code:n = \@@_set_independent: ,
626     rl .code:n = \@@_set_independent: ,
627     lr .code:n = \@@_set_independent: ,
628     i .code:n = \@@_set_independent: ,
629     rr .default:n = NoValue ,
630     ll .default:n = NoValue ,
631     rl .default:n = NoValue ,
632     lr .default:n = NoValue ,
633     i .default:n = NoValue ,
634     new-group .value_forbidden:n = true ,
635     new-group .code:n =
636       \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
637       { \str_set:Nn \l_@@_status_arrow_str { new-group } }
638       { \@@_error:n { new-group~without~groups } } ,
639     o .code:n =
640       \str_if_empty:NTF \l_@@_previous_key_str
641       {
642         \int_compare:nNnTF \l_@@_pos_arrow_int < 6
643         { \@@_error:n { invalid-key-o } }
644         {
645           \str_set:Nn \l_@@_status_arrow_str { over }
646           \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
647         }
648       }
649       { \@@_error:n { Incompatible~options~in~Arrow } } ,

```

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.

```

650 tikz-code .code:n = \prg_do_nothing: ,
651 tikz-code .value_required:n = true ,
652 tikz .code:n = \prg_do_nothing: ,
653 tikz .value_required:n = true ,
654 start-adjust .code:n = \prg_do_nothing: ,
655 start-adjust .value_required:n = true ,
656 end-adjust .code:n = \prg_do_nothing: ,
657 end-adjust .value_required:n = true ,

```



```

658 adjust .code:n = \prg_do_nothing: ,
659 adjust .value_required:n = true ,
660 xoffset .code:n = ,
661 unknown .code:n =
662   \@@_sort_seq:N \l_@@_options_Arrow_seq
663   \seq_if_in:NoTF \l_@@_options_WithArrows_seq \l_keys_key_str
664   {
665     \str_set:Nn \l_tmpa_str
666     { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
667   }
668   { \str_clear:N \l_tmpa_str }
669   \@@_error:n { Unknown~option~in~Arrow }
670 }

```

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

```

671 \seq_new:N \l_@@_options_Arrow_seq
672 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_seq
673 {
674   adjust, end-adjust, i, jump, ll, lr, o , rl, rr, start-adjust, tikz,
675   tikz-code, xoffset
676 }

```

```

677 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1
678 {
679   \str_if_empty:NT \l_@@_previous_key_str
680   {
681     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
682     \int_set:Nn \l_@@_pos_arrow_int { #1 }
683   }
684 }

```

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.

```

685 \keys_define:nn {WithArrows / Arrow / SecondPass }
686 {
687   tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
688   tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
689   tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
690   rr .code:n = \@@_fix_pos_arrow:n 3 ,
691   ll .code:n = \@@_fix_pos_arrow:n 1 ,
692   rl .code:n = \@@_fix_pos_arrow:n 2 ,
693   lr .code:n = \@@_fix_pos_arrow:n 0 ,
694   i .code:n = \@@_fix_pos_arrow:n 5 ,
695   o .code:n = \str_set:Nn \l_@@_previous_key_str { o } ,

```

The option `xoffset` is not allowed when the option `group` or the option `groups` is used except, if the arrow is independent or if there is only one arrow.

```

696 xoffset .code:n =
697   \bool_lazy_all:nTF
698   {
699     { \int_compare_p:nNn \g_@@_arrow_int > 1 }
700     { \int_compare_p:nNn \l_@@_pos_arrow_int > 5 }
701     { ! \str_if_eq_p:on \l_@@_status_arrow_str { independent } }
702   }
703   { \@@_error:n { Option~xoffset~forbidden } }
704   { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
705 xoffset .value_required:n = true ,
706 start-adjust .dim_set:N = \l_@@_start_adjust_dim,
707 end-adjust .dim_set:N = \l_@@_end_adjust_dim,

```

```

708   adjust .code:n =
709     \dim_set:Nn \l_@@_start_adjust_dim { #1 }
710     \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
711   }

```

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

```

712 {*LaTeX}
713 \NewDocumentCommand \WithArrowsOptions { m }
714 {/LaTeX}
715 {*plain-TeX}
716 \cs_set_protected:Npn \WithArrowsOptions #1
717 {/plain-TeX}
718 {
719   \str_clear_new:N \l_@@_previous_key_str
720   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
721 }

```

## 12.8 The command `\Arrow`

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

```

722 {*LaTeX}
723 \cs_new_protected:Npn \@@_Arrow
724   { \@@_collect_options:n { \@@_Arrow_iii } }
725 \bool_if:NTF \g_@@_beamer_bool
726   {
727     \NewDocumentCommand \@@_Arrow_iii { m d < > m ! 0 { } }
728     {
729       \tl_if_novalue:nTF { #2 }
730         { \@@_Arrow_ii { #1 } { #3 } [ #4 ] }
731         { \only <#2> { \@@_Arrow_ii { #1 } { #3 } [ #4 ] } }
732     }
733   }
734   {
735     \NewDocumentCommand \@@_Arrow_iii { m m ! 0 { } }
736     { \@@_Arrow_ii { #1 } { #2 } [ #3 ] }
737   }
738 \NewDocumentCommand \@@_Arrow_ii { m m ! 0 { } }
739 {/LaTeX}
740 {*plain-TeX}
741 \cs_new_protected:Npn \@@_Arrow
742   {
743     \peek_meaning:NTF [
744       { \@@_Arrow_i }
745       { \@@_Arrow_i [ ] }
746     }
747 \cs_new_protected:Npn \@@_Arrow_i [ #1 ] #2
748   {
749     \peek_meaning:NTF [
750       { \@@_Arrow_ii [ #1 ] { #2 } }
751       { \@@_Arrow_ii [ #1 ] { #2 } [ ] }
752     }
753 \cs_new_protected:Npn \@@_Arrow_ii [ #1 ] #2 [ #3 ]
754 {/plain-TeX}
755   {

```

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.

```
756 \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 `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`.

```
757 \str_clear_new:N \l_@@_previous_key_str
758 \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:

```
759 \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`):

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

3. The “status” of the arrow, with 4 possible values: `empty`, `independent`, `new-group` or `over`.

```
762 \prop_put:Nno \l_tmpa_prop { status } \l_@@_status_arrow_str
```

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

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

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

```
764 \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 some error messages).

```
765 \prop_put:Nne \l_tmpa_prop { input-line } \msg_line_number:
```

7. The total width of the arrow (with the label)... but we don't know it now and that's why we put `0 pt`. There are used for the arrows of type `o`.

```
766 \prop_put:Nnn \l_tmpa_prop { width } { 0 pt }
```

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.

```
767 \prop_gclear_new:c
768 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
769 \prop_gset_eq:cN
770 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
771 \l_tmpa_prop
772 }
```

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

```
773 \cs_new_protected:Npn \@@_Arrow_first_columns:
774 { \@@_error:n { Arrow-not-in-last-column } \@@_Arrow }
```

## 12.9 The environments `{WithArrows}` and `{DispWithArrows}`

### 12.9.1 Code before the `\halign`

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

```
775 \cs_new_protected:Npn \@@_pre_halign:n #1
```

First, the initialization 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.

```
776 {
777 <*LaTeX>
778   \str_clear_new:N \l_@@_type_env_str
779   \str_set:NV \l_@@_type_env_str \@currentenv
780 </LaTeX>
```

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

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

783   \tikzset { arrows = [ flex ] } % https://texnique.fr/osqa/questions/12199
```

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.

```
784   \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.

```
785   \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.

```
786   \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 some error messages.

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

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

```
788   \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
789   \int_gzero:N \g_@@_arrow_int
790   \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
791   \int_gzero:N \g_@@_line_int
792   \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
793   \int_gzero:N \g_@@_col_int
794   \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
795   \int_gzero:N \g_@@_static_col_int
```

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

```
\int_gincr:N \g_@@_col_int
\int_set:Nn \g_@@_static_col_int 3
```

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

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

```
796 \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).

```
797 \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
798 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
799 \str_clear_new:N \l_@@_prefix_str
800 \str_set:Ne \l_@@_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

```
801 \cs_set_eq:NN \ \ \@@_cr:
802 \dim_zero:N \mathsurround
```

These counters will be used later as variables.

```
803 \int_zero_new:N \l_@@_initial_int
804 \int_zero_new:N \l_@@_final_int
805 \int_zero_new:N \l_@@_arrow_int
806 \int_zero_new:N \l_@@_pos_of_arrow_int
807 \int_zero_new:N \l_@@_jump_int
```

The counter `\l_@@_jump_int` corresponds to the option `jump`. Now, we set the initial value for this option.

```
808 \int_set:Nn \l_@@_jump_int 1
```

The string `\l_@@_format_str` corresponds to the option `format`. Now, we set the initial value for this option.

```
809 \str_set:Nn \l_@@_format_str { rL }
```

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

```
810 (*LaTeX)
811 \seq_clear_new:N \l_@@_labels_seq
812 \bool_set_false:N \l_@@_tag_next_line_bool
813 (/LaTeX)
```

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

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

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

```
815 \tl_clear_new:N \l_@@_code_before_tl
816 \tl_clear_new:N \l_@@_code_after_tl
```

---

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

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

```

817 \str_clear_new:N \l_@@_previous_key_str
818 \bool_if:NT \l_@@_in_WithArrows_bool
819   { \keys_set:nn { WithArrows / WithArrows } { #1 } }
820 \bool_if:NT \l_@@_in_DispWithArrows_bool
821   { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }

```

The dimension `\g_@@_overlap_x_dim` will be the maximal overlap on the right of the arrows (and their labels) drawn in the environment `{WithArrows}`. The dimension `\l_@@_delta_x_dim` will be the difference of abscissa between the right side of the alignment (`\halign`) and the left side of the arrow.

```

822 \bool_if:NF \l_@@_right_overlap_bool
823   {
824     \bool_if:NT \l_@@_in_WithArrows_bool
825       {
826         \dim_gzero_new:N \g_@@_overlap_x_dim
827         \dim_zero_new:N \l_@@_delta_x_dim
828       }
829   }

```

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

```

830 \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_first_columns:

```

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

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

```

831 \int_set:Nn \l_@@_nb_cols_int { \str_count:N \l_@@_format_str }

```

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

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

```

832 \int_gset_eq:NN \g_@@_col_int \l_@@_nb_cols_int

```

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

```

833 \seq_clear_new:N \l_@@_format_seq
834 \seq_set_split:NnV \l_@@_format_seq { } \l_@@_format_str

```

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

```

835  $\LaTeX$ 
836   \bool_if:NT \c_@@_footnote_bool { \begin { savenotes } }
837  $\LaTeX$ 

```

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

```

838 \l_@@_code_before_tl

```

```

839 <*LaTeX>
840   \cs_set_eq:NN \notag \@@_notag:
841   \cs_set_eq:NN \nonumber \@@_nonumber:
842   \cs_set_eq:NN \tag \@@_tag
843   \cs_set_eq:NN \@@_old_label \label
844   \cs_set_eq:NN \label \@@_label:n
845   \cs_set_eq:NN \tagnextline \@@_tagnextline:
846 </LaTeX>
847 }

```

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

## 12.9.2 The construction of the preamble of the `\halign`

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

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

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

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

```

848 \cs_new_protected:Npn \@@_construct_halign:
849 {
850   \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
851   {

```

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

```

852   \use:e
853   {

```

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

We begin the construction of a generic column.

```

854       \int_gdecr:N \g_@@_col_int
855       \@@_construct_halign:
856       \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
857       {

```

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

```

858         \cs_set_eq:cN { \l_@@_command_name_str } \@@_Arrow
859 <*LaTeX>
860         \bool_if:NT \l_@@_in_DispWithArrows_bool
861         {

```

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

```
862         \@@_test_if_to_tag:
```

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

```
863         \IfPackageLoadedT { amsmath } { \@@_set_qedhere: }
864     }
865 </LaTeX>
866 }
867 \str_if_eq:onT \l_@@_type_col_str { c } \hfil
868 \str_if_eq:onT \l_@@_type_col_str { C } \hfil
869 \str_if_eq:onT \l_@@_type_col_str { r } \hfill
870 \str_if_eq:onT \l_@@_type_col_str { R } \hfill
871 \int_gincr:N \g_@@_col_int
872 \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
873 \c_math_toggle_token
874 \str_if_eq:onT \l_@@_type_col_str { C } { { } }
875 \str_if_eq:onT \l_@@_type_col_str { L } { { } }
876 \bool_if:NT \l_@@_displaystyle_bool \displaystyle
877 ##
878 \str_if_eq:onT \l_@@_type_col_str { C } { { } }
879 \str_if_eq:onT \l_@@_type_col_str { R } { { } }
880 \c_math_toggle_token
881 \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
882     \@@_construct_nodes:
883     {
```

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

```
884         \str_if_eq:onT \l_@@_type_col_str { l } \hfil
885         \str_if_eq:onT \l_@@_type_col_str { L } \hfil
886         \str_if_eq:onT \l_@@_type_col_str { c } \hfil
887         \str_if_eq:onT \l_@@_type_col_str { C } \hfil
888         \bool_if:NT \l_@@_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
889     &
890     }
891 }
892 }
```

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

```
893     {
894     \bool_if:NTF \l_@@_in_WithArrows_bool
895     {
896     \ialign
897     \bgroup
898     }
899     {
900     \halign to \l_@@_linewidth_dim
901     \bgroup
902     \bool_if:NT \l_@@_fleqn_bool
903     { \skip_horizontal:N \l_@@_mathindent_skip }
904     }
905     \int_gincr:N \g_@@_line_int
906     \int_gzero:N \g_@@_col_int
907     \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
908     {
909     \skip_horizontal:n
910     { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
911     }
912     \strut
913 }
914 }
```



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

```
915 \cs_new_protected:Npn \@@_construct_nodes:
916   {
```

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).

```
917   \tikz [ remember-picture , overlay ]
918     \node
919       [
920         node-contents = { } ,
921         @@_node_style ,
922         name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
923       ]
924     ;
925   \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).

```
926   \tikz [ remember-picture , overlay ]
927     \node
928       [
929         node-contents = { } ,
930         @@_node_style ,
931         name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
932       ]
933     ;
934   \str_if_empty:NF \l_@@_name_str
935     {
936     \pgfpicture
937     \pgfnodealias
938     { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
939     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l }
940     \pgfnodealias
941     { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
942     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
943     \endpgfpicture
944   }
945   \bool_if:NT \l_@@_show_node_names_bool
946     {
947     \hbox_overlap_right:n
948     { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
949   }
950 }
```

### 12.9.3 The environment `{WithArrows}`

```
951 <LaTeX>
952 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
953 </LaTeX>
954 <*plain-TeX>
955 \cs_new_protected:Npn \WithArrows
956   {
957   \group_begin:
958   \peek_meaning:NTF [
959     { \WithArrows_i }
960     { \WithArrows_i [ ] }
961   ]
962   \cs_new_protected:Npn \WithArrows_i [ #1 ]
963 </plain-TeX>
964   {
965     \bool_set_true:N \l_@@_in-WithArrows_bool
```

```

966   \bool_set_false:N \l_@@_in_DisWithArrows_bool
967 (*plain-TeX)
968   \str_clear_new:N \l_@@_type_env_str
969   \str_set:Nn \l_@@_type_env_str { WithArrows }
970 
```

plain-TeX

```

971   \@@_pre_halign:n { #1 }
972   \if_mode_math: \else:
973     \@@_error:n { WithArrows~outside~math~mode }
974   \fi:
975   \box_clear_new:N \l_@@_env_box
976   \hbox_set:Nw \l_@@_env_box

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`<sup>32</sup> 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 mode<sup>33</sup> and therefore, we can use `\vcenter`.

```

977   \int_compare:nNnT \l_@@_pos_env_int = 1 \c_math_toggle_token
978   \int_case:nn \l_@@_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
979   \bgroup

```

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

```

980   \spread@equation

```

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

```

981   \@@_construct_halign:

```

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

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

```

982   &&
983   \@@_error:n { Too~much~columns~in~WithArrows }
984   \c_math_toggle_token
985   \bool_if:NT \l_@@_displaystyle_bool \displaystyle
986   { ## }
987   \c_math_toggle_token
988   \cr
989   }

```

We begin the second part of the environment `{WithArrows}`. We have three `\egroup`: one for the `\halign`, one for the `\vtop` (or `\vcenter` or `\vbox`) and one for the `\hbox_set:Nn \l_@@_env_box`.

```

990 (*plain-TeX)
991 \cs_new_protected:Npn \endWithArrows
992 
```

plain-TeX

```

993   {
994     \
995     \egroup
996     \egroup
997     \int_compare:nNnT \l_@@_pos_env_int = 1 \c_math_toggle_token
998     \hbox_set_end:
999     \@@_post_halign:

```

We want to add white space on the right side of the box in order to take into account the arrows and their labels.

```

1000   \bool_if:NF \l_@@_right_overlap_bool

```

<sup>32</sup>Notice that the use of `\vtop` seems color-safe here...

<sup>33</sup>An error is raised if the environment is used outside math mode.

```

1001     {
1002     \box_set_wd:Nn \l_@@_env_box
1003     { \g_@@_overlap_x_dim + \box_wd:N \l_@@_env_box }
1004     }
1005     \box_use_drop:N \l_@@_env_box

```

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`).

```

1006 <*LaTeX>
1007     \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
1008 </LaTeX>
1009 <*plain-TeX>
1010     \group_end:
1011 </plain-TeX>
1012 }

```

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

#### 12.9.4 After the construction of the `\halign`

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

```

1013 \cs_new_protected:Npn \@@_post_halign:

```

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

```

1014 {
1015     \cs_set:Npn \WithArrowsRightX { \g_@@_right_x_dim }

```

We use `\normalbaselines` of plain-TeX because we have used `\spread@equation` (of `amsmath` or defined directly if `amsmath` is not loaded) and you don't want `\spread@equation` to have effects in the labels of the arrows.

```

1016     \normalbaselines

```

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

```

1017     \int_if_zero:nF \g_@@_arrow_int
1018     {

```

If there is only one arrow, the options `group` and `groups` do not really make sense and it will be quicker to act as if we were in option `i` (moreover, it allows the option `xoffset` for the unique arrow).

```

1019         \int_compare:nNnT \g_@@_arrow_int = 1
1020         {
1021             \int_compare:nNnT \l_@@_pos_arrow_int > 5
1022             { \int_set:Nn \l_@@_pos_arrow_int 5 }
1023         }
1024         \@@_scan_arrows:
1025     }

```

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

```

1026     \group_begin:
1027     \tikzset { every~picture / .style = @@_standard }

```

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

```

1028     \cs_set:Npn \WithArrowsNbLines { \int_use:N \g_@@_line_int }

```

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

```

1029     \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
1030     \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
1031     \bool_set_true:N \l_@@_in_code_after_bool
1032     \l_@@_code_after_tl
1033     \group_end:

```

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

```

1034     \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
1035     \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
1036     \seq_gput_right:Ne \g_@@_position_in_the_tree_seq
1037     { \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`.

```

1038     \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
1039     { \int_gincr:N \g_@@_last_env_int }

```

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

```

1040     \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
1041     \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
1042     \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
1043     \int_gset:Nn \g_@@_line_int \l_tmpa_tl
1044     \seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
1045     \int_gset:Nn \g_@@_col_int \l_tmpa_tl
1046     \seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
1047     \int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
1048 }

```

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

### 12.9.5 The command of end of row

We give now the definition of `\@@_cr:` which is the definition of `\` in an environment `{WithArrows}`. The two 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).

```

1049 \cs_new_protected:Npn \@@_cr:
1050 {
1051     \scan_stop:

```

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

```

1052     \int_compare:nNnF \g_@@_col_int = \g_@@_static_col_int
1053     { \@@_error:n { omit~probably~used } }
1054     \prg_replicate:nn { \l_@@_nb_cols_int - \g_@@_static_col_int } { & { } }
1055     \group_align_safe_begin:
1056     \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
1057 }

```

---

<sup>34</sup>As of now, `\MultiArrow` has no option, and that’s why its internal name is a name of L3 with the signature `:nn` whereas `\Arrow` in `code-after` provides options and has the name of a function defined with `\NewDocumentCommand`.

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.

```
1058 \cs_new_protected:Npn \@@_cr_i:
1059   { \peek_meaning:NTF [ \@@_cr_ii: { \@@_cr_ii: [ \c_zero_dim ] } }
```

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

```
1060 <*LaTeX>
1061 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1062   {
1063     \peek_remove_spaces:n
1064     {
1065       \peek_meaning:NTF \end
1066       {
1067         \@@_cr_iii:n { #1 }
```

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

```
1068         \@@_analyze_end:Nn
1069         }
1070         { \@@_cr_iii:n { #1 } }
1071       }
1072     }

1073 \cs_new_protected:Npn \@@_cr_iii:n #1
1074 </LaTeX>
1075 <*plain-TeX>
1076 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1077 </plain-TeX>
1078   {
1079     \group_align_safe_end:
```

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

```
1080   \bool_if:NT \l_@@_in_DispWithArrows_bool
```

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

```
1081   {
1082 <*LaTeX>
1083     \clist_if_in:NnTF \l_@@_tags_clist { all }
1084     {
```

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).

```
1085     \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 `&`).

```
1086     \cs_gset:Npe \g_tmpa_tl
1087     { \tl_if_empty:NTF \l_@@_tag_tl \theequation \l_@@_tag_tl }
```

---

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

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`.

```
1088     \seq_if_empty:NF \l_@@_labels_seq
1089     {
```

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`.

```
1090     \cs_set:Npe \@currentlabel { \p@equation \g_tmpa_tl }
```

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

```
1091     \IfPackageLoadedT { hyperref }
1092     {
1093         % the following line is probably pointless (2022/05/16)
1094         % \str_set:Nn \This@name { equation }
1095         \hyper@refstepcounter { equation }
1096     }
```

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.

```
1097     \IfPackageLoadedT { cleveref }
1098     {
1099         \cref@constructprefix { equation } \cref@result
1100         \protected@edef \cref@currentlabel
1101         {
1102             [
1103                 \cs_if_exist:NTF \cref@equation@alias
1104                 \cref@equation@alias
1105                 { equation }
1106             ]
1107             [ \arabic { equation } ] [ \cref@result ]
1108             \p@equation \g_tmpa_tl
1109         }
1110     }
```

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`).

```
1111     \seq_map_function:NN \l_@@_labels_seq \@_old_label
1112     }
```

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.

```
1113     \@_save:N \l_@@_tag_star_bool
1114     \@_save:N \l_@@_qedhere_bool
1115     \bool_if:NT \l_@@_tag_next_line_bool
1116     {
1117         \openup -\jot
1118         \bool_set_false:N \l_@@_tag_next_line_bool
1119         \notag \&
1120     }
1121     &
1122     \@_restore:N \l_@@_tag_star_bool
1123     \@_restore:N \l_@@_qedhere_bool
1124     \bool_if:NT \l_@@_qedhere_bool
1125     { \hbox_overlap_left:n \@_qedhere_i: }
1126     \cs_set_eq:NN \theequation \g_tmpa_tl
1127     \bool_if:NT \l_@@_tag_star_bool
1128     { \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`.

```
1129     \hbox_overlap_left:n
```

```

1130     {
1131         \bool_if:NF \c_@@_leqno_bool
1132         {
1133             \pgfpicture
1134             \pgfrememberpicturepositiononpagetrue
1135             \pgfcoordinate
1136             { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1137             \pgfpointorigin
1138             \endpgfpicture
1139         }
1140         \quad
1141         \@eqnnum
1142     }
1143     \bool_if:NT \c_@@_leqno_bool
1144     {
1145         \pgfpicture
1146         \pgfrememberpicturepositiononpagetrue
1147         \pgfcoordinate
1148         { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1149         \pgfpointorigin
1150         \endpgfpicture
1151     }
1152 }
1153 {
1154     \@@_save:N \l_@@_qedhere_bool
1155 </LaTeX>
1156 &
1157 <*LaTeX>
1158     \@@_restore:N \l_@@_qedhere_bool
1159     \bool_if:NT \l_@@_qedhere_bool
1160     { \hbox_overlap_left:n \@@_qedhere_i: }
1161 </LaTeX>
1162     \pgfpicture
1163     \pgfrememberpicturepositiononpagetrue
1164     \pgfcoordinate
1165     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1166     \pgfpointorigin
1167     \endpgfpicture
1168 <*LaTeX>
1169 }
1170 </LaTeX>
1171 }
1172     \dim_compare:nNnT { #1 } < \c_zero_dim
1173     { \@@_error:n { option-of-cr-negative } }
1174
1175     \cr
1176     \noalign
1177     {
1178         \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1179         \skip_vertical:N \l_tmpa_dim
1180         \skip_vertical:N \l_@@_interline_skip
1181         \scan_stop:
1182     }
1183 }

```

According to the documentation of L3, the previous addition in “#1 + \l\_@@\_interline\_skip” is really an addition of skips (=glues).

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

```

1184 <*LaTeX>

```

```

1185 \cs_new_protected:Npn \@_analyze_end:Nn #1 #2
1186   {
1187     \str_if_eq:onT \l_@@_type_env_str { #2 }
1188     {
1189       \@_error:n { newline-at-the-end-of-env }
1190       \group_begin:
1191       \globaldefs = 1
1192       \@_msg_redirect_name:nn { newline-at-the-end-of-env } { none }
1193       \group_end:
1194     }

```

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

```

1195     \end { #2 }
1196   }
1197 \</LaTeX>

```

### 12.9.6 The environment `{DispWithArrows}`

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

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

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

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

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

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

```

1198 \<LaTeX>
1199 \bool_new:N \l_@@_in_label_or_minipage_bool
1200 \</LaTeX>
1201 \<LaTeX>
1202 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1203 \</LaTeX>
1204 \<plain-TeX>
1205 \cs_new_protected:Npn \DispWithArrows
1206   {
1207     \group_begin:
1208     \peek_meaning:NTF <
1209     { \DispWithArrows_i }
1210     { \DispWithArrows_i < \c_novalue_tl > }
1211   }
1212 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1213   {
1214     \peek_meaning:NTF [
1215     { \DispWithArrows_ii < #1 > }
1216     { \DispWithArrows_ii < #1 > [ ] }
1217   }
1218 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1219 \</plain-TeX>
1220   {
1221     \bool_set_true:N \l_@@_in_DispWithArrows_bool
1222 \<plain-TeX>
1223     \str_clear_new:N \l_@@_type_env_str
1224     \str_set:Nn \l_@@_type_env_str { DispWithArrows }
1225 \</plain-TeX>

```

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



beginning of the environment (in particular, because it must be done before the execution of the `code-before`<sup>36</sup>).

```

1226 (*LaTeX)
1227   \bool_if:NF \l_@@_sbwi_bool
1228     {
1229       \legacy_if:nT { @inlabel }
1230         { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1231       \legacy_if:nT { @minipage }
1232         { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1233     }
1234 (/LaTeX)

```

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).

```

1235 (*LaTeX)
1236   \IfPackageLoadedT { mathtools }
1237     {
1238       \MH_if_boolean:nT { show_only_refs }
1239         {
1240           \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`.

```

1241           \MH_set_boolean_T:n { show_only_refs }
1242         }
1243     }

```

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`.

```

1244   \IfPackageLoadedT { typedref }
1245     { \str_set:Nn \sr@name { equation } }

```

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

```

1246   \IfPackageLoadedT { amsmath } { \intertext@ }
1247 (/LaTeX)
1248   \tl_if_novalue:oF { #1 }
1249     { \cs_set_nopar:Npn \l_@@_left_brace_tl { #1 } }
1250   \@@_pre_halign:n { #2 }

```

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

```

1251 (*LaTeX)
1252   \bool_if:NT \l_@@_subequations_bool { \begin { subequations } }
1253 (/LaTeX)

1254   \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
1255     {

```

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

```

1256       \hbox_set:Nn \l_tmpa_box
1257         {

```

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

```

1258           \group_begin:
1259           \dim_zero:N \nulldelimiterspace
1260           \c_math_toggle_token
1261           \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1262           \c_math_toggle_token
1263           \group_end:
1264         }

```

---

<sup>36</sup>The `code-before` is not meant to contains typesetting material. However, it may contain, for example, a `{tikzpicture}` with options `overlay` and `remember picture` in order to draw nodes *under* some elements of the environment `{DispWithArrows}`.

```

1265     \dim_zero_new:N \l_@@_delim_wd_dim
1266     \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
1267     \box_clear_new:N \l_@@_left_brace_box
1268     \hbox_set:Nn \l_@@_left_brace_box
1269     {
1270         \group_begin:
1271         \cs_set_eq:NN \label \@@_old_label
1272         \c_math_toggle_token
1273         \bool_if:NT \l_@@_displaystyle_bool \displaystyle
1274         \l_@@_left_brace_tl
1275         { }
1276         \c_math_toggle_token
1277     \group_end:
1278     }
1279 }

```

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

```

1280 <*LaTeX>
1281     \tl_clear_new:N \l_@@_tag_tl

1282     \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.

```

1283     \bool_set_false:N \l_@@_tag_star_bool
1284 </LaTeX>

1285     \if_mode_math:
1286         \@@_fatal:n { DispNetArrows~in~math~mode }
1287     \fi:

```

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

```

1288 <*plain-TeX>
1289     \dim_zero_new:N \linewidth
1290     \dim_set_eq:NN \linewidth \displaywidth
1291 </plain-TeX>
1292 <*LaTeX>
1293     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1294     {
1295         \noindent
1296         \c_math_toggle_token
1297     }
1298     {
1299 </LaTeX>

```

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.

```

1300     \if_mode_vertical:
1301     \nointerlineskip
1302     \hbox_to_wd:nn { .6 \linewidth } { }
1303     \fi:
1304     \c_math_toggle_token \c_math_toggle_token
1305 <*LaTeX>
1306     }
1307 </LaTeX>

1308     \dim_zero_new:N \l_@@_linewidth_dim
1309 <*LaTeX>
1310     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1311     { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
1312     { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
1313 </LaTeX>

```

```

1314 ⟨*plain-TeX⟩
1315   \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
1316 ⟨/plain-TeX⟩

1317   \box_clear_new:N \l_@@_halign_box
1318   \setbox \l_@@_halign_box \vtop \bgroup
1319   \tabskip =
1320     \bool_if:NTF \l_@@_fleqn_bool
1321     \c_zero_skip
1322     { 0 pt plus 1000 pt minus 1000 pt }

```

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).

```

1323   \spread@equation
1324   \@@_construct_halign:
1325   \tabskip = 0 pt plus 1000 pt minus 1000 pt
1326   &

```

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

```

1327   $ ## $
1328   \tabskip = \c_zero_skip
1329   &&
1330   \@@_fatal:n { Too~much~columns~in~DispWithArrows }
1331   \bool_if:nT \c_false_bool { ## }
1332   \cr
1333   }

```

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

```

1334 ⟨*plain-TeX⟩
1335 \cs_new_protected:Npn \endDispWithArrows
1336 ⟨/plain-TeX⟩
1337 {
1338 ⟨*LaTeX⟩
1339   \clist_if_in:NnT \l_@@_tags_clist { last }
1340   { \clist_set:Nn \l_@@_tags_clist { all } }
1341 ⟨/LaTeX⟩
1342   \

```

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

```

1343   \egroup
1344   \unskip \unpenalty \unskip \unpenalty
1345   \box_set_to_last:N \l_tmpa_box
1346   \nointerlineskip
1347   \box_use:N \l_tmpa_box
1348   \dim_gzero_new:N \g_@@_alignment_dim
1349   \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
1350   \box_clear_new:N \l_@@_new_box
1351   \hbox_set:Nn \l_@@_new_box { \hbox_unpack_drop:N \l_tmpa_box }
1352   \dim_compare:nNnT
1353     { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
1354     { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }

```

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

```

1355   \egroup
1356   \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
1357     { \box_use_drop:N \l_@@_halign_box }
1358     {
1359       \hbox_to_wd:nn \l_@@_linewidth_dim

```

```

1360     {
1361       \bool_if:NTF \l_@@_fleqn_bool
1362       { \skip_horizontal:N \l_@@_mathindent_skip }
1363       \hfil
1364       \hbox_to_wd:nn \g_@@_alignment_dim
1365       {
1366         \box_use_drop:N \l_@@_left_brace_box

```

Here, you should use `\box_ht_plus_dp:N` when TeXLive 2021 will be available on Overleaf.

```

1367         \dim_set:Nn \l_tmpa_dim
1368         {
1369           \box_ht:N \l_@@_halign_box
1370           + \box_dp:N \l_@@_halign_box
1371         }
1372         \group_begin:
1373         \dim_zero:N \nulldelimiterspace
1374         \c_math_toggle_token
1375         \left \l_@@_replace_left_brace_by_tl
1376         \vcenter to \l_tmpa_dim { \vfil }
1377         \right.
1378         \c_math_toggle_token
1379         \group_end:
1380         \hfil
1381       }
1382     \hfil
1383   }
1384   \skip_horizontal:N -\l_@@_linewidth_dim
1385   \vcenter { \box_use_drop:N \l_@@_halign_box }
1386 }

```

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.

```

1387   \dim_gzero_new:N \g_@@_right_x_dim
1388   \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
1389   \pgfpicture
1390   \pgfrememberpicturepositiononpagetrue
1391   \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
1392   {
1393     \cs_if_free:cTF
1394     { pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
1395     { \@@_fatal:n { Inexistent-v-node } }
1396     {
1397       \pgfpointanchor
1398       { wa - \l_@@_prefix_str - \l_tmpa_int - v }
1399       { center }
1400       \dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
1401       { \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x }
1402     }
1403   }
1404   \endpgfpicture

```

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

```

1405   \@@_post_halign:

```

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

```

1406   <*LaTeX>
1407   \IfPackageLoadedT { mathtools }
1408   { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }

```

```

1409   \bool_if:NTF \l_@@_in_label_or_minipage_bool
1410     {
1411       \c_math_toggle_token
1412       \skip_vertical:N \belowdisplayskip
1413     }
1414     { \c_math_toggle_token \c_math_toggle_token }
1415 </LaTeX>
1416 <*plain-TeX>
1417     \c_math_toggle_token \c_math_toggle_token
1418 </plain-TeX>
1419 <*LaTeX>
1420   \bool_if:NT \l_@@_subequations_bool { \end { subequations } }

```

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`).

```

1421   \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
1422 </LaTeX>
1423 <*plain-TeX>
1424   \group_end:
1425 </plain-TeX>
1426 <*LaTeX>
1427   \ignorespacesafterend
1428 </LaTeX>
1429 }

```

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.

```

1430 <*LaTeX>
1431 \NewDocumentEnvironment { DispWithArrows* } { }
1432 {
1433   \WithArrowsOptions { notag }
1434   \DispWithArrows
1435 }
1436 \endDispWithArrows
1437 </LaTeX>

```

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

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

```

1438 \cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
1439 {
1440   \bool_if:NTF \l_@@_in_WithArrows_bool
1441     { \@@_error:nn { Not~allowed~in~WithArrows } { #1 } }
1442     {
1443       \int_compare:nNnTF \g_@@_col_int < \l_@@_nb_cols_int
1444         { \@@_error:nn { Not~allowed~in~DispWithArrows } { #1 } }
1445         { #2 }
1446     }
1447 }

```

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

```

1448 <*LaTeX>
1449 \cs_new_protected:Npn \@@_notag:
1450   { \@@_if_in_last_col_of_disp:Nn \notag { \clist_clear:N \l_@@_tags_clist } }

```

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

```

1451 \cs_new_protected:Npn \@@_nonumber:
1452   { \@@_if_in_last_col_of_disp:Nn \nonumber { \clist_clear:N \l_@@_tags_clist } }

```

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

```

1453 \NewDocumentCommand \@@_tag { s m }
1454   {
1455     \@@_if_in_last_col_of_disp:Nn \tag
1456     {
1457       \tl_if_empty:NF \l_@@_tag_tl
1458       { \@@_error:nn { Multiple~tags } { #2 } }
1459       \clist_set:Nn \l_@@_tags_clist { all }
1460       \IfPackageLoadedT { mathtools }
1461       {
1462         \MH_if_boolean:nT { show_only_refs }
1463         {
1464           \MH_if_boolean:nF { show_manual_tags }
1465           { \clist_clear:N \l_@@_tags_clist }
1466         }
1467       }
1468       \tl_set:Nn \l_@@_tag_tl { #2 }
1469       \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`.<sup>37</sup>

```

1470     \bool_if:nT { #1 }
1471     {
1472       \IfPackageLoadedF { amsmath }
1473       { \@@_error:n { tag*~without~amsmath } }
1474     }
1475   }
1476 }

```

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`.

```

1477 \cs_new_protected:Npn \@@_label:n #1
1478   {
1479     \@@_if_in_last_col_of_disp:Nn \label
1480     {
1481       \seq_if_empty:NF \l_@@_labels_seq
1482       {
1483         \IfPackageLoadedTF { cleveref }
1484         { \@@_error:n { Multiple~labels~with~cleveref } }
1485         { \@@_error:n { Multiple~labels } }
1486       }
1487       \seq_put_right:Nn \l_@@_labels_seq { #1 }
1488       \IfPackageLoadedT { mathtools }
1489       {
1490         \MH_if_boolean:nT { show_only_refs }

```

---

<sup>37</sup>There are two versions of `@eqnnum`, a standard version and a version for the option `leqno`.

```

1491     {
1492       \cs_if_exist:cTF { MT_r_#1 }
1493       { \clist_set:Nn \l_@@_tags_clist { all } }
1494       { \clist_clear:N \l_@@_tags_clist }
1495     }
1496   }
1497   \IfPackageLoadedT { autonum }
1498   {
1499     \cs_if_exist:cTF { autonum@#1Referenced }
1500     { \clist_set:Nn \l_@@_tags_clist { all } }
1501     { \clist_clear:N \l_@@_tags_clist }
1502   }
1503 }
1504 }

```

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

```

1505 \cs_new_protected:Npn \@@_tagnextline:
1506 {
1507   \@@_if_in_last_col_of_disp:Nn \tagnextline
1508   { \bool_set_true:N \l_@@_tag_next_line_bool }
1509 }

```

The environments `{DispWithArrows}` and `{DispWithArrows*}` are compliant with the command `\qedhere` of `amsthm`. However, this compatibility requires a special version of `\qedhere`. This special version is called `\@@_qedhere:` and will be linked with `\qedhere` in the last column of the environment `{DispWithArrows}` (only if the package `amsthm` has been loaded). `\@@_qedhere:` raises the boolean `\l_@@_qedhere_bool`.

```

1510 \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }
1511 \cs_new_protected:Npn \@@_set_qedhere: { \cs_set_eq:NN \qedhere \@@_qedhere: }

```

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

```

1512 \cs_new_protected:Npn \@@_qedhere_i:
1513 {
1514   \group_begin:
1515   \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.

```

1516   \cs_set_eq:NN \qed@elt \setQED@elt
1517   \QED@stack \relax \relax
1518   \group_end:
1519 }
1520 </LaTeX>

```

## 12.11 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.

```

1521 \cs_new_protected:Npn \@@_scan_arrows:
1522 {
1523   \group_begin:
1524   \int_compare:nNnT \l_@@_pos_arrow_int = 7
1525     {
1526       \@@_scan_arrows_i:
1527       \int_set:Nn \l_@@_pos_arrow_int 8
1528     }
1529   \@@_scan_arrows_i:
1530   \group_end:
1531 }

1532 \cs_new_protected:Npn \@@_scan_arrows_i:
1533 {

```

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

```

1534   \int_zero_new:N \l_@@_first_arrow_of_group_int
1535   \int_zero_new:N \l_@@_first_line_of_group_int
1536   \int_zero_new:N \l_@@_last_line_of_group_int
1537   \seq_clear_new:N \l_@@_first_arrows_seq
1538   \seq_clear_new:N \l_@@_last_arrows_seq

```

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

```

1539   \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.

```

1540   \int_set:Nn \l_@@_arrow_int 1
1541   \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1542   {

```

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.

```

1543     \prop_get:cnN
1544     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1545     { initial } \l_tmpa_tl
1546     \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1547     \prop_get:cnN
1548     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1549     { final } \l_tmpa_tl
1550     \int_set:Nn \l_@@_final_int \l_tmpa_tl
1551     \prop_get:cnN
1552     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1553     { status } \l_@@_status_arrow_str
1554     \prop_get:cnN
1555     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1556     { 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`.

```

1557     \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1558     {
1559         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1560         { \@@_error:n { Too-few-lines-for-an-arrow } }
1561     }
1562     \@@_treat_an_arrow_in_scan:

```

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

```

1563     \int_incr:N \l_@@_arrow_int
1564 }

```

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 arrive after the last line of the environment).

```

1565     \bool_lazy_and:nnT
1566     { ! \int_compare_p:nNn \l_@@_pos_arrow_int = 7 }
1567     { \int_compare_p:nNn \l_@@_first_arrow_of_group_int > \c_zero_int }
1568     { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1569 }

```

The following command is only for the lisibility of the code. It's used only once. Its name may be misleading. Indeed, it treats an arrow in the scan but it *may* trigger the construction of all arrows of a group if it detects that a group has just been completed (with `\@@_draw_arrows:nn`)

```

1570 \cs_new_protected:Npn \@@_treat_an_arrow_in_scan:
1571 {

```

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

```

1572     \bool_lazy_and:nnT
1573     { \int_compare_p:nNn \l_@@_arrow_int > 1 }
1574     {
1575         \bool_lazy_or:p:nn
1576         {
1577             \bool_lazy_and_p:nn
1578             {
1579                 \int_compare_p:nNn
1580                 \l_@@_initial_int > \l_@@_last_line_of_group_int
1581             }
1582             { \bool_not_p:n { \int_compare_p:nNn \l_@@_pos_arrow_int = 7 } }
1583         }
1584         { \str_if_eq_p:on \l_@@_status_arrow_str { new-group } }
1585     }
1586     {
1587         \int_if_zero:nF \l_@@_first_arrow_of_group_int
1588         {
1589             \@@_draw_arrows:nn
1590             \l_@@_first_arrow_of_group_int
1591             { \l_@@_arrow_int - 1 }
1592         }
1593         \bool_set_true:N \l_@@_new_group_bool
1594     }

```

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 conditional, 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`.

```

1595     \bool_if:nTF \l_@@_new_group_bool
1596     {
1597         \bool_set_false:N \l_@@_new_group_bool
1598         \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1599         \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1600         \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1601         \seq_clear:N \l_@@_first_arrows_seq
1602         \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
1603         \seq_clear:N \l_@@_last_arrows_seq
1604         \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.

```

1605         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1606         { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1607     }

```

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

```

1608     {

```

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

```

1609         \str_if_eq:onF \l_@@_status_arrow_str { independent }
1610         {

```

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

```

1611             \int_compare:nNnT \l_@@_initial_int = \l_@@_first_line_of_group_int
1612             { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1613             \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1614             {
1615                 \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1616                 \seq_clear:N \l_@@_last_arrows_seq
1617                 \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
1618             }
1619             {
1620                 \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1621                 { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1622             }
1623         }
1624     }

```

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.

```

1625         \str_if_eq:onF \l_@@_status_arrow_str { independent }
1626         {
1627             \int_compare:nNnF \l_@@_pos_arrow_int = 8
1628             { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1629         }
1630     }

```

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`.

```

1631 \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1632   {
1633     \group_begin:
1634     \int_zero_new:N \l_@@_first_arrow_int
1635     \int_set:Nn \l_@@_first_arrow_int { #1 }
1636     \int_zero_new:N \l_@@_last_arrow_int
1637     \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.

```

1638     \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1639     \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1640     {

```

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.

```

1641         \prop_get:cnN
1642         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1643         { initial } \l_tmpa_tl
1644         \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1645         \prop_get:cnN
1646         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1647         { final } \l_tmpa_tl
1648         \int_set:Nn \l_@@_final_int \l_tmpa_tl
1649         \prop_get:cnN
1650         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1651         { status } \l_@@_status_arrow_str

```

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).

```

1652         \int_compare:nNnF \l_@@_final_int > \g_@@_line_int

```

If the arrow is of type `over` (key `o`), we don’t draw that arrow now (those arrows will be drawn after all the other arrows).

```

1653     {
1654         \str_if_eq:onTF \l_@@_status_arrow_str { over }
1655         { \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
1656         \@@_draw_arrow:
1657     }
1658     \int_incr:N \l_@@_arrow_int
1659   }
1660   \@@_draw_o_arrows_of_the_group:
1661   \group_end:
1662 }

```

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).

```

1663 \cs_new_protected:Npn \@@_draw_arrow:
1664   {
1665     \group_begin:

```

We process the options of the current arrow.

```

1666 \prop_get:cnN
1667   { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1668   { options }
1669   \l_tmpa_tl
1670 \str_clear_new:N \l_@@_previous_key_str

```

We will expand the second argument of `\keys_set_known:nn` exactly three times. Maybe that an e-expansion would be possible but, in the past, there were problems with fragile commands such as `\bfseries` in the option `font` of the option `tikz` (it seems that this no longer the case).

```

1671 \exp_args:NNNno \exp_args:NNno \exp_args:Nno
1672 \keys_set_known:nn
1673   { WithArrows / Arrow / SecondPass }
1674   { \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`:

```

1675 \bool_set_false:N \l_@@_initial_r_bool
1676 \bool_set_false:N \l_@@_final_r_bool
1677 \int_case:nn \l_@@_pos_arrow_int
1678   {
1679     0 { \bool_set_true:N \l_@@_final_r_bool }
1680     2 { \bool_set_true:N \l_@@_initial_r_bool }
1681     3
1682     {
1683       \bool_set_true:N \l_@@_initial_r_bool
1684       \bool_set_true:N \l_@@_final_r_bool
1685     }
1686   }

```

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

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

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

```

1687 \int_compare:nNnT \l_@@_pos_arrow_int = 5
1688   {
1689     \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1690     \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1691   }

```

`\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`.

```

1692 \tl_set:Ne \l_@@_initial_tl
1693   { \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl }
1694 \tl_set:Ne \l_@@_final_tl
1695   { \int_use:N \l_@@_final_int - \bool_if:NTF \l_@@_final_r_bool rl }

```

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

```

1696 \prop_get:cnN
1697   { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1698   { label }
1699   \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 of the command `\seq_if_in:NnTF` which is *not* expandable.

```

1700   \seq_if_in:NeTF \l_@@_first_arrows_seq
1701     { \int_use:N \l_@@_arrow_int }
1702     { \bool_set_true:N \l_tmpa_bool }
1703     { \bool_set_false:N \l_tmpa_bool }
1704   \seq_if_in:NeTF \l_@@_last_arrows_seq
1705     { \int_use:N \l_@@_arrow_int }
1706     { \bool_set_true:N \l_tmpb_bool }
1707     { \bool_set_false:N \l_tmpb_bool }
1708   \int_compare:nNnT \l_@@_pos_arrow_int = 5
1709     {
1710       \bool_set_true:N \l_tmpa_bool
1711       \bool_set_true:N \l_tmpb_bool
1712     }

```

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`).

```

1713   \dim_gzero_new:N \g_@@_x_initial_dim
1714   \dim_gzero_new:N \g_@@_x_final_dim
1715   \dim_gzero_new:N \g_@@_y_initial_dim
1716   \dim_gzero_new:N \g_@@_y_final_dim
1717   \pgfpicture
1718     \pgfrememberpicturepositiononpagetrue
1719     \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south }
1720     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1721     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1722     \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north }
1723     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1724     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1725   \endpgfpicture
1726   \bool_lazy_and:nnTF
1727     {
1728       \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
1729         > \l_@@_max_length_of_arrow_dim
1730     }
1731     { \int_compare_p:nNn { \l_@@_final_int - \l_@@_initial_int } = 1 }
1732     {
1733       \tl_gset:Ne \g_tmpa_tl
1734         {
1735           \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1736             { \dim_use:N \g_@@_x_initial_dim }
1737             { \dim_use:N \l_@@_x_dim } ,
1738           \dim_eval:n
1739             {
1740               ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1741               + 0.5 \l_@@_max_length_of_arrow_dim
1742             }
1743         }
1744       \tl_gset:Ne \g_tmpb_tl
1745         {

```

```

1746     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1747       { \dim_use:N \g_@@_x_final_dim }
1748       { \dim_use:N \l_@@_x_dim } ,
1749     \dim_eval:n
1750     {
1751       ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1752       - 0.5 \l_@@_max_length_of_arrow_dim
1753     }
1754   }
1755 }
1756 {
1757   \tl_gset:Ne \g_tmpa_tl
1758   {
1759     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1760       { \dim_use:N \g_@@_x_initial_dim }
1761       { \dim_use:N \l_@@_x_dim } ,
1762     \bool_if:NTF \l_tmpa_bool
1763       { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
1764       { \dim_use:N \g_@@_y_initial_dim }
1765   }
1766   \tl_gset:Ne \g_tmpb_tl
1767   {
1768     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1769       { \dim_use:N \g_@@_x_final_dim }
1770       { \dim_use:N \l_@@_x_dim } ,
1771     \bool_if:NTF \l_tmpb_bool
1772       { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
1773       { \dim_use:N \g_@@_y_final_dim }
1774   }
1775 }

```

The dimension `\l_@@_delta_x_dim` is the difference of abscissa between the right side of the alignment (`\halign`) and the left side of the arrow.

```

1776   \bool_if:NF \l_@@_right_overlap_bool
1777   {
1778     \bool_if:NT \l_@@_in_WithArrows_bool
1779     {
1780       \pgfpicture
1781       \pgfrememberpicturepositiononpagetrue
1782       \pgfpointanchor { wa - \l_@@_prefix_str - 1 - r } { south }
1783       \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1784       {
1785         \dim_set:Nn \l_@@_delta_x_dim
1786         {
1787           \pgf@x -
1788           ( \dim_min:nn \g_@@_x_initial_dim \g_@@_x_final_dim )
1789         }
1790       }
1791       { \dim_set:Nn \l_@@_delta_x_dim { \pgf@x - \l_@@_x_dim } }
1792     \endpgfpicture
1793   }
1794 }

```

Eventually, we can draw the arrow with the code in `\l_@@_tikz_code_tl`. We recall that the value by default for this token list is: “`\draw (#1) to node {#3} (#2) ;`”. This value can be modified with the option `tikz-code`. We use the variant `\@@_draw_arrow:nno` of the macro `\@@_draw_arrow:nnn` because of the characters *underscore* in the name `\l_tmpa_tl`: if the user uses the Tikz library `babel`, the third argument of the command `\@@_draw_arrow:nno` will be rescanned because this third argument will be in the argument of a command `node` of an instruction `\draw` of Tikz... and we will have an error because of the characters *underscore*.<sup>38</sup>

---

<sup>38</sup>There were other solutions: use another name without *underscore* (like `\ltmpat1`) or use the package `underscore`

```
1795 \@@_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).

```
1796 \group_end:
1797 }
```

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_arrow:n` which will create the function `\@@_arrow:nnn`.

```
1798 \cs_generate_variant:Nn \@@_def_function_arrow:n { o }
1799 \cs_new_protected:Npn \@@_def_function_arrow:n #1
1800 {
1801   \cs_set:Npn \@@_arrow:nnn ##1 ##2 ##3
1802   {
1803     <*LaTeX>
1804     \begin{tikzpicture}
1805     </LaTeX>
1806     <*plain-TeX>
1807     \tikzpicture
1808     </plain-TeX>
1809     [ @@_standard_arrow ]
```

You keep track of the bounding box because we want to compute the total width of the arrow (with the label) for the arrows of type `over` and also for the actualization of `\g_@@_overlap_x_dim`.

```
1810 \pgf@relevantforpicturesizetrue
1811 #1
1812 \dim_compare:nNnTF \pgf@picminx = { 16000 pt }
1813 { \dim_zero:N \l_tmpa_dim }
1814 { \dim_set:Nn \l_tmpa_dim { \pgf@picmaxx - \pgf@picminx } }
1815 \dim_add:Nn \l_tmpa_dim \l_@@_xoffset_dim
```

`\l_@@_arrow_int = 0` probably means that we have an arrow in the code-after.

```
1816 \int_compare:nNnT \l_@@_arrow_int > 0 % added 2024/10/01
1817 {
1818   \prop_gput:cnV
1819   { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1820   { width }
1821   \l_tmpa_dim
1822 }
```

Now, the actualization of `\g_@@_overlap_x_dim`.

```
1823 \bool_if:NF \l_@@_right_overlap_bool
1824 {
1825   \bool_if:NT \l_@@_in_WithArrows_bool
1826   {
1827     \dim_gset:Nn \g_@@_overlap_x_dim
1828     {
1829       \dim_max:nn
1830       \g_@@_overlap_x_dim
1831       { \l_tmpa_dim - \l_@@_delta_x_dim }
1832     }
1833   }
1834 }
1835 \pgfresetboundingbox
1836 <*LaTeX>
1837 \end{tikzpicture}
1838 </LaTeX>
1839 <*plain-TeX>
1840 \endtikzpicture
1841 </plain-TeX>
```

---

(with this package, the characters *underscore* will be rescanned without errors, even in text mode).

```

1842     }
1843 }

```

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

```

1844 \cs_generate_variant:Nn \@@_draw_arrow:nnn { n n o }
1845 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1846 {

```

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

```

1847     \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
1848     { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }

```

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

```

1849     \@@_def_function_arrow:o \l_@@_tikz_code_tl
1850     \@@_arrow:nnn { #1 } { #2 } { #3 }
1851 }

```

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

```

1852 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1853 {
1854     \pgfset { inner~sep = 0pt }

```

First, we draw the arrow without the label.

```

1855     \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;

```

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

```

1856     \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }

```

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

```

1857     \dim_set:Nn \l_tmpa_dim { \g_@@_right_x_dim - \pgf@x - 0.33333 em }

```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “text width”.<sup>39</sup>

```

1858     \path \pgfextra { \tl_gset:Ne \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`).

```

1859     \tl_if_empty:NF \g_tmpa_tl
1860     {
1861         \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1862         \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1863         { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1864     }

```

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

```

1865     \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1866     {
1867         \path ( @@_label.west )
1868         <LaTeX>
1869         node [ anchor = west ]
1870         {
1871             \skip_horizontal:n { 0.33333 em }
1872             \begin { minipage } { \l_tmpa_dim }
1873             \tikz@text@action
1874             \pgfkeysgetvalue { / tikz / node~halign~header } \l_tmpa_tl
1875             \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@left@header }
1876             { \pgfutil@raggedright }

```

---

<sup>39</sup>In 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.



```

1877         {
1878             \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@right@header }
1879             { \pgfutil@raggedleft }
1880             {
1881                 \tl_if_eq:NnT \l_tmpa_tl { \tikz@align@center@header }
1882                 { \centering }
1883             }
1884         }
1885     #3
1886     \end { minipage }
1887 } ;
1888 \end { LaTeX }
1889 \end { *plain-TeX }
1890 node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1891 { \skip_horizontal:n { 0.33333 em } #3 } ;
1892 \end { plain-TeX }
1893 }
1894 }

```

### 12.11.1 The command `update_x`

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`).

```

1895 \cs_new_protected:Npn \@@_update_x:nn #1 #2
1896 {
1897     \dim_gset_eq:NN \g_tmpa_dim \l_@@_x_dim
1898     \pgfpicture
1899     \pgfrememberpicturepositiononpagetrue
1900     \int_step_inline:nnn { #1 } { #2 }
1901     {
1902         \pgfpointanchor { wa - \l_@@_prefix_str - ##1 - 1 } { center }
1903         \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \pgf@x }
1904     }
1905     \endpgfpicture
1906     \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1907 }

```

### 12.11.2 We draw the arrows of type `o`

We recall that the arrows of type `o` will be drawn *over* (hence the letter `o`) the other arrows. The arrows of type `o` are available only when the option `group` or the option `groups` is in force. The arrows of type `o` will be drawn group by group. The command `\@@_draw_o_arrows_of_the_group:` is called after the construction of the (other) arrows of the group.

```

1908 \cs_new_protected:Npn \@@_draw_o_arrows_of_the_group:
1909 {

```

The numbers of the arrows of type `o` we have to draw are in the sequence `\l_@@_o_arrows_seq`. We have to sort that sequence because the order in which these arrows will be drawn matters.

- The arrows which arrive first must be drawn first.
- For arrows with the same final line, the arrows with lower initial line must be drawn after (because they encompass the previous ones).

The second point ensures the expected output in situations such as in the following example:

```


$$\begin{array}{l}
A = B \\
= C \\
= D \\
= E + E
\end{array}$$


```

$\xrightarrow{\text{one}}$ 
 $\xrightarrow{\text{two}}$ 
 $\xrightarrow{\text{three}}$

```

1910 \seq_sort:Nn \l_@@_o_arrows_seq
1911 {
1912   \prop_get:cnN
1913   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1914   { final } \l_tmpa_tl

```

We recall that `\prop_get:cnN` retrieves token lists (here `\l_tmpa_tl` and `\l_tmpb_tl`). We don't need to do an explicit conversion in L3 integers because such token lists can be used directly in `\int_compare:nNnTF`.

```

1915   \prop_get:cnN
1916   { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1917   { final } \l_tmpb_tl
1918   \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1919   \sort_return_same:
1920   {
1921     \int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
1922     \sort_return_swapped:
1923     {
1924       \prop_get:cnN
1925       { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1926       { initial } \l_tmpa_tl
1927       \prop_get:cnN
1928       { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1929       { initial } \l_tmpb_tl
1930       \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1931       \sort_return_swapped:
1932       \sort_return_same:
1933     }
1934   }
1935 }

```

Now, we can draw the arrows of type `o` of the group in the order of the sequence.

```

1936 \seq_map_inline:Nn \l_@@_o_arrows_seq
1937 {

```

We retrieve the initial row and the final row of the arrow.

```

1938   \prop_get:cnN
1939   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1940   { initial } \l_tmpa_tl
1941   \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1942   \prop_get:cnN
1943   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1944   { final } \l_tmpa_tl
1945   \int_set:Nn \l_@@_final_int \l_tmpa_tl

```

The string `\l_@@_input_line_str` will be used only in some error messages.

```

1946   \prop_get:cnN
1947   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1948   { input-line } \l_@@_input_line_str

```

We have to compute the maximal width of all the arrows (with their labels) which are covered by our arrow. We will compute that dimension in `\g_tmpa_dim`. We need a global dimension because we will have to exit a `\pgfpicture`.

```

1949   \dim_gzero:N \g_tmpa_dim

```

We will raise the boolean `\g_tmpa_bool` if we find an arrow “under” our arrow (we should find at least once since you are drawing an arrow of type o: if not, we will raise an error<sup>40</sup>).

```

1950     \bool_set_false:N \g_tmpa_bool
1951     \pgfpicture
1952     \pgfrememberpicturepositiononpagetrue
1953     \int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
1954     {
1955         \prop_get:cnN
1956         { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1957         { initial } \l_tmpa_tl
1958         \prop_get:cnN
1959         { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1960         { final } \l_tmpb_tl
1961         \prop_get:cnN
1962         { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1963         { status } \l_@@_status_arrow_str
1964         \bool_lazy_any:nF
1965         {
1966             { \int_compare_p:n { ##1 = #####1 } }
1967             { \int_compare_p:nNn \l_@@_initial_int > \l_tmpa_tl }
1968             { \int_compare_p:nNn \l_tmpb_tl > \l_@@_final_int }

```

We don’t take into account the independent arrows because we have only computed the *width* of the arrows and that’s why our arrow of type o will be positioned only relatively to the current group.

```

1969         { \str_if_eq_p:on \l_@@_status_arrow_str { independent } }
1970     }
1971     {

```

The total width of the arrow (with its label) has been stored in a “field” of the arrow.

```

1972         \bool_gset_true:N \g_tmpa_bool
1973         \prop_get:cnN
1974         { g_@@ _ arrow _ \l_@@_prefix_str _ #####1 _ prop }
1975         { width }
1976         \l_tmpa_tl

```

We have to do a global affectation in order to exit the `pgfpicture`.

```

1977         \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \l_tmpa_tl }
1978     }
1979 }
1980 \endpgfpicture

```

The boolean `\g_tmpa_bool` is raised if at least one arrow has been found “under” our arrow (it should be the case since we are drawing an arrow of type o).

```

1981     \bool_if:NTF \g_tmpa_bool
1982     {
1983         \int_set:Nn \l_@@_arrow_int { ##1 }
1984         \dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
1985         \dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
1986         \@@_draw_arrow:
1987     }
1988     { \@@_error:n { o~arrow~with~no~arrow~under } }
1989 }
1990 }

```

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.

```

1991 <LaTeX>
1992 \NewExpandableDocumentCommand \WithArrowsLastEnv { }

```

---

<sup>40</sup>Maybe we will change that in future versions.

```

1993 { \int_use:N \g_@@_last_env_int }
1994 </LaTeX>
1995 <*plain-TeX>
1996 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
1997 </plain-TeX>

```

## 12.12 The command `\Arrow` in `code-after`

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

```

1998 \keys_define:nn { WithArrows / Arrow / code-after }
1999 {
2000   tikz      .code:n =
2001     \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
2002   tikz      .value_required:n = true ,
2003   rr        .value_forbidden:n = true ,
2004   rr        .code:n = \@@_fix_pos_option:n 0 ,
2005   ll        .value_forbidden:n = true ,
2006   ll        .code:n = \@@_fix_pos_option:n 1 ,
2007   rl        .value_forbidden:n = true ,
2008   rl        .code:n = \@@_fix_pos_option:n 2 ,
2009   lr        .value_forbidden:n = true ,
2010   lr        .code:n = \@@_fix_pos_option:n 3 ,
2011   v         .value_forbidden:n = true ,
2012   v         .code:n = \@@_fix_pos_option:n 4 ,
2013   tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
2014   tikz-code .value_required:n = true ,
2015   xoffset   .dim_set:N = \l_@@_xoffset_dim ,
2016   xoffset   .value_required:n = true ,
2017   unknown   .code:n =
2018     \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
2019     \@@_error:n { Unknown-option-Arrow-in-code-after }
2020 }

```

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

```

2021 \seq_new:N \l_@@_options_Arrow_code_after_seq
2022 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq
2023 { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }

2024 <*LaTeX>
2025 \NewDocumentCommand \@@_Arrow_code_after { 0 { } m m m ! 0 { } }
2026 </LaTeX>
2027 <*plain-TeX>
2028 \cs_new_protected:Npn \@@_Arrow_code_after
2029 {
2030   \peek_meaning:NTF [
2031     { \@@_Arrow_code_after_i }
2032     { \@@_Arrow_code_after_i [ ] }
2033   }
2034 \cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4
2035 {
2036   \peek_meaning:NTF [
2037     { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
2038     { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
2039   }
2040 \cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
2041 </plain-TeX>

```

```

2042 {
2043   \int_set_eq:NN \l_@@_pos_arrow_int \c_one_int
2044   \str_clear_new:N \l_@@_previous_key_str
2045   \group_begin:
2046     \keys_set:nn { WithArrows / Arrow / code-after }
2047     { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
2048     \bool_set_false:N \l_@@_initial_r_bool
2049     \bool_set_false:N \l_@@_final_r_bool
2050     \int_case:nn \l_@@_pos_arrow_int
2051     {
2052       0
2053       {
2054         \bool_set_true:N \l_@@_initial_r_bool
2055         \bool_set_true:N \l_@@_final_r_bool
2056       }
2057       2 { \bool_set_true:N \l_@@_initial_r_bool }
2058       3 { \bool_set_true:N \l_@@_final_r_bool }
2059     }

```

We prevent drawing an arrow from a line to itself.

```

2060   \tl_if_eq:nnTF { #2 } { #3 }
2061   { \@@_error:nn { Both-lines-are-equal } { #2 } }

```

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

```

2062 {
2063   \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - 1 }
2064   { \@@_error:ne { Wrong-line-in-Arrow } { #2 } }
2065   {
2066     \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - 1 }
2067     { \@@_error:ne { Wrong-line-in-Arrow } { #3 } }
2068     {
2069       \int_compare:nNnTF \l_@@_pos_arrow_int = 4
2070       {
2071         \pgfpicture
2072         \pgfrememberpicturepositiononpagetrue
2073         \pgfpointanchor { wa - \l_@@_prefix_str - #2 - 1 }
2074         { south }
2075         \dim_set_eq:NN \l_tmpa_dim \pgf@x
2076         \dim_set_eq:NN \l_tmpb_dim \pgf@y
2077         \pgfpointanchor { wa - \l_@@_prefix_str - #3 - 1 }
2078         { north }
2079         \dim_set:Nn \l_tmpa_dim
2080         { \dim_max:nn \l_tmpa_dim \pgf@x }
2081         \tl_gset:Ne \g_tmpa_tl
2082         { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
2083         \tl_gset:Ne \g_tmpb_tl
2084         { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
2085         \endpgfpicture
2086       }
2087     {
2088       \pgfpicture
2089       \pgfrememberpicturepositiononpagetrue
2090       \pgfpointanchor
2091       {
2092         wa - \l_@@_prefix_str -
2093         #2 - \bool_if:NTF \l_@@_initial_r_bool r l
2094       }
2095       { south }
2096       \tl_gset:Ne \g_tmpa_tl
2097       { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
2098       \pgfpointanchor
2099       {
2100         wa - \l_@@_prefix_str -

```

```

2101             #3 - \bool_if:NTF \l_@@_final_r_bool r l
2102             }
2103             { north }
2104             \tl_gset:Ne \g_tmpb_tl
2105             { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
2106             \endpgfpicture
2107             }
2108             \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
2109         }
2110     }
2111 }
2112 \group_end:
2113 }

```

## 12.13 The command `\MultiArrow` in code-after

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

```

2114 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
2115 {

```

The user of the command `\MultiArrow` (in code-after) will be able to specify the list of lines with the same syntax as the loop `\foreach` of `pgffor`. First, we test with a regular expression whether the format of the list of lines is correct.

```

2116     \exp_args:Nne
2117     \regex_match:nnTF
2118     { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
2119     { #1 }
2120     { \@@_MultiArrow_i:nn { #1 } { #2 } }
2121     { \@@_error:ne { Invalid-specification-for-MultiArrow } { #1 } }
2122 }
2123 \cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
2124 {

```

That’s why we construct a “clist” of L3 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 will construct the list in `\g_tmpa_clist`.

```

2125     \foreach \x in { #1 }
2126     {
2127         \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
2128         { \@@_error:ne { Wrong-line-specification-in-MultiArrow } \x }
2129         { \clist_gput_right:Ne \g_tmpa_clist \x }
2130     }

```

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

```

2131     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
2132     { \@@_error:n { Too-small-specification-for-MultiArrow } }
2133     {
2134         \clist_sort:Nn \g_tmpa_clist
2135         {
2136             \int_compare:nNnTF { ##1 } > { ##2 }
2137             \sort_return_swapped:
2138             \sort_return_same:
2139         }

```

We extract the minimum in `\l_tmpa_tl` (it must be an integer but we store it in a token list of L3).

```

2140     \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...

```

2141     \clist_reverse:N \g_tmpa_clist
2142     \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).

```
2143 \exp_args:No \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
2144 <*LaTeX>
2145 \begin { tikzpicture }
2146 </LaTeX>
2147 <*plain-TeX>
2148 \tikzpicture
2149 </plain-TeX>
2150 [
2151 \@@_standard ,
2152 every~path / .style = { WithArrows / arrow }
2153 ]
2154 \draw [<->] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
2155 -- ++(5mm,0)
2156 -- node (@@_label) {}
2157 ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
2158 -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south) ;
2159 \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }
2160 \dim_set:Nn \l_tmpa_dim { 20 cm }
2161 \path \pgfextra { \tl_gset:Ne \g_tmpa_tl \tikz@text@width } ;
2162 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
2163 \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
2164 {
2165 \dim_set:Nn \l_tmpb_dim
2166 { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
2167 \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
2168 { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
2169 }
2170 \path (@@_label.west)
2171 node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
2172 <*LaTeX>
2173 \end { tikzpicture }
2174 </LaTeX>
2175 <*plain-TeX>
2176 \endtikzpicture
2177 </plain-TeX>
2178 }
2179 }

2180 \cs_new_protected:Npn \@@_MultiArrow_i:n #1
2181 {
2182 <*LaTeX>
2183 \begin { tikzpicture }
2184 </LaTeX>
2185 <*plain-TeX>
2186 \tikzpicture
2187 </plain-TeX>
2188 [
2189 \@@_standard ,
2190 every~path / .style = { WithArrows / arrow }
2191 ]
2192 \foreach \k in { #1 }
2193 {
2194 \draw [ <- ]
2195 ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
2196 } % ;
2197 <*LaTeX>
2198 \end{tikzpicture}
2199 </LaTeX>
```

```

2200 <*plain-TeX>
2201     \endtikzpicture
2202 </plain-TeX>
2203   }

```

## 12.14 The error messages of the package

```

2204 \bool_if:NTF \c_@@_messages_for_Overleaf_bool
2205   { \str_const:Nn \c_@@_available_keys_str { } }
2206   {
2207     \str_const:Nn \c_@@_available_keys_str
2208     { For~a~list~of~the~available~keys,~type~H~<return>. }
2209   }

```

```

2210 \str_new:N \l_witharrows_body_str

```

The following commands must *not* be protected since they will be used in error messages.

```

2211 \cs_new:Npn \@@_potential_body_i:
2212   {
2213     \str_if_empty:NF \l_witharrows_body_str
2214     { \ If~you~want~to~see~the~body~of~the~environment,~type~H~<return>. }
2215   }

```

```

2216 \cs_new:Npn \@@_potential_body_ii:
2217   {
2218     \str_if_empty:NTF \l_witharrows_body_str
2219     { No~further~help~available }
2220     {
2221       The~body~of~your~environment~was:\\
2222       \l_witharrows_body_str
2223     }
2224   }

```

```

2225 \str_const:Nn \c_@@_option_ignored_str
2226   { If~you~go~on,~this~option~will~be~ignored. }
2227 \str_const:Nn \c_@@_command_ignored_str
2228   { If~you~go~on,~this~command~will~be~ignored. }

```

```

2229 <*LaTeX>
2230 \@@_msg_new:nn { amsmath-not-loaded }
2231   {
2232     amsmath-not-loaded.\\
2233     You~can't~use~the~option~'\l_keys_key_str'~because~the~
2234     package~'amsmath'~has~not~been~loaded.\\
2235     If~you~go~on,~this~option~will~be~ignored~in~the~rest~
2236     of~the~document.
2237   }
2238 </LaTeX>

```

```

2239 \@@_msg_new:nn { Bad~value~for~replace~brace~by }
2240   {
2241     Incorrect~value.\\
2242     Bad~value~for~the~option~'\l_keys_key_str'.~The~value~must~begin~
2243     with~an~extensible~left~delimiter.~The~possible~values~are:~.,
2244     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
2245     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
2246     \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
2247     \token_to_str:N \lfloor\ and~\token_to_str:N \lceil\
2248     (and~\token_to_str:N \lvert\ and~\token_to_str:N \lVert\
2249     if~amsmath~or~unicode~math~is~loaded~in~LaTeX).\\
2250     \c_@@_option_ignored_str
2251   }

```

```

2252 \@@_msg_new:nn { option-of-cr-negative }
2253   {
2254     Bad~value.\\

```



```

2255 The~argument~of~the~command~\token_to_str:N\~
2256 should~be~positive~in~the~row~\int_use:N \g_@@_line_int\
2257 of~your~environment~\{\l_@@_type_env_str\}.\
2258 \c_@@_option_ignored_str
2259 }
2260 \@@_msg_new:nn { omit~probably~used }
2261 {
2262   Strange~problem.\
2263   Maybe~you~have~used~a~command~
2264   \token_to_str:N\omit\ in~the~line~\int_use:N \g_@@_line_int\
2265   (or~another~line)~of~your~environment~\{\l_@@_type_env_str\}.\
2266   You~can~go~on~but~you~may~have~others~errors.
2267 }
2268 <*LaTeX>
2269 \@@_msg_new:nnn { newline~at~the~end~of~env }
2270 {
2271   Incorrect~end.\
2272   The~environments~of~witharrows~(\{WithArrows\}~and~
2273   \{DispWithArrows\})~should~not~end~by~\token_to_str:N \.\
2274   However,~you~can~go~on~for~this~time.~No~similar~error~will~be~
2275   raised~in~this~document.
2276   \@@_potential_body_i:
2277 }
2278 { \@@_potential_body_ii: }
2279 </LaTeX>
2280 \@@_msg_new:nnn { Invalid~option~format }
2281 {
2282   Invalide~value.\
2283   The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
2284   must~not~be~empty.\
2285   \c_@@_option_ignored_str
2286   \@@_potential_body_i:
2287 }
2288 { \@@_potential_body_ii: }
2289 \@@_msg_new:nnn { invalid~key~o }
2290 {
2291   Invalid~use~of~a~key.\
2292   The~key~'o'~for~individual~arrows~can~be~used~only~in~mode~
2293   'group'~or~in~mode~'groups'.\
2294   \c_@@_option_ignored_str
2295   \@@_potential_body_i:
2296 }
2297 { \@@_potential_body_ii: }
2298 \@@_msg_new:nnn { Value~for~a~key }
2299 {
2300   Misuse~of~a~key.\
2301   The~key~'\l_keys_key_str'~should~be~used~without~value. \
2302   However,~you~can~go~on~for~this~time.
2303   \@@_potential_body_i:
2304 }
2305 { \@@_potential_body_ii: }
2306 \@@_msg_new:nnn { Unknown~option~in~Arrow }
2307 {
2308   Unknown~option.\
2309   The~key~'\l_keys_key_str'~is~unknown~for~the~command~
2310   \l_@@_string_Arrow_for_msg_str\ in~the~row~
2311   \int_use:N \g_@@_line_int\ of~your~environment~
2312   \{\l_@@_type_env_str\}. \l_tmpa_str \
2313   \c_@@_option_ignored_str \
2314   \c_@@_available_keys_str
2315 }

```

```

2316 {
2317   The~available~keys~are~(in~alphabetic~order):~
2318   \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
2319 }
2320 \@@_msg_new:nnn { Unknown~option~WithArrows }
2321 {
2322   Unknown~option.\\
2323   The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
2324   \c_@@_option_ignored_str \\
2325   \c_@@_available_keys_str
2326 }
2327 {
2328   The~available~keys~are~(in~alphabetic~order):~
2329   \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.
2330 }
2331 \@@_msg_new:nnn { Unknown~option~DispWithArrows }
2332 {
2333   Unknown~option.\\
2334   The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
2335   \c_@@_option_ignored_str \\
2336   \c_@@_available_keys_str
2337 }
2338 {
2339   The~available~keys~are~(in~alphabetic~order):~
2340   \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.
2341 }
2342 \@@_msg_new:nnn { Unknown~option~WithArrowsOptions }
2343 {
2344   Unknown~option.\\
2345   The~key~'\l_keys_key_str'~is~unknown~in~
2346   \token_to_str:N \WithArrowsOptions. \\
2347   \c_@@_option_ignored_str \\
2348   \c_@@_available_keys_str
2349 }
2350 {
2351   The~available~keys~are~(in~alphabetic~order):~
2352   \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
2353 }
2354 \@@_msg_new:nnn { Unknown~option~Arrow~in~code~after }
2355 {
2356   Unknown~option.\\
2357   The~key~'\l_keys_key_str'~is~unknown~in~
2358   \token_to_str:N \Arrow\ in~code~after. \\
2359   \c_@@_option_ignored_str \\
2360   \c_@@_available_keys_str
2361 }
2362 {
2363   The~available~keys~are~(in~alphabetic~order):~
2364   \seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2365 }
2366 \@@_msg_new:nnn { Too~much~columns~in~WithArrows }
2367 {
2368   Too~much~columns.\\
2369   Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
2370   \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2371   Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2372   If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2373   the~option~'more~columns'~at~a~global~level~or~for~an~environment. \\
2374   However,~you~can~go~one~for~this~time.
2375   \@@_potential_body_i:
2376 }
2377 { \@@_potential_body_ii: }

```

```

2378 \@@_msg_new:nnn { Too-much-columns-in-DispWithArrows }
2379 {
2380   Too-much-columns.\\
2381   Your-environment-\{\l_@@_type_env_str\}-has-\int_use:N
2382   \l_@@_nb_cols_int\ columns-and-you-try-to-use-one-more.~
2383   Maybe-you-have-forgotten-a-\c_backslash_str\c_backslash_str\
2384   at-the-end-of-row-\int_use:N \g_@@_line_int. \\
2385   This-error-is-fatal.
2386   \@@_potential_body_i:
2387 }
2388 { \@@_potential_body_ii: }

2389 \@@_msg_new:nn { Negative-jump }
2390 {
2391   Incorrect-value.\\
2392   You-can't-use-a-negative-value-for-the-option-'jump'-of-command~
2393   \l_@@_string_Arrow_for_msg_str\
2394   in-the-row-\int_use:N \g_@@_line_int\
2395   of-your-environment-\{\l_@@_type_env_str\}.~
2396   You-can-create-an-arrow-going-backwards-with-the-option~'<'-of-Tikz. \\
2397   \c_@@_option_ignored_str
2398 }

2399 \@@_msg_new:nn { new-group-without-groups }
2400 {
2401   Misuse-of-a-key.\\
2402   You-can't-use-the-option-'new-group'~for~the~command~
2403   \l_@@_string_Arrow_for_msg_str\
2404   because-you-are-not-in-'groups'~mode.~Try-to-use-the-option~
2405   'groups'~in-your-environment-\{\l_@@_type_env_str\}. \\
2406   \c_@@_option_ignored_str
2407 }

2408 \@@_msg_new:nnn
2409 { Too-few-lines-for-an-arrow }
2410 {
2411   Impossible-arrow.\\
2412   Line-\l_@@_input_line_str\
2413   :~an-arrow-specified-in-the-row-\int_use:N \l_@@_initial_int\
2414   of-your-environment-\{\l_@@_type_env_str\}~can't-be-drawn~
2415   because~it~arrives~after~the~last~row~of~the~environment. \\
2416   If-you-go-on,~this~arrow~will~be~ignored.
2417   \@@_potential_body_i:
2418 }
2419 { \@@_potential_body_ii: }

2420 \@@_msg_new:nn { o-arrow-with-no-arrow-under }
2421 {
2422   Problem-with-the-key~'o'.\\
2423   Line-\l_@@_input_line_str\
2424   :~there~is~no~arrow~'under'~your~arrow~of~type~'o'.\\
2425   If-you-go-on,~this~arrow~won't~be~drawn.
2426 }

2427 \@@_msg_new:nnn { WithArrows-outside-math-mode }
2428 {
2429   You-are-outside-math-mode.\\
2430   The-environment-\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
2431   like~the~environment-\{aligned\}~of~amsmath. \\
2432   Nevertheless,~you~can~go~on.
2433   \@@_potential_body_i:
2434 }
2435 { \@@_potential_body_ii: }

2436 \@@_msg_new:nnn { DispWithArrows-in-math-mode }
2437 {
2438   You-are-in-math-mode.\\

```

```

2439 The~environment~\{\l_@@_type_env_str}\}~should~be~used~only~outside~math~
2440 mode~like~the~environments~\{align}\}~and~\{align*}\}~of~amsmath. \\\
2441 This~error~is~fatal.
2442 \@@_potential_body_i:
2443 }
2444 { \@@_potential_body_ii: }
2445 \@@_msg_new:nn { Incompatible~options~in~Arrow }
2446 {
2447   Incompatible~options.\\
2448   You~try~to~use~the~option~'\l_keys_key_str'~but~
2449   this~option~is~incompatible~or~redundant~with~the~option~
2450   '\l_@@_previous_key_str'~set~in~the~same~command~
2451   \l_@@_string_Arrow_for_msg_str. \\\
2452   \c_@@_option_ignored_str
2453 }
2454 \@@_msg_new:nn { Incompatible~options a}
2455 {
2456   Incompatible~options.\\
2457   You~try~to~use~the~option~'\l_keys_key_str'~but~
2458   this~option~is~incompatible~or~redundant~with~the~option~
2459   '\l_@@_previous_key_str'~set~in~the~same~command~
2460   \bool_if:NT \l_@@_in_code_after_bool
2461   {
2462     \l_@@_string_Arrow_for_msg_str\
2463     in~the~code~after~of~your~environment~\{\l_@@_type_env_str}\}
2464   }. \\\
2465   \c_@@_option_ignored_str
2466 }
2467 \@@_msg_new:nnn { Arrow~not~in~last~column }
2468 {
2469   Bad~use~of~\l_@@_string_Arrow_for_msg_str.\\
2470   You~should~use~the~command~\l_@@_string_Arrow_for_msg_str\
2471   only~in~the~last~column~(column~\int_use:N\l_@@_nb_cols_int)~
2472   in~the~row~\int_use:N \g_@@_line_int\
2473   of~your~environment~\{\l_@@_type_env_str}\}. \\\
2474   However~you~can~go~on~for~this~time.
2475   \@@_potential_body_i:
2476 }
2477 { \@@_potential_body_ii: }
2478 \@@_msg_new:nn { Wrong~line~in~Arrow }
2479 {
2480   Wrong~line.\\
2481   The~specification~of~line~'#1'~you~use~in~the~command~
2482   \l_@@_string_Arrow_for_msg_str\
2483   in~the~'code~after'~of~\{\l_@@_type_env_str}\}~doesn't~exist. \\\
2484   \c_@@_option_ignored_str
2485 }
2486 \@@_msg_new:nn { Both~lines~are~equal }
2487 {
2488   Both~lines~are~equal.\\
2489   In~the~'code~after'~of~\{\l_@@_type_env_str}\}~you~try~to~
2490   draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\\
2491   \c_@@_option_ignored_str
2492 }
2493 \@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }
2494 {
2495   Wrong~line~specification.\\
2496   The~specification~of~line~'#1'~doesn't~exist. \\\
2497   If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
2498 }
2499 \@@_msg_new:nn { Too~small~specification~for~MultiArrow }

```

```

2500 {
2501   Too~small~specification.\\
2502   The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
2503   is~too~small:~you~need~at~least~two~lines. \\
2504   \c_@@_command_ignored_str
2505 }
2506 \@@_msg_new:nn { Not~allowed-in~DispWithArrows }
2507 {
2508   Forbidden~command.\\
2509   The~command~\token_to_str:N #1
2510   is~allowed~only~in~the~last~column~
2511   (column~\int_use:N\l_@@_nb_cols_int)~of~\{\l_@@_type_env_str\}. \\
2512   \c_@@_option_ignored_str
2513 }
2514 \@@_msg_new:nn { Not~allowed-in~WithArrows }
2515 {
2516   Forbidden~command.\\
2517   The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
2518   (it's~allowed~in~the~last~column~of~\{DispWithArrows\}). \\
2519   \c_@@_option_ignored_str
2520 }
2521 ⟨*LaTeX⟩
2522 \@@_msg_new:nn { tag*~without~amsmath }
2523 {
2524   amsmath~not~loaded.\\
2525   We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2526   (or~mathtools). \\
2527   If~you~go~on,~the~command~\token_to_str:N\tag\
2528   will~be~used~instead.
2529 }
2530 \@@_msg_new:nn { Multiple~tags }
2531 {
2532   Multiple~tags.\\
2533   You~can't~use~twice~the~command~\token_to_str:N\tag\
2534   in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2535   If~you~go~on,~the~tag~'#1'~will~be~used.
2536 }
2537 \@@_msg_new:nn { Multiple~labels }
2538 {
2539   Multiple~labels.\\
2540   Normally,~we~can't~use~the~command~\token_to_str:N\label\
2541   twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2542   However,~you~can~go~on.~
2543   \IfPackageLoadedT { showlabels }
2544     { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
2545   If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
2546   'allow-multiple-labels'~at~the~global~or~environment~level.
2547 }
2548 \@@_msg_new:nn { Multiple~labels~with~cleveref }
2549 {
2550   Multiple~labels.\\
2551   Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
2552   twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2553   If~you~go~on,~you~may~have~undefined~references.
2554 }
2555 ⟨/LaTeX⟩
2556 \@@_msg_new:nn { Inexistent~v-node }
2557 {
2558   There~is~a~problem.\\
2559   Maybe~you~have~put~a~command~\token_to_str:N\cr\
2560   instead~of~a~command~\token_to_str:N\~at~the~end~of~

```

```

2561 the~row~\l_tmpa_int\
2562 of~your~environment~\{\l_@@_type_env_str\}. \\
2563 This~error~is~fatal.
2564 }

```

The following error when the user tries to use the option `xoffset` in mode `group` or `groups` (in fact, it's possible to use the option `xoffset` if there is only *one* arrow: of course, the option `group` and `groups` do not make sense in this case but, maybe, the option was set in a `\WithArrowsOptions`).

```

2565 \@@_msg_new:nn { Option-xoffset~forbidden }
2566 {
2567   Incorrect~key.\\
2568   You~can't~use~the~option~'xoffset'~in~the~command~
2569   \l_@@_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g_@@_line_int\
2570   of~your~environment~\{\l_@@_type_env_str\}~
2571   because~you~are~using~the~option~
2572   ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
2573     { group }
2574     { groups } '~It's~possible~for~an~independent~arrow~or~if~there~is~
2575     only~one~arrow. \\
2576     \c_@@_option_ignored_str
2577 }
2578 \@@_msg_new:nnn { Duplicate~name }
2579 {
2580   Duplicate~name.\\
2581   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2582   the~same~environment~name~twice.~You~can~go~on,~but,~
2583   maybe,~you~will~have~incorrect~results. \\
2584   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2585   If~you~don't~want~to~see~this~message~again,~use~the~option~
2586   'allow~duplicate~names'.
2587 }
2588 {
2589   The~names~already~defined~in~this~document~are:~
2590   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2591 }
2592 \@@_msg_new:nn { Invalid~specification~for~MultiArrow }
2593 {
2594   Invalid~specification.\\
2595   The~specification~of~rows~for~\token_to_str:N\MultiArrow\
2596   (i.e.~#1)~is~invalid. \\
2597   \c_@@_command_ignored_str
2598 }

```

## 12.15 The command `\WithArrowsNewStyle`

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

```

2599 <*LaTeX>
2600 \NewDocumentCommand \WithArrowsNewStyle { m m }
2601 </LaTeX>
2602 <*plain-TeX>
2603 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2604 </plain-TeX>
2605 {
2606   \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2607   { \@@_error:nn { Key~already~defined } { #1 } }
2608   {

```

First, we detect whether there is unknown keys in `#2` by storing in `\l_tmpa_seq` the list of the unknown keys.

```

2609     \seq_clear:N \l_tmpa_seq
2610     \keyval_parse:NNn \@@_valid_key:n \@@_valid_key:nn { #2 }
2611     \seq_if_empty:NTF \l_tmpa_seq
2612     {

```

```

2613     \seq_put_right:Ne \l_@@_options_WithArrows_seq
2614     { \tl_to_str:n { #1 } }
2615     \seq_put_right:Ne \l_@@_options_DispWithArrows_seq
2616     { \tl_to_str:n { #1 } }
2617     \seq_put_right:Ne \l_@@_options_WithArrowsOptions_seq
2618     { \tl_to_str:N { #1 } }
2619     \keys_precompile:nnc
2620     { WithArrows / WithArrowsOptions }
2621     { #2 }
2622     { @@ _ style _ #1 _ tl }
2623     \keys_define:nn { WithArrows / Global }
2624     { #1 .code:n = \use:c { @@ _ style _ #1 _ tl } }
2625     }
2626     { \@@_error:nn { Impossible~style } { #1 } }
2627   }
2628 }
2629 \@@_msg_new:nn { Impossible~style }
2630 {
2631   Impossible~style.\
2632   It's~impossible~to~define~the~style~'#1'~
2633   because~it~contains~unknown~keys:~'
2634   \seq_use:Nnnn \l_tmpa_seq { '~and~' } { ',~' } { ',~and~' }.
2635 }
2636 \cs_new_protected:Npn \@@_valid_key:n #1
2637 {
2638   \keys_if_exist:nnF { WithArrows / Global } { #1 }
2639   { \seq_put_right:Nn \l_tmpa_seq { #1 } }
2640 }
2641 \cs_new_protected:Npn \@@_valid_key:nn #1 #2
2642 {
2643   \keys_if_exist:nnF { WithArrows / Global } { #1 }
2644   { \seq_put_right:Nn \l_tmpa_seq { #1 } }
2645 }
2646 \@@_msg_new:nn { Key~already~defined }
2647 {
2648   Key~already~defined.\
2649   The~key~'#1'~is~already~defined. \
2650   If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2651   will~be~ignored.
2652 }

```

## 12.16 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 the Tikz library `calc`. That's why we have decided not to load by default this library. If it is not loaded, the user will have an error only when using the option `up` or the option `down`.

The keys `up` and `down` can be used with a value. This value is a list of pairs key-value specific to the options `up` and `down`.

- The key `radius` is the radius of the rounded corner of the arrow.
- The key `width` is the width of the horizontal part of the arrow. The corresponding dimension is `\l_@@_arrow_width_dim`. By convention, a value of 0 pt for `\l_@@_arrow_width_dim` means that the option `width` has been used with the special value `min` and a value of `\c_max_dim` means that it has been used with the value `max`.

```

2653 \keys_define:nn { WithArrows / up-and-down }
2654 {
2655   radius .dim_set:N = \l_@@_up_and_down_radius_dim ,

```

```

2656 radius .value_required:n = true ,
2657 width .code:n =
2658   \str_case:nnF { #1 }
2659   {
2660     { min } { \dim_zero:N \l_@@_arrow_width_dim }
2661     { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
2662   }
2663   { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
2664 width .value_required:n = true ,
2665 unknown .code:n = \@@_error:n { Option-unknown~for~up-and-down }
2666 }
2667 \@@_msg_new:nn { Option-unknown~for~up-and-down }
2668 {
2669   Unknown~option.\\
2670   The~option~'\l_keys_key_str'~is~unknown.~\c_@@_option_ignored_str
2671 }

```

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

```

2672 (*LaTeX)
2673 \tl_const:Nn \c_@@_tikz_code_up_tl
2674 {

```

First the case when the key `up` is used with `width=max` (that's the default behaviour).

```

2675   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2676   {
2677     \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2678     let \p1 = ( #1 ) , \p2 = ( #2 )
2679     in (\p1) -- node
2680     {
2681       \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2682       \begin { varwidth } \l_tmpa_dim

```

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

```

2683         \narrowragged
2684         #3
2685       \end { varwidth }
2686     }
2687     (\x2,\y1) -- (\p2) ;
2688   }

```

Now the case where the key `up` is used with `width=value` with `value` equal to `min` or a numeric value. The instruction `\path` doesn't draw anything: its aim is to compute the natural width of the label of the arrow. We can't use `\pgfextra` here because of the `\hbox_gset:Nn`.

```

2689   {
2690     \path
2691     let \p1 = ( #1 ) , \p2 = ( #2 )
2692     in node
2693     {

```

The length `\l_tmpa_dim` will be the maximal width of the box composed by the environment `{varwidth}`.

```

2694       \dim_set:Nn \l_tmpa_dim
2695       { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
2696       \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
2697       {
2698         \dim_set:Nn \l_tmpa_dim
2699         { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
2700       }

```



Now, the length `\l_tmpa_dim` is computed. We can compose the label in the box `\g_tmpa_box`. We have to do a global affectation to be able to exit the node.

```

2701         \hbox_gset:Nn \g_tmpa_box
2702         {
2703             \begin { varwidth } \l_tmpa_dim
2704             \narrowragged
2705             #3
2706             \end { varwidth }
2707         }

```

The length `\g_tmpa_dim` will be the width of the arrow (+ the radius of the corner).

```

2708         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2709         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2710         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2711         \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2712     } ;
2713 \draw
2714   let \p1 = ( #1 ) , \p2 = ( #2 )
2715   in (\x2-\g_tmpa_dim,\y1)
2716     -- node { \box_use:N \g_tmpa_box }
2717     (\x2-\l_@@_up_and_down_radius_dim,\y1)
2718     [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2719     -| (\p2) ;
2720 }
2721 }
2722 </LaTeX>
2723 <*plain-TeX>
2724 \tl_const:Nn \c_@@_tikz_code_up_tl
2725 {
2726   \dim_case:nnF \l_@@_arrow_width_dim
2727   {
2728     \c_max_dim
2729     {
2730       \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2731       let \p1 = ( #1 ) , \p2 = ( #2 )
2732       in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
2733     }
2734     \c_zero_dim
2735     {
2736       \path node
2737       {
2738         \hbox_gset:Nn \g_tmpa_box { #3 }
2739         \dim_gset:Nn \g_tmpa_dim
2740         { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2741       } ;
2742       \draw
2743       let \p1 = ( #1 ) , \p2 = ( #2 )
2744       in (\x2-\g_tmpa_dim,\y1)
2745         -- node { \box_use:N \g_tmpa_box }
2746         (\x2-\l_@@_up_and_down_radius_dim,\y1)
2747         [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2748         -| (\p2) ;
2749     }
2750   }
2751   {
2752     \draw
2753     let \p1 = ( #1 ) , \p2 = ( #2 )
2754     in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
2755       -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
2756       [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2757       -| (\p2) ;
2758   }
2759 }
2760 </plain-TeX>

```

The code for an arrow of type down is similar to the previous code (for an arrow of type up).

```

2761 <*LaTeX>
2762 \tl_const:Nn \c_@@_tikz_code_down_tl
2763 {
2764   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2765     {
2766       \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2767         let \p1 = ( #1 ) , \p2 = ( #2 )
2768         in (\p1) -- (\x1,\y2) -- node
2769           {
2770             \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2771             \begin { varwidth } \l_tmpa_dim
2772               \narrowragged
2773               #3
2774             \end { varwidth }
2775           }
2776         (\p2) ;
2777     }
2778   {
2779     \path
2780     let \p1 = ( #1 ) , \p2 = ( #2 )
2781     in node
2782       {
2783         \hbox_gset:Nn \g_tmpa_box
2784         {
2785           \dim_set:Nn \l_tmpa_dim

```

The 2 mm are for the tip of the arrow. We don't want the label of the arrow too close to the tip of arrow (we assume that to the tip of the arrow has its standard position, that is at the end of the arrow.).

```

2786           { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
2787           \begin { varwidth } \l_tmpa_dim
2788             \narrowragged
2789             #3
2790           \end { varwidth }
2791         }
2792         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2793         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2794         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2795         \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2796       } ;
2797
2798   \draw
2799   let \p1 = ( #1 ) , \p2 = ( #2 )
2800   in (\p1)
2801     { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2802     -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2803     -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2804     -- ++ (-2mm,0) ;
2805   }
2806 }
2807 </LaTeX>
2808 %
2809 <*plain-TeX>
2810 \tl_const:Nn \c_@@_tikz_code_down_tl
2811 {
2812   \dim_case:nnF \l_@@_arrow_width_dim
2813   {
2814     \c_max_dim
2815     {
2816       \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2817       let \p1 = ( #1 ) , \p2 = ( #2 )
2818       in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
2819     }

```

```

2820 \c_zero_dim
2821 {
2822   \path node
2823   {
2824     \hbox_gset:Nn \g_tmpa_box { #3 }
2825     \dim_gset:Nn \g_tmpa_dim
2826     { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2827   } ;
2828   \draw
2829   let \p1 = ( #1 ) , \p2 = ( #2 )
2830   in (\p1)
2831   { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2832   -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2833   -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2834   -- ++ (-2mm,0) ;
2835 }
2836 }
2837 {
2838   \draw
2839   let \p1 = ( #1 ) , \p2 = ( #2 )
2840   in (\p1)
2841   { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2842   -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2843   -- node { #3 }
2844     (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
2845   -- ++ (-2mm,0) ;
2846 }
2847 }
2848 </plain-TeX>

```

We recall that the options of the individual arrows are scanned twice. First, when are scanned when the command `\Arrow` occurs (we try to know whether the arrow is “individual”, etc.). That’s the first pass.

```

2849 \keys_define:nn { WithArrows / Arrow / FirstPass }
2850 {
2851   up .code:n = \@@_set_independent_bis: ,
2852   down .code:n = \@@_set_independent_bis: ,
2853   up .default:n = NoValue ,
2854   down .default:n = NoValue
2855 }

```

The options are scanned a second time when the arrow is actually drawn. That’s the second pass.

```

2856 \keys_define:nn { WithArrows / Arrow / SecondPass }
2857 {
2858   up .code:n =
2859     \str_if_empty:NT \l_@@_previous_key_str
2860     {
2861       \str_set:Nn \l_@@_previous_key_str { up }
2862       \cs_if_exist:NTF \tikz@library@calc@loaded
2863       {
2864         \keys_set:no { WithArrows / up-and-down } \l_keys_value_tl
2865         \int_set:Nn \l_@@_pos_arrow_int 1

```

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 `tikz-code` that will overwrite our affectation.

```

2866     \bool_set_false:N \l_@@_wrap_lines_bool

```

The main action occurs now. We change the value of the `tikz-code`.

```

2867     \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
2868   }
2869   { \@@_error:n { calc-not-loaded } }
2870 } ,

```

```

2871 down .code:n =
2872   \str_if_empty:NT \l_@@_previous_key_str
2873   {
2874     \str_set:Nn \l_@@_previous_key_str { down }
2875     \cs_if_exist:NTF \tikz@library@calc@loaded
2876     {
2877       \keys_set:no { WithArrows / up-and-down } \l_keys_value_tl
2878       \int_set:Nn \l_@@_pos_arrow_int 1
2879       \bool_set_false:N \l_@@_wrap_lines_bool
2880       \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
2881     }
2882     { \@@_error:n { calc-not-loaded } }
2883   }
2884 }

2885 \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
2886 \seq_put_right:Nn \l_@@_options_Arrow_seq { up }

2887 \@@_msg_new:nn { calc-not-loaded }
2888 {
2889   calc-not-loaded.\@
2890   You~can't~use~the~option~'\l_keys_key_str'~because~you~don't~have~loaded~the~
2891   Tikz~library~'calc'.~You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
2892   ~in~the~preamble~of~your~document. \@
2893   \c_@@_option_ignored_str
2894 }

2895 <*plain-TeX>
2896 \catcode \@ = 12
2897 \ExplSyntaxOff
2898 </plain-TeX>

```

## 13 History

### Changes between 2.8 and 2.9

Argument `<...>` for the command `\Arrow` in the class Beamer.

### Changes between 2.7 and 2.8

New key `right-overlap`

### Changes between 2.6b and 2.7

Correction of a bug: when the key `wrap-lines` was in force, the content of the annotations was not “flush left” by default as it should be (but justified).

### Changes between 2.6 and 2.6a (and 2.6b)

Replacement of `\hbox_unpack_clear:N` by `\hbox_unpack_drop:N` since `\hbox_unpack_clear:N` is now deprecated in L3.

Version 2.6d: correction of a bug (cf. question 628461 on TeX StackExchange).

### Changes between 2.5 and 2.5.1

Correction of the erroneous programming of the nodes aliases.

### Changes between 2.4 and 2.5

Arrows of type `o` which are *over* other arrows.

`witharrows` now requires and loads `varwidth`

## Changes between 2.3 and 2.4

Correction of a bug with `{DispWithArrows}` : cf. question 535989 on TeX StackExchange.

## Changes between 2.2 and 2.3

Two options for the arrows of type up and down: `width` and `radius`.

## Changes between 2.1 and 2.2

Addition of `\normalbaselines` at the beginning of `\@@_post_halign:`.

The warning for an environment ending by `\` has been transformed in `error`.

## Changes between 2.0 and 2.1

Option `max-length-of-arrow`.

Validation with regular expression for the first argument of `\MultiArrow`.

## Changes between 1.18 and 2.0

A version of `witharrows` is available for plain-TeX.

## Changes between 1.17 and 1.18

New option `<...>` for `{DispWithArrows}`.

Option `subequations`.

Warning when `{WithArrows}` or `{DispWithArrows}` ends by `\`.

No space before an environment `{DispWithArrows}` if we are at the beginning of a `{minipage}`.

## Changes between 1.16 and 1.17

Option `format`.

## Changes between 1.15 and 1.16

Option `no-arrows`

The behaviour of `{DispWithArrows}` after an `\item` of a LaTeX list has been changed : no vertical is added. The previous behaviour can be restored with the option `standard-behaviour-with-items`. A given name can no longer be used for two distinct environments. However, it's possible to deactivate this control with the option `allow-duplicate-names`.

## Changes between 1.14 and 1.15

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

Tikz externalization is now deactivated in the environments of the extension `witharrows`.<sup>41</sup>

---

<sup>41</sup>Before 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`.

## Changes between 1.13 and 1.14

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

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments: [//www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)).

Modification of the code of `\WithArrowsNewStyle` following a correction of a bug in `l3keys` in the version of `l3kernel` of 2019/01/28.

New error message `Inexistent~v-node` to avoid a `pgf` error.

The error `Option incompatible with 'group(s)'` was suppressed in the version 1.12 but this was a mistake since this error is used with the option `xoffset` at the local level. The error is put back.

## Changes between 1.12 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 1.11 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 1.10 and 1.11

New commands `\WithArrowsNewStyle` and `\WithArrowsRightX`.

## 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 1.8 and 1.9

New option `wrap-lines` for the 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 (`autonom`, `cleveref`, `fancyref`, `hyperref`, `prettyref`, `refstyle`, `typedref` and `varioref`) and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.

## Changes between 1.6 and 1.7

New environments `{DispWithArrows}` and `{DispWithArrows*}`.

## Changes between versions 1.5 and 1.6

The code has been improved to be faster and the Tikz library `calc` is no longer required. A new option `name` is available for the environments `{WithArrows}`.

## Changes between versions 1.4 and 1.5

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

## Changes between versions 1.3 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 1.2 and 1.3

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

## Changes between versions 1.1 and 1.2

The package `witharrows` can now be loaded without having loaded previously `tikz` and the libraries `arrow.meta` and `bending` (this extension and these libraries are loaded silently by `witharrows`). New option `groups` (with a `s`)

## Changes between versions 1.0 and 1.1

Option for the command `\\` and option `interline`  
Compatibility with `\usetikzlibrary{babel}`  
Possibility of nested environments `{WithArrows}`

## Contents

<b>1</b>	<b>Options for the shape of the arrows</b>	<b>2</b>
<b>2</b>	<b>Numbers of columns</b>	<b>6</b>
<b>3</b>	<b>Precise positioning of the arrows</b>	<b>7</b>
<b>4</b>	<b>The option 'o' for individual arrows</b>	<b>9</b>
<b>5</b>	<b>The options 'up' and 'down' for individual arrows</b>	<b>10</b>
<b>6</b>	<b>Comparison with the environment <code>{aligned}</code></b>	<b>11</b>
<b>7</b>	<b>Arrows in nested environments</b>	<b>14</b>
<b>8</b>	<b>Arrows from outside environments <code>{WithArrows}</code></b>	<b>16</b>
<b>9</b>	<b>The environment <code>{DispWithArrows}</code></b>	<b>17</b>
9.1	The option <code>&lt;...&gt;</code> of <code>DispWithArrows</code> . . . . .	22

<b>10</b>	<b>Advanced features</b>	<b>23</b>
10.1	Utilisation with Beamer . . . . .	23
10.2	Use with plain-Tex . . . . .	23
10.3	The option tikz-code : how to change the shape of the arrows . . . . .	24
10.4	The command <code>\WithArrowsNewStyle</code> . . . . .	24
10.5	The key <code>right-overlap</code> . . . . .	25
10.6	Vertical positioning of the arrows . . . . .	25
10.7	Footnotes in the environments of <code>witharrows</code> . . . . .	27
10.8	Option <code>no-arrows</code> . . . . .	27
10.9	Note for the users of AUCTeX . . . . .	27
10.10	Note for the developers . . . . .	28
<b>11</b>	<b>Examples</b>	<b>28</b>
11.1	<code>\MoveEqLeft</code> . . . . .	28
11.2	A command <code>\DoubleArrow</code> . . . . .	28
11.3	Modifying the shape of the nodes . . . . .	29
11.4	Examples with the option tikz-code . . . . .	29
11.4.1	Example 1 . . . . .	30
11.4.2	Example 2 . . . . .	30
11.4.3	Example 3 . . . . .	31
11.5	Automatic numbered loop . . . . .	32
<b>12</b>	<b>Implementation</b>	<b>33</b>
12.1	Declaration of the package and extensions loaded . . . . .	33
12.2	The packages <code>footnote</code> and <code>footnotehyper</code> . . . . .	33
12.3	The class option <code>legno</code> . . . . .	35
12.4	Collecting options . . . . .	36
12.5	Some technical definitions . . . . .	36
12.6	Variables . . . . .	39
12.7	The definition of the options . . . . .	42
12.8	The command <code>\Arrow</code> . . . . .	50
12.9	The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code> . . . . .	52
12.9.1	Code before the <code>\halign</code> . . . . .	52
12.9.2	The construction of the preamble of the <code>\halign</code> . . . . .	55
12.9.3	The environment <code>{WithArrows}</code> . . . . .	57
12.9.4	After the construction of the <code>\halign</code> . . . . .	59
12.9.5	The command of end of row . . . . .	60
12.9.6	The environment <code>{DispWithArrows}</code> . . . . .	64
12.10	The commands <code>\tag</code> , <code>\notag</code> , <code>\label</code> , <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code> . . . . .	69
12.11	We draw the arrows . . . . .	71
12.11.1	The command <code>update_x</code> . . . . .	81
12.11.2	We draw the arrows of type <code>o</code> . . . . .	81
12.12	The command <code>\Arrow</code> in code-after . . . . .	84
12.13	The command <code>\MultiArrow</code> in code-after . . . . .	86
12.14	The error messages of the package . . . . .	88
12.15	The command <code>\WithArrowsNewStyle</code> . . . . .	94
12.16	The options up and down . . . . .	95
<b>13</b>	<b>History</b>	<b>100</b>