

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

F. Pantigny
fpantigny@wanadoo.fr

October 21, 2019

Abstract

The LaTeX package **witharrows** provides environments `{WithArrows}` and `{DispWithArrows}` similar to the environments `{aligned}` and `{align}` of **amsmath** but with the possibility to draw arrows on the right side of the alignment. These arrows are usually used to give explanations concerning the mathematical calculus presented.

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

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

```
$\begin{WithArrows}
A \& = (a+1)^2 \Arrow{we expand} \\\
& = a^2 + 2a + 1 \quad \% don't put \\\ here
\end{WithArrows}$
```

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

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

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

1 Options for the shape of the arrows

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

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

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

*This document corresponds to the version 2.1 of **witharrows**, at the date of 2019/10/21.

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

It’s possible to put commands `\` in the text to force new lines³. However, if we put a `\`, a command of font placed in the beginning of the text will have effect only until the first command `\` (like in an environment `{tabular}`). That’s why Tikz gives an option `font` to modify the font of the whole text. Nevertheless, if we use the option `tikz={font={\bfseries}}`, the default specification of `\small` and `\itshape` will be overwritten.

²It’s possible to avoid the hyphenations of the words: use the Tikz option “`align = flush left`” in LaTeX and “`align = {flushleft,nothyphenated}`” in ConTeXt.

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

```

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

```

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

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

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

```

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

```

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

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

Without the option `displaystyle`:

```

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

```

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

The same example with the option `displaystyle`:

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

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

Almost all the options can also be set at the document level with the command `\WithArrowsOptions`. In this case, the scope of the declarations is the current TeX group (these declarations are “semi-global”). For example, if we want all the environments `{WithArrows}` composed in `\displaystyle` with blue arrows, we can write `\WithArrowsOptions{displaystyle,tikz=blue}`.⁵

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

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

The command `\Arrow` is recognized only in the environments `{WithArrows}`. If we have a command `\Arrow` previously defined, it’s possible to go on using it outside the environments `{WithArrows}`. However, a previously defined command `\Arrow` may still be useful in an environment `{WithArrows}`. If we want to use it in such an environment, it’s possible to change the name of the command `\Arrow` of the package `witharrows`: there is an option `command-name`⁶ for this purpose. The new name of the command must be given to the option *without* the leading backslash.

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

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

The environment `{WithArrows}` provides also two options `code-before` and `code-after`⁷ for LaTeX code that will be executed at the beginning and at the end of the environment. These options are not designed to be hooks (they are available only at the environment level and they do not apply to the nested environments).

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

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

Special commands are available in `code-after`: a command `\WithArrowsNbLines` which gives the number of lines (=rows) of the current environment (this is a command and not a counter), a special form of the command `\Arrow` and the command `\MultiArrow`: these commands are described in the section concerning the nested environments, p. 12.

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

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

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

2 Numbers of columns

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

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

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

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

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

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

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

3 Precise positioning of the arrows

The environment `{WithArrows}` defines, during the composition of the array, two series of nodes materialized in red in the following example.⁸

⁸The option `show-nodes` can be used to materialize the nodes. The nodes are in fact Tikz nodes of shape “rectangle”, but with zero width. An arrow between two nodes starts at the *south* anchor of the first node and arrives at the *north* anchor of the second node.

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

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.

```

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

```

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

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

```

 $\begin{WithArrows}[displaystyle,group]
2xy'-3y=\sqrt{x}
& \Leftrightarrow 2x(K'y_0+Ky'_0)-3Ky_0 = \sqrt{x} \quad \backslash\backslash
& \Leftrightarrow 2xK'y_0 + K(2xy'_0-3y_0) = \sqrt{x} \quad \backslash\backslash
& \Leftrightarrow 2xK'y_0 = \sqrt{x} \quad \backslash\backslash \text{Arrow}\{...\}\backslash\backslash
\dots
\end{WithArrows}$ 

```

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

The environment `{WithArrows}` gives also a `groups` option (with a *s* in the name). With this option, the arrows are divided into several “groups”. Each group is a set of connected⁹ arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

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

In an environment which uses the option `group` or the option `groups`, it’s still possible to give an option of position (`ll`, `lr`, `rl`, `rr` or `i`) to an individual arrow¹⁰. Such arrow will be drawn irrespective of the groups. It’s also possible to start a new group by applying the option `new-group` to an given arrow.

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

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

⁹More precisely: for each arrow *a*, we note *i(a)* the number of its initial row and *f(a)* the number of its final row; for two arrows *a* and *b*, we say that *a* ~ *b* when $\llbracket i(a), f(a) \rrbracket \cap \llbracket i(b), f(b) \rrbracket \neq \emptyset$; the groups are the equivalence classes of the transitive closure of ~.

¹⁰Such an arrow will be called *independent* in the technical documentation

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

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

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

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

```

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

```

$$\begin{aligned}
A &= B && \text{an arrow of type up} \\
&= C + C + C + C + C + C + C + C + C \\
&= C + C + C + C + C + C + C + C \\
&= E + E && \text{an arrow of type down}
\end{aligned}$$

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

5 Comparison with the environment `{aligned}`

`{WithArrows}` bears similarities with the environment `{aligned}` of the extension `amsmath`. These are only similarities because `{WithArrows}` has not been written upon the environment `{aligned}`.¹¹

¹¹In fact, it's possible to use the package `witharrows` without the package `amsmath`.

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

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

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

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

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

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

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

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

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

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.

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

```

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

```

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

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

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

```

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

```

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

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

```

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

```

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

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

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

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

The following is composed with `{WithArrows}[c,displaystyle]`. The results are strictly identical.¹³

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

6 Arrows in nested environments

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

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

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

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 :

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

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 ;

¹³In versions of `amsmath` older than the 5 nov. 2016, a thin space was added on the left of an environment `{aligned}`. The new versions do not add this space and neither do `{WithArrows}`.

- the label of the arrow.

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

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

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

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

The arrow we want to draw starts in the 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}$
```

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

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}$
```

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

```


$$\begin{array}{l} A \rightarrow B \\ A \rightarrow C \\ A \rightarrow D \\ A \rightarrow E \\ A \rightarrow F \end{array}$$


```

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

For illustrative purposes, we give an example of nested environments `{WithArrows}`, and, for each “right node”, the name of that node.¹⁴

$$\begin{aligned}
A &\triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B + B + B_{\text{wa-40-1-r}} \\
&\triangleleft \begin{cases} C \triangleleft D_{\text{wa-40-1-1-r}} \\ E \triangleleft F_{\text{wa-40-1-2-r}} \end{cases} \quad \text{wa-40-2-r} \\
&\triangleleft \begin{cases} G \triangleleft H + H + H + H + H + H + H_{\text{wa-40-2-1-r}} \\ I \triangleleft \begin{cases} J \triangleleft K_{\text{wa-40-2-1-1-r}} \\ L \triangleleft M_{\text{wa-40-2-1-2-r}} \end{cases} \end{cases} \quad \begin{matrix} \text{wa-40-3-r} \\ \text{wa-40-2-2-r} \end{matrix} \\
&\triangleleft \begin{cases} N \triangleleft O_{\text{wa-40-3-1-r}} \\ P \triangleleft Q_{\text{wa-40-3-2-r}} \end{cases} \quad \text{wa-40-4-r}
\end{aligned}$$

- the command `\WithArrowsLastEnv` gives the number of the last environment of level 0 (*i.e.* which is not included in another environment of the package `witharrows`);
- a name can be given to a given environment with the option `name` and, in this case, the nodes created in the environment will have aliases constructed with this name;
- the Tikz style `WithArrows/arrow` is the style used by `witharrows` when drawing an arrow¹⁵;

14

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

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

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

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

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

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

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

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

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

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

8 The environment `{DispWithArrows}`

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

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

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

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

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}
```

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

A link to the equation (\star) .¹⁶

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.¹⁷

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

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

It's also possible to suppress all the autogenerated numbers with the boolean option `notag` (or `nonumber`), at the global or environment level. There is also an environment `{DispWithArrows*}` which suppresses all these numbers.¹⁸

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

¹⁶In this document, the references has been customized with `\labelformat{equation}{(#1)}`

¹⁷Notice that the environment `{DispWithArrows}` is compatible with the command `\qedhere` of `amsthm`.

¹⁸Even in this case, it's possible to put a “manual tag” with the command `\tag`.

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

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}
```

$$\begin{aligned} A &= A_1 \\ &= A_2 \\ &= A_3 \end{aligned} \quad \begin{array}{l} \downarrow \text{first stage} \\ \downarrow \text{second stage} \end{array} \quad (3)$$

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 default value of this parameter is 25 pt.

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

$$\begin{aligned} A &= (a+1)^2 \\ &= a^2 + 2a + 1 \end{aligned} \quad \downarrow \text{we expand} \quad \begin{array}{l} (4) \\ (5) \end{array}$$

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*}`.¹⁹

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

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

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

¹⁹The package `amsmath` has an option `leqno` but `witharrows`, of course, is not aware of that option: `witharrows` only checks the option `leqno` of the document class.

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

First environment.

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

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

Second environment.

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

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

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

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

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

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

```
\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) \\
& \quad \operatorname{Arrow}{\text{sum of terms of a geometric progression of ratio } e^{i \frac{2\pi}{n}}} \\
& = \frac{1}{n} \operatorname{Re} \left( \frac{1 - \operatorname{bigl}(e^{i \frac{\pi}{2n}} \operatorname{bigr})^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
& \quad \operatorname{Arrow}{\text{This line has been wrapped automatically.}} \\
& = \frac{1}{n} \operatorname{Re} \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right) \\
\end{DispWithArrows*}
```

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

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.²⁰

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{\mathbf{\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}

```

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

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`.

²⁰It'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`.²¹ It is not compatible with `showkeys` (not all the labels are shown).

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

The environment `{DispWithArrows}` provides an option `left-brace`. When present, the value of this option is composed on the left, followed by a curly brace (hence the name) and the body of the environment.²²

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.²³

```
\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}
```

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

(15)

(16)

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

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

²¹We recall that `varioref`, `hyperref`, `cleveref` and `autonum` must be loaded in this order. The package `witharrows` can be loaded anywhere.

²²The option `left-brace` can also be used without value: in this case, only the brace is drawn...

²³The 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 provided by `hyperref`. It's a variant of `\ref` which doesn't create interactive link.

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

9 Advanced features

9.1 Utilisation with plain-Tex

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

```
\input{witharrows}
```

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

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

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

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

The option `tikz-code`²⁴ allows the user to change the shape of the arrows.²⁵

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

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

By default, the value is the following:

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

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

²⁴For historical reasons, the option `tikz-code` has an alias: `TikzCode`.

²⁵If the option `wrap-lines` is used in an environment `{DispWithArrows}` or `{DispWithArrows*}`, the option `tikz-code` will have no effect for the arrows of this environment but only for the arrows in the nested environments `{WithArrows}`.

```

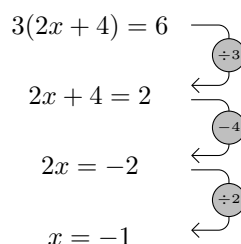
\begin{WithArrows}[format=c,ygap=5pt,interline=4mm,
  tikz-code = {\draw[rounded corners]
    (#1) -- ([xshift=5mm]#1)
    -- node[circle,
      draw,
      auto = false,
      fill = gray!50,
      inner sep = 1pt] {\tiny #3}
    ([xshift=5mm]#2)
    -- (#2) ; }]}
3 (2x+4) = 6   \Arrow{$\div 3$} \\  

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

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

x = -1
\end{WithArrows}

```



The environments `{DispWithArrows}` and its starred version `{DispWithArrows*}` provide a command `\WithArrowsRightX` which can be used in a definition of `tikz-code`. This command gives the x -value of the right side of the composition box (taking into account the eventual tags of the equations). For an example of use, see p. 27.

9.3 The command `\WithArrowsNewStyle`

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

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

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

For an example of use, see p. 27.

9.4 Vertical positioning of the arrows

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

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

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

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

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

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

```

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

```

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

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

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

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

Since the version 2.1 of `witharrows`, an arrow of `jump` equal to 1 has a maximal length²⁶ equal to the parameter `max-length-of-arrow`. The default value of this parameter est 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 \ ici
} \\
& =
\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}

```

²⁶We call *length* of an arrow the difference between the *y*-value of its start point and the *y* value of its end point.

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

9.5 Footnotes in the environments of witharrows

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

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

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

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

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

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

$$\begin{aligned}
A &= (a+b)^2 \\
&= a^2 + b^2 + 2ab \quad \searrow \textit{We expand}^{27}
\end{aligned}$$

9.6 Option no-arrows

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

9.7 Note for developers

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

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

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

If you use this environment `{DWA}` in math mode, you will have the following error message:

The environment `{DWA}` should be used only outside math mode.

²⁷A footnote.

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

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

10 Examples

10.1 `\MoveEqLeft`

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

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

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

10.2 Modifying the shape of the nodes

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

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

  3 (2x+4) = 6 \Arrow{\$div 3} \\\
  2x+4 = 2 \Arrow{\$-4} \\\
  2x = -2 \Arrow{\$div 2} \\\
  2x = -1

\end{WithArrows}
```

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

10.3 Examples with the option tikz-code

We recall that the option `tikz-code` is the Tikz code used by `witharrows` to draw the arrows.²⁸

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

10.3.1 Example 1

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

```

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

```

| | |
|--|----------------------------------|
| $S_n = \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right)$ | $\cos x = \Re(e^{ix})$ |
| $= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i \frac{k\pi}{2n}}\right)$ | \leftarrow |
| $= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right)$ | $\Re(z + z') = \Re(z) + \Re(z')$ |
| $= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right)$ | \leftarrow |
| $= \frac{1}{n} \Re\left(\frac{1 - \left(e^{i \frac{\pi}{2n}}\right)^n}{1 - e^{i \frac{\pi}{2n}}}\right)$ | \leftarrow |
| $= \frac{1}{n} \Re\left(\frac{1 - i}{1 - e^{i \frac{\pi}{2n}}}\right)$ | \leftarrow |

²⁸If 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).

10.3.2 Example 2

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

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

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

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

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

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right) \quad \boxed{\exp \text{ is a morphism for } \times \text{ et } +}$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right) \quad \boxed{\text{sum of terms of a geometric progression of ratio } e^{i \frac{2\pi}{n}}}$$

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

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

10.3.3 Example 3

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

```
\begin{WithArrows}[ll,interline=5mm,xoffset=5mm,
  tikz-code = {\draw[rounded corners,
                    every node/.style = {circle,
                                          draw,
                                          auto = false,

```

```

inner sep = 1pt,
fill = gray!50,
font = \tiny }]

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

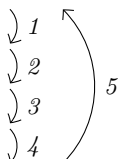
E & \Longlefttrightarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\Arrow{$\times 15$}\\
& \Longlefttrightarrow 5(x+4) + 3(5x+3) = 105 \\
& \Longlefttrightarrow 5x+20 + 15x+9 = 105 \\
& \Longlefttrightarrow 20x+29 = 105
\Arrow{$-29$}\\
& \Longlefttrightarrow 20x = 76
\Arrow{$\div 20$}\\
& \Longlefttrightarrow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

```

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

```
e.\;& f \text{ is lipschitzian}
\end{WithArrows}$
```

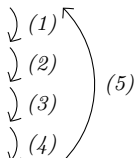
a. f est continuous on E
b. f est continuous in 0
c. f is bounded on the unit sphere
d. $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K\|x\|$
e. f is lipschitzian



As usual, it's possible to change the characteristic of both arrows and nodes with the option `tikz`. However, if we want to change the style to have, for example, numbers in 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. $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K\|x\|$
e. f is lipschitzian



11 Implementation

11.1 Declaration of the package and extensions loaded

First, `tikz` and some Tikz libraries are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.²⁹

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

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

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

²⁹cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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

```

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

```

11.2 The packages `footnote` and `footnotehyper`

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 2.1), there are two such options: `footnote` and `footnotehyper`. With the option `footnote`, `witharrows` loads `footnote` and uses it to extract the footnotes from the environments `{WithArrows}`. Idem for the option `footnotehyper`.

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

```

24 \<LaTeX>
25 \bool_new:N \g__wa_footnotehyper_bool

```

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

```

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

```

```

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

```

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

```

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

```

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

```

51 \ProcessKeysOptions { WithArrows / package }

52 \__wa_msg_new:nn { Option-incompatible-with-Beamer }
53 {
54   The~option~'\l_keys_key_tl'~is~incompatible~
55   with~Beamer~because~Beamer~has~its~own~system~to~extract~footnotes.
56 }

```

```

57 \_wa_msg_new:nn { footnote-with-footnotehyper-package }
58 {
59   You~can't~use~the~option~'footnote'~because~the~package~
60   footnotehyper~has~already~been~loaded.~
61   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
62   within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
63   of~the~package~footnotehyper.\\
64   If~you~go~on,~the~package~footnote~won't~be~loaded.
65 }
66 \_wa_msg_new:nn { footnotehyper-with-footnote-package }
67 {
68   You~can't~use~the~option~'footnotehyper'~because~the~package~
69   footnote~has~already~been~loaded.~
70   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
71   within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
72   of~the~package~footnote.\\
73   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
74 }

75 \bool_if:NT \g__wa_footnote_bool
76 {
77   \@ifclassloaded { beamer }
78   { \msg_info:nn { witharrows } { Option~incompatible~with~Beamer } }
79   {
80     \@ifpackageloaded { footnotehyper }
81     { \_wa_error:n { footnote~with~footnotehyper~package } }
82     { \usepackage { footnote } }
83   }
84 }

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

```

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

11.3 The class option `leqno`

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

```

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

```

11.4 Some technical definitions

```

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

```

We create booleans in order to know if some packages are loaded. For example, for the package `amsmath`, the boolean is called `\c_@@_amsmath_loaded_bool`.³⁰

```

103 \AtBeginDocument
104 {
105   \clist_map_inline:nn
106   {
107     amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
108     typedref, unicode-math, varwidth
109   }
110   {
111     \bool_new:c { c__wa_#1_loaded_bool }
112     (*LaTeX)
113     \ifpackageloaded { #1 }
114       { \bool_set_true:c { c__wa_#1_loaded_bool } }
115       { }
116     (/LaTeX)
117     (*plain-TeX)
118     \bool_set_false:c { c__wa_#1_loaded_bool }
119     (/plain-TeX)
120   }
121 }

```

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

```

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

```

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

```

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

```

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

```

146 \cs_new_protected:Npn \__wa_convert_to_str_seq:N #1

```

³⁰It's not possible to use `\ifpackageloaded` in the core of the functions because `\ifpackageloaded` is available only in the preamble.


```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

If the user uses the option `show-nodes` (it's a `l3keys` option), the Tikz options `draw` and `red` will be appended to this style. This feature may be useful for debugging.³¹

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

```

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

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

```

205 \tikzset
206 {
207   WithArrows / arrow / tips / .style =
208   { > = { Straight~Barb [ scale = 1.2 , bend ] } }
209 }
```

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

```

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

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

```

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

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

```

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

³¹The `v`-nodes, created near the end of line in `{DispWithArrows}` and `{DispWithArrows*}` are not shown with the option `show-nodes`.

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

Nevertheless, we want the extension `witharrows` available without `amsmath`. That's why we give a definition of `\spread@equation` if `amsmath` is not loaded (we put the code in a `\AtBeginDocument` because the flag `\c_@@_amsmath_loaded_bool` is itself set in a `\AtBeginDocument`).

```

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

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

```

11.5 Variables

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

```

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

```

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

```

248 \seq_new:N \g__wa_position_in_the_tree_seq
249 \seq_gput_right:Nn \g__wa_position_in_the_tree_seq 1

```

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

```

250 \int_new:N \g__wa_last_env_int

```

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

```

251 \int_new:N \l__wa_pos_env_int

```

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

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

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

```

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

```

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

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

273 \seq_new:N \g__wa_names_seq

```

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

```

274 \LaTeX
275 \bool_new:N \l__wa_sbwi_bool
276 \LaTeX

```

```

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

```

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

```

289 \str_new:N \l__wa_format_str

```

The option boolean corresponds to the option `subequations`.

```

290 \*LaTeX
291 \bool_new:N \l__wa_subequations_bool
292 \*LaTeX

```

11.6 The definition of the options

There are four levels where options can be set:

- with `\usepackage[...]{witharrows}`: this level will be called *package* level;
- with `\WithArrowsOptions{...}`: this level will be called *global* level³²;
- 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).

```

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

302 \cs_new_protected:Npn \__wa_fix_pos_option:n #1
303 { \__wa_eval_if_allowed:n { \int_set:Nn \l__wa_pos_arrow_int { #1 } } }

```

³²This 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.

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

```

304 \keys_define:nn { WithArrows / Global }
305 {
306   max-length-of-arrow .dim_set:N = \l__wa_max_length_of_arrow_dim ,
307   max-length-of-arrow .value_required:n = true ,
308   max-length-of-arrow .initial:n = 2 cm ,
309   ygap .dim_set:N = \l__wa_ygap_dim ,
310   ygap .value_required:n = true ,
311   ygap .initial:n = 0.4 ex ,
312   ystart .dim_set:N = \l__wa_ystart_dim ,
313   ystart .value_required:n = true ,
314   ystart .initial:n = 0.4 ex ,
315   more-columns .code:n =
316     \__wa_msg_redirect_name:nn { Too-much-columns-in-WithArrows } { none } ,
317   more-columns .value_forbidden:n = true,
318   command-name .code:n =
319     \str_set:Nn \l__wa_command_name_str { #1 }
320     \str_set:Nx \l__wa_string_Arrow_for_msg_str
321       { \c_backslash_str Arrow-alias~\c_backslash_str #1 } ,
322   command-name .value_required:n = true ,
323   tikz-code .tl_set:N = \l__wa_tikz_code_tl,
324   tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
325   tikz-code .value_required:n = true ,
326   TikzCode .meta:n = { tikz-code = #1 } ,
327   displaystyle .bool_set:N = \l__wa_displaystyle_bool ,
328   displaystyle .default:n = true ,
329   show-nodes .code:n =
330     \tikzset { __wa_node_style / .append~style = { draw , red } } ,
331   show-nodes .value_forbidden:n = true,
332   show-node-names .bool_set:N = \l__wa_show_node_names_bool ,
333   show-node-names .default:n = true ,
334   group .code:n =
335     \str_if_empty:NTF \l__wa_previous_key_str
336       {
337         \str_set:Nn \l__wa_previous_key_str { group }
338         \seq_remove_all:Nn \l__wa_options_Arrow_seq { xoffset }
339         \int_set:Nn \l__wa_pos_arrow_int 7
340       }
341       { \__wa_error:n { Incompatible-options } } ,
342   group .value_forbidden:n = true ,
343   groups .code:n =
344     \str_if_empty:NTF \l__wa_previous_key_str
345       {
346         \str_set:Nn \l__wa_previous_key_str { groups }
347         \seq_if_in:NnF \l__wa_options_Arrow_seq { new-group }
348           { \seq_put_right:Nn \l__wa_options_Arrow_seq { new-group } }
349         \seq_remove_all:Nn \l__wa_options_Arrow_seq { xoffset }
350         \int_set:Nn \l__wa_pos_arrow_int 6
351       }
352       { \__wa_error:n { Incompatible-options } } ,
353   groups .value_forbidden:n = true ,
354   tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
355   tikz .initial:n = \c_empty_tl ,
356   tikz .value_required:n = true ,
357   rr .value_forbidden:n = true ,
358   rr .code:n = \__wa_fix_pos_option:n 3 ,
359   ll .value_forbidden:n = true ,
360   ll .code:n = \__wa_fix_pos_option:n 1 ,
361   rl .value_forbidden:n = true ,
362   rl .code:n = \__wa_fix_pos_option:n 2 ,
363   lr .value_forbidden:n = true ,
364   lr .code:n = \__wa_fix_pos_option:n 0 ,
365   i .value_forbidden:n = true ,

```

```

366 i .code:n = \__wa_fix_pos_option:n 5 ,
367 xoffset .dim_set:N = \l__wa_xoffset_dim ,
368 xoffset .value_required:n = true ,
369 xoffset .initial:n = 3 mm ,
370 jot .dim_set:N = \jot ,
371 jot .value_required:n = true ,
372 interline .skip_set:N = \l__wa_interline_skip ,
373 interline .value_required:n = true ,
374 start-adjust .dim_set:N = \l__wa_start_adjust_dim ,
375 start-adjust .value_required:n = true ,
376 start-adjust .initial:n = 0.4 ex ,
377 end-adjust .dim_set:N = \l__wa_end_adjust_dim ,
378 end-adjust .value_required:n = true ,
379 end-adjust .initial:n = 0.4 ex ,
380 adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
381 adjust .value_required:n = true ,

```

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

```

382 no-arrows .code:n =
383   \cs_set_eq:NN \__wa_draw_arrows:nn \use_none:nn
384   \cs_set_eq:NN \__wa_draw_arrow:nnn \use_none:nnn ,
385 no-arrows .value_forbidden:n = true ,
386 }

```

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`.

```

387 \keys_define:nn { WithArrows / WithArrowsSpecific }
388 {
389   t .code:n = \int_set:Nn \l__wa_pos_env_int 0 ,
390   t .value_forbidden:n = true ,
391   c .code:n = \int_set:Nn \l__wa_pos_env_int 1 ,
392   c .value_forbidden:n = true ,
393   b .code:n = \int_set:Nn \l__wa_pos_env_int 2 ,
394   b .value_forbidden:n = true
395 }

```

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

```

396 \clist_new:N \c__wa_extensible_delimiters_clist
397 \clist_set:Nn \c__wa_extensible_delimiters_clist
398 {
399   ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
400 }
401 \*LaTeX
402 \AtBeginDocument
403 {
404   \bool_if:nT
405     { \c__wa_amsmath_loaded_bool || \use:c { c__wa_unicode-math_loaded_bool } }
406     {
407       \clist_put_right:Nn \c__wa_extensible_delimiters_clist { \lvert, \lVert }
408     }
409 }
410 \*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`.

```

411 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
412 {
413   fleqn .bool_set:N = \l__wa_fleqn_bool ,

```

```

414   fleqn .default:n = true ,
415   mathindent .dim_set:N = \l__wa_mathindent_dim ,
416   mathindent .value_required:n = true ,
417   mathindent .initial:n = 25 pt ,
418 <*LaTeX>
419   notag .code:n =
420     \str_if_eq:nnTF { #1 } { true }
421     { \clist_clear:N \l__wa_tags_clist }
422     { \clist_set:Nn \l__wa_tags_clist { all } } ,
423   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.

```

424   subequations .code:n =
425     \bool_if:NTF \c__wa_amsmath_loaded_bool
426     { \bool_set_true:N \l__wa_subequations_bool }
427     {
428       \__wa_error:n { amsmath-not-loaded }
429       \group_begin:
430       \globaldefs = 1
431       \__wa_msg_redirect_name:nn { amsmath-not-loaded } { info }
432       \group_end:
433     } ,
434   subequations .default:n = true ,
435   subequations .value_forbidden:n = true ,
436   nonumber .meta:n = notag ,
437   allow-multiple-labels .code:n =
438     \__wa_msg_redirect_name:nn { Multiple~labels } { none } ,
439   allow-multiple-labels .value_forbidden:n = true ,
440   tagged-lines .code:n =
441     \clist_set:Nn \l__wa_tags_clist { #1 }
442     \clist_if_in:NnT \l__wa_tags_clist { first }
443     {
444       \clist_remove_all:Nn \l__wa_tags_clist { first }
445       \clist_put_left:Nn \l__wa_tags_clist \c_one_int
446     } ,
447   tagged-lines .value_required:n = true ,
448 </LaTeX>
449   wrap-lines .bool_set:N = \l__wa_wrap_lines_bool ,
450   wrap-lines .default:n = true ,
451   replace-left-brace-by .code:n =
452     {
453       \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
454       \clist_if_in:NVTF
455       \c__wa_extensible_delimiters_clist
456       \l_tmpa_tl
457       { \tl_set:Nn \l__wa_replace_left_brace_by_tl { #1 } }
458       { \__wa_error:n { Bad~value~for~replace~brace~by } }
459     } ,
460   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`).

```

461 <*LaTeX>
462   standard-behaviour-with-items .bool_set:N = \l__wa_sbwi_bool ,
463   standard-behaviour-with-items .default:n = true
464 </LaTeX>
465 }

```

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

```

466 \keys_define:nn { WithArrows / Env }

```



```

467 {
468   name .code:n =

```

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

```

469   \str_set:Nn \l_tmpa_str { #1 }
470   \seq_if_in:NVTF \g__wa_names_seq \l_tmpa_str
471     { \__wa_error:n { Duplicate~name } }
472     { \seq_gput_left:NV \g__wa_names_seq \l_tmpa_str }
473   \str_set_eq:NN \l__wa_name_str \l_tmpa_str ,
474   name .value_required:n = true ,
475   code-before .code:n = \tl_put_right:Nn \l__wa_code_before_tl { #1 } ,
476   code-before .value_required:n = true ,
477   CodeBefore .meta:n = { code-before = #1 } ,
478   code-after .code:n = \tl_put_right:Nn \l__wa_code_after_tl { #1 } ,
479   code-after .value_required:n = true ,
480   CodeAfter .meta:n = { code-after = #1 } ,
481   format .code:n =
482     \tl_if_empty:nTF { #1 }
483       { \__wa_error:n { Invalid~option~format } }
484       {
485         \regex_match:nnTF { \A[rcl]*Z } { #1 }
486         { \tl_set:Nn \l__wa_format_str { #1 } }
487         { \__wa_error:n { Invalid~option~format } }
488       } ,
489   format .value_required:n = true ,
490 }

```

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.

```

491 \keys_define:nn { WithArrows }
492 {
493   WithArrows .inherit:n =
494   {
495     WithArrows / Global ,
496     WithArrows / WithArrowsSpecific ,
497     WithArrows / Env
498   } ,
499   DispWithArrows .inherit:n =
500   {
501     WithArrows / DispWithArrowsSpecific ,
502     WithArrows / Global ,
503     WithArrows / Env ,
504   } ,
505   WithArrowsOptions .inherit:n =
506   {
507     WithArrows / Global ,
508     WithArrows / WithArrowsSpecific ,
509     WithArrows / DispWithArrowsSpecific
510   }
511 }

```

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

```

512 \seq_new:N \l__wa_options-WithArrows_seq
513 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options-WithArrows_seq
514 {
515   adjust, b, c, code-after, code-before, command-name,
516   displaystyle, end-adjust,
517   format, group, groups, i,
518   interline, jot, ll,
519   lr, max-length-of-arrow, more-columns, name,

```

```

520     no-arrows, rl, rr,
521     show-node-names, show-nodes, start-adjust,
522     t, tikz, tikz-code,
523     xoffset, ygap, ystart
524 }
525 \__wa_convert_to_str_seq:N \l__wa_options_WithArrows_seq

526 \keys_define:nn { WithArrows / WithArrows }
527 {
528     unknown .code:n =
529         \__wa_sort_seq:N \l__wa_options_WithArrows_seq
530         \__wa_error:n { Unknown~option~WithArrows }
531 }

532 \keys_define:nn { WithArrows / DispWithArrows }
533 {
534     left-brace .tl_set:N = \l__wa_left_brace_tl ,
535     unknown .code:n =
536         \__wa_sort_seq:N \l__wa_options_DispWithArrows_seq
537         \__wa_error:n { Unknown~option~DispWithArrows }
538 }

```

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

```

539 \seq_new:N \l__wa_options_DispWithArrows_seq
540 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_DispWithArrows_seq
541 {
542     code-after, code-before, command-name, tikz-code, adjust,
543     displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
544     left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
545     replace-left-brace-by, rl, rr, show-node-names, show-nodes, start-adjust,
546     tikz, wrap-lines, xoffset, ygap, ystart,
547 <LaTeX>
548     allow-multiple-labels, tagged-lines, nonumber, notag
549 </LaTeX>
550 }

551 \keys_define:nn { WithArrows / WithArrowsOptions }
552 {
553     allow-duplicate-names .code:n =
554         \__wa_msg_redirect_name:nn { Duplicate~name } { none } ,
555     allow-duplicate-names .value_forbidden:n = true ,
556     unknown .code:n =
557         \__wa_sort_seq:N \l__wa_options_WithArrowsOptions_seq
558         \__wa_error:n { Unknown~option~WithArrowsOptions }
559 }

```

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

```

560 \seq_new:N \l__wa_options_WithArrowsOptions_seq
561 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_WithArrowsOptions_seq
562 {
563     allow-duplicate-names, b, c, command-name, more-columns, tikz-code, adjust,
564     displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
565     mathindent, max-length-of-arrow, no-arrows, rl, rr, show-node-names,
566     show-nodes, start-adjust, t, tikz, wrap-lines, xoffset, ygap, ystart,
567 <LaTeX>
568     allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
569     tagged-lines
570 </LaTeX>
571 }

```

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

```

572 \cs_new_protected:Npn \__wa_set_independent:
573 {
574   \str_if_empty:NTF \l__wa_previous_key_str
575   {
576     \str_set_eq:NN \l__wa_previous_key_str \l_keys_key_tl
577     \str_set:Nn \l__wa_status_arrow_str { independent }
578     \str_if_eq:VnF \l_keys_value_tl { NoValue }
579     { \__wa_error:n { Value-for-a-key } }
580   }
581   { \__wa_error:n { Incompatible~options~in~Arrow } }
582 }

```

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

```

583 \keys_define:nn { WithArrows / Arrow / FirstPass }
584 {
585   jump .code:n =
586     \int_compare:nTF { #1 > 0 }
587     { \int_set:Nn \l__wa_jump_int { #1 } }
588     { \__wa_error:n { Negative~jump } } ,
589   jump .value_required:n = true,
590   rr .code:n = \__wa_set_independent: ,
591   ll .code:n = \__wa_set_independent: ,
592   rl .code:n = \__wa_set_independent: ,
593   lr .code:n = \__wa_set_independent: ,
594   i .code:n = \__wa_set_independent: ,
595   rr .default:n = NoValue ,
596   ll .default:n = NoValue ,
597   rl .default:n = NoValue ,
598   lr .default:n = NoValue ,
599   i .default:n = NoValue ,
600   new-group .value_forbidden:n = true,
601   new-group .code:n =
602     \int_compare:nTF { \l__wa_pos_arrow_int = 6 }
603     { \str_set:Nn \l__wa_status_arrow_str { new-group } }
604     { \__wa_error:n { new-group~without~groups } } ,

```

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

```

605   tikz-code .code:n = \prg_do_nothing: ,
606   tikz-code .value_required:n = true ,
607   tikz .code:n = \prg_do_nothing: ,
608   tikz .value_required:n = true ,
609   start-adjust .code:n = \prg_do_nothing: ,
610   start-adjust .value_required:n = true ,
611   end-adjust .code:n = \prg_do_nothing: ,
612   end-adjust .value_required:n = true ,
613   adjust .code:n = \prg_do_nothing: ,
614   adjust .value_required:n = true ,
615   xoffset .code:n = ,
616   unknown .code:n =
617     \__wa_sort_seq:N \l__wa_options_Arrow_seq
618     \seq_if_in:NVTf \l__wa_options_WithArrows_seq \l_keys_key_tl
619     {

```

```

620     \str_set:Nn \l_tmpa_str
621     { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
622   }
623   { \str_clear:N \l_tmpa_str }
624   \__wa_error:n { Unknown~option~in~Arrow }
625 }

```

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

```

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

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

```

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

```

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

```

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

```

651   xoffset .code:n =
652     \bool_if:nTF
653     {
654       \int_compare_p:nNn \g__wa_arrow_int > 1
655       &&
656       \int_compare_p:nNn \l__wa_pos_arrow_int > 5
657       &&
658       ! \str_if_eq_p:Vn \l__wa_status_arrow_str { independent }
659     }
660     { \__wa_error:n { Option~xoffset~forbidden } }
661     { \dim_set:Nn \l__wa_xoffset_dim { #1 } } ,
662   xoffset .value_required:n = true ,
663   start-adjust .dim_set:N = \l__wa_start_adjust_dim,
664   end-adjust .dim_set:N = \l__wa_end_adjust_dim,
665   adjust .code:n =
666     \dim_set:Nn \l__wa_start_adjust_dim { #1 }
667     \dim_set:Nn \l__wa_end_adjust_dim { #1 } ,
668 }

```

`\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`.

```

669 \*LaTeX
670 \NewDocumentCommand \WithArrowsOptions { m }
671 \*LaTeX
672 \*plain-TeX
673 \cs_set_protected:Npn \WithArrowsOptions #1
674 \*plain-TeX
675 {
676   \str_clear_new:N \l__wa_previous_key_str
677   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
678 }

```

11.7 The command `\Arrow`

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

```

679 \*LaTeX
680 \NewDocumentCommand \__wa_Arrow { 0 { } m ! 0 { } }
681 \*LaTeX
682 \*plain-TeX
683 \cs_new_protected:Npn \__wa_Arrow
684 {
685   \peek_meaning:NTF [
686     { \__wa_Arrow_i }
687     { \__wa_Arrow_i [ ] }
688   }
689   \cs_new_protected:Npn \__wa_Arrow_i [ #1 ] #2
690   {
691     \peek_meaning:NTF [
692       { \__wa_Arrow_ii [ #1 ] { #2 } }
693       { \__wa_Arrow_ii [ #1 ] { #2 } [ ] }
694     }
695     \cs_new_protected:Npn \__wa_Arrow_ii [ #1 ] #2 [ #3 ]
696   \*plain-TeX
697   {

```

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.

```

698   \int_gincr:N \g__wa_arrow_int

```

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

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

```

699   \str_clear_new:N \l__wa_previous_key_str
700   \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:

```
701 \prop_put:NnV \l_tmpa_prop { initial } \g__wa_line_int
```

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

```
702 \int_set:Nn \l_tmpa_int { \g__wa_line_int + \l__wa_jump_int }
703 \prop_put:NnV \l_tmpa_prop { final } \l_tmpa_int
```

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

```
704 \prop_put:NnV \l_tmpa_prop { status } \l__wa_status_arrow_str
```

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

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

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

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

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

```
707 \prop_put:Nnx \l_tmpa_prop { input-line } \msg_line_number:
```

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

```
708 \prop_gclear_new:c
709 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \g__wa_arrow_int _ prop }
710 \prop_gset_eq:cN
711 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \g__wa_arrow_int _ prop }
712 \l_tmpa_prop
713 }
```

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.

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

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

11.8.1 Code before the `\halign`

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

```
716 \cs_new_protected:Npn \__wa_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.

```
717 {
718 <*LaTeX>
719 \str_clear_new:N \l__wa_type_env_str
720 \str_set:NV \l__wa_type_env_str \@currenvir
721 </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`.

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

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

```
724 \str_clear_new:N \l__wa_name_str
```

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

```
725 \str_clear_new:N \l__wa_status_arrow_str
```

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

```
726 \dim_zero_new:N \l__wa_x_dim
```

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

```
727 \str_clear_new:N \l__wa_input_line_str
```

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

```
728 \seq_gput_right:NV \g__wa_arrow_int_seq \g__wa_arrow_int
729 \int_gzero:N \g__wa_arrow_int
730 \seq_gput_right:NV \g__wa_line_int_seq \g__wa_line_int
731 \int_gzero:N \g__wa_line_int
732 \seq_gput_right:NV \g__wa_col_int_seq \g__wa_col_int
733 \int_gzero:N \g__wa_col_int
```

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

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

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

```
734 \int_gzero_new:N \g__wa_static_col_int
```

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

```
735 \seq_gput_right:Nn \g__wa_position_in_the_tree_seq 1
```

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

```
736 \seq_set_eq:NN \l_tmpa_seq \g__wa_position_in_the_tree_seq
737 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
738 \str_clear_new:N \l__wa_prefix_str
739 \str_set:Nx \l__wa_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

```
740 \cs_set_eq:NN \l \l__wa_cr:
741 \dim_zero:N \mathsurround
```

These counters will be used later as variables.

```
742 \int_zero_new:N \l__wa_initial_int
743 \int_zero_new:N \l__wa_final_int
744 \int_zero_new:N \l__wa_arrow_int
745 \int_zero_new:N \l__wa_pos_of_arrow_int
746 \int_zero_new:N \l__wa_jump_int
```

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

```
747 \int_set:Nn \l__wa_jump_int \c_one_int
```

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

```
748 \str_set:Nn \l__wa_format_str { rl }
```

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

```
749 \LaTeX
750 \seq_clear_new:N \l__wa_labels_seq
751 \bool_set_false:N \l__wa_tag_next_line_bool
752 \LaTeX
```

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.³³

```
753 \skip_zero:N \l__wa_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`.

```
754 \tl_clear_new:N \l__wa_code_before_tl
755 \tl_clear_new:N \l__wa_code_after_tl
```

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

```
756 \str_clear_new:N \l__wa_previous_key_str
757 \bool_if:NT \l__wa_in_WithArrows_bool
758 { \keys_set:nn { WithArrows / WithArrows } { #1 } }
759 \bool_if:NT \l__wa_in_DispWithArrows_bool
760 { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

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

```
761 \cs_set_eq:cN \l__wa_command_name_str \l__wa_Arrow_first_columns:
```

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

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

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

³³It's recalled that, by design, the option `interline` of an environment doesn't apply in the nested environments.

- 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.

```
763 \int_gset_eq:NN \g__wa_col_int \l__wa_nb_cols_int
```

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

```
764 \seq_clear_new:N \l__wa_format_seq
765 \seq_set_split:NnV \l__wa_format_seq { } \l__wa_format_str
```

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

```
766 \*LaTeX
767 \bool_if:NT \g__wa_footnote_bool { \begin { savenotes } }
768 \*LaTeX
```

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

```
769 \l__wa_code_before_tl
```

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

```
770 \spread@equation
771 \*LaTeX
772 \cs_set_eq:NN \notag \__wa_notag:
773 \cs_set_eq:NN \nonumber \__wa_nonumber:
774 \cs_set_eq:NN \tag \__wa_tag
775 \cs_set_eq:NN \__wa_old_label \label
776 \cs_set_eq:NN \label \__wa_label:n
777 \cs_set_eq:NN \tagnextline \__wa_tagnextline:
778 \*LaTeX
779 }
```

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

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

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

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

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

- The construction of the preamble is done by reading backwards the format `\l_@@_format_str` and adding the corresponding tokens in the input stream of TeX. That means that the part of the preamble concerning the last cell will be constructed first.
- The function `\@@_construct_halign:` is recursive in order to treat succesively all the letters of the preamble.

- Each part of the preamble is created with a `\use:x` function. This expansion of the preamble gives the ability of controlling which parts of the code will be expanded during the construction of the preamble (other parts will be expanded and executed only during the execution of the `\halign`).
- The counter `\g_@@_col_int` is used during the loop of the construction of the preamble but, it will also appears in the preamble (we could have chosen two different counters but this way saves a counter).

```

780 \cs_new_protected:Npn \__wa_construct_halign:
781 {
782   \seq_pop_right:NNTF \l__wa_format_seq \l__wa_type_col_str
783   {

```

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

```

784   \use:x
785   {

```

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:x`.

We begin the construction of a generic column.

```

786       \int_gdecr:N \g__wa_col_int
787       \__wa_construct_halign:
788       \int_compare:nNnT \g__wa_col_int = \l__wa_nb_cols_int
789       {

```

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

```

790       \cs_set_eq:cN { \l__wa_command_name_str } \__wa_Arrow
791   < *LaTeX >
792       \bool_if:NT \l__wa_in_DispWithArrows_bool
793       {

```

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).

```

794       \__wa_test_if_to_tag:

```

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

```

795       \bool_if:NT \c__wa_amsthm_loaded_bool \__wa_set_qedhere:
796   }
797 < /LaTeX >
798   }
799   \str_if_eq:VnT \l__wa_type_col_str { c } \hfil
800   \str_if_eq:VnT \l__wa_type_col_str { r } \hfill
801   \int_gincr:N \g__wa_col_int
802   \int_gset:Nn \g__wa_static_col_int { \int_use:N \g__wa_col_int }
803   \c_math_toggle_token
804   {
805     { }
806     \bool_if:NT \l__wa_displaystyle_bool \displaystyle
807     ####
808   }
809   \c_math_toggle_token
810   \int_compare:nNnTF \g__wa_col_int = \l__wa_nb_cols_int
811   { \__wa_construct_nodes: }
812   {

```

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:`).

```

813         \str_if_eq:VnT \l__wa_type_col_str { l } \hfil
814         \str_if_eq:VnT \l__wa_type_col_str { c } \hfil
815         \bool_if:NT \l__wa_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
816     &
817 }
818 }
819 }

```

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

```

820 {
821     \bool_if:NTF \l__wa_in_WithArrows_bool
822     {
823         \ialign
824         \bgroup
825     }
826     {
827         \halign to \l__wa_linewidth_dim
828         \bgroup
829         \bool_if:NT \l__wa_fleqn_bool
830         { \skip_horizontal:N \l__wa_mathindent_dim }
831     }
832     \int_gincr:N \g__wa_line_int
833     \int_gzero:N \g__wa_col_int
834     \tl_if_eq:NNF \l__wa_left_brace_tl \c_novalue_tl
835     {
836         \skip_horizontal:n
837         { \box_wd:N \l__wa_left_brace_box + \l__wa_delim_wd_dim }
838     }
839     \strut
840 }
841 }

```

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.

```

842 \cs_new_protected:Npn \__wa_construct_nodes:
843 {

```

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).

```

844     \tikz [ remember-picture , overlay ]
845     \node
846     [
847         node~contents = { } ,
848         __wa_node_style ,
849         name = wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - 1 ,
850         alias =
851         {
852             \str_if_empty:NF \l__wa_name_str
853             { \l__wa_name_str - \int_use:N \g__wa_line_int - 1 }
854         }
855     ]
856     ;
857     \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).

```

858     \tikz [ remember-picture , overlay ]
859     \node
860     [

```

```

861     node~contents = { } ,
862     __wa_node_style ,
863     name = wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - r ,
864     alias =
865     {
866         \str_if_empty:NF \l__wa_name_str
867         { \l__wa_name_str - \int_use:N \g__wa_line_int - r }
868     }
869 ]
870 ;
871 \bool_if:NT \l__wa_show_node_names_bool
872 {
873     \hbox_overlap_right:n
874     { \small wa - \l__wa_prefix_str - \int_use:N \g__wa_line_int - r }
875 }
876 }

```

11.8.3 The environment {WithArrows}

```

877 <*LaTeX>
878 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
879 </LaTeX>
880 <*plain-TeX>
881 \cs_new_protected:Npn \WithArrows
882 {
883     \group_begin:
884     \peek_meaning:NTF [
885     { \WithArrows_i }
886     { \WithArrows_i [ ] }
887 }
888 \cs_new_protected:Npn \WithArrows_i [ #1 ]
889 </plain-TeX>
890 {
891     \bool_set_true:N \l__wa_in_WithArrows_bool
892     \bool_set_false:N \l__wa_in_DispWithArrows_bool
893 <*plain-TeX>
894     \str_clear_new:N \l__wa_type_env_str
895     \str_set:Nn \l__wa_type_env_str { WithArrows }
896 </plain-TeX>
897     \__wa_pre_halign:n { #1 }
898     \if_mode_math: \else:
899         \__wa_error:n { WithArrows~outside~math~mode }
900     \fi:

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`³⁴ 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 mode³⁵ and therefore, we can use `\vcenter`.

```

901     \int_case:nn \l__wa_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
902     \bgroup

```

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

```

903     \__wa_construct_halign:

```

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

³⁴Notice that the use of `\vtop` seems color-safe here...

³⁵An error is raised if the environment is used outside math mode.

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`.

```

904    &&
905    \__wa_error:n { Too-much-columns-in-WithArrows }
906    \c_math_toggle_token
907    \bool_if:NT \l__wa_displaystyle_bool \displaystyle
908    { ## }
909    \c_math_toggle_token
910    \cr
911  }

```

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

```

912  \<plain-TeX>
913  \cs_new_protected:Npn \endWithArrows
914  \</plain-TeX>
915  {
916    \\\
917    \egroup
918    \egroup
919    \__wa_post_halign:

```

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

```

920  \<LaTeX>
921    \bool_if:NT \g__wa_footnote_bool { \end { savenotes } }
922  \</LaTeX>
923  \<plain-TeX>
924    \group_end:
925  \</plain-TeX>
926  }

```

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

11.8.4 After the construction of the `\halign`

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

```

927  \cs_new_protected:Npn \__wa_post_halign:

```

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

```

928  {
929    \cs_set:Npn \WithArrowsRightX { \g__wa_right_x_dim }

```

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

```

930    \int_compare:nNnT \g__wa_arrow_int > 0
931    {

```

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).

```

932      \int_compare:nNnT \g__wa_arrow_int = 1
933      {
934        \int_compare:nNnT \l__wa_pos_arrow_int > 5
935        { \int_set:Nn \l__wa_pos_arrow_int 5 }
936      }
937      \__wa_scan_arrows:
938    }

```

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

```
939 \group_begin:
940 \tikzset { every-picture / .style = __wa_standard }
```

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

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

The command `\MultiArrow` is available in `code-after`, and we have a special version of `\Arrow`, called “`\Arrow` in `code-after`” in the documentation.³⁶

```
942 \cs_set_eq:NN \MultiArrow \__wa_MultiArrow:nn
943 \cs_set_eq:cN \l__wa_command_name_str \__wa_Arrow_code_after
944 \bool_set_true:N \l__wa_in_code_after_bool
945 \l__wa_code_after_tl
946 \group_end:
```

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

```
947 \seq_gpop_right:NN \g__wa_position_in_the_tree_seq \l_tmpa_tl
948 \seq_gpop_right:NN \g__wa_position_in_the_tree_seq \l_tmpa_tl
949 \seq_gput_right:Nx \g__wa_position_in_the_tree_seq
950 { \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`.

```
951 \int_compare:nNnT { \seq_count:N \g__wa_position_in_the_tree_seq } = 1
952 { \int_gincr:N \g__wa_last_env_int }
```

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

```
953 \seq_gpop_right:NN \g__wa_arrow_int_seq \l_tmpa_tl
954 \int_gset:Nn \g__wa_arrow_int \l_tmpa_tl
955 \seq_gpop_right:NN \g__wa_line_int_seq \l_tmpa_tl
956 \int_gset:Nn \g__wa_line_int \l_tmpa_tl
957 \seq_gpop_right:NN \g__wa_col_int_seq \l_tmpa_tl
958 \int_gset:Nn \g__wa_col_int \l_tmpa_tl
959 }
```

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

11.8.5 The command of end of row

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

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

```
960 \cs_new_protected:Npn \__wa_cr:
961 {
962 \scan_stop:
```

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

```
963 \int_compare:nNnF \g__wa_col_int = \g__wa_static_col_int
964 { \__wa_error:n { omit-probably-used } }
965 \prg_replicate:nn { \l__wa_nb_cols_int - \g__wa_static_col_int } { & { } }
966 \group_align_safe_begin:
967 \peek_meaning_remove:NTF * \__wa_cr_i: \__wa_cr_i:
968 }
```

³⁶As for now, `\MultiArrow` has no option, and that's why its internal name is a name of `expl3` with the signature `:nn` whereas `\Arrow` in `code-after` provides options and has the name of a function defined with `\NewDocumentCommand`.

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.

```
969 \cs_new_protected:Npn \__wa_cr_i:
970   { \peek_meaning:NTF [ \__wa_cr_ii: { \__wa_cr_ii: [ \c_zero_dim ] } }
```

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

```
971 <*LaTeX>
972 \cs_new_protected:Npn \__wa_cr_ii: [ #1 ]
973   {
974     \peek_meaning_ignore_spaces:NTF \end
975     {
976       \__wa_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`.

```
977       \__wa_analyze_end:Nn
978     }
979     { \__wa_cr_iii:n { #1 } }
980   }
981 \cs_new_protected:Npn \__wa_cr_iii:n #1
982 </LaTeX>
983 <*plain-TeX>
984 \cs_new_protected:Npn \__wa_cr_ii: [ #1 ]
985 </plain-TeX>
986   {
987     \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.³⁷

```
988   \bool_if:NT \l__wa_in_DispWithArrows_bool
```

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

```
989   {
990 <*LaTeX>
991     \clist_if_in:NnTF \l__wa_tags_clist { all }
992     {
```

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).

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

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

```
994       \cs_gset:Npx \g_tmpa_tl
995       { \tl_if_empty:NTF \l__wa_tag_tl \theequation \l__wa_tag_tl }
```

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

```
996       \seq_if_empty:NF \l__wa_labels_seq
997       {
```

³⁷The v-nodes are used to compute the abscissa of the right margin, used by the option `wrap-lines`.

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`.

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

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

```
999 \bool_if:NT \c__wa_hyperref_loaded_bool
1000 {
1001   \str_set:Nn \This@name { equation }
1002   \hyper@refstepcounter { equation }
1003 }
```

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.

```
1004 \bool_if:NT \c__wa_cleveref_loaded_bool
1005 {
1006   \cref@constructprefix { equation } \cref@result
1007   \protected@edef \cref@currentlabel
1008   {
1009     [
1010       \cs_if_exist:NTF \cref@equation@alias
1011       \cref@equation@alias
1012       { equation }
1013     ]
1014     [ \arabic { equation } ] [ \cref@result ]
1015     \p@equation \g_tmpa_tl
1016   }
1017 }
```

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`).

```
1018 \seq_map_function:NN \l__wa_labels_seq \__wa_old_label
1019 }
```

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.

```
1020 \__wa_save:N \l__wa_tag_star_bool
1021 \__wa_save:N \l__wa_qedhere_bool
1022 \bool_if:NT \l__wa_tag_next_line_bool
1023 {
1024   \openup -\jot
1025   \bool_set_false:N \l__wa_tag_next_line_bool
1026   \notag \&
1027 }
1028 &
1029 \__wa_restore:N \l__wa_tag_star_bool
1030 \__wa_restore:N \l__wa_qedhere_bool
1031 \bool_if:NT \l__wa_qedhere_bool
1032 { \hbox_overlap_left:n \__wa_qedhere_i: }
1033 \cs_set_eq:NN \theequation \g_tmpa_tl
1034 \bool_if:NT \l__wa_tag_star_bool
1035 { \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`.

```
1036 \hbox_overlap_left:n
1037 {
1038   \bool_if:NF \c__wa_leqno_bool
1039   {
1040     \tikz [ __wa_standard ]
1041       \coordinate ( \int_use:N \g__wa_line_int - v ) ;
```



```

1042         }
1043         \quad
1044         \@eqnnum
1045     }
1046     \bool_if:NT \c__wa_leqno_bool
1047     {
1048         \tikz [ __wa_standard ]
1049             \coordinate ( \int_use:N \g__wa_line_int - v ) ;
1050     }
1051 }
1052 {
1053     \__wa_save:N \l__wa_qedhere_bool
1054 </LaTeX>
1055     &
1056 <*LaTeX>
1057     \__wa_restore:N \l__wa_qedhere_bool
1058     \bool_if:NT \l__wa_qedhere_bool
1059     { \hbox_overlap_left:n \__wa_qedhere_i: }
1060 </LaTeX>
1061     \tikz [ __wa_standard ]
1062         \coordinate ( \int_use:N \g__wa_line_int - v ) ;
1063 <*LaTeX>
1064 }
1065 </LaTeX>
1066 }
1067 \dim_compare:nNnT { #1 } < \c_zero_dim
1068 { \__wa_error:n { option~of~cr~negative } }
1069
1070 \cr
1071 \noalign
1072 {
1073     \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1074     \skip_vertical:n { \l_tmpa_dim + \l__wa_interline_skip }
1075     \scan_stop:
1076 }
1077 }

```

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

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

```

1078 <*LaTeX>
1079 \cs_new_protected:Npn \__wa_analyze_end:Nn #1 #2
1080 {
1081     \exp_args:NV \str_if_eq:nnT \l__wa_type_env_str { #2 }
1082     {
1083         \__wa_warning:n { newline~at~the~end~of~env }
1084         \group_begin:
1085         \globaldefs = 1
1086         \__wa_msg_redirect_name:nn { newline~at~the~end~of~env } { none }
1087         \group_end:
1088     }

```

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

```

1089     \end { #2 }
1090 }
1091 </LaTeX>

```

11.8.6 The environment `{DispWithArrows}`

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

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

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

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

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

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

```

1092 <*LaTeX>
1093 \bool_new:N \l__wa_in_label_or_minipage_bool
1094 </LaTeX>

1095 <*LaTeX>
1096 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1097 </LaTeX>
1098 <*plain-TeX>
1099 \cs_new_protected:Npn \DispWithArrows
1100 {
1101   \group_begin:
1102   \peek_meaning:NTF <
1103     { \DispWithArrows_i }
1104     { \DispWithArrows_i < \c_novalue_tl > }
1105   }
1106 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1107 {
1108   \peek_meaning:NTF [
1109     { \DispWithArrows_ii < #1 > }
1110     { \DispWithArrows_ii < #1 > [ ] }
1111   }
1112 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1113 </plain-TeX>
1114 {
1115   \bool_set_true:N \l__wa_in_DispWithArrows_bool
1116 <*plain-TeX>
1117   \str_clear_new:N \l__wa_type_env_str
1118   \str_set:Nn \l__wa_type_env_str { DispWithArrows }
1119 </plain-TeX>

```

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

```

1120 <*LaTeX>
1121 \bool_if:nT \c__wa_mathtools_loaded_bool
1122 {
1123   \MH_if_boolean:nT { show_only_refs }
1124   {
1125     \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`.

```

1126     \MH_set_boolean:T:n { show_only_refs }
1127   }
1128 }

```

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`.

```

1129 \bool_if:NT \c__wa_typedref_loaded_bool { \str_set:Nn \sr@name { equation } }

```

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

```

1130 \bool_if:NT \c__wa_amsmath_loaded_bool \intertext@
1131 </LaTeX>
1132 \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l__wa_left_brace_tl { #1 } }

```

```
1133 \__wa_pre_halign:n { #2 }
```

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

```
1134 (*LaTeX)
1135 \bool_if:NT \l__wa_subequations_bool { \begin { subequations } }
1136 </LaTeX>
```

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

```
1137 (*LaTeX)
1138 \bool_if:NF \l__wa_sbwi_bool
1139 {
1140   \if@inlabel
1141     \bool_set_true:N \l__wa_in_label_or_minipage_bool
1142   \fi
1143   \if@minipage
1144     \bool_set_true:N \l__wa_in_label_or_minipage_bool
1145   \fi
1146 }
1147 </LaTeX>

1148 \tl_if_eq:NNF \l__wa_left_brace_tl \c_novaluel_tl
1149 {
```

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

```
1150 \hbox_set:Nn \l_tmpa_box
1151 {
```

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

```
1152 \group_begin:
1153 \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1154 \c_math_toggle_token
1155 \left \l__wa_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1156 \c_math_toggle_token
1157 \group_end:
1158 }
1159 \dim_zero_new:N \l__wa_delim_wd_dim
1160 \dim_set:Nn \l__wa_delim_wd_dim { \box_wd:N \l_tmpa_box }
1161 \box_clear_new:N \l__wa_left_brace_box
1162 \hbox_set:Nn \l__wa_left_brace_box
1163 {
1164   \group_begin:
1165     \cs_set_eq:NN \label \__wa_old_label
1166     \c_math_toggle_token
1167     \bool_if:NT \l__wa_displaystyle_bool \displaystyle
1168     \l__wa_left_brace_tl
1169     { }
1170     \c_math_toggle_token
1171   \group_end:
1172 }
1173 }
```

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

```
1174 (*LaTeX)
1175 \tl_clear_new:N \l__wa_tag_tl

1176 \bool_set_false:N \l__wa_qedhere_bool
```

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

```
1177 \bool_set_false:N \l__wa_tag_star_bool
1178 </LaTeX>
```

```

1179 \if_mode_math:
1180     \__wa_fatal:n { DispWithArrows~in~math~mode }
1181 \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`.

```

1182 <*plain-TeX>
1183     \dim_zero_new:N \linewidth
1184     \dim_set_eq:NN \linewidth \displaywidth
1185 </plain-TeX>
1186 <*LaTeX>
1187     \bool_if:NTF \l__wa_in_label_or_minipage_bool
1188         { \c_math_toggle_token }
1189         {
1190 </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.

```

1191     \if_mode_vertical:
1192     \nointerlineskip
1193     \hbox_to_wd:nn { .6 \linewidth } { }
1194     \fi:
1195     \c_math_toggle_token \c_math_toggle_token
1196 <*LaTeX>
1197     }
1198 </LaTeX>
1199     \dim_zero_new:N \l__wa_linewidth_dim
1200 <*LaTeX>
1201     \bool_if:NTF \l__wa_in_label_or_minipage_bool
1202         { \dim_set_eq:NN \l__wa_linewidth_dim \linewidth }
1203         { \dim_set_eq:NN \l__wa_linewidth_dim \displaywidth }
1204 </LaTeX>
1205 <*plain-TeX>
1206     \dim_set_eq:NN \l__wa_linewidth_dim \displaywidth
1207 </plain-TeX>
1208     \box_clear_new:N \l__wa_halign_box
1209     \setbox \l__wa_halign_box \vtop \bgroup
1210     \tabskip =
1211     \bool_if:NTF \l__wa_fleqn_bool
1212         \c_zero_skip
1213         { 0 pt plus 1000 pt minus 1000 pt }
1214     \__wa_construct_halign:
1215     \tabskip = 0 pt plus 1000 pt minus 1000 pt
1216     &

```

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.

```

1217     $ ## $
1218     \tabskip = \c_zero_skip
1219     &&
1220     \__wa_fatal:n { Too~much~columns~in~DispWithArrows }
1221     \bool_if:nT \c_false_bool { ## }
1222     \cr
1223 }

```

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

```

1224 <*plain-TeX>
1225 \cs_new_protected:Npn \endDispWithArrows

```

```

1226 </plain-TeX>
1227 {
1228 <*LaTeX>
1229   \clist_if_in:NnT \l__wa_tags_clist { last }
1230   { \clist_set:Nn \l__wa_tags_clist { all } }
1231 </LaTeX>
1232 \}

```

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

```

1233 \egroup
1234 \unskip \unpenalty \unskip \unpenalty
1235 \box_set_to_last:N \l_tmpa_box
1236 \nointerlineskip
1237 \box_use:N \l_tmpa_box
1238 \dim_gzero_new:N \g__wa_alignment_dim
1239 \dim_gset:Nn \g__wa_alignment_dim { \box_wd:N \l_tmpa_box }
1240 \box_clear_new:N \l__wa_new_box
1241 \hbox_set:Nn \l__wa_new_box { \hbox_unpack_clear:N \l_tmpa_box }
1242 \dim_compare:nNnT
1243   { \box_wd:N \l__wa_new_box } < \g__wa_alignment_dim
1244   { \dim_gset:Nn \g__wa_alignment_dim { \box_wd:N \l__wa_new_box } }

```

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

```

1245 \egroup
1246 \tl_if_eq:NNTF \l__wa_left_brace_tl \c_novalue_tl
1247   { \box_use_drop:N \l__wa_halign_box }
1248   {
1249     \hbox_to_wd:nn \l__wa_linewidth_dim
1250     {
1251       \bool_if:NNTF \l__wa_fleqn_bool
1252         { \skip_horizontal:n \l__wa_mathindent_dim }
1253       \hfil
1254       \hbox_to_wd:nn \g__wa_alignment_dim
1255       {
1256         \box_use_drop:N \l__wa_left_brace_box
1257         \dim_set:Nn \l_tmpa_dim
1258         {
1259           \box_ht:N \l__wa_halign_box
1260           + \box_dp:N \l__wa_halign_box
1261         }
1262         \group_begin:
1263         \dim_set_eq:NN \nullldelimiterspace \c_zero_dim
1264         \c_math_toggle_token
1265         \left \l__wa_replace_left_brace_by_tl
1266         \vcenter to \l_tmpa_dim { \vfil }
1267         \right.
1268         \c_math_toggle_token
1269         \group_end:
1270         \hfil
1271       }
1272       \hfil
1273     }
1274     \skip_horizontal:n { - \l__wa_linewidth_dim }
1275     \vcenter { \box_use_drop:N \l__wa_halign_box }
1276   }

```

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.

```

1277 \dim_gzero_new:N \g__wa_right_x_dim
1278 \dim_gset_eq:NN \g__wa_right_x_dim \c_max_dim
1279 <*LaTeX>

```

```

1280 \begin { tikzpicture } [ __wa_standard ]
1281 </LaTeX>
1282 <*plain-TeX>
1283 \tikzpicture [ __wa_standard ]
1284 </plain-TeX>
1285 \int_step_variable:nNn \g__wa_line_int \l_tmpa_int
1286 {
1287 \cs_if_free:cTF
1288 { pgf@sh@ns@wa - \l__wa_prefix_str - \l_tmpa_int - v }
1289 { \__wa_fatal:n { Inexistent~v-node } }
1290 {
1291 \tikz@parse@node\pgfutil@firstofone ( \l_tmpa_int - v )
1292 \dim_set:Nn \l_tmpa_dim \pgf@x
1293 \dim_compare:nNnT \l_tmpa_dim < \g__wa_right_x_dim
1294 { \dim_gset:Nn \g__wa_right_x_dim \l_tmpa_dim }
1295 }
1296 }
1297 <*LaTeX>
1298 \end { tikzpicture }
1299 </LaTeX>
1300 <*plain-TeX>
1301 \endtikzpicture
1302 </plain-TeX>

```

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

```

1303 \__wa_post_halign:

```

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

```

1304 <*LaTeX>
1305 \bool_if:nT \c__wa_mathtools_loaded_bool
1306 { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1307 \bool_if:NTF \l__wa_in_label_or_minipage_bool
1308 {
1309 \c_math_toggle_token
1310 \skip_vertical:N \belowdisplayskip
1311 }
1312 { \c_math_toggle_token \c_math_toggle_token }
1313 </LaTeX>
1314 <*plain-TeX>
1315 \c_math_toggle_token \c_math_toggle_token
1316 </plain-TeX>
1317 <*LaTeX>
1318 \bool_if:NT \l__wa_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`).

```

1319 \bool_if:NT \g__wa_footnote_bool { \end { savenotes } }
1320 </LaTeX>
1321 <*plain-TeX>
1322 \group_end:
1323 </plain-TeX>
1324 <*LaTeX>
1325 \ignorespacesafterend
1326 </LaTeX>
1327 }

```

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

```

1328 <*LaTeX>
1329 \NewDocumentEnvironment { DispWithArrows* } { }
1330 {
1331     \WithArrowsOptions { notag }
1332     \DispWithArrows
1333 }
1334 \endDispWithArrows
1335 </LaTeX>

```

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

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

```

1336 \cs_new_protected:Npn \__wa_if_in_last_col_of_disp:Nn #1 #2
1337 {
1338     \bool_if:NTF \l__wa_in-WithArrows_bool
1339     { \__wa_error:nn { Not-allowed-in-WithArrows } { #1 } }
1340     {
1341         \int_compare:nNnTF \g__wa_col_int < \l__wa_nb_cols_int
1342         { \__wa_error:nn { Not-allowed-in-DispWithArrows } { #1 } }
1343         { #2 }
1344     }
1345 }

```

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

```

1346 <*LaTeX>
1347 \cs_new_protected:Npn \__wa_notag:
1348 { \__wa_if_in_last_col_of_disp:Nn \notag { \clist_clear:N \l__wa_tags_clist } }

```

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

```

1349 \cs_new_protected:Npn \__wa_nonumber:
1350 { \__wa_if_in_last_col_of_disp:Nn \nonumber { \clist_clear:N \l__wa_tags_clist } }

```

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

```

1351 \NewDocumentCommand \__wa_tag { s m }
1352 {
1353     \__wa_if_in_last_col_of_disp:Nn \tag
1354     {
1355         \tl_if_empty:NF \l__wa_tag_tl
1356         { \__wa_error:nn { Multiple-tags } { #2 } }
1357         \clist_set:Nn \l__wa_tags_clist { all }
1358         \bool_if:nT \c__wa_mathtools_loaded_bool
1359         {
1360             \MH_if_boolean:nT { show_only_refs }
1361             {
1362                 \MH_if_boolean:nF { show_manual_tags }
1363                 { \clist_clear:N \l__wa_tags_clist }
1364             }
1365         }
1366         \tl_set:Nn \l__wa_tag_tl { #2 }
1367         \bool_set:Nn \l__wa_tag_star_bool { #1 }

```

The starred version `\tag*` can't be used if `amsmath` has not been loaded because this version does the job by deactivating the command `\tagform@` inserted by `amsmath` in the (two versions of the) command `\@eqnnum`.³⁸

```

1368     \bool_if:nT { #1 && ! \bool_if_p:N \c__wa_amsmath_loaded_bool }
1369     { \__wa_error:n { tag*~without~amsmath } }
1370   }
1371 }

```

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__wa_labels_seq`.

```

1372 \cs_new_protected:Npn \__wa_label:n #1
1373 {
1374   \__wa_if_in_last_col_of_disp:Nn \label
1375   {
1376     \seq_if_empty:NF \l__wa_labels_seq
1377     {
1378       \bool_if:NTF \c__wa_cleveref_loaded_bool
1379       { \__wa_error:n { Multiple~labels-with~cleveref } }
1380       { \__wa_error:n { Multiple~labels } }
1381     }
1382     \seq_put_right:Nn \l__wa_labels_seq { #1 }
1383     \bool_if:nT \c__wa_mathtools_loaded_bool
1384     {
1385       \MH_if_boolean:nT { show_only_refs }
1386       {
1387         \cs_if_exist:cTF { MT_r_#1 }
1388         { \clist_set:Nn \l__wa_tags_clist { all } }
1389         { \clist_clear:N \l__wa_tags_clist }
1390       }
1391     }
1392     \bool_if:nT \c__wa_autonum_loaded_bool
1393     {
1394       \cs_if_exist:cTF { autonum@#1Referenced }
1395       { \clist_set:Nn \l__wa_tags_clist { all } }
1396       { \clist_clear:N \l__wa_tags_clist }
1397     }
1398   }
1399 }

```

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

```

1400 \cs_new_protected:Npn \__wa_tagnextline:
1401 {
1402   \__wa_if_in_last_col_of_disp:Nn \tagnextline
1403   { \bool_set_true:N \l__wa_tag_next_line_bool }
1404 }

```

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__wa_qedhere_bool`.

```

1405 \cs_new_protected:Npn \__wa_qedhere: { \bool_set_true:N \l__wa_qedhere_bool }
1406 \cs_new_protected:Npn \__wa_set_qedhere: { \cs_set_eq:NN \qedhere \__wa_qedhere: }

```

³⁸There are two versions of `@eqnnum`, a standard version and a version for the option `leqno`.

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`.

```

1407 \cs_new_protected:Npn \__wa_qedhere_i:
1408 {
1409   \group_begin:
1410   \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.

```

1411   \cs_set_eq:NN \qed@elt \setQED@elt
1412   \QED@stack \relax \relax
1413   \group_end:
1414 }
1415 </LaTeX>

```

11.10 We draw the arrows

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

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

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

```

1416 \cs_new_protected:Npn \__wa_scan_arrows:
1417 {
1418   \group_begin:
1419   \int_compare:nNnT \l__wa_pos_arrow_int = 7
1420   {
1421     \__wa_scan_arrows_i:
1422     \int_set:Nn \l__wa_pos_arrow_int 8
1423   }
1424   \__wa_scan_arrows_i:
1425   \group_end:
1426 }

1427 \cs_new_protected:Npn \__wa_scan_arrows_i:
1428 {

```

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

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

`\l_@@_first_arrows_seq` will be the list the arrows of the group starting at the first line of the group (we may have several arrows starting from the same line). We have to know all these arrows because of the ajustement by `\l_@@_start_adjust_dim`.

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

`\l_@@_last_arrows_seq` will be the list of all the arrows of the group ending at the last line of the group (impossible to guess in advance).

```

1429   \int_zero_new:N \l__wa_first_arrow_of_group_int
1430   \int_zero_new:N \l__wa_first_line_of_group_int
1431   \int_zero_new:N \l__wa_last_line_of_group_int
1432   \seq_clear_new:N \l__wa_first_arrows_seq
1433   \seq_clear_new:N \l__wa_last_arrows_seq

```

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

```
1434 \bool_set_true:N \l__wa_new_group_bool
```

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

```
1435 \int_set:Nn \l__wa_arrow_int \c_one_int
1436 \int_until_do:nNnn \l__wa_arrow_int > \g__wa_arrow_int
1437 {
```

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.

```
1438 \prop_get:cnN
1439 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1440 { initial } \l_tmpa_tl
1441 \int_set:Nn \l__wa_initial_int \l_tmpa_tl
1442 \prop_get:cnN
1443 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1444 { final } \l_tmpa_tl
1445 \int_set:Nn \l__wa_final_int \l_tmpa_tl
1446 \prop_get:cnN
1447 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1448 { status } \l__wa_status_arrow_str
1449 \prop_get:cnN
1450 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1451 { input-line } \l__wa_input_line_str
```

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

```
1452 \int_compare:nNnTF \l__wa_final_int > \g__wa_line_int
1453 {
1454 \int_compare:nNnF \l__wa_pos_arrow_int = 8
1455 { \__wa_error:n { Too~few~lines~for~an~arrow } }
1456 }
1457 \__wa_code_for_possible_arrow:
```

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

```
1458 \int_incr:N \l__wa_arrow_int
1459 }
```

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).

```
1460 \bool_if:nT
1461 {
1462 \int_compare_p:n { \l__wa_pos_arrow_int != 7 }
1463 &&
1464 \int_compare_p:nNn \l__wa_first_arrow_of_group_int > 0
1465 }
1466 { \__wa_draw_arrows:nn \l__wa_first_arrow_of_group_int \g__wa_arrow_int }
1467 }
```

```
1468 \cs_new_protected:Npn \__wa_code_for_possible_arrow:
1469 {
```

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`).

```

1470 \bool_if:nT
1471 {
1472   \int_compare_p:nNn \l__wa_arrow_int > \c_one_int
1473   &&
1474   ( \int_compare_p:n { \l__wa_initial_int > \l__wa_last_line_of_group_int }
1475     &&
1476     \int_compare_p:n { \l__wa_pos_arrow_int != 7 }
1477     ||
1478     \str_if_eq_p:Vn \l__wa_status_arrow_str { new-group }
1479   )
1480 }
1481 {
1482   \int_compare:nNnF \l__wa_first_arrow_of_group_int = \c_zero_int
1483   {
1484     \__wa_draw_arrows:nn
1485       \l__wa_first_arrow_of_group_int
1486       { \l__wa_arrow_int - 1 }
1487   }
1488   \bool_set_true:N \l__wa_new_group_bool
1489 }

```

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

```

1490 \bool_if:nTF \l__wa_new_group_bool
1491 {
1492   \bool_set_false:N \l__wa_new_group_bool
1493   \int_set_eq:NN \l__wa_first_arrow_of_group_int \l__wa_arrow_int
1494   \int_set_eq:NN \l__wa_first_line_of_group_int \l__wa_initial_int
1495   \int_set_eq:NN \l__wa_last_line_of_group_int \l__wa_final_int
1496   \seq_clear:N \l__wa_first_arrows_seq
1497   \seq_put_left:Nv \l__wa_first_arrows_seq \l__wa_arrow_int
1498   \seq_clear:N \l__wa_last_arrows_seq
1499   \seq_put_left:Nv \l__wa_last_arrows_seq \l__wa_arrow_int

```

If we are in option `group` and in the second step of treatment (`\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.

```

1500   \int_compare:nT { \l__wa_pos_arrow_int != 8 }
1501   { \dim_set:Nn \l__wa_x_dim { - \c_max_dim } }
1502 }

```

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

```

1503 {

```

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

```

1504   \bool_if:nF
1505   { \str_if_eq_p:Vn \l__wa_status_arrow_str { independent } }
1506   {

```

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

```

1507       \int_compare:nT
1508       { \l__wa_initial_int = \l__wa_first_line_of_group_int }

```

```

1509         { \seq_put_left:NV \l__wa_first_arrows_seq \l__wa_arrow_int }
1510     \int_compare:nNnTF \l__wa_final_int > \l__wa_last_line_of_group_int
1511     {
1512         \int_set_eq:NN \l__wa_last_line_of_group_int \l__wa_final_int
1513         \seq_clear:N \l__wa_last_arrows_seq
1514         \seq_put_left:NV \l__wa_last_arrows_seq \l__wa_arrow_int
1515     }
1516     {
1517         \int_compare:nNnT \l__wa_final_int = \l__wa_last_line_of_group_int
1518         { \seq_put_left:NV \l__wa_last_arrows_seq \l__wa_arrow_int }
1519     }
1520 }
1521 }

```

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

```

1522     \bool_if:nF { \str_if_eq_p:Vn \l__wa_status_arrow_str { independent } }
1523     {
1524         \int_compare:nT { \l__wa_pos_arrow_int != 8 }
1525         { \l__wa_update_x:nn \l__wa_initial_int \l__wa_final_int }
1526     }
1527 }

```

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

```

1528 \cs_generate_variant:Nn \keys_set:nn { n o }
1529 \cs_new_protected:Npn \__wa_keys_set:
1530 { \keys_set:known:no { WithArrows / Arrow / SecondPass } }

```

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

```

1531 \cs_new_protected:Npn \__wa_draw_arrows:nn #1 #2
1532 {
1533     \group_begin:
1534     \int_zero_new:N \l__wa_first_arrow_int
1535     \int_set:Nn \l__wa_first_arrow_int { #1 }
1536     \int_zero_new:N \l__wa_last_arrow_int
1537     \int_set:Nn \l__wa_last_arrow_int { #2 }

```

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

```

1538     \int_set:Nn \l__wa_arrow_int \l__wa_first_arrow_int
1539     \int_until_do:nNnn \l__wa_arrow_int > \l__wa_last_arrow_int
1540     {

```

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.

```

1541         \prop_get:cnN
1542         { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1543         { initial } \l_tmpa_tl
1544         \int_set:Nn \l__wa_initial_int \l_tmpa_tl
1545         \prop_get:cnN
1546         { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1547         { final } \l_tmpa_tl
1548         \int_set:Nn \l__wa_final_int \l_tmpa_tl

```

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

```

1549     \int_compare:nT { \l__wa_final_int <= \g__wa_line_int } \__wa_draw_arrows_i:
1550     \int_incr:N \l__wa_arrow_int
1551   }
1552   \group_end:
1553 }

```

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

```

1554 \cs_new_protected:Npn \__wa_draw_arrows_i:
1555 {
1556   \group_begin:

```

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

```

1557   \prop_get:cnN
1558     { g__wa_arrow _\l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1559     { options } \l_tmpa_tl
1560   \str_clear_new:N \l__wa_previous_key_str
1561   \exp_args:NNo \exp_args:No
1562   \__wa_keys_set: { \l_tmpa_tl , tikz = { xshift = \l__wa_xoffset_dim } }

```

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

```

1563   \bool_set_false:N \l__wa_initial_r_bool
1564   \bool_set_false:N \l__wa_final_r_bool
1565   \int_case:nn \l__wa_pos_arrow_int
1566   {
1567     0 { \bool_set_true:N \l__wa_final_r_bool }
1568     2 { \bool_set_true:N \l__wa_initial_r_bool }
1569     3
1570     {
1571       \bool_set_true:N \l__wa_initial_r_bool
1572       \bool_set_true:N \l__wa_final_r_bool
1573     }
1574   }

```

| 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).

```

1575   \int_compare:nNnT \l__wa_pos_arrow_int = 5
1576   {
1577     \dim_set:Nn \l__wa_x_dim { - \c_max_dim }
1578     \__wa_update_x:nn \l__wa_initial_int \l__wa_final_int
1579   }

```

`\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`.

```

1580 \tl_set:Nx \l__wa_initial_tl
1581 {
1582     \int_use:N \l__wa_initial_int - \bool_if:NTF \l__wa_initial_r_bool rl
1583     .south
1584 }
1585 \tl_set:Nx \l__wa_final_tl
1586 { \int_use:N \l__wa_final_int - \bool_if:NTF \l__wa_final_r_bool rl .north }

```

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

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

```

1587 \prop_get:cnN
1588 { g__wa_arrow _ \l__wa_prefix_str _ \int_use:N \l__wa_arrow_int _ prop }
1589 { label }
1590 \l_tmpa_tl

```

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

```

1591 \seq_if_in:NxTF \l__wa_first_arrows_seq
1592 { \int_use:N \l__wa_arrow_int }
1593 { \bool_set_true:N \l_tmpa_bool }
1594 { \bool_set_false:N \l_tmpa_bool }
1595 \seq_if_in:NxTF \l__wa_last_arrows_seq
1596 { \int_use:N \l__wa_arrow_int }
1597 { \bool_set_true:N \l_tmpb_bool }
1598 { \bool_set_false:N \l_tmpb_bool }
1599 \int_compare:nNnT \l__wa_pos_arrow_int = 5
1600 {
1601     \bool_set_true:N \l_tmpa_bool
1602     \bool_set_true:N \l_tmpb_bool
1603 }

```

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 ajustement 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`).

```

1604 \dim_gzero_new:N \g__wa_x_initial_dim
1605 \dim_gzero_new:N \g__wa_x_final_dim
1606 \dim_gzero_new:N \g__wa_y_initial_dim
1607 \dim_gzero_new:N \g__wa_y_final_dim
1608 <LaTeX>
1609 \begin { tikzpicture } [ __wa_standard ]
1610 </LaTeX>
1611 <plain-TeX>
1612 \tikzpicture [ __wa_standard ]
1613 </plain-TeX>
1614 \tikz@scan@one@point \pgfutil@firstofone ( \l__wa_initial_tl )
1615 \dim_gset:Nn \g__wa_x_initial_dim \pgf@x
1616 \dim_gset:Nn \g__wa_y_initial_dim \pgf@y
1617 \tikz@scan@one@point \pgfutil@firstofone ( \l__wa_final_tl )
1618 \dim_gset:Nn \g__wa_x_final_dim \pgf@x

```

```

1619 \dim_gset:Nn \g__wa_y_final_dim \pgf@y
1620 (*LaTeX)
1621 \end { tikzpicture }
1622 (/LaTeX)
1623 (*plain-TeX)
1624 \endtikzpicture
1625 (/plain-TeX)
1626 \bool_if:nTF
1627 { \dim_compare_p:nNn { \g__wa_y_initial_dim - \g__wa_y_final_dim }
1628 > \l__wa_max_length_of_arrow_dim
1629 &&
1630 \int_compare_p:nNn { \l__wa_final_int - \l__wa_initial_int } = 1
1631 }
1632 {
1633 \tl_gset:Nx \g_tmpa_tl
1634 {
1635 \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1636 { \dim_use:N \g__wa_x_initial_dim }
1637 { \dim_use:N \l__wa_x_dim } ,
1638 \dim_eval:n
1639 {
1640 ( \g__wa_y_initial_dim + \g__wa_y_final_dim ) / 2
1641 + ( \l__wa_max_length_of_arrow_dim / 2 )
1642 }
1643 }
1644 \tl_gset:Nx \g_tmpb_tl
1645 {
1646 \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1647 { \dim_use:N \g__wa_x_final_dim }
1648 { \dim_use:N \l__wa_x_dim } ,
1649 \dim_eval:n
1650 {
1651 ( \g__wa_y_initial_dim + \g__wa_y_final_dim ) / 2
1652 - ( \l__wa_max_length_of_arrow_dim / 2 )
1653 }
1654 }
1655 }
1656 {
1657 \tl_gset:Nx \g_tmpa_tl
1658 {
1659 \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1660 { \dim_use:N \g__wa_x_initial_dim }
1661 { \dim_use:N \l__wa_x_dim } ,
1662 \bool_if:NTF \l_tmpa_bool
1663 { \dim_eval:n { \g__wa_y_initial_dim + \l__wa_start_adjust_dim } }
1664 { \dim_use:N \g__wa_y_initial_dim }
1665 }
1666 \tl_gset:Nx \g_tmpb_tl
1667 {
1668 \int_compare:nNnTF \l__wa_pos_arrow_int < 5
1669 { \dim_use:N \g__wa_x_final_dim }
1670 { \dim_use:N \l__wa_x_dim } ,
1671 \bool_if:NTF \l_tmpb_bool
1672 { \dim_eval:n { \g__wa_y_final_dim - \l__wa_end_adjust_dim } }
1673 { \dim_use:N \g__wa_y_final_dim }
1674 }
1675 }

```

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*.³⁹

```
1676 \__wa_draw_arrow:nno \g_tmpa_tl \g_tmpb_tl \l_tmpa_tl
```

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

```
1677 \group_end:
1678 }
```

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

```
1679 \cs_new_protected:Npn \__wa_def_function_tmpa:n #1
1680 {
1681   \cs_set:Npn \__wa_tmpa:nnn ##1 ##2 ##3
1682   {
1683     <*LaTeX>
1684     \begin{tikzpicture}
1685     </LaTeX>
1686     <*plain-TeX>
1687     \tikzpicture
1688     </plain-TeX>
1689     [
1690       __wa_standard ,
1691       every~path / .style = WithArrows / arrow
1692     ]
1693     #1
1694     <*LaTeX>
1695     \end{tikzpicture}
1696     </LaTeX>
1697     <*plain-TeX>
1698     \endtikzpicture
1699     </plain-TeX>
1700   }
1701 }
```

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

```
1702 \cs_new_protected:Npn \__wa_draw_arrow:nnn #1 #2 #3
1703 {
```

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`).

```
1704 \bool_if:nT { \l__wa_wrap_lines_bool && \l__wa_in_DisWithArrows_bool }
1705 { \tl_set_eq:NN \l__wa_tikz_code_tl \c__wa_tikz_code_wrap_lines_tl }
```

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

```
1706 \exp_args:NV \__wa_def_function_tmpa:n \l__wa_tikz_code_tl
1707 \__wa_tmpa:nnn { #1 } { #2 } { #3 }
1708 }
1709 \cs_generate_variant:Nn \__wa_draw_arrow:nnn { n n o }
```

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

```
1710 \tl_const:Nn \c__wa_tikz_code_wrap_lines_tl
1711 {
```

³⁹There were other solutions: use another name without *underscore* (like `\ltmpatl`) or use the package `underscore` (with this package, the characters *underscore* will be rescanned without errors, even in text mode).

First, we draw the arrow without the label.

```
1712 \draw ( #1 ) to node ( __wa_label ) { } ( #2 ) ;
```

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

```
1713 \tikz@parse@node \pgfutil@firstofone ( __wa_label.west )
```

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

```
1714 \dim_set:Nn \l_tmpa_dim
1715 { \g_@@_right_x_dim - \pgf@x - \pgfkeysvalueof { / pgf / inner~xsep } }
```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “text width”.⁴⁰

```
1716 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
```

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

```
1717 \tl_if_empty:NF \g_tmpa_tl
1718 {
1719   \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1720   \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1721     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1722 }
```

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

```
1723 \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1724 {
1725   \path ( __wa_label.west )
1726     node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1727       { #3 } ;
1728 }
1729 }
```

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`).

```
1730 \cs_new_protected:Npn \__wa_update_x:nn #1 #2
1731 {
1732   \int_step_inline:nnn { #1 } { #2 }
1733   {
1734     \*LaTeX
1735     \begin { tikzpicture } [ __wa_standard ]
1736     \*LaTeX
1737     \*plain-TeX
1738     \tikzpicture [ __wa_standard ]
1739     \*plain-TeX
1740     \tikz@scan@one@point \pgfutil@firstofone ( ##1 - 1 )
1741     \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \l_@@_x_dim \pgf@x }
1742     \*LaTeX
1743     \end { tikzpicture }
1744     \*LaTeX
1745     \*plain-TeX
1746     \endtikzpicture
1747     \*plain-TeX
1748     \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1749   }
1750 }
```

⁴⁰In 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.

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.

```
1751 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g__wa_last_env_int }
```

11.11 The command `\Arrow` in `code-after`

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

```
1752 \keys_define:nn { WithArrows / Arrow / code-after }
1753 {
1754   tikz      .code:n =
1755     \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
1756   tikz      .value_required:n = true ,
1757   rr        .value_forbidden:n = true ,
1758   rr        .code:n      = \__wa_fix_pos_option:n 0 ,
1759   ll        .value_forbidden:n = true ,
1760   ll        .code:n      = \__wa_fix_pos_option:n 1 ,
1761   rl        .value_forbidden:n = true ,
1762   rl        .code:n      = \__wa_fix_pos_option:n 2 ,
1763   lr        .value_forbidden:n = true ,
1764   lr        .code:n      = \__wa_fix_pos_option:n 3 ,
1765   v         .value_forbidden:n = true ,
1766   v         .code:n      = \__wa_fix_pos_option:n 4 ,
1767   tikz-code .tl_set:N      = \l__wa_tikz_code_tl ,
1768   tikz-code .value_required:n = true ,
1769   xoffset   .dim_set:N      = \l__wa_xoffset_dim ,
1770   xoffset   .value_required:n = true ,
1771   unknown   .code:n =
1772     \__wa_sort_seq:N \l__wa_options_Arrow_code_after_seq
1773     \__wa_error:n { Unknown~option~Arrow~in~code-after }
1774 }
```

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.

```
1775 \seq_new:N \l__wa_options_Arrow_code_after_seq
1776 \__wa_set_seq_of_str_from_clist:Nn \l__wa_options_Arrow_code_after_seq
1777 { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }
```

```
1778 <*LaTeX>
1779 \NewDocumentCommand \__wa_Arrow_code_after { 0 { } m m m ! 0 { } }
1780 </LaTeX>
1781 <*plain-TeX>
1782 \cs_new_protected:Npn \__wa_Arrow_code_after
1783 {
1784   \peek_meaning:NTF [
1785     { \__wa_Arrow_code_after_i }
1786     { \__wa_Arrow_code_after_i [ ] }
1787   }
1788   \cs_new_protected:Npn \__wa_Arrow_code_after_i [ #1 ] #2 #3 #4
1789   {
1790     \peek_meaning:NTF [
1791       { \__wa_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1792       { \__wa_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
1793     }
1794     \cs_new_protected:Npn \__wa_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
```

```

1795 </plain-TeX>
1796 {
1797   \int_set:Nn \l__wa_pos_arrow_int 1
1798   \str_clear_new:N \l__wa_previous_key_str
1799   \group_begin:
1800     \keys_set:nn { WithArrows / Arrow / code-after }
1801     { #1, #5, tikz = { xshift = \l__wa_xoffset_dim } }
1802     \bool_set_false:N \l__wa_initial_r_bool
1803     \bool_set_false:N \l__wa_final_r_bool
1804     \int_case:nn \l__wa_pos_arrow_int
1805       {
1806         0
1807         {
1808           \bool_set_true:N \l__wa_initial_r_bool
1809           \bool_set_true:N \l__wa_final_r_bool
1810         }
1811         2 { \bool_set_true:N \l__wa_initial_r_bool }
1812         3 { \bool_set_true:N \l__wa_final_r_bool }
1813       }

```

We prevent drawing an arrow from a line to itself.

```

1814   \tl_if_eq:nnTF { #2 } { #3 }
1815     { \__wa_error:nn { Both~lines~are~equal } { #2 } }

```

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

```

1816   {
1817     \cs_if_free:cTF { pgf@sh@ns@wa - \l__wa_prefix_str - #2 - 1 }
1818       { \__wa_error:nx { Wrong~line~in~Arrow } { #2 } }
1819     {
1820       \cs_if_free:cTF { pgf@sh@ns@wa - \l__wa_prefix_str - #3 - 1 }
1821         { \__wa_error:nx { Wrong~line~in~Arrow } { #3 } }
1822       {
1823         \int_compare:nNnTF \l__wa_pos_arrow_int = 4
1824           {
1825             <*LaTeX>
1826               \begin { tikzpicture } [ __wa_standard ]
1827             </LaTeX>
1828             <*plain-TeX>
1829               \tikzpicture [ __wa_standard ]
1830             </plain-TeX>
1831               \tikz@scan@one@point \pgfutil@firstofone (#2-1.south)
1832               \dim_set_eq:NN \l_tmpa_dim \pgf@x
1833               \dim_set_eq:NN \l_tmpb_dim \pgf@y
1834               \tikz@scan@one@point \pgfutil@firstofone (#3-1.north)
1835               \dim_set:Nn \l_tmpa_dim
1836                 { \dim_max:nn \l_tmpa_dim \pgf@x }
1837               \tl_gset:Nx \g_tmpa_tl
1838                 { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1839               \tl_gset:Nx \g_tmpb_tl
1840                 { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1841             <*LaTeX>
1842               \end { tikzpicture }
1843             </LaTeX>
1844             <*plain-TeX>
1845               \endtikzpicture
1846             </plain-TeX>
1847           }
1848         {
1849           <*LaTeX>
1850             \begin { tikzpicture } [ __wa_standard ]
1851           </LaTeX>
1852           <*plain-TeX>
1853             \tikzpicture [ __wa_standard ]

```

```

1854 </plain-TeX>
1855         \tikz@scan@one@point \pgfutil@firstofone
1856         ( #2-\bool_if:NTF\l__wa_initial_r_bool rl .south )
1857         \tl_gset:Nx \g_tmpa_tl
1858         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1859         \tikz@scan@one@point \pgfutil@firstofone
1860         ( #3-\bool_if:NTF\l__wa_final_r_bool rl .north )
1861         \tl_gset:Nx \g_tmpb_tl
1862         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1863 <*LaTeX>
1864         \end { tikzpicture }
1865 </LaTeX>
1866 <*plain-TeX>
1867         \endtikzpicture
1868 </plain-TeX>
1869     }
1870     \__wa_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
1871 }
1872 }
1873 }
1874 \group_end:
1875 }

```

11.12 The command `\MultiArrow` in code-after

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

```

1876 \cs_new_protected:Npn \__wa_MultiArrow:nn #1 #2
1877 {

```

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.

```

1878     \exp_args:Nnx
1879     \regex_match:nnTF
1880     { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
1881     { #1 }
1882     { \__wa_MultiArrow_i:nn { #1 } { #2 } }
1883     { \__wa_error:nx { Invalid-specification-for-MultiArrow } { #1 } }
1884 }
1885 \cs_new_protected:Npn \__wa_MultiArrow_i:nn #1 #2
1886 {

```

That's why we construct a “clist” of `expl3` from the specification of list given by the user. The construction of the “clist” must be global in order to exit the `\foreach` and that's why we will construct the list in `\g_tmpa_clist`.

```

1887     \foreach \x in { #1 }
1888     {
1889         \cs_if_free:cTF { pgf@sh@ns@wa - \l__wa_prefix_str - \x - 1 }
1890         { \__wa_error:nx { Wrong-line-specification-in-MultiArrow } \x }
1891         { \clist_gput_right:Nx \g_tmpa_clist \x }
1892     }

```

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

```

1893     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
1894     { \__wa_error:n { Too-small-specification-for-MultiArrow } }
1895     {
1896         \clist_sort:Nn \g_tmpa_clist
1897         {
1898             \int_compare:nTF { ##1 > ##2 }
1899             \sort_return_swapped:
1900             \sort_return_same:
1901         }

```

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

```
1902 \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...

```
1903 \clist_reverse:N \g_tmpa_clist
1904 \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).

```
1905 \exp_args:NV \_wa_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
1906 \*LaTeX
1907 \begin { tikzpicture }
1908 \*LaTeX
1909 \*plain-TeX
1910 \tikzpicture
1911 \*plain-TeX
1912 [
1913   __wa_standard ,
1914   every~path / .style = { WithArrows / arrow }
1915 ]
1916 \draw [ <-> ] ([xshift = \l__wa_xoffset_dim]\l_tmpa_tl-r.south)
1917   -- ++(5mm,0)
1918   -- node (__wa_label) {}
1919     ([xshift = \l__wa_xoffset_dim+5mm]\l_tmpb_tl-r.south)
1920   -- ([xshift = \l__wa_xoffset_dim]\l_tmpb_tl-r.south) ;
1921 \tikz@parse@node \pgfutil@firstofone (__wa_label.west)
1922 \dim_set:Nn \l_tmpa_dim { 20 cm }
1923 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
1924 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
1925 \bool_if:nT { \l__wa_wrap_lines_bool && \l__wa_in_DispWithArrows_bool }
1926 {
1927   \dim_set:Nn \l_tmpb_dim
1928     { \g__wa_right_x_dim - \pgf@x - 0.3333 em }
1929   \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1930     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1931 }
1932 \path (__wa_label.west)
1933   node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
1934 \*LaTeX
1935 \end{tikzpicture}
1936 \*LaTeX
1937 \*plain-TeX
1938 \endtikzpicture
1939 \*plain-TeX
1940 }
1941 }
1942 \cs_new_protected:Npn \_wa_MultiArrow_i:n #1
1943 {
1944 \*LaTeX
1945 \begin { tikzpicture }
1946 \*LaTeX
1947 \*plain-TeX
1948 \tikzpicture
1949 \*plain-TeX
1950 [
1951   __wa_standard ,
1952   every~path / .style = { WithArrows / arrow }
1953 ]
```

```

1954     \foreach \k in { #1 }
1955     {
1956         \draw [ <- ]
1957             ( [xshift = \l__wa_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
1958     } ;
1959 \*LaTeX
1960 \end{tikzpicture}
1961 \*plain-TeX
1962 \endtikzpicture
1963 \*plain-TeX
1964 }
1965

```

11.13 The error messages of the package

```

1966 \str_const:Nn \c__wa_option_ignored_str
1967 { If~you~go~on,~this~option~will~be~ignored. }
1968 \str_const:Nn \c__wa_command_ignored_str
1969 { If~you~go~on,~this~command~will~be~ignored. }
1970 \*LaTeX
1971 \__wa_msg_new:nn { amsmath~not~loaded }
1972 {
1973     You~can't~use~the~option~'\l_keys_key_tl'~because~the~
1974     package~'amsmath'~has~not~been~loaded.\\
1975     If~you~go~on,~this~option~will~be~ignored~in~the~rest~
1976     of~the~document.
1977 }
1978 \*LaTeX
1979 \__wa_msg_new:nn { Bad~value~for~replace~brace~by }
1980 {
1981     Bad~value~for~the~option~'\l_keys_key_tl'.~The~value~must~begin~
1982     with~an~extensible~left~delimiter.~The~possible~values~are:~,~,
1983     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
1984     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
1985     \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
1986     \token_to_str:N \lfloor~and~\token_to_str:N \lceil~
1987     (and~\token_to_str:N \lvert~and~\token_to_str:N \lVert~
1988     if~amsmath~or~unicode-math~is~loaded~in~LaTeX).\\
1989     \c__wa_option_ignored_str
1990 }
1991 \__wa_msg_new:nn { option~of~cr~negative }
1992 {
1993     The~argument~of~the~command~\token_to_str:N~~
1994     should~be~positive~in~the~row~\int_use:N \g__wa_line_int~
1995     of~your~environment~\{\l__wa_type_env_str\}.\\
1996     \c__wa_option_ignored_str
1997 }
1998 \__wa_msg_new:nn { omit~probably~used }
1999 {
2000     There~is~a~problem.~Maybe~you~have~used~a~command~
2001     \token_to_str:N\omit~in~the~line~\int_use:N \g__wa_line_int~
2002     (or~another~line)~of~your~environment~\{\l__wa_type_env_str\}.\\
2003     You~can~go~on~but~you~may~have~others~errors.
2004 }
2005 \*LaTeX
2006 \__wa_msg_new:nn { newline~at~the~end~of~env }
2007 {
2008     The~environments~of~witharrows~(\{WithArrows\})~and~
2009     \{DispWithArrows\}~should~not~end~by~\token_to_str:N \\\\.\\
2010     This~warning~might~become~an~error~in~a~future~version.

```

```

2011 }
2012 </LaTeX>
2013 \_wa_msg_new:nn { Invalid~option~format }
2014 {
2015     The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
2016     must~not~be~empty.\\
2017     \c_wa_option_ignored_str
2018 }
2019 \_wa_msg_new:nn { Value~for~a~key }
2020 {
2021     The~key~'\l_keys_key_tl'~should~be~used~without~value. \\
2022     However,~you~can~go~on~for~this~time.
2023 }
2024 \_wa_msg_new:nnn { Unknown~option~in~Arrow }
2025 {
2026     The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
2027     \l_wa_string_Arrow_for_msg_str\ in~the~row~
2028     \int_use:N \g_wa_line_int\ of~your~environment~
2029     \{\l_wa_type_env_str\}. \l_tmpa_str \\
2030     \c_wa_option_ignored_str \\
2031     For~a~list~of~the~available~keys,~type~H~<return>.
2032 }
2033 {
2034     The~available~keys~are~(in~alphabetic~order):~
2035     \seq_use:Nnnn \l_wa_options_Arrow_seq {~and~} {,~} {~and~}.
2036 }
2037 \_wa_msg_new:nnn { Unknown~option~WithArrows }
2038 {
2039     The~key~'\l_keys_key_tl'~is~unknown~in~\{\l_wa_type_env_str\}. \\
2040     \c_wa_option_ignored_str \\
2041     For~a~list~of~the~available~keys,~type~H~<return>.
2042 }
2043 {
2044     The~available~keys~are~(in~alphabetic~order):~
2045     \seq_use:Nnnn \l_wa_options_WithArrows_seq {~and~} {,~} {~and~}.
2046 }
2047 \_wa_msg_new:nnn { Unknown~option~DispWithArrows }
2048 {
2049     The~key~'\l_keys_key_tl'~is~unknown~in~\{\l_wa_type_env_str\}. \\
2050     \c_wa_option_ignored_str \\
2051     For~a~list~of~the~available~keys,~type~H~<return>.
2052 }
2053 {
2054     The~available~keys~are~(in~alphabetic~order):~
2055     \seq_use:Nnnn \l_wa_options_DispWithArrows_seq {~and~} {,~} {~and~}.
2056 }
2057 \_wa_msg_new:nnn { Unknown~option~WithArrowsOptions }
2058 {
2059     The~key~'\l_keys_key_tl'~is~unknown~in~
2060     \token_to_str:N \WithArrowsOptions. \\
2061     \c_wa_option_ignored_str \\
2062     For~a~list~of~the~available~keys,~type~H~<return>.
2063 }
2064 {
2065     The~available~keys~are~(in~alphabetic~order):~
2066     \seq_use:Nnnn \l_wa_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
2067 }
2068 \_wa_msg_new:nnn { Unknown~option~Arrow~in~code~after }
2069 {
2070     The~key~'\l_keys_key_tl'~is~unknown~in~
2071     \token_to_str:N \Arrow\ in~code~after. \\

```

```

2072 \c_wa_option_ignored_str \
2073 For~a~list~of~the~available~keys,~type~H~<return>.
2074 }
2075 {
2076 The~available~keys~are~(in~alphabetic~order):~
2077 \seq~use:Nnnn \l_wa_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2078 }
2079 \__wa_msg_new:nn { Too~much~columns~in~WithArrows }
2080 {
2081 Your~environment~\{\l_wa_type_env_str\}~has~\int~use:N
2082 \l_wa_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2083 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2084 If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2085 the~option~'more~columns'~at~a~global~level~or~for~an~environment. \
2086 However,~you~can~go~one~for~this~time.
2087 }
2088 \__wa_msg_new:nn { Too~much~columns~in~DispWithArrows }
2089 {
2090 Your~environment~\{\l_wa_type_env_str\}~has~\int~use:N
2091 \l_wa_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2092 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\
2093 at~the~end~of~row~\int~use:N \g_wa_line_int. \
2094 This~error~is~fatal.
2095 }
2096 \__wa_msg_new:nn { Negative~jump }
2097 {
2098 You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
2099 \l_wa_string_Arrow_for_msg_str\
2100 in~the~row~\int~use:N \g_wa_line_int\
2101 of~your~environment~\{\l_wa_type_env_str\}.~
2102 You~can~create~an~arrow~going~backwards~with~the~option~'<-'~of~Tikz. \
2103 \c_wa_option_ignored_str
2104 }
2105 \__wa_msg_new:nn { new~group~without~groups }
2106 {
2107 You~can't~use~the~option~'new~group'~for~the~command~
2108 \l_wa_string_Arrow_for_msg_str\
2109 because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
2110 'groups'~in~your~environment~\{\l_wa_type_env_str\}. \
2111 \c_wa_option_ignored_str
2112 }
2113 \__wa_msg_new:nn
2114 { Too~few~lines~for~an~arrow }
2115 {
2116 Line~\l_wa_input_line_str\
2117 :~an~arrow~specified~in~the~row~\int~use:N \l_wa_initial_int\
2118 of~your~environment~\{\l_wa_type_env_str\}~can't~be~drawn~
2119 because~it~arrives~after~the~last~row~of~the~environment. \
2120 If~you~go~on,~this~arrow~will~be~ignored.
2121 }
2122 \__wa_msg_new:nn { WithArrows~outside~math~mode }
2123 {
2124 The~environment~\{\l_wa_type_env_str\}~should~be~used~only~in~math~mode~
2125 like~the~environment~\{aligned\}~of~amsmath. \
2126 Nevertheless,~you~can~go~on.
2127 }
2128 \__wa_msg_new:nn { DispWithArrows~in~math~mode }
2129 {
2130 The~environment~\{\l_wa_type_env_str\}~should~be~used~only~outside~math~
2131 mode~like~the~environment~\{align\}~of~amsmath. \
2132 This~error~is~fatal.

```



```

2133     }
2134 \_wa_msg_new:nn { Incompatible~options~in~Arrow }
2135 {
2136     You~try~to~use~the~option~'\l_keys_key_tl'~but~
2137     this~option~is~incompatible~or~redundant~with~the~option~
2138     '\l_wa_previous_key_str'~set~in~the~same~command~
2139     \l_wa_string_Arrow_for_msg_str. \\
2140     \c_wa_option_ignored_str
2141 }
2142 \_wa_msg_new:nn { Incompatible~options }
2143 { You~try~to~use~the~option~'\l_keys_key_tl'~but~
2144     this~option~is~incompatible~or~redundant~with~the~option~
2145     '\l_wa_previous_key_str'~set~in~the~same~command~
2146     \bool_if:NT \l_wa_in_code_after_bool
2147     {
2148         \l_wa_string_Arrow_for_msg_str\
2149         in~the~code~after~of~your~environment~\{\l_wa_type_env_str\}
2150     }. \\
2151     \c_wa_option_ignored_str
2152 }
2153 \_wa_msg_new:nn { Arrow~not~in~last~column }
2154 {
2155     You~should~use~the~command~\l_wa_string_Arrow_for_msg_str\
2156     only~in~the~last~column~(column~\int_use:N\l_wa_nb_cols_int)~
2157     of~your~environment~\{\l_wa_type_env_str\}.\\
2158     However~you~can~go~on~for~this~time.
2159 }
2160 \_wa_msg_new:nn { Wrong~line~in~Arrow }
2161 {
2162     The~specification~of~line~'#1'~you~use~in~the~command~
2163     \l_wa_string_Arrow_for_msg_str\
2164     in~the~'code~after'~of~\{\l_wa_type_env_str\}~doesn't~exist. \\
2165     \c_wa_option_ignored_str
2166 }
2167 \_wa_msg_new:nn { Both~lines~are~equal }
2168 {
2169     In~the~'code~after'~of~\{\l_wa_type_env_str\}~you~try~to~
2170     draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\
2171     \c_wa_option_ignored_str
2172 }
2173 \_wa_msg_new:nn { Wrong~line~specification~in~MultiArrow }
2174 {
2175     The~specification~of~line~'#1'~doesn't~exist. \\
2176     If~you~go~on~,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
2177 }
2178 \_wa_msg_new:nn { Too~small~specification~for~MultiArrow }
2179 {
2180     The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
2181     is~too~small:~you~need~at~least~two~lines. \\
2182     \c_wa_command_ignored_str
2183 }
2184 \_wa_msg_new:nn { Not~allowed~in~DispWithArrows }
2185 {
2186     The~command~\token_to_str:N #1
2187     is~allowed~only~in~the~last~column~
2188     (column~\int_use:N\l_wa_nb_cols_int)~of~\{\l_wa_type_env_str\}. \\
2189     \c_wa_option_ignored_str
2190 }
2191 \_wa_msg_new:nn { Not~allowed~in~WithArrows }
2192 {

```

```

2193 The~command~\token_to_str:N #1 is~not~allowed~in~\{\l__wa_type_env_str\}~
2194 (it's~allowed~in~the~last~column~of~\{\DispWithArrows\}). \\\
2195 \c__wa_option_ignored_str
2196 }
2197 (*LaTeX)
2198 \__wa_msg_new:nn { tag*~without~amsmath }
2199 {
2200   We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2201   (or~mathtools). \\\
2202   If~you~go~on,~the~command~\token_to_str:N\tag\
2203   will~be~used~instead.
2204 }
2205 \__wa_msg_new:nn { Multiple~tags }
2206 {
2207   You~can't~use~twice~the~command~\token_to_str:N\tag\
2208   in~a~line~of~the~environment~\{\l__wa_type_env_str\}. \\\
2209   If~you~go~on,~the~tag~'#1'~will~be~used.
2210 }
2211 \__wa_msg_new:nn { Multiple~labels }
2212 {
2213   Normally,~we~can't~use~the~command~\token_to_str:N\label\
2214   twice~in~a~line~of~the~environment~\{\l__wa_type_env_str\}. \\\
2215   However,~you~can~go~on.~
2216   \bool_if:NT \c__wa_showlabels_loaded_bool
2217     { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
2218   If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
2219   'allow~multiple~labels'~at~the~global~or~environment~level.
2220 }
2221 \__wa_msg_new:nn { Multiple~labels~with~cleveref }
2222 {
2223   Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
2224   twice~in~a~line~of~the~environment~\{\l__wa_type_env_str\}. \\\
2225   If~you~go~on,~you~may~have~undefined~references.
2226 }
2227 (/LaTeX)
2228 \__wa_msg_new:nn { Inexistent~v-node }
2229 {
2230   There~is~a~problem.~Maybe~you~have~put~a~command~\token_to_str:N\cr\
2231   instead~of~a~command~\token_to_str:N\\~at~the~end~of~
2232   the~row~\l_tmpa_int\
2233   of~your~environment~\{\l__wa_type_env_str\}. \\\
2234   This~error~is~fatal.
2235 }

```

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`).

```

2236 \__wa_msg_new:nn { Option~xoffset~forbidden }
2237 {
2238   You~can't~use~the~option~'xoffset'~in~the~command~
2239   \l__wa_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g__wa_line_int\
2240   of~your~environment~\{\l__wa_type_env_str\}~
2241   because~you~are~using~the~option~
2242   ' \int_compare:nNnTF \l__wa_pos_arrow_int = 7
2243     { group }
2244     { groups } ' .~It's~possible~for~an~independent~arrow~or~if~there~is~
2245   only~one~arrow. \\\
2246   \c__wa_option_ignored_str
2247 }
2248 \__wa_msg_new:nnn { Duplicate~name }
2249 {

```

```

2250 The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2251 the~same~environment~name~twice.~You~can~go~on,~but,~
2252 maybe,~you~will~have~incorrect~results. \\
2253 For~a~list~of~the~names~already~used,~type~H~<return>. \\
2254 If~you~don't~want~to~see~this~message~again,~use~the~option~
2255 'allow-duplicate-names'.
2256 }
2257 {
2258 The~names~already~defined~in~this~document~are:~
2259 \seq_use:Nnnn \g__wa_names_seq { ,~ } { ,~ } { ~and~ }.
2260 }

```

11.14 The command `\WithArrowsNewStyle`

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

```

2261 (*LaTeX)
2262 \NewDocumentCommand \WithArrowsNewStyle { m m }
2263 (/LaTeX)
2264 (*plain-TeX)
2265 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2266 (/plain-TeX)
2267 {
2268   \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2269   { \__wa_error:nn { Key-already-defined } { #1 } }
2270   {
2271     \keys_define:nn { WithArrows / Global }
2272     {
2273       #1 .code:n =
2274       { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
2275     }
2276     \seq_put_right:Nx \l__wa_options-WithArrows_seq { \tl_to_str:n { #1 } }
2277     \seq_put_right:Nx \l__wa_options-DispWithArrows_seq
2278     { \tl_to_str:n { #1 } }
2279     \seq_put_right:Nx \l__wa_options-WithArrowsOptions_seq
2280     { \tl_to_str:N { #1 } }

```

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

```

2281   \group_begin:
2282   \msg_set:nnn { witharrows } { Unknown-option-WithArrowsOptions }
2283   {
2284     The~key~'\l_keys_key_tl'~can't~be~set~in~the~
2285     definition~of~a~style.~You~can~go~on~for~this~time~
2286     but~you~should~suppress~this~key.
2287   }
2288   \WithArrowsOptions { #2 }
2289   \group_end:
2290 }
2291 }
2292 \__wa_msg_new:nn { Key-already-defined }
2293 {
2294   The~key~'#1'~is~already~defined. \\
2295   If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2296   will~be~ignored.
2297 }

```

11.15 The options up and down

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

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

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

```

2298 \tl_const:Nn \c_wa_tikz_code_up_tl
2299 {
2300   \draw [ rounded-corners ]
2301     let \p1 = (#1) ,
2302         \p2 = (#2)
2303     in (\p1) -- node {
2304   <*LaTeX>
2305               \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2306               \begin { varwidth } \l_tmpa_dim

```

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

```

2307               \narrowragged
2308               #3
2309           \end { varwidth }
2310   </LaTeX>
2311   <*plain-TeX>
2312               #3
2313   </plain-TeX>
2314       }
2315   (\x2,\y1) -- (\p2) ;
2316 }

```

Idem for the option down.

```

2317 \tl_const:Nn \c_wa_tikz_code_down_tl
2318 {
2319   \draw [ rounded-corners ]
2320     let \p1 = (#1) ,
2321         \p2 = (#2)
2322     in (\p1) -- (\x1,\y2) --
2323         node {
2324   <*LaTeX>
2325               \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2326               \begin { varwidth } \l_tmpa_dim
2327                   \narrowragged
2328                   #3
2329               \end { varwidth }
2330   </LaTeX>
2331   <*plain-TeX>
2332               #3
2333   </plain-TeX>
2334       }
2335   (\p2) ;
2336 }

```

```

2337 \keys_define:nn { WithArrows / Arrow / FirstPass }
2338 {
2339   up .code:n = \__wa_set_independent: ,
2340   down .code:n = \__wa_set_independent: ,
2341   up .default:n = NoValue ,
2342   down .default:n = NoValue
2343 }

```

```

2344 \keys_define:nn { WithArrows / Arrow / SecondPass }
2345 {
2346   up .code:n =
2347     \str_if_empty:NT \l__wa_previous_key_str
2348     {
2349       \str_set:Nn \l__wa_previous_key_str { up }
2350   <*LaTeX>
2351     \bool_if:NTF \c_wa_varwidth_loaded_bool
2352     {

```

```

2353 </LaTeX>
2354         \cs_if_exist:cTF { tikz@library@calc@loaded }
2355         {
2356             \int_set:Nn \l__wa_pos_arrow_int \c_one_int

```

We have to set `\l__wa_wrap_lines_bool` to false because, otherwise, if the option `wrap_lines` is used at a higher level (global or environment), we will have a special affectation to `tikz-code` that will overwrite our affectation.

```

2357             \bool_set_false:N \l__wa_wrap_lines_bool
2358             \tl_set_eq:NN \l__wa_tikz_code_tl
2359             \c__wa_tikz_code_up_tl
2360         }
2361         { \__wa_error:n { calc-not-loaded } }
2362 <*LaTeX>
2363     }
2364     { \__wa_error:n { varwidth-not-loaded } }
2365 </LaTeX>
2366 } ,
2367 down .code:n =
2368     \str_if_empty:NT \l__wa_previous_key_str
2369     {
2370         \str_set:Nn \l__wa_previous_key_str { down }
2371 <*LaTeX>
2372         \bool_if:NTF \c__wa_varwidth_loaded_bool
2373         {
2374 </LaTeX>
2375             \cs_if_exist:cTF { tikz@library@calc@loaded }
2376             {
2377                 \int_set:Nn \l__wa_pos_arrow_int \c_one_int
2378                 \bool_set_false:N \l__wa_wrap_lines_bool
2379                 \tl_set_eq:NN \l__wa_tikz_code_tl
2380                 \c__wa_tikz_code_down_tl
2381             }
2382             { \__wa_error:n { calc-not-loaded } }
2383 <*LaTeX>
2384         }
2385         { \__wa_error:n { varwidth-not-loaded } }
2386 </LaTeX>
2387     }
2388 }
2389 \seq_put_right:Nn \l__wa_options_Arrow_seq { down }
2390 \seq_put_right:Nn \l__wa_options_Arrow_seq { up }
2391 \__wa_msg_new:nn { varwidth-not-loaded }
2392 {
2393     You~can't~use~the~option~'\l_keys_key_tl'~because~
2394     you~don't~have~loaded~the~package~'varwidth'. \\\
2395     \c__wa_option_ignored_str
2396 }
2397 \__wa_msg_new:nn { calc-not-loaded }
2398 {
2399     You~can't~use~the~option~'\l_keys_key_tl'~because~you~don't~have~loaded~the~
2400     Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
2401     ~in~the~preamble~of~your~document. \\\
2402     \c__wa_option_ignored_str
2403 }
2404 \__wa_msg_new:nn { Invalid~specification~for~MultiArrow }
2405 {
2406     The~specification~of~rows~for~'\token_to_str:N\MultiArrow\
2407     (i.e.~#1)~is~invalid. \\\
2408     \c__wa_command_ignored_str
2409 }

```

```

2410 <*plain-TeX>
2411 \catcode ` \@ = 12
2412 \ExplSyntaxOff
2413 </plain-TeX>

```

12 History

Changes between versions 1.0 and 1.1

Option for the command `\` and option `interline`
 Compatibility with `\usetikzlibrary{babel}`
 Possibility of nested environments `{WithArrows}`

Changes between versions 1.1 and 1.2

The package `witharrows` can now be loaded without having loaded previously `tikz` and the libraries `arrow.meta` and `bending` (this extension and these libraries are loaded silently by `witharrows`).
 New option `groups` (with a `s`)

Changes between versions 1.2 and 1.3

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

Changes between versions 1.3 and 1.4

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

Changes between versions 1.4 and 1.5

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

Changes between versions 1.5 and 1.6

The code has been improved to be faster and the Tikz library `calc` is no longer required.
 A new option `name` is available for the environments `{WithArrows}`.

Changes between 1.6 and 1.7

New environments `{DispWithArrows}` and `{DispWithArrows*}`.

Changes between 1.7 and 1.8

The numbers and tags of the environment `{DispWithArrows}` are now compatible with all the major LaTeX packages concerning references (`autonum`, `cleveref`, `fancyref`, `hyperref`, `prettyref`, `refstyle`, `typedref` and `varioref`) and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.

Changes between 1.8 and 1.9

New option `wrap-lines` for the environments `{DispWithArrows}` and `{DispWithArrows*}`.

Changes between 1.9 and 1.10

If the option `wrap-lines` is used, the option “`text width`” of Tikz is still active: if the value given to “`text width`” is lower than the width computed by `wrap-lines`, this value is used to wrap the lines.

The option `wrap-lines` is now fully compatible with the class option `leqno`.

Correction of a bug: `\nointerlineskip` and `\makebox[.6\linewidth]{}` should be inserted in `{DispWithArrows}` only in vertical mode.

Changes between 1.10 and 1.11

New commands `\WithArrowsNewStyle` and `\WithArrowsRightX`.

Changes between 1.11 and 1.12

New command `\tagnextline`.

New option `tagged-lines`.

An option of position (`ll`, `lr`, `rl`, `rr` or `i`) is now allowed at the local level even if the option `group` or the option `groups` is used at the global or environment level.

Compatibility of `{DispWithArrows}` with `\qedhere` of `amsthm`.

Compatibility with the packages `refcheck`, `showlabels` and `listbls`.

The option `\AllowLineWithoutAmpersand` is deprecated because lines without ampersands are now always allowed.

Changes between 1.12 and 1.13

Options `start-adjust`, `end-adjust` and `adjust`.

This version is not strictly compatible with previous ones. To restore the behaviour of the previous versions, one has to use the option `adjust` with the value 0 pt:

```
\WithArrowsOptions{adjust = 0pt}
```

Changes between 1.13 and 1.14

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

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments: [//www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)).

Modification of the code of `\WithArrowsNewStyle` following a correction of a bug in `l3keys` in the version of `l3kernel` of 2019/01/28.

New error message `Inexistent~v-node` to avoid a pgf error.

The error `Option incompatible with 'group(s)'` was suppressed in the version 1.12 but this was a mistake since this error is used with the option `xoffset` at the local level. The error is put back.

Changes between 1.14 and 1.15

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

Tikz externalization is now deactivated in the environments of the extension `witharrows`.⁴¹

⁴¹ 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 1.15 and 1.16

Option `no-arrows`

The behaviour of `{DispWithArrows}` after an `\item` of a LaTeX list has been changed : no vertical is added. The previous behaviour can be restored with the option `standard-behaviour-with-items`. A given name can no longer be used for two distinct environments. However, it's possible to deactivate this control with the option `allow-duplicate-names`.

Changes between 1.16 and 1.17

Option `format`.

Changes between 1.17 and 1.18

New option `<...>` for `{DispWithArrows}`.

Option `subequations`.

Warning when `{WithArrows}` or `{DispWithArrows}` ends by `\\`.

No space before an environment `{DispWithArrows}` if we are at the beginning of a `{minipage}`.

Changes between 1.18 and 2.0

A version of `witharrows` is available for plain-TeX.

Changes between 2.0 and 2.1

Option `max-length-of-arrow`.

Validation with regular expression for the first argument of `\MultiArrow`.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

| Symbols | | |
|--------------------------------|--|---|
| <code>\,</code> | | 1880 |
| <code>\.</code> | | 1880 |
| <code>\\</code> | | 63, 72, 740, 916, 1026, 1232, 1974, 1988, 1993, 1995, 2002, 2009, 2016, 2021, 2029, 2030, 2039, 2040, 2049, 2050, 2060, 2061, 2071, 2072, 2085, 2093, 2102, 2110, 2119, 2125, 2131, 2139, 2150, 2157, 2164, 2170, 2175, 2181, 2188, 2194, 2201, 2208, 2214, 2224, 2231, 2233, 2245, 2252, 2253, 2294, 2394, 2401, 2407 |
| <code>\{</code> | 399, 1983, 1995, 2002, 2008, 2009, 2029, 2039, 2049, 2081, 2090, 2101, 2110, 2118, 2124, 2125, 2130, 2131, 2149, 2157, 2164, 2169, 2188, 2193, 2194, 2208, 2214, 2224, 2233, 2240 | |
| <code>\}</code> | | 1995, 2002, 2008, 2009, 2029, 2039, 2049, 2081, 2090, 2101, 2110, 2118, 2124, 2125, 2130, 2131, 2149, 2157, 2164, 2169, 2188, 2193, 2194, 2208, 2214, 2224, 2233, 2240 |
| <code>\</code> | | 54, 1986, 1987, 1994, 2001, 2027, 2028, 2071, 2082, 2091, 2092, 2099, 2100, 2108, 2116, 2117, 2148, 2155, 2163, 2180, 2202, 2207, 2213, 2223, 2230, 2232, 2239, 2295, 2406 |
| A | | |
| <code>\A</code> | | 485, 1880 |
| \arabic | | 1014 |
| <code>\Arrow</code> | | 272, 2071 |
| Arrow internal commands: | | |
| <code>__wa_Arrow</code> | | 680, 683, 715, 790 |
| <code>\AtBeginDocument</code> | | 103, 225, 402 |
| B | | |
| <code>\begin</code> | | 767, 1135, 1280, 1609, 1684, 1735, 1826, 1850, 1907, 1945, 2306, 2326 |
| <code>\belowdisplayskip</code> | | 1310 |
| <code>\bgroup</code> | | 824, 828, 902, 1209 |
| bool commands: | | |
| <code>\bool_gset_true:N</code> | | 94 |
| <code>\bool_if:NTF</code> | | 75, 85, 227, 425, 757, 759, 767, 792, 795, 806, 815, 821, 829, 871, 907, 921, 988, 999, 1004, 1022, 1031, 1034, 1038, 1046, 1058, 1129, 1130, 1135, 1138, 1167, 1187, 1201, 1211, 1251, 1307, 1318, 1319, 1338, 1378, 1582, 1586, 1662, 1671, 1856, 1860, 2146, 2216, 2351, 2372 |
| <code>\bool_if:nTF</code> | | 404, 652, 1121, 1221, 1305, 1358, 1368, 1383, 1392, 1460, 1470, 1490, 1504, 1522, 1626, 1704, 1925 |
| <code>\bool_if_p:N</code> | | 1368 |
| <code>\bool_new:N</code> | | |
| | 25, 26, 96, 111, 245, 246, 247, 275, 278, 279, 280, 282, 283, 284, 285, 291, 1093 | |

`\bool_set:Nn` 1367
`\bool_set_false:N`
 .. 118, 751, 892, 1025, 1176, 1177, 1492,
 1563, 1564, 1594, 1598, 1802, 1803, 2357, 2378
`\bool_set_true:N` 97,
 114, 426, 891, 944, 1115, 1141, 1144, 1403,
 1405, 1434, 1488, 1567, 1568, 1571, 1572,
 1593, 1597, 1601, 1602, 1808, 1809, 1811, 1812
`\c_false_bool` 1221
`\l_tmpa_bool` 1593, 1594, 1601, 1662
`\l_tmpb_bool` 1597, 1598, 1602, 1671
box commands:
`\box_clear_new:N` 1161, 1208, 1240
`\box_dp:N` 1260
`\box_ht:N` 1259
`\box_set_to_last:N` 1235
`\box_use:N` 1237
`\box_use_drop:N` 1247, 1256, 1275
`\box_wd:N` 837, 1160, 1239, 1243, 1244
`\l_tmpa_box` 1150, 1160, 1235, 1237, 1239, 1241

C

`\catcode` 22, 2411
char commands:
`\char_generate:nn` 163, 179
clist commands:
`\clist_clear:N`
 421, 1348, 1350, 1363, 1389, 1396
`\clist_count:N` 1893
`\clist_gput_right:Nn` 1891
`\clist_if_in:NnTF` .. 265, 442, 454, 991, 1229
`\clist_map_inline:nn` 105
`\clist_new:N` 261, 396
`\clist_pop:NN` 1902, 1904
`\clist_put_left:Nn` 445
`\clist_put_right:Nn` 407
`\clist_remove_all:Nn` 444
`\clist_reverse:N` 1903
`\clist_set:Nn` 262,
 266, 397, 422, 441, 1230, 1357, 1388, 1395
`\clist_sort:Nn` 1896
`\g_tmpa_clist`
 1891, 1893, 1896, 1902, 1903, 1904, 1905
`\coordinate` 1041, 1049, 1062
`\cr` 910, 1070, 1222, 2230
cs commands:
`\cs_generate_variant:Nn`
 36, 101, 102, 1528, 1709
`\cs_gset:Npx` 994
`\cs_if_exist:NnTF`
 722, 1010, 1387, 1394, 2354, 2375
`\cs_if_free:NnTF` 1287, 1817, 1820, 1889
`\cs_new:Npn` 1751
`\cs_new_protected:Npn` 28,
 29, 30, 32, 33, 34, 35, 124, 128, 131, 146,
 155, 160, 176, 234, 263, 293, 302, 572,
 632, 683, 689, 695, 714, 716, 780, 842,
 881, 888, 913, 927, 960, 969, 972, 981, 984,
 1079, 1099, 1106, 1112, 1225, 1336, 1347,
 1349, 1372, 1400, 1405, 1406, 1407, 1416,
 1427, 1468, 1529, 1531, 1554, 1679, 1702,
 1730, 1782, 1788, 1794, 1876, 1885, 1942, 2265
`\cs_set:Npn` 929, 941, 1681
`\cs_set:Npx` 998

`\cs_set_eq:NN` 237, 383, 384,
 740, 761, 772, 773, 774, 775, 776, 777, 790,
 942, 943, 1033, 1035, 1165, 1406, 1410, 1411
`\cs_set_protected:Npn` 673
`\cs_to_str:N` 164, 180

D

`\d` 1880
`\DeclareOption` 97, 98
dim commands:
`\dim_compare:nNnTF`
 1067, 1242, 1293, 1720, 1723, 1929
`\dim_compare_p:nNn` 1627
`\dim_eval:n` 1638, 1649, 1663, 1672
`\dim_gset:Nn`
 1239, 1244, 1294, 1615, 1616, 1618, 1619, 1741
`\dim_gset_eq:NN` 1278
`\dim_gzero_new:N`
 1238, 1277, 1604, 1605, 1606, 1607
`\dim_max:nn` 1073, 1741, 1836
`\dim_set:Nn` 661, 666,
 667, 1073, 1160, 1257, 1292, 1501, 1577,
 1714, 1719, 1835, 1922, 1924, 1927, 2305, 2325
`\dim_set_eq:NN` 1153, 1184, 1202,
 1203, 1206, 1263, 1721, 1748, 1832, 1833, 1930
`\dim_use:N` 1636,
 1637, 1647, 1648, 1660, 1661, 1664, 1669,
 1670, 1673, 1726, 1838, 1840, 1858, 1862, 1933
`\dim_zero:N` 741
`\dim_zero_new:N` 726, 1159, 1183, 1199
`\c_max_dim` 1278, 1501, 1577
`\g_tmpa_dim` 1741, 1748
`\l_tmpa_dim` 1073, 1074, 1257, 1266,
 1292, 1293, 1294, 1714, 1720, 1721, 1723,
 1726, 1832, 1835, 1836, 1838, 1840, 1922,
 1924, 1929, 1930, 1933, 2305, 2306, 2325, 2326
`\l_tmpb_dim`
 1719, 1720, 1721, 1833, 1838, 1927, 1929, 1930
`\c_zero_dim`
 . 191, 192, 970, 1067, 1073, 1153, 1263, 1723
`\displaystyle` 806, 907, 1167
`\displaywidth` 1184, 1203, 1206
`\DispWithArrows` 1099, 1332
DispWithArrows commands:
`\DispWithArrows_i` 1103, 1104, 1106
`\DispWithArrows_ii` 1109, 1110, 1112
`\draw` 324, 643, 1712, 1916, 1956, 2300, 2319

E

`\egroup` 917, 918, 1233, 1245
else commands:
`\else:` 898
`\end` .. 921, 974, 1089, 1298, 1318, 1319, 1621,
 1695, 1743, 1842, 1864, 1935, 1960, 2309, 2329
`\endDispWithArrows` 1225, 1334
`\endtikzpicture`
 1301, 1624, 1698, 1746, 1845, 1867, 1938, 1963
`\endWithArrows` 913
exp commands:
`\exp_args:NNo` 1561
`\exp_args:Nnx` 1878
`\exp_args:No` 1132, 1561
`\exp_args:NV` 1081, 1706, 1905
`\ExplSyntaxOff` 2412

| | | | |
|--------------------------|---|------------------------|---|
| \ExplSyntaxOn | 21 | \int_use:N | 709, 711, 802, 849, 853, 863, 867, 874, 941, 1041, 1049, 1062, 1439, 1443, 1447, 1450, 1542, 1546, 1558, 1582, 1586, 1588, 1592, 1596, 1751, 1994, 2001, 2028, 2081, 2090, 2093, 2100, 2117, 2156, 2188, 2239 |
| F | | | |
| \fi | 1142, 1145 | \int_zero_new:N | 742, 743, 744, 745, 746, 1429, 1430, 1431, 1534, 1536 |
| fi commands: | | \c_one_int | 445, 747, 1435, 1472, 2356, 2377 |
| \fi: | 900, 1181, 1194 | \l_tmpa_int | 702, 703, 1285, 1288, 1291, 2232 |
| \foreach | 1887, 1954 | \c_zero_int | 1482 |
| G | | | |
| \globaldefs | 430, 1085 | \itshape | 217 |
| group commands: | | J | |
| \group_align_safe_begin: | 966 | \jot | 236, 370, 1024 |
| \group_align_safe_end: | 987 | K | |
| \group_begin: | | \k | 1954, 1957 |
| | 429, 883, 939, 1084, 1101, 1152, 1164, 1262, 1409, 1418, 1533, 1556, 1799, 2281 | keys commands: | |
| \group_end: | 432, 924, 946, 1087, 1157, 1171, 1269, 1322, 1413, 1425, 1552, 1677, 1874, 2289 | \keys_define:nn | 38, 304, 387, 411, 466, 491, 526, 532, 551, 583, 640, 1752, 2271, 2337, 2344 |
| H | | | |
| \halign | 827 | \keys_if_exist:nnTF | 2268 |
| hbox commands: | | \l_keys_key_tl | 47, 54, 297, 576, 618, 636, 1973, 1981, 2021, 2026, 2039, 2049, 2059, 2070, 2136, 2143, 2284, 2393, 2399 |
| \hbox_overlap_left:n | 1032, 1036, 1059 | \keys_set:nn | 677, 700, 758, 760, 1528, 1800 |
| \hbox_overlap_right:n | 873 | \keys_set_known:nn | 1530, 2274 |
| \hbox_set:Nn | 1150, 1162, 1241 | \l_keys_value_tl | 578, 2250 |
| \hbox_to_wd:nn | 1193, 1249, 1254 | L | |
| \hbox_unpack_clear:N | 1241 | \label | 775, 776, 1165, 1374, 2213, 2223 |
| \hfil | 799, 813, 814, 857, 1253, 1270, 1272 | \langle | 399, 1985 |
| \hfill | 800 | \lbrace | 399, 460, 1983 |
| I | | | |
| \ialign | 823 | \lbrack | 399, 1984 |
| if commands: | | \lceil | 399, 1986 |
| \if_mode_math: | 898, 1179 | \left | 1155, 1265 |
| \if_mode_vertical: | 1191 | \lfloor | 399, 1986 |
| \ignorespacesafterend | 1325 | \lgroup | 399, 1984 |
| \input | 7, 8 | \linewidth | 1183, 1184, 1193, 1202 |
| int commands: | | \lmoustache | 399, 1985 |
| \int_case:nn | 901, 1565, 1804 | lua commands: | |
| \int_compare:nNnTF | | \lua_now:n | 125 |
| | 135, 788, 810, 930, 932, 934, 951, 963, 1341, 1419, 1452, 1454, 1482, 1510, 1517, 1575, 1599, 1635, 1646, 1659, 1668, 1823, 2242 | \lVert | 407, 1987 |
| \int_compare:nTF | | \lvert | 407, 1987 |
| | 586, 602, 1500, 1507, 1524, 1549, 1893, 1898 | M | |
| \int_compare_p:n | 1462, 1474, 1476 | math commands: | |
| \int_compare_p:nNn | 654, 656, 1464, 1472, 1630 | \c_math_toggle_token | |
| \int_eval:n | 950 | | 803, 809, 906, 909, 1154, 1156, 1166, 1170, 1188, 1195, 1264, 1268, 1309, 1312, 1315 |
| \int_gdecr:N | 786 | \mathsurround | 741 |
| \int_gincr:N | 698, 801, 832, 952, 993 | MH commands: | |
| \int_gset:Nn | 802, 954, 956, 958 | \MH_if_boolean:nTF | |
| \int_gset_eq:NN | 763 | | 1123, 1306, 1360, 1362, 1385 |
| \int_gzero:N | 729, 731, 733, 833 | \MH_set_boolean_T:n | 1126 |
| \int_gzero_new:N | 734 | msg commands: | |
| \int_incr:N | 1458, 1550 | \msg_error:nn | 32 |
| \int_new:N | 250, 251, 252, 255, 257, 259, 288 | \msg_error:nnn | 35 |
| \int_set:Nn | 253, 303, 339, 350, 389, 391, 393, 587, 637, 702, 747, 762, 935, 1422, 1435, 1441, 1445, 1535, 1537, 1538, 1544, 1548, 1797, 2356, 2377 | \msg_fatal:nn | 34 |
| \int_set_eq:NN | 1493, 1494, 1495, 1512 | \msg_info:nn | 78 |
| \int_step_inline:nnn | 1732 | \msg_line_number: | 707 |
| \int_step_variable:nNn | 1285 | \msg_new:nnn | 28 |
| \int_until_do:nNnn | 1436, 1539 | \msg_new:nnnn | 29 |
| | | \msg_redirect_name:nnn | 31 |
| | | \msg_set:nnn | 2282 |

| | | |
|--|---------------------------------------|--|
| sys commands: | | |
| \sys_if_engine luatex:TF | 122 | |
| T | | |
| \tabskip | 815, 1210, 1215, 1218 | |
| \tag | 774, 1353, 2200, 2202, 2207 | |
| tag internal commands: | | |
| _wa_tag | 774, 1351 | |
| \tagnextline | 777, 1402 | |
| TeX and L ^A T _E X 2 _ε commands: | | |
| \@ | 22, 2411 | |
| \@currentlabel | 998 | |
| \@currenvir | 720 | |
| \@eqnnum | 1044 | |
| \@ifclassloaded | 77, 87 | |
| \@ifpackageloaded | 80, 90, 113 | |
| \@cequation | 993 | |
| \cref@constructprefix | 1006 | |
| \cref@currentlabel | 1007 | |
| \cref@equation@alias | 1010, 1011 | |
| \cref@result | 1006, 1014 | |
| \hyper@refstepcounter | 1002 | |
| \if@inlabel | 1140 | |
| \if@minipage | 1143 | |
| \intertext@ | 1130 | |
| \p@equation | 998, 1015 | |
| \pgf@x | 1292, 1615, | |
| 1618, 1715, 1741, 1832, 1836, 1858, 1862, 1928 | | |
| \pgf@y | 1616, 1619, 1833, 1840, 1858, 1862 | |
| \pgfutil@firstofone | 1291, 1614, | |
| 1617, 1713, 1740, 1831, 1834, 1855, 1859, 1921 | | |
| \protected@edef | 1007 | |
| \qed@elt | 1411 | |
| \QED@stack | 1412 | |
| \setQED@elt | 1411 | |
| \spread@equation | 234, 237, 770 | |
| \sr@name | 1129 | |
| \tagform@ | 1035 | |
| \This@name | 1001 | |
| \tikz@library@external@loaded | 722 | |
| \tikz@parse@node | 1291, 1713, 1921 | |
| \tikz@scan@one@point | | |
| 1614, 1617, 1740, 1831, 1834, 1855, 1859 | | |
| \tikz@text@width | 1716, 1923 | |
| tex commands: | | |
| \tex_strcmp:D | 129 | |
| \theequation | 995, 1033 | |
| \tikz | 844, 858, 1040, 1048, 1061 | |
| \tikzpicture | | |
| 1283, 1612, 1687, 1738, 1829, 1853, 1910, 1948 | | |
| \tikzset | 186, | |
| 196, 205, 210, 330, 354, 644, 723, 940, 1755 | | |
| tl commands: | | |
| \c_empty_tl | 355, 645 | |
| \c_novalue_tl | 244, 834, 1104, 1148, 1246 | |
| \tl_clear_new:N | 754, 755, 1175 | |
| \tl_const:Nn | 1710, 2298, 2317 | |
| \tl_gset:Nn | 1633, 1644, | |
| 1657, 1666, 1716, 1837, 1839, 1857, 1861, 1923 | | |
| \tl_head:n | 453 | |
| \tl_if_empty:NTF | 993, 995, 1355, 1717, 1924 | |
| \tl_if_empty:nTF | 482 | |
| \tl_if_eq:NNTF | 834, 1148, 1246 | |
| \tl_if_eq:nnTF | 1814 | |
| \tl_if_novalue:nTF | 1132 | |
| \tl_new:N | 243, 286, 287 | |
| \tl_put_right:Nn | 101, 475, 478 | |
| \tl_set:Nn | 453, 457, 486, 1132, 1366, 1580, 1585 | |
| \tl_set_eq:NN | 244, 1705, 2358, 2379 | |
| \tl_to_str:N | 2280 | |
| \tl_to_str:n | 151, 2276, 2278 | |
| \g_tmpa_tl | | |
| 994, 998, 1015, 1033, 1633, 1657, 1676, | | |
| 1716, 1717, 1719, 1837, 1857, 1870, 1923, 1924 | | |
| \l_tmpa_tl | 165, 181, 453, 456, | |
| 737, 947, 948, 950, 953, 954, 955, 956, 957, | | |
| 958, 1440, 1441, 1444, 1445, 1543, 1544, | | |
| 1547, 1548, 1559, 1562, 1590, 1676, 1902, 1916 | | |
| \g_tmpb_tl | 1644, 1666, 1676, 1839, 1861, 1870 | |
| \l_tmpb_tl | 1904, 1919, 1920 | |
| token commands: | | |
| \token_to_str:N | 49, 272, 1983, 1984, | |
| 1985, 1986, 1987, 1993, 2001, 2009, 2060, | | |
| 2071, 2176, 2180, 2186, 2193, 2200, 2202, | | |
| 2207, 2213, 2223, 2230, 2231, 2295, 2400, 2406 | | |
| U | | |
| \unpenalty | 1234 | |
| \unskip | 1234 | |
| use commands: | | |
| \use:N | 167, 170, 173, 183, 405 | |
| \use:n | 784 | |
| \use_none:nn | 383 | |
| \use_none:nnn | 384 | |
| \usepackage | 82, 92 | |
| \usetikzlibrary | 10, 2400 | |
| V | | |
| \vbox | 901 | |
| \vcenter | 901, 1155, 1266, 1275 | |
| \vfil | 1266 | |
| \vtop | 901, 1209 | |
| W | | |
| wa internal commands: | | |
| \g_wa_alignment_dim | | |
| 1238, 1239, 1243, 1244, 1254 | | |
| \c_wa_amsmath_loaded_bool | | |
| 227, 405, 425, 1130, 1368 | | |
| \c_wa_amsthm_loaded_bool | 795 | |
| _wa_analyze_end:Nn | 977, 1079 | |
| _wa_Arrow_code_after | 943, 1779, 1782 | |
| _wa_Arrow_code_after_i | 1785, 1786, 1788 | |
| _wa_Arrow_code_after_ii | 1791, 1792, 1794 | |
| _wa_Arrow_first_columns: | 714, 761 | |
| _wa_Arrow_i | 686, 687, 689 | |
| _wa_Arrow_ii | 692, 693, 695 | |
| \g_wa_arrow_int | 255, 654, 698, | |
| 709, 711, 728, 729, 930, 932, 954, 1436, 1466 | | |
| \l_wa_arrow_int | 744, 1435, 1436, | |
| 1439, 1443, 1447, 1450, 1458, 1472, 1486, | | |
| 1493, 1497, 1499, 1509, 1514, 1518, 1538, | | |
| 1539, 1542, 1546, 1550, 1558, 1588, 1592, 1596 | | |
| \g_wa_arrow_int_seq | 254, 728, 953 | |
| \c_wa_autonum_loaded_bool | 1392 | |
| \c_wa_cleveref_loaded_bool | 1004, 1378 | |
| \l_wa_code_after_tl | 478, 755, 945 | |
| \l_wa_code_before_tl | 475, 754, 769 | |
| _wa_code_for_possible_arrow: | 1457, 1468 | |

| | |
|--|--|
| <code>\g_wa_col_int</code> | 259, 732, 733, 763, |
| | 786, 788, 801, 802, 810, 833, 958, 963, 1341 |
| <code>\g_wa_col_int_seq</code> | 258, 732, 957 |
| <code>\c_wa_command_ignored_str</code> | 1968, 2182, 2408 |
| <code>\l_wa_command_name_str</code> | |
| | 269, 270, 319, 761, 790, 943 |
| <code>_wa_construct_halign:</code> | 780, 787, 903, 1214 |
| <code>_wa_construct_nodes:</code> | 811, 842 |
| <code>_wa_convert_to_str_seq:N</code> .. | 146, 158, 525 |
| <code>_wa_cr:</code> | 740, 960 |
| <code>_wa_cr_i:</code> | 967, 969 |
| <code>_wa_cr_ii:</code> | 970, 972, 984 |
| <code>_wa_cr_iii:n</code> | 976, 979, 981 |
| <code>_wa_def_function_tmpa:n</code> | 1679, 1706 |
| <code>\l_wa_delim_wd_dim</code> | 837, 1159, 1160 |
| <code>\l_wa_displaystyle_bool</code> | 327, 806, 907, 1167 |
| <code>_wa_draw_arrow:nnn</code> | |
| | 384, 1676, 1702, 1709, 1870 |
| <code>_wa_draw_arrows:nn</code> .. | 383, 1466, 1484, 1531 |
| <code>_wa_draw_arrows_i:</code> | 1549, 1554 |
| <code>\l_wa_end_adjust_dim</code> .. | 377, 664, 667, 1672 |
| <code>_wa_error:n</code> | 32, |
| | 81, 91, 300, 341, 352, 428, 458, 471, 483, |
| | 487, 530, 537, 558, 579, 581, 588, 604, 624, |
| | 660, 715, 899, 905, 964, 1068, 1369, 1379, |
| | 1380, 1455, 1773, 1894, 2361, 2364, 2382, 2385 |
| <code>_wa_error:nn</code> | 35, 36, 1339, |
| | 1342, 1356, 1815, 1818, 1821, 1883, 1890, 2269 |
| <code>_wa_eval_if_allowed:n</code> | 293, 303 |
| <code>\c_wa_extensible_delimiters_clist</code> .. | |
| | 396, 397, 407, 455 |
| <code>_wa_fatal:n</code> | 34, 43, 1180, 1220, 1289 |
| <code>\l_wa_final_int</code> | |
| | 743, 1445, 1452, 1495, 1510, |
| | 1512, 1517, 1525, 1548, 1549, 1578, 1586, 1630 |
| <code>\l_wa_final_r_bool</code> | 285, |
| | 1564, 1567, 1572, 1586, 1803, 1809, 1812, 1860 |
| <code>\l_wa_final_tl</code> | 287, 1585, 1617 |
| <code>\l_wa_first_arrow_int</code> ... | 1534, 1535, 1538 |
| <code>\l_wa_first_arrow_of_group_int</code> | |
| | 1429, 1464, 1466, 1482, 1485, 1493 |
| <code>\l_wa_first_arrows_seq</code> | |
| | 1432, 1496, 1497, 1509, 1591 |
| <code>\l_wa_first_line_of_group_int</code> | |
| | 1430, 1494, 1508 |
| <code>_wa_fix_pos_arrow:n</code> | |
| | 632, 646, 647, 648, 649, 650 |
| <code>_wa_fix_pos_option:n</code> ... | 302, 358, 360, |
| | 362, 364, 366, 1758, 1760, 1762, 1764, 1766 |
| <code>\l_wa_fleqn_bool</code> | 413, 829, 1211, 1251 |
| <code>\g_wa_footnote_bool</code> | |
| | 26, 40, 75, 94, 767, 921, 1319 |
| <code>\g_wa_footnotehyper_bool</code> | 25, 41, 85 |
| <code>\l_wa_format_seq</code> | 764, 765, 782 |
| <code>\l_wa_format_str</code> ... | 289, 486, 748, 762, 765 |
| <code>\l_wa_halign_box</code> | |
| | 1208, 1209, 1247, 1259, 1260, 1275 |
| <code>\c_wa_hyperref_loaded_bool</code> | 999 |
| <code>_wa_if_in_last_col_of_disp:Nn</code> | |
| | 1336, 1348, 1350, 1353, 1374, 1402 |
| <code>\l_wa_in_code_after_bool</code> .. | 247, 944, 2146 |
| <code>\l_wa_in_DispWithArrows_bool</code> | 246, |
| | 759, 792, 815, 892, 988, 1115, 1704, 1925 |
| <code>\l_wa_in_first_columns_bool</code> | 282 |
| <code>\l_wa_in_label_or_minipage_bool</code> | |
| | 1093, 1141, 1144, 1187, 1201, 1307 |
| <code>\l_wa_in_WithArrows_bool</code> | |
| | 245, 757, 821, 891, 1338 |
| <code>_wa_info:n</code> | 88 |
| <code>\l_wa_initial_int</code> | 742, 1441, 1474, |
| | 1494, 1508, 1525, 1544, 1578, 1582, 1630, 2117 |
| <code>\l_wa_initial_r_bool</code> | 284, |
| | 1563, 1568, 1571, 1582, 1802, 1808, 1811, 1856 |
| <code>\l_wa_initial_tl</code> | 286, 1580, 1614 |
| <code>\l_wa_input_line_str</code> | 727, 1451, 2116 |
| <code>\l_wa_interline_skip</code> | 372, 753, 1074 |
| <code>\l_wa_jump_int</code> | 587, 702, 746, 747 |
| <code>_wa_keys_set:</code> | 1529, 1562 |
| <code>_wa_label:n</code> | 776, 1372 |
| <code>\l_wa_labels_seq</code> | 750, 996, 1018, 1376, 1382 |
| <code>\l_wa_last_arrow_int</code> | 1536, 1537, 1539 |
| <code>\l_wa_last_arrows_seq</code> | |
| | 1433, 1498, 1499, 1513, 1514, 1518, 1595 |
| <code>\g_wa_last_env_int</code> | 250, 952, 1751 |
| <code>\l_wa_last_line_of_group_int</code> | |
| | 1431, 1474, 1495, 1510, 1512, 1517 |
| <code>\l_wa_left_brace_box</code> . | 837, 1161, 1162, 1256 |
| <code>\l_wa_left_brace_tl</code> | |
| | 243, 244, 534, 834, 1132, 1148, 1168, 1246 |
| <code>\c_wa_leqno_bool</code> | 96, 97, 1038, 1046 |
| <code>\g_wa_line_int</code> | 257, |
| | 265, 701, 702, 730, 731, 832, 849, 853, 863, |
| | 867, 874, 941, 956, 1041, 1049, 1062, 1285, |
| | 1452, 1549, 1994, 2001, 2028, 2093, 2100, 2239 |
| <code>\g_wa_line_int_seq</code> | 256, 730, 955 |
| <code>\l_wa_linewidth_dim</code> | |
| | 827, 1199, 1202, 1203, 1206, 1249, 1274 |
| <code>\l_wa_mathindent_dim</code> | 415, 830, 1252 |
| <code>\c_wa_mathtools_loaded_bool</code> | |
| | 1121, 1305, 1358, 1383 |
| <code>\l_wa_max_length_of_arrow_dim</code> | |
| | 306, 1628, 1641, 1652 |
| <code>_wa_msg_new:nn</code> | 28, |
| | 45, 52, 57, 66, 1971, 1979, 1991, 1998, |
| | 2006, 2013, 2019, 2079, 2088, 2096, 2105, |
| | 2113, 2122, 2128, 2134, 2142, 2153, 2160, |
| | 2167, 2173, 2178, 2184, 2191, 2198, 2205, |
| | 2211, 2221, 2228, 2236, 2292, 2391, 2397, 2404 |
| <code>_wa_msg_new:nnn</code> | |
| | 29, 2024, 2037, 2047, 2057, 2068, 2248 |
| <code>_wa_msg_redirect_name:nn</code> | |
| | 30, 316, 431, 438, 554, 1086 |
| <code>_wa_MultiArrow:nn</code> | 942, 1876 |
| <code>_wa_MultiArrow_i:n</code> | 1905, 1942 |
| <code>_wa_MultiArrow_i:nn</code> | 1882, 1885 |
| <code>\l_wa_name_str</code> . | 473, 724, 852, 853, 866, 867 |
| <code>\g_wa_names_seq</code> | 273, 470, 472, 2259 |
| <code>\l_wa_nb_cols_int</code> | 288, 762, 763, |
| | 788, 810, 965, 1341, 2082, 2091, 2156, 2188 |
| <code>\l_wa_new_box</code> | 1240, 1241, 1243, 1244 |
| <code>\l_wa_new_group_bool</code> | |
| | 283, 1434, 1488, 1490, 1492 |
| <code>_wa_nonumber:</code> | 773, 1349 |
| <code>_wa_notag:</code> | 772, 1347 |
| <code>_wa_old_label</code> | 775, 1018, 1165 |

| | |
|--|--|
| <code>\c__wa_option_ignored_str</code> | <code>\l__wa_string_Arrow_for_msg_str</code> |
| 1966, 1989, 1996, 2017, 2030, 2040, 2050, 2061, 2072, 2103, 2111, 2140, 2151, 2165, 2171, 2189, 2195, 2246, 2395, 2402 | 271, 272, 320, 2027, 2099, 2108, 2139, 2148, 2155, 2163, 2239 |
| <code>\l__wa_options_Arrow_code_after_seq</code> .. | <code>\l__wa_subequations_bool</code> 291, 426, 1135, 1318 |
| 1772, 1775, 1776, 2077 | <code>\l__wa_tag_next_line_bool</code> |
| <code>\l__wa_options_Arrow_seq</code> | 279, 751, 1022, 1025, 1403 |
| 338, 347, 348, 349, 617, 626, 627, 2035, 2389, 2390 | <code>\l__wa_tag_star_bool</code> |
| <code>\l__wa_options_DispWithArrows_seq</code> ... | 278, 1020, 1029, 1034, 1177, 1367 |
| 230, 536, 539, 540, 2055, 2277 | <code>\l__wa_tag_tl</code> 993, 995, 1175, 1355, 1366 |
| <code>\l__wa_options_WithArrows_seq</code> | <code>__wa_tagnextline:</code> |
| 512, 513, 525, 529, 618, 2045, 2276 | 777, 1400 |
| <code>\l__wa_options_WithArrowsOptions_seq</code> . | <code>\l__wa_tags_clist</code> 261, 262, 265, 266, 421, 422, 441, 442, 444, 445, 991, 1229, 1230, 1348, 1350, 1357, 1363, 1388, 1389, 1395, 1396 |
| 229, 557, 560, 561, 2066, 2279 | <code>__wa_test_if_to_tag:</code> |
| <code>\l__wa_pos_arrow_int</code> | 263, 794 |
| .. 252, 253, 303, 339, 350, 602, 637, 656, 934, 935, 1419, 1422, 1454, 1462, 1476, 1500, 1524, 1565, 1575, 1599, 1635, 1646, 1659, 1668, 1797, 1804, 1823, 2242, 2356, 2377 | <code>\c__wa_tikz_code_down_tl</code> |
| <code>\l__wa_pos_env_int</code> .. 251, 389, 391, 393, 901 | 2317, 2380 |
| <code>\l__wa_pos_of_arrow_int</code> | <code>\l__wa_tikz_code_tl</code> |
| 745 | 323, 642, 1705, 1706, 1767, 2358, 2379 |
| <code>\g__wa_position_in_the_tree_seq</code> | <code>\c__wa_tikz_code_up_tl</code> |
| 248, 249, 735, 736, 947, 948, 949, 951 | 2298, 2359 |
| <code>__wa_post_halign:</code> | <code>\c__wa_tikz_code_wrap_lines_tl</code> . 1705, 1710 |
| 919, 927, 1303 | <code>__wa_tmpa:nnn</code> |
| <code>__wa_pre_halign:n</code> | 1681, 1707 |
| 716, 897, 1133 | <code>\l__wa_type_col_str</code> .. 782, 799, 800, 813, 814 |
| <code>\l__wa_prefix_str</code> ... 202, 709, 711, 738, 739, 849, 863, 874, 1288, 1439, 1443, 1447, 1450, 1542, 1546, 1558, 1588, 1817, 1820, 1889 | <code>\l__wa_type_env_str</code> |
| <code>\l__wa_previous_key_str</code> | 719, 720, 894, 895, 1081, 1117, 1118, 1995, 2002, 2029, 2039, 2049, 2081, 2090, 2101, 2110, 2118, 2124, 2130, 2149, 2157, 2164, 2169, 2188, 2193, 2208, 2214, 2224, 2233, 2240 |
| 295, 297, 335, 337, 344, 346, 574, 576, 634, 636, 676, 699, 756, 1560, 1798, 2138, 2145, 2347, 2349, 2368, 2370 | <code>\c__wa_typedref_loaded_bool</code> |
| <code>__wa_qedhere:</code> | 1129 |
| 1405, 1406 | <code>__wa_update_x:nn</code> |
| <code>\l__wa_qedhere_bool</code> | 1525, 1578, 1730 |
| 280, 1021, 1030, 1031, 1053, 1057, 1058, 1176, 1405 | <code>\c__wa_varwidth_loaded_bool</code> |
| <code>__wa_qedhere_i:</code> | 2351, 2372 |
| 1032, 1059, 1407 | <code>__wa_warning:n</code> |
| <code>\l__wa_replace_left_brace_by_tl</code> | 33, 1083 |
| 457, 1155, 1265 | <code>\l__wa_wrap_lines_bool</code> |
| <code>__wa_restore:N</code> | 449, 1704, 1925, 2357, 2378 |
| 176, 1029, 1030, 1057 | <code>\l__wa_x_dim</code> |
| <code>\g__wa_right_x_dim</code> | 726, 1501, 1577, 1637, 1648, 1661, 1670, 1741, 1748 |
| 929, 1277, 1278, 1293, 1294, 1715, 1928 | <code>\g__wa_x_final_dim</code> .. 1605, 1618, 1647, 1669 |
| <code>__wa_save:N</code> | <code>\g__wa_x_initial_dim</code> . 1604, 1615, 1636, 1660 |
| 160, 1020, 1021, 1053 | <code>\l__wa_xoffset_dim</code> |
| <code>\l__wa_sbwi_bool</code> | 367, 661, 1562, 1769, 1801, 1916, 1919, 1920, 1957 |
| 275, 462, 1138 | <code>\g__wa_y_final_dim</code> |
| <code>__wa_scan_arrows:</code> | ... 1607, 1619, 1627, 1640, 1651, 1672, 1673 |
| 937, 1416 | <code>\g__wa_y_initial_dim</code> |
| <code>__wa_scan_arrows_i:</code> | ... 1606, 1616, 1627, 1640, 1651, 1663, 1664 |
| 1421, 1424, 1427 | <code>\l__wa_ygap_dim</code> |
| <code>__wa_set_independent:</code> | 193, 309 |
| 572, 590, 591, 592, 593, 594, 2339, 2340 | <code>\l__wa_ystart_dim</code> |
| <code>__wa_set_qedhere:</code> | 190, 312 |
| 795, 1406 | <code>\WithArrows</code> |
| <code>__wa_set_seq_of_str_from_clist:Nn</code> .. | 881 |
| 155, 513, 540, 561, 627, 1776 | <code>\WithArrows commands:</code> |
| <code>\l__wa_show_node_names_bool</code> | <code>\WithArrows_i</code> |
| 332, 871 | 885, 886, 888 |
| <code>\c__wa_showlabels_loaded_bool</code> | <code>\WithArrowsLastEnv</code> |
| 2216 | 1751 |
| <code>__wa_sort_seq:N</code> 131, 529, 536, 557, 617, 1772 | <code>\WithArrowsNbLines</code> |
| <code>\l__wa_start_adjust_dim</code> 374, 663, 666, 1663 | 941 |
| <code>\g__wa_static_col_int</code> ... 734, 802, 963, 965 | <code>\WithArrowsNewStyle</code> |
| <code>\l__wa_status_arrow_str</code> | 2262, 2265, 2295 |
| 577, 603, 658, 704, 725, 1448, 1478, 1505, 1522 | <code>\WithArrowsOptions</code> 49, 670, 673, 1331, 2060, 2288 |
| <code>__wa_strcmp:nn</code> | <code>\WithArrowsRightX</code> |
| 124, 128, 137 | 929 |
| | X |
| | <code>\x</code> . 1887, 1889, 1890, 1891, 2305, 2315, 2322, 2325 |
| | Y |
| | <code>\y</code> |
| | 2315, 2322 |
| | Z |
| | <code>\Z</code> |
| | 485, 1880 |

Contents

| | | |
|-----------|--|-----------|
| 1 | Options for the shape of the arrows | 1 |
| 2 | Numbers of columns | 6 |
| 3 | Precise positioning of the arrows | 6 |
| 4 | The options 'up' and 'down' for individual arrows | 9 |
| 5 | Comparison with the environment <code>{aligned}</code> | 9 |
| 6 | Arrows in nested environments | 12 |
| 7 | Arrows from outside environments <code>{WithArrows}</code> | 14 |
| 8 | The environment <code>{DispWithArrows}</code> | 16 |
| 8.1 | The option <code><...></code> of <code>DispWithArrows</code> | 20 |
| 9 | Advanced features | 21 |
| 9.1 | Utilisation with plain-TeX | 21 |
| 9.2 | The option <code>tikz-code</code> : how to change the shape of the arrows | 21 |
| 9.3 | The command <code>\WithArrowsNewStyle</code> | 22 |
| 9.4 | Vertical positioning of the arrows | 22 |
| 9.5 | Footnotes in the environments of <code>witharrows</code> | 24 |
| 9.6 | Option <code>no-arrows</code> | 24 |
| 9.7 | Note for developpers | 24 |
| 10 | Examples | 25 |
| 10.1 | <code>\MoveEqLeft</code> | 25 |
| 10.2 | Modifying the shape of the nodes | 25 |
| 10.3 | Examples with the option <code>tikz-code</code> | 26 |
| 10.3.1 | Example 1 | 26 |
| 10.3.2 | Example 2 | 27 |
| 10.3.3 | Example 3 | 27 |
| 10.4 | Automatic numbered loop | 28 |
| 11 | Implementation | 29 |
| 11.1 | Declaration of the package and extensions loaded | 29 |
| 11.2 | The packages <code>footnote</code> and <code>footnotehyper</code> | 30 |
| 11.3 | The class option <code>legno</code> | 31 |
| 11.4 | Some technical definitions | 32 |
| 11.5 | Variables | 35 |
| 11.6 | The definition of the options | 37 |
| 11.7 | The command <code>\Arrow</code> | 45 |
| 11.8 | The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code> | 46 |
| 11.8.1 | Code before the <code>\halign</code> | 46 |
| 11.8.2 | The construction of the preamble of the <code>\halign</code> | 49 |
| 11.8.3 | The environment <code>{WithArrows}</code> | 52 |
| 11.8.4 | After the construction of the <code>\halign</code> | 53 |
| 11.8.5 | The command of end of row | 54 |
| 11.8.6 | The environment <code>{DispWithArrows}</code> | 58 |
| 11.9 | The commands <code>\tag</code> , <code>\notag</code> , <code>\label</code> , <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code> | 63 |
| 11.10 | We draw the arrows | 65 |
| 11.11 | The command <code>\Arrow</code> in code-after | 74 |
| 11.12 | The command <code>\MultiArrow</code> in code-after | 76 |
| 11.13 | The error messages of the package | 78 |
| 11.14 | The command <code>\WithArrowsNewStyle</code> | 83 |
| 11.15 | The options <code>up</code> and <code>down</code> | 83 |

12 History

86

Index

88