

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

F. Pantigny  
fpantigny@wanadoo.fr

December 27, 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. 22). 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:

```


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


```

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

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

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

## 1 Options for the shape of the arrows

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

The option `jump` gives the number<sup>1</sup> of rows the arrow must jump (the default value is, of course, 1).

```


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


```

---

\*This document corresponds to the version 2.3 of `witharrows`, at the date of 2019/12/27.

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

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

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 stucked on the text). The initial 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}

```

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

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

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

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

In fact, it’s possible to change more drastically the shape or the arrows with the option `tikz-code` (presented p. 22).

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

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

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

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. 19.

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

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

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

It’s possible to put commands `\` in the text to force new lines<sup>3</sup>. However, if we put a `\`, a command of font placed in the beginning of the text will have effect only until the first command `\` (like in an environment `{tabular}`). That’s why Tikz 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.

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

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

```

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

```

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

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.<sup>4</sup>

```

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

```

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

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}

```

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

The same example with the option `displaystyle`:

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

---

<sup>4</sup>They 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}`.<sup>5</sup>

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

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

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` 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` 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. 13.

---

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

## 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 initial 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*}
```

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

## 3 Precise positioning of the arrows

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

<sup>6</sup>The option `show-nodes` can be used to materialize the nodes. The nodes are in fact Tikz nodes of shape “rectangle”, but with zero width. An arrow between two nodes starts at the *south* anchor of the first node and arrives at the *north* anchor of the second node.

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

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.

```

 $\begin{WithArrows}[displaystyle,group]
2xy'-3y=\sqrt{x}
& \Leftrightarrow 2x(K'y_0+Ky_0')-3Ky_0 = \sqrt{x} \quad \backslash\backslash
& \Leftrightarrow 2xK'y_0 + K(2xy_0'-3y_0) = \sqrt{x} \quad \backslash\backslash
& \Leftrightarrow 2xK'y_0 = \sqrt{x} \quad \backslash\backslash \text{Arrow}\{...\}\backslash\backslash
\dots
\end{WithArrows}$ 

```

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

The environment `{WithArrows}` gives also a `groups` option (with a *s* in the name). With this option, the arrows are divided into several “groups”. Each group is a set of connected<sup>7</sup> arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

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

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 arrow<sup>8</sup>. Such arrow will be drawn irrespective of the groups. It’s also possible to start a new group by applying the option `new-group` to an given arrow.

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

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).

<sup>7</sup>More precisely: for each arrow *a*, we note *i(a)* the number of its initial row and *f(a)* the number of its final row; for two arrows *a* and *b*, we say that *a* ~ *b* when  $\llbracket i(a), f(a) \rrbracket \cap \llbracket i(b), f(b) \rrbracket \neq \emptyset$ ; the groups are the equivalence classes of the transitive closure of ~.

<sup>8</sup>Such 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 individual arrows, called “up” and “down”. The following example illustrates these types of arrows:

```

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

```

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

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

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

- The key `radius` is the radius of the rounded corner of the arrow.<sup>9</sup>

<sup>9</sup>The initial value of this parameter is 4 pt, which is the default value of the “`rounded corners`” of Tikz.

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

```

 $\begin{WithArrows}$ 
 $A \& = B$ 
 $\xrightarrow[\text{we try}]{} \\ \& = C + C + C + C + C + C + C + C + C$ 
 $\end{WithArrows}$ 

```

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

```

 $\begin{WithArrows}$ 
 $A \& = B$ 
 $\xrightarrow[\text{we try}]{} \\ \& = C + C + C + C + C + C + C + C + C$ 
 $\end{WithArrows}$ 

```

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

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

```

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

```

## 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}`.<sup>10</sup>

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

```

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

```

---

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

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

In the environments of `amsmath` (or `mathtools`), the spacing between rows is fixed by a parameter called `\jot` (it's a dimension and not a skip). That's also the case for the environment `{WithArrows}`. An option `jot` has been given to the environment `{WithArrows}` in order to change the value of this parameter `\jot` for a given environment.<sup>11</sup>

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

`\WithArrowsOptions{displaystyle}`.

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

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

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

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

<sup>12</sup>In 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}`.

## 6 Arrows in nested environments

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

```

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

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

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 :

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

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

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

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

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

The specification of the row is constructed with the position of the concerned environment in the nesting tree, followed (after an hyphen) by the number of 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 $2$} ]$ 
 $\varphi(x, y) = 0$ 
 $\quad \& \Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0 \quad \backslash \backslash$ 
.....
 $\end{WithArrows}$ 

```

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

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 $2$}]$ 
 $\varphi(x, y) = 0$ 
 $\quad \& \Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0 \quad \backslash \backslash$ 
.....
 $\end{WithArrows}$ 

```

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

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

```

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

```

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

## 7 Arrows from outside environments `{WithArrows}`

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

$$\begin{aligned}
A &\triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B + B + B^{\text{wa-42-1-r}} \\
&\triangleleft \begin{cases} C \triangleleft D^{\text{wa-42-1-1-r}} \\ E \triangleleft F^{\text{wa-42-1-2-r}} \end{cases} \quad \text{wa-42-2-r} \\
&\triangleleft \begin{cases} G \triangleleft H + H + H + H + H + H + H^{\text{wa-42-2-1-r}} \\ I \triangleleft \begin{cases} J \triangleleft K^{\text{wa-42-2-1-1-r}} \\ L \triangleleft M^{\text{wa-42-2-1-2-r}} \end{cases} \end{cases} \quad \begin{matrix} \text{wa-42-3-r} \\ \text{wa-42-2-2-r} \end{matrix} \\
&\triangleleft \begin{cases} N \triangleleft O^{\text{wa-42-3-1-r}} \\ P \triangleleft Q^{\text{wa-42-3-2-r}} \end{cases} \quad \text{wa-42-4-r}
\end{aligned}$$

- 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 arrow<sup>14</sup>;
- the Tikz style `WithArrows/arrow/tips` is the style for the tip of the arrow (loaded by `WithArrows/arrow`).

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

15

```

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

```

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

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

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

```

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

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

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

```

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

## 8 The environment `{DispWithArrows}`

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

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



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

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

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}
```

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

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

If `amsmath` (or `mathtools`) is loaded, it's also possible to use `\tag*` which, as in `amsmath`, typesets the tag without the parentheses. For example, it's possible to use it to put the symbol `\square` of `amssymb`. This symbol is often used to mark the end of a proof.<sup>16</sup>

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

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

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.<sup>17</sup>

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

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

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 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}
```

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

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

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

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

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

$$A = (a + 1)^2 \tag{4}$$

$$= a^2 + 2a + 1 \quad \downarrow \text{we expand} \tag{5}$$

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

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

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

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

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

First environment.

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

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

Second environment.

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

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

---

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

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}
```

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

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

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

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

$\left. \begin{array}{l} \text{sum of terms of a geometric progres-} \\ \text{sion of ratio } e^{i \frac{\pi}{2n}} \end{array} \right\}$   
 $\left. \begin{array}{l} \text{This line has been wrapped automati-} \\ \text{cally.} \end{array} \right\}$

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

We have said that the environments `{DispWithArrows}` and `{DispWithArrows*}` should be used in horizontal mode and not in vertical mode. However, there is an exception. These environments can

be used directly after a `\item` of a LaTeX list. In this case, no vertical space is added before the environment.<sup>19</sup>

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

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

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

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

Concerning the references, the package `witharrows` is compatible with the extensions `autonum`, `cleveref`, `fancyref`, `hyperref`, `listbls`, `prettyref`, `refcheck`, `refstyle`, `showlabels`, `smartref`, `typedref` and `varioref`, and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.<sup>20</sup>

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

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

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

## 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.<sup>21</sup>

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

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

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

$$\binom{n}{p} = \begin{cases} 0 & \text{if } p > n \\ \frac{n(n-1)\cdots(n-p+1)}{p!} & \text{if } 0 \leq p \leq n \\ 0 & \text{if } p < 0 \end{cases} \quad \begin{matrix} (14) \\ (15) \\ (16) \end{matrix}$$

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

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

$$(17) \Leftrightarrow \begin{cases} x + y + z = -3 \\ xy + xz + yz = -2 \\ xyz = -15 \end{cases} \quad \begin{matrix} (17a) \\ (17b) \\ (17c) \end{matrix}$$

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.

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

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

## 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 fonctionnalités of the LaTeX version. In particular, the fonctionnalités 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` allows the user to change the shape of the arrows.<sup>23</sup>

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).

```
\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}
```

---

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

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

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

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

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}$

```

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

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

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

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

Since the version 2.1 of `witharrows`, an arrow of `jump` equal to 1 has a maximal length<sup>24</sup> equal to the parameter `max-length-of-arrow`. The initial value of this parameter is 2 cm.

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

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

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

$$= \begin{vmatrix} 1 & a & a^2 & a^3 & a^4 \\ 0 & b-a & b^2-a^2 & b^3-a^3 & b^4-a^4 \\ 0 & c-a & c^2-a^2 & c^3-a^3 & c^4-a^4 \\ 0 & d-a & d^2-a^2 & d^3-a^3 & d^4-a^4 \\ 0 & e-a & e^2-a^2 & e^3-a^3 & e^4-a^4 \end{vmatrix}$$

<sup>24</sup>We call *length* of an arrow the difference between the *y*-value of its start point and the *y* value of its end point.



## 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:

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

## 9.6 Option no-arrows

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

## 9.7 Note for developers

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

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

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

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

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

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

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

---

<sup>25</sup>A footnote.

## 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 - \bigl(\frac{4}{5}\bigr)^2}
\end{WithArrows}$ 

```

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

### 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.

```

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

```

$$\begin{array}{lcl}
 3(2x+4) = 6 & & \begin{array}{c} \circlearrowleft 3 \\ \downarrow \end{array} \\
 2x+4 = 2 & & \begin{array}{c} \circlearrowleft -4 \\ \downarrow \end{array} \\
 2x = -2 & & \begin{array}{c} \circlearrowleft 2 \\ \downarrow \end{array} \\
 2x = -1 & & \begin{array}{c} \circlearrowleft 2 \\ \downarrow \end{array}
 \end{array}$$

### 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.<sup>26</sup>

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

#### 10.3.1 Example 1

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

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

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

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

#### 10.3.2 Example 2

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

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

```

\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\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) \\
  &\quad \text{\Arrow{\$ \cos x = \Re(e^{ix})\$}} \\
  &\dots\dots\dots

```

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

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

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right) \quad \overleftarrow{\exp \text{ is a morphism for } \times \text{ et } +} \quad (20)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right) \quad \overleftarrow{\text{sum of terms of a geometric}} \quad (21)$$

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

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

### 10.3.3 Example 3

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

```

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

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

```

```

(\p1) -- (\x2,\y1) -- node {#3} (\p2)
\fi ;}]
E & \Longleftarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\Arrow{$\times 15$}\\
& \Longleftarrow 5(x+4) + 3(5x+3) = 105 \\
& \Longleftarrow 5x+20 + 15x+9 = 105 \\
& \Longleftarrow 20x+29 = 105
\Arrow{$-29$}\\
& \Longleftarrow 20x = 76
\Arrow{$\div 20$}\\
& \Longleftarrow x = \frac{38}{10}
\end{WithArrows}

```

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

## 10.4 Automatic numbered loop

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

```

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

```

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

```

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

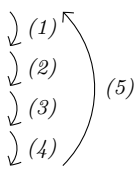
```

$$\begin{aligned}
 &a. f \text{ est continuous on } E \\
 &b. f \text{ est continuous in } 0 \\
 &c. f \text{ is bounded on the unit sphere} \\
 &d. \exists K > 0 \quad \forall x \in E \quad \|f(x)\| \leq K \|x\| \\
 &e. f \text{ is lipschitzian}
 \end{aligned}
 \begin{array}{c}
 \downarrow 1 \\
 \downarrow 2 \\
 \downarrow 3 \\
 \downarrow 4
 \end{array}
 \left. \vphantom{\begin{array}{c} \downarrow 1 \\ \downarrow 2 \\ \downarrow 3 \\ \downarrow 4 \end{array}} \right\} 5$$

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.<sup>27</sup>

```
<@@=witharrows>
```

```

1 <*LaTeX>
2 \RequirePackage{tikz}
3 \RequirePackage{expl3}[2019/07/01]
4 </LaTeX>
5 <*plain-TeX>
6 \input tikz.tex
7 \input expl3-generic.tex
8 </plain-TeX>
9 \usetikzlibrary{arrows.meta,bending}

```

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

```

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

```

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

```

17 \RequirePackage { xparse } [ 2019-01-01 ]
18 </LaTeX>
19 <*plain-TeX>
20 \ExplSyntaxOn
21 \catcode ` \@ = 11
22 </plain-TeX>

```

<sup>27</sup>cf. [tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails](https://tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails)

## 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 2.3), 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.

```
23 \*LaTeX
24 \bool_new:N \g_@@_footnotehyper_bool
```

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

```
25 \bool_new:N \g_@@_footnote_bool
26 \*LaTeX
```

```
27 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
28 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { witharrows } }
29 \cs_new_protected:Npn \@@_msg_redirect_name:nn
30   { \msg_redirect_name:nnn { witharrows } }
31 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
32 \cs_new_protected:Npn \@@_warning:n { \msg_warning:nn { witharrows } }
33 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { witharrows } }
34 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { witharrows } }
35 \cs_generate_variant:Nn \@@_error:nn { n x }
```

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

```
36 \*LaTeX
37 \keys_define:nn { WithArrows / package }
38   {
39     footnote .bool_gset:N = \g_@@_footnote_bool ,
40     footnotehyper .bool_gset:N = \g_@@_footnotehyper_bool ,
41     unknown .code:n =
42       \@@_fatal:n { Option~unknown~for~package }
43   }
44 \@@_msg_new:nn { Option~unknown~for~package }
45   {
46     You~can't~use~the~option~'\l_keys_key_tl'~when~loading~the~
47     package~witharrows.~Try~to~use~the~command~
48     \token_to_str:N\WithArrowsOptions.
49   }
```

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

```
50 \ProcessKeysOptions { WithArrows / package }

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

56 \@@_msg_new:nn { footnote~with~footnotehyper~package }
57   {
58     You~can't~use~the~option~'footnote'~because~the~package~
59     footnotehyper~has~already~been~loaded.~
60     If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
61     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
62     of~the~package~footnotehyper.\\
63     If~you~go~on,~the~package~footnote~won't~be~loaded.
64   }
```

```

65 \@@_msg_new:nn { footnotehyper~with~footnote~package }
66 {
67   You~can't~use~the~option~'footnotehyper'~because~the~package~
68   footnote~has~already~been~loaded.~
69   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
70   within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
71   of~the~package~footnote.\@
72   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
73 }

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

84 \bool_if:NT \g_@@_footnotehyper_bool
85 {
86   \@ifclassloaded { beamer }
87   { \@_info:n { Option~incompatible~with~Beamer } }
88   {
89     \@ifpackageloaded { footnote }
90     { \@_error:n { footnotehyper~with~footnote~package } }
91     { \usepackage { footnotehyper } }
92   }
93   \bool_gset_true:N \g_@@_footnote_bool
94 }

```

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

### 11.3 The class option `leqno`

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

```

95 \bool_new:N \c_@@_leqno_bool
96 \DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
97 \DeclareOption* { }
98 \ProcessOptions*
99 </LaTeX>

```

### 11.4 Some technical definitions

```

100 \cs_generate_variant:Nn \tl_put_right:Nn { N v }
101 \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`.<sup>28</sup>

```

102 \AtBeginDocument
103 {
104   \clist_map_inline:nn
105   {
106     amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
107     typedref, unicode-math, varwidth
108   }
109   {
110     \bool_new:c { c_@@_#1_loaded_bool }
111   }
112   \ifpackageloaded { #1 }
113   { \bool_set_true:c { c_@@_#1_loaded_bool } }
114   { }
115 \end{LaTeX}
116 \ifplain
117 { \bool_set_false:c { c_@@_#1_loaded_bool } }
118 \end{plain-LaTeX}
119 }
120 }
```

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

```

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

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

```

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

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`).

```

145 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
146 {
147   \seq_clear:N \l_tmpa_seq
148   \seq_map_inline:Nn #1
149   {
```

---

<sup>28</sup>It's not possible to use `\ifpackageloaded` in the core of the functions because `\ifpackageloaded` is available only in the preamble.

```

150     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
151   }
152   \seq_set_eq:NN #1 \l_tmpa_seq
153 }
154 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
155 {
156   \seq_set_from_clist:Nn #1 { #2 }
157   \@@_convert_to_str_seq:N #1
158 }

```

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.

```

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

```

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

```

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

```

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

```

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

```

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.

```

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

```

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

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

```

195 \tikzset
196 {
197   @@_standard / .style =
198   {
199     remember~picture ,
200     overlay ,
201     name~prefix = wa - \l_@@_prefix_str -
202   }
203 }
```

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

```

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

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

```

209 \tikzset
210 {
211   WithArrows / arrow / .style =
212   {
213     align = left ,
214     auto = left ,
215     <*LaTeX>
216     font = \small \itshape ,
217     </LaTeX>
218     WithArrows / arrow / tips ,
219     bend~left = 45 ,
220     ->
221   }
222 }
```

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}`.

```

223 <*LaTeX>
224 \AtBeginDocument
225 {
226   \bool_if:NTF \c_@@_amsmath_loaded_bool
227   {
228     \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
229     \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
230   }
231 }
```

---

<sup>29</sup>The 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`).

```

231     {
232     </LaTeX>
233     \cs_new_protected:Npn \spread@equation
234     {
235         \openup \jot
236         \cs_set_eq:NN \spread@equation \prg_do_nothing:
237     }
238     <*LaTeX>
239     }
240 }
241 </LaTeX>

242 \tl_new:N \l_@@_left_brace_tl
243 \tl_set_eq:NN \l_@@_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`).

```

244 \bool_new:N \l_@@_in-WithArrows_bool
245 \bool_new:N \l_@@_in-DispWithArrows_bool
246 \bool_new:N \l_@@_in_code_after_bool

```

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

```

247 \seq_new:N \g_@@_position_in_the_tree_seq
248 \seq_gput_right:Nn \g_@@_position_in_the_tree_seq 1

```

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

```

249 \int_new:N \g_@@_last_env_int

```

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

```

250 \int_new:N \l_@@_pos_env_int

```

The integer `\l_@@_pos_arrow_int` indicates the position of the arrow with the following code (the option `v` is accessible only for the arrows in `code-after` where the options `i`, `group` 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).

```

251 \int_new:N \l_@@_pos_arrow_int
252 \int_set:Nn \l_@@_pos_arrow_int 3

```

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

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

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

```
253 \seq_new:N \g_@@_arrow_int_seq
254 \int_new:N \g_@@_arrow_int
255 \seq_new:N \g_@@_line_int_seq
256 \int_new:N \g_@@_line_int
257 \seq_new:N \g_@@_col_int_seq
258 \int_new:N \g_@@_col_int
```

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

```
259 \seq_new:N \g_@@_static_col_int_seq
260 \int_new:N \g_@@_static_col_int
```

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

```
261 \*LaTeX
262 \clist_new:N \l_@@_tags_clist
263 \clist_set:Nn \l_@@_tags_clist { all }
```

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

```
264 \cs_new_protected:Npn \@@_test_if_to_tag:
265 {
266   \clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int
267   { \clist_set:Nn \l_@@_tags_clist { all } }
268 }
269 \*LaTeX
```

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

```
270 \str_new:N \l_@@_command_name_str
271 \str_set:Nn \l_@@_command_name_str { Arrow }
```

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

```
272 \str_new:N \l_@@_string_Arrow_for_msg_str
273 \str_set:Nx \l_@@_string_Arrow_for_msg_str { \token_to_str:N \Arrow }
```

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

```
274 \seq_new:N \g_@@_names_seq
```

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

```

275 \*LaTeX
276 \bool_new:N \l_@@_sbwi_bool
277 \*LaTeX

278 \*LaTeX
279 \bool_new:N \l_@@_tag_star_bool
280 \bool_new:N \l_@@_tag_next_line_bool
281 \bool_new:N \l_@@_qedhere_bool
282 \*LaTeX
283 \bool_new:N \l_@@_in_first_columns_bool
284 \bool_new:N \l_@@_new_group_bool
285 \bool_new:N \l_@@_initial_r_bool
286 \bool_new:N \l_@@_final_r_bool
287 \tl_new:N \l_@@_initial_tl
288 \tl_new:N \l_@@_final_tl
289 \int_new:N \l_@@_nb_cols_int

```

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

```

290 \str_new:N \l_@@_format_str

```

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

```

291 \*LaTeX
292 \bool_new:N \l_@@_subequations_bool
293 \*LaTeX

```

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

```

294 \dim_new:N \l_@@_arrow_width_dim
295 \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim

```

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

```

296 \dim_new:N \l_@@_up_and_down_radius_dim
297 \dim_set:Nn \l_@@_up_and_down_radius_dim { 4 pt }

```

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

---

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

When we scan a list of options, we want to be able to raise an error if two options of position (l1, r1, i, etc.) of the arrows are present. That's why we keep the first option of position in a variable called `\l_@@_previous_key_str`. The following function `\@@_eval_if_allowed:n` will execute its argument only if a first key of position has not been set (and raise an error elsewhere).

```

298 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
299 {
300   \str_if_empty:NTF \l_@@_previous_key_str
301   {
302     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
303     #1
304   }
305   { \@@_error:n { Incompatible~options } }
306 }

307 \cs_new_protected:Npn \@@_fix_pos_option:n #1
308 { \@@_eval_if_allowed:n { \int_set:Nn \l_@@_pos_arrow_int { #1 } } }

```

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

```

309 \keys_define:nn { WithArrows / Global }
310 {
311   max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
312   max-length-of-arrow .value_required:n = true ,
313   max-length-of-arrow .initial:n = 2 cm ,
314   ygap .dim_set:N = \l_@@_ygap_dim ,
315   ygap .initial:n = 0.4 ex ,
316   ystart .dim_set:N = \l_@@_ystart_dim ,
317
318   ystart .initial:n = 0.4 ex ,
319   more-columns .code:n =
320     \@@_msg_redirect_name:nn { Too~much~columns~in~WithArrows } { none } ,
321   command-name .code:n =
322     \str_set:Nn \l_@@_command_name_str { #1 }
323     \str_set:Nx \l_@@_string_Arrow_for_msg_str
324     { \c_backslash_str Arrow~alias~\c_backslash_str #1 } ,
325   tikz-code .tl_set:N = \l_@@_tikz_code_tl,
326   tikz-code .initial:n = \draw~( #1 )~to~node{ #3 }~( #2 )~; ,
327   TikzCode .meta:n = { tikz-code = #1 } ,
328   displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
329   displaystyle .default:n = true ,
330   show-nodes .code:n =
331     \tikzset { @@_node_style / .append~style = { draw , red } } ,
332   show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
333   show-node-names .default:n = true ,
334   group .code:n =
335     \str_if_empty:NTF \l_@@_previous_key_str
336     {
337       \str_set:Nn \l_@@_previous_key_str { group }
338       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
339       \int_set:Nn \l_@@_pos_arrow_int 7
340     }
341     { \@@_error:n { Incompatible~options } } ,
342   groups .code:n =
343     \str_if_empty:NTF \l_@@_previous_key_str
344     {
345       \str_set:Nn \l_@@_previous_key_str { groups }
346       \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
347       { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
348       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
349       \int_set:Nn \l_@@_pos_arrow_int 6
350     }
351     { \@@_error:n { Incompatible~options } } ,
352   tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
353   tikz .initial:n = \c_empty_tl ,

```

```

354 rr .code:n = \@@_fix_pos_option:n 3 ,
355 ll .code:n = \@@_fix_pos_option:n 1 ,
356 rl .code:n = \@@_fix_pos_option:n 2 ,
357 lr .code:n = \@@_fix_pos_option:n 0 ,
358 i .code:n = \@@_fix_pos_option:n 5 ,
359 xoffset .dim_set:N = \l_@@_xoffset_dim ,
360 xoffset .initial:n = 3 mm ,
361 jot .dim_set:N = \jot ,
362 interline .skip_set:N = \l_@@_interline_skip ,
363 start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
364 start-adjust .initial:n = 0.4 ex ,
365 end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
366 end-adjust .initial:n = 0.4 ex ,
367 adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
368 up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,

```

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”.

```

369 no-arrows .code:n =
370   \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn
371   \cs_set_eq:NN \@@_draw_arrow:nnn \use_none:nnn ,

```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). We maintain these lines in the DTX only for the case of a modification of `13keys`.

```

372 {*comment}
373 ygap .value_required:n = true ,
374 ystart .value_required:n = true ,
375 more-columns .value_forbidden:n = true,
376 command-name .value_required:n = true ,
377 tikz-code .value_required:n = true ,
378 group .value_forbidden:n = true ,
379 groups .value_forbidden:n = true ,
380 tikz .value_required:n = true ,
381 show-nodes .value_forbidden:n = true,
382 ll .value_forbidden:n = true ,
383 rr .value_forbidden:n = true ,
384 rl .value_forbidden:n = true ,
385 lr .value_forbidden:n = true ,
386 i .value_forbidden:n = true ,
387 xoffset .value_required:n = true ,
388 jot .value_required:n = true ,
389 interline .value_required:n = true ,
390 start-adjust .value_required:n = true ,
391 end-adjust .value_required:n = true ,
392 adjust .value_required:n = true ,
393 up-and-down .value_required:n = true ,
394 no-arrows .value_forbidden:n = true
395 {/comment}
396 }

```

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`.

```

397 \keys_define:nn { WithArrows / WithArrowsSpecific }
398 {
399   t .code:n = \int_set:Nn \l_@@_pos_env_int 0 ,
400   c .code:n = \int_set:Nn \l_@@_pos_env_int 1 ,
401   b .code:n = \int_set:Nn \l_@@_pos_env_int 2 ,

```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). We maintain these lines in the DTX only for the case of a modification of `13keys`.



```

402 <*comment>
403   t .value_forbidden:n = true ,
404   c .value_forbidden:n = true ,
405   b .value_forbidden:n = true
406 </comment>
407 }

```

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

```

408 \clist_new:N \c_@@_extensible_delimiters_clist
409 \clist_set:Nn \c_@@_extensible_delimiters_clist
410 {
411   ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
412 }
413 <*LaTeX>
414 \AtBeginDocument
415 {
416   \bool_if:nT
417     { \c_@@_amsmath_loaded_bool || \use:c { c_@@_unicode-math_loaded_bool } }
418     {
419       \clist_put_right:Nn \c_@@_extensible_delimiters_clist { \lvert, \lVert }
420     }
421 }
422 </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`.

```

423 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
424 {
425   fleqn .bool_set:N = \l_@@_fleqn_bool ,
426   fleqn .default:n = true ,
427   mathindent .dim_set:N = \l_@@_mathindent_dim ,
428   mathindent .initial:n = 25 pt ,
429 <*LaTeX>
430   notag .code:n =
431     \str_if_eq:nnTF { #1 } { true }
432     { \clist_clear:N \l_@@_tags_clist }
433     { \clist_set:Nn \l_@@_tags_clist { all } } ,
434   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.

```

435   subequations .code:n =
436     \bool_if:NTF \c_@@_amsmath_loaded_bool
437     { \bool_set_true:N \l_@@_subequations_bool }
438     {
439       \@@_error:n { amsmath-not-loaded }
440       \group_begin:
441       \globaldefs = 1
442       \@@_msg_redirect_name:nn { amsmath-not-loaded } { info }
443       \group_end:
444     } ,
445   subequations .default:n = true ,
446   nonumber .meta:n = notag ,
447   allow-multiple-labels .code:n =
448     \@@_msg_redirect_name:nn { Multiple-labels } { none } ,
449   tagged-lines .code:n =
450     \clist_set:Nn \l_@@_tags_clist { #1 }
451     \clist_if_in:NnT \l_@@_tags_clist { first }
452     {
453       \clist_remove_all:Nn \l_@@_tags_clist { first }
454       \clist_put_left:Nn \l_@@_tags_clist \c_one_int

```

```

455     } ,
456 \LaTeX
457     wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
458     wrap-lines .default:n = true ,
459     replace-left-brace-by .code:n =
460     {
461         \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
462         \clist_if_in:NVTF
463             \c_@@_extensible_delimiters_clist
464             \l_tmpa_tl
465             { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }
466             { \@@_error:n { Bad-value-for-replace-brace-by } }
467     } ,
468     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`).

```

469 \LaTeX
470     standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
471     standard-behaviour-with-items .default:n = true ,
472 \LaTeX

```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). We maintain these lines in the DTX only for the case of a modification of `l3keys`.

```

473 \comment
474     mathindent .value_required:n = true ,
475     subequations .value_forbidden:n = true ,
476     allow-multiple-labels .value_forbidden:n = true ,
477     tagged-lines .value_required:n = true
478 \comment
479 }

```

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

```

480 \keys_define:nn { WithArrows / Env }
481 {
482     name .code:n =

```

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

```

483     \str_set:Nn \l_tmpa_str { #1 }
484     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
485     { \@@_error:n { Duplicate-name } }
486     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
487     \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
488     code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
489     CodeBefore .meta:n = { code-before = #1 } ,
490     code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
491     CodeAfter .meta:n = { code-after = #1 } ,
492     format .code:n =
493     \tl_if_empty:nTF { #1 }
494     { \@@_error:n { Invalid-option-format } }
495     {
496         \regex_match:nnTF { \A[rcl]*Z } { #1 }
497         { \tl_set:Nn \l_@@_format_str { #1 } }
498         { \@@_error:n { Invalid-option-format } }
499     } ,

```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). Maybe we should delete these lines.

```

500     code-before .value_required:n = true,

```

```

501     code-after .value_required:n = true ,
502     name .value_required:n = true ,
503     format .value_required:n = true ,
504 }

```

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.

```

505 \keys_define:nn { WithArrows }
506 {
507     WithArrows .inherit:n =
508     {
509         WithArrows / Global ,
510         WithArrows / WithArrowsSpecific ,
511         WithArrows / Env
512     } ,
513     WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
514     DispWithArrows .inherit:n =
515     {
516         WithArrows / DispWithArrowsSpecific ,
517         WithArrows / Global ,
518         WithArrows / Env ,
519     } ,
520     DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
521     WithArrowsOptions .inherit:n =
522     {
523         WithArrows / Global ,
524         WithArrows / WithArrowsSpecific ,
525         WithArrows / DispWithArrowsSpecific ,
526     } ,
527     WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
528 }

```

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

```

529 \seq_new:N \l_@@_options-WithArrows_seq
530 \@@_set_seq_of_str_from_clist:Nn \l_@@_options-WithArrows_seq
531 {
532     adjust, b, c, code-after, code-before, command-name,
533     displaystyle, end-adjust,
534     format, group, groups, i,
535     interline, jot, ll,
536     lr, max-length-of-arrow, more-columns, name,
537     no-arrows, rl, rr, up-and-down,
538     show-node-names, show-nodes, start-adjust,
539     t, tikz, tikz-code,
540     xoffset, ygap, ystart
541 }
542 \@@_convert_to_str_seq:N \l_@@_options-WithArrows_seq

543 \keys_define:nn { WithArrows / WithArrows }
544 {
545     unknown .code:n =
546         \@@_sort_seq:N \l_@@_options-WithArrows_seq
547         \@@_error:n { Unknown-option-WithArrows }
548 }

549 \keys_define:nn { WithArrows / DispWithArrows }
550 {
551     left-brace .tl_set:N = \l_@@_left_brace_tl ,
552     unknown .code:n =

```

```

553 \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
554 \@@_error:n { Unknown~option~DispWithArrows } ,
555 }

```

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

```

556 \seq_new:N \l_@@_options_DispWithArrows_seq
557 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
558 {
559   code-after, code-before, command-name, tikz-code, adjust,
560   displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
561   left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
562   up-and-down, replace-left-brace-by, rl, rr, show-node-names,
563   show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
564   (*LaTeX)
565   allow-multiple-labels, tagged-lines, nonumber, notag
566   (/LaTeX)
567 }
568 \keys_define:nn { WithArrows / WithArrowsOptions }
569 {
570   allow-duplicate-names .code:n =
571     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
572   allow-duplicate-names .value_forbidden:n = true ,
573   unknown .code:n =
574     \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
575     \@@_error:n { Unknown~option~WithArrowsOptions }
576 }

```

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

```

577 \seq_new:N \l_@@_options_WithArrowsOptions_seq
578 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrowsOptions_seq
579 {
580   allow-duplicate-names, b, c, command-name, more-columns, tikz-code, adjust,
581   displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
582   mathindent, max-length-of-arrow, no-arrows, up-and-down, rl, rr,
583   show-node-names, show-nodes, start-adjust, t, tikz, wrap-lines, xoffset,
584   ygap, ystart,
585   (*LaTeX)
586   allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
587   tagged-lines
588   (/LaTeX)
589 }

```

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

```

590 \cs_new_protected:Npn \@@_set_independent:
591 {
592   \str_if_eq:VnF \l_keys_value_tl { NoValue }
593     { \@@_error:n { Value~for~a~key } }
594   \@@_set_independent_bis:
595 }

```

The command \@@\_set\_independent\_bis: is the same as \@@\_set\_independent: except that the key may be used with a value.

```

596 \cs_new_protected:Npn \@@_set_independent_bis:
597 {
598   \str_if_empty:NTF \l_@@_previous_key_str

```

```

599     {
600       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
601       \str_set:Nn \l_@@_status_arrow_str { independent }
602     }
603     { \@@_error:n { Incompatible~options-in~Arrow } }
604   }

```

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”.

```

605 \keys_define:nn { WithArrows / Arrow / FirstPass }
606 {
607   jump .code:n =
608     \int_compare:nTF { #1 > 0 }
609     { \int_set:Nn \l_@@_jump_int { #1 } }
610     { \@@_error:n { Negative~jump } } ,
611   jump .value_required:n = true,
612   rr .code:n = \@@_set_independent: ,
613   ll .code:n = \@@_set_independent: ,
614   rl .code:n = \@@_set_independent: ,
615   lr .code:n = \@@_set_independent: ,
616   i .code:n = \@@_set_independent: ,
617   rr .default:n = NoValue ,
618   ll .default:n = NoValue ,
619   rl .default:n = NoValue ,
620   lr .default:n = NoValue ,
621   i .default:n = NoValue ,
622   new-group .value_forbidden:n = true,
623   new-group .code:n =
624     \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
625     { \str_set:Nn \l_@@_status_arrow_str { new-group } }
626     { \@@_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.

```

627   tikz-code .code:n = \prg_do_nothing: ,
628   tikz-code .value_required:n = true ,
629   tikz .code:n = \prg_do_nothing: ,
630   tikz .value_required:n = true ,
631   start-adjust .code:n = \prg_do_nothing: ,
632   start-adjust .value_required:n = true ,
633   end-adjust .code:n = \prg_do_nothing: ,
634   end-adjust .value_required:n = true ,
635   adjust .code:n = \prg_do_nothing: ,
636   adjust .value_required:n = true ,
637   xoffset .code:n = ,
638   unknown .code:n =
639     \@@_sort_seq:N \l_@@_options_Arrow_seq
640     \seq_if_in:NVTF \l_@@_options_WithArrows_seq \l_keys_key_tl
641     {
642       \str_set:Nn \l_tmpa_str
643       { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
644     }
645     { \str_clear:N \l_tmpa_str }
646     \@@_error:n { Unknown~option-in~Arrow }
647   }

```

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

```

648 \seq_new:N \l_@@_options_Arrow_seq
649 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_seq
650 {
651     adjust, end-adjust, i, jump, ll, lr, rl, rr, start-adjust, tikz, tikz-code,
652     xoffset
653 }

654 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1
655 {
656     \str_if_empty:NT \l_@@_previous_key_str
657     {
658         \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_tl
659         \int_set:Nn \l_@@_pos_arrow_int { #1 }
660     }
661 }

```

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.

```

662 \keys_define:nn {WithArrows / Arrow / SecondPass }
663 {
664     tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
665     tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
666     tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
667     tikz .initial:n = \c_empty_tl ,
668     rr .code:n = \@@_fix_pos_arrow:n 3 ,
669     ll .code:n = \@@_fix_pos_arrow:n 1 ,
670     rl .code:n = \@@_fix_pos_arrow:n 2 ,
671     lr .code:n = \@@_fix_pos_arrow:n 0 ,
672     i .code:n = \@@_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.

```

673     xoffset .code:n =
674         \bool_if:nTF
675         {
676             \int_compare_p:nNn \g_@@_arrow_int > 1
677             &&
678             \int_compare_p:nNn \l_@@_pos_arrow_int > 5
679             &&
680             ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
681         }
682         { \@@_error:n { Option~xoffset~forbidden } }
683         { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
684     xoffset .value_required:n = true ,
685     start-adjust .dim_set:N = \l_@@_start_adjust_dim,
686     end-adjust .dim_set:N = \l_@@_end_adjust_dim,
687     adjust .code:n =
688         \dim_set:Nn \l_@@_start_adjust_dim { #1 }
689         \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
690 }

```

`\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`.

```

691 {*LaTeX}
692 \NewDocumentCommand \WithArrowsOptions { m }
693 {/LaTeX}
694 {*plain-TeX}

```

```

695 \cs_set_protected:Npn \WithArrowsOptions #1
696 </plain-TeX>
697 {
698   \str_clear_new:N \l_@@_previous_key_str
699   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
700 }

```

## 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}`.

```

701 <*LaTeX>
702 \NewDocumentCommand \@@_Arrow { 0 { } m ! 0 { } }
703 </LaTeX>
704 <*plain-TeX>
705 \cs_new_protected:Npn \@@_Arrow
706 {
707   \peek_meaning:NTF [
708     { \@@_Arrow_i }
709     { \@@_Arrow_i [ ] }
710   }
711   \cs_new_protected:Npn \@@_Arrow_i [ #1 ] #2
712   {
713     \peek_meaning:NTF [
714       { \@@_Arrow_ii [ #1 ] { #2 } }
715       { \@@_Arrow_ii [ #1 ] { #2 } [ ] }
716     }
717     \cs_new_protected:Npn \@@_Arrow_ii [ #1 ] #2 [ #3 ]
718   </plain-TeX>
719   {

```

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.

```

720   \int_gincr:N \g_@@_arrow_int

```

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

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

```

721   \str_clear_new:N \l_@@_previous_key_str
722   \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:

```

723   \prop_put:NnV \l_tmpa_prop { initial } \g_@@_line_int

```

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

```

724   \int_set:Nn \l_tmpa_int { \g_@@_line_int + \l_@@_jump_int }
725   \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.

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

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

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

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

```
728 \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).

```
729 \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.

```
730 \prop_gclear_new:c
731 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
732 \prop_gset_eq:cN
733 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
734 \l_tmpa_prop
735 }
```

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.

```
736 \cs_new_protected:Npn \@@_Arrow_first_columns:
737 { \@@_error:n { Arrow-not-in-last-column } \@@_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.

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

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

```
739 {
740 <*LaTeX>
741 \str_clear_new:N \l_@@_type_env_str
742 \str_set:NV \l_@@_type_env_str \currenenvir
743 </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`.

```
744 \cs_if_exist:NT \tikz@library@external@loaded
745 { \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.

```
746 \str_clear_new:N \l_@@_name_str
```



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

```
747 \str_clear_new:N \l_@@_status_arrow_str
```

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

```
748 \dim_zero_new:N \l_@@_x_dim
```

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

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

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

```
750 \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
751 \int_gzero:N \g_@@_arrow_int
752 \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
753 \int_gzero:N \g_@@_line_int
754 \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
755 \int_gzero:N \g_@@_col_int
756 \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
757 \int_gzero:N \g_@@_static_col_int
```

In the preamble of the `\halign`, there will be *two* counters of the columns. The aim of this programming is to detect the 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_set:Nn \g_@@_static_col_int 3
```

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

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

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

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

```
759 \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
760 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
761 \str_clear_new:N \l_@@_prefix_str
762 \str_set:Nx \l_@@_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

```
763 \cs_set_eq:NN \ \@@_cr:
764 \dim_zero:N \mathsurround
```

These counters will be used later as variables.

```
765 \int_zero_new:N \l_@@_initial_int
766 \int_zero_new:N \l_@@_final_int
767 \int_zero_new:N \l_@@_arrow_int
768 \int_zero_new:N \l_@@_pos_of_arrow_int
769 \int_zero_new:N \l_@@_jump_int
```

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

```
770 \int_set:Nn \l_@@_jump_int \c_one_int
```

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

```
771 \str_set:Nn \l_@@_format_str { rl }
```

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

```
772 <*LaTeX>
773 \seq_clear_new:N \l_@@_labels_seq
774 \bool_set_false:N \l_@@_tag_next_line_bool
775 </LaTeX>
```

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

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

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

```
777 \tl_clear_new:N \l_@@_code_before_tl
778 \tl_clear_new:N \l_@@_code_after_tl
```

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

```
779 \str_clear_new:N \l_@@_previous_key_str
780 \bool_if:NT \l_@@_in_WithArrows_bool
781 { \keys_set:nn { WithArrows / WithArrows } { #1 } }
782 \bool_if:NT \l_@@_in_DispWithArrows_bool
783 { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

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

```
784 \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_first_columns:
```

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

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

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

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

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

```
786 \int_gset_eq:NN \g_@@_col_int \l_@@_nb_cols_int
```

---

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

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.

```

787 \seq_clear_new:N \l_@@_format_seq
788 \seq_set_split:NnV \l_@@_format_seq { } \l_@@_format_str

```

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

```

789 <*LaTeX>
790 \bool_if:NT \g_@@_footnote_bool { \begin { savenotes } }
791 </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`).

```

792 \l_@@_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).

```

793 \spread@equation
794 <*LaTeX>
795 \cs_set_eq:NN \notag \@@_notag:
796 \cs_set_eq:NN \nonumber \@@_nonumber:
797 \cs_set_eq:NN \tag \@@_tag
798 \cs_set_eq:NN \@@_old_label \label
799 \cs_set_eq:NN \label \@@_label:n
800 \cs_set_eq:NN \tagnextline \@@_tagnextline:
801 </LaTeX>
802 }

```

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

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

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

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

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

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

```

803 \cs_new_protected:Npn \@@_construct_halign:
804 {
805   \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
806   {

```

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.

```

807   \use:x
808   {

```

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.

```

809       \int_gdecr:N \g_@@_col_int
810       \@@_construct_halign:
811       \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
812       {

```

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`.

```

813         \cs_set_eq:cn { \l_@@_command_name_str } \@@_Arrow
814   < *LaTeX >
815         \bool_if:NT \l_@@_in_DispWithArrows_bool
816         {

```

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).

```

817         \@@_test_if_to_tag:

```

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

```

818         \bool_if:NT \c_@@_amsthm_loaded_bool \@@_set_qedhere:
819       }
820 < /LaTeX >
821   }
822   \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
823   \str_if_eq:VnT \l_@@_type_col_str { r } \hfill
824   \int_gincr:N \g_@@_col_int
825   \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
826   \c_math_toggle_token
827   {
828     { }
829     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
830     ####
831   }
832   \c_math_toggle_token
833   \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
834   { \@@_construct_nodes: }
835   {

```

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:`).

```

836       \str_if_eq:VnT \l_@@_type_col_str { l } \hfil
837       \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
838       \bool_if:NT \l_@@_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
839     &
840   }
841 }
842 }

```

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

```

843 {
844   \bool_if:NTF \l_@@_in-WithArrows_bool
845   {
846     \ialign
847     \bgroup
848   }
849   {
850     \halign to \l_@@_linewidth_dim
851     \bgroup
852     \bool_if:NT \l_@@_fleqn_bool
853     { \skip_horizontal:N \l_@@_mathindent_dim }
854   }
855   \int_gincr:N \g_@@_line_int
856   \int_gzero:N \g_@@_col_int
857   \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
858   {
859     \skip_horizontal:n
860     { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
861   }
862   \strut
863 }
864 }
```

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.

```

865 \cs_new_protected:Npn \@@_construct_nodes:
866 {
```

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).

```

867   \tikz [ remember~picture , overlay ]
868   \node
869   [
870     node~contents = { } ,
871     @@_node_style ,
872     name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
873     alias =
874     {
875       \str_if_empty:NF \l_@@_name_str
876       { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
877     }
878   ]
879   ;
880   \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).

```

881   \tikz [ remember~picture , overlay ]
882   \node
883   [
884     node~contents = { } ,
885     @@_node_style ,
886     name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
887     alias =
888     {
889       \str_if_empty:NF \l_@@_name_str
890       { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
891     }
892   ]
893   ;
```

```

894 \bool_if:NT \l_@@_show_node_names_bool
895 {
896     \hbox_overlap_right:n
897     { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
898 }
899 }

```

### 11.8.3 The environment `{WithArrows}`

```

900 <*LaTeX>
901 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
902 </LaTeX>
903 <*plain-TeX>
904 \cs_new_protected:Npn \WithArrows
905 {
906     \group_begin:
907     \peek_meaning:NTF [
908     { \WithArrows_i }
909     { \WithArrows_i [ ] }
910 }
911 \cs_new_protected:Npn \WithArrows_i [ #1 ]
912 </plain-TeX>
913 {
914     \bool_set_true:N \l_@@_in-WithArrows_bool
915     \bool_set_false:N \l_@@_in-DispWithArrows_bool
916 <*plain-TeX>
917     \str_clear_new:N \l_@@_type_env_str
918     \str_set:Nn \l_@@_type_env_str { WithArrows }
919 </plain-TeX>
920     \@@_pre_halign:n { #1 }
921     \if_mode_math: \else:
922         \@@_error:n { WithArrows~outside~math~mode }
923     \fi:

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`<sup>32</sup> depending of the value of `\l_@@_pos_env_int` (fixed by the options `t`, `c` or `b`). The environment `{WithArrows}` must be used in math mode<sup>33</sup> and therefore, we can use `\vcenter`.

```

924     \int_case:nn \l_@@_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
925     \bgroup

```

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

```

926     \@@_construct_halign:

```

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

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

```

927     &&
928     \@@_error:n { Too~much~columns~in~WithArrows }
929     \c_math_toggle_token
930     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
931     { ## }
932     \c_math_toggle_token
933     \cr
934 }

```

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

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

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`).

```

935 <*plain-TeX>
936 \cs_new_protected:Npn \endWithArrows
937 </plain-TeX>
938 {
939     \\\
940     \egroup
941     \egroup
942     \@@_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`).

```

943 <*LaTeX>
944     \bool_if:NT \g_@@_footnote_bool { \end { savenotes } }
945 </LaTeX>
946 <*plain-TeX>
947     \group_end:
948 </plain-TeX>
949 }

```

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}`.

```

950 \cs_new_protected:Npn \@@_post_halign:

```

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

```

951 {
952     \cs_set:Npn \WithArrowsRightX { \g_@@_right_x_dim }

```

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

```

953     \normalbaselines

```

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

```

954     \int_compare:nNnT \g_@@_arrow_int > 0
955     {

```

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).

```

956         \int_compare:nNnT \g_@@_arrow_int = 1
957         {
958             \int_compare:nNnT \l_@@_pos_arrow_int > 5
959             { \int_set:Nn \l_@@_pos_arrow_int 5 }
960         }
961         \@@_scan_arrows:
962     }

```

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

```

963     \group_begin:
964     \tikzset { every-picture / .style = @@_standard }

```

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

```

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

```

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

```

966 \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
967 \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
968 \bool_set_true:N \l_@@_in_code_after_bool
969 \l_@@_code_after_tl
970 \group_end:

```

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

```

971 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
972 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
973 \seq_gput_right:Nx \g_@@_position_in_the_tree_seq
974 { \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`.

```

975 \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
976 { \int_gincr:N \g_@@_last_env_int }

```

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

```

977 \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
978 \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
979 \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
980 \int_gset:Nn \g_@@_line_int \l_tmpa_tl
981 \seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
982 \int_gset:Nn \g_@@_col_int \l_tmpa_tl
983 \seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
984 \int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
985 }

```

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).

```

986 \cs_new_protected:Npn \@@_cr:
987 {
988 \scan_stop:

```

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

```

989 \int_compare:nNnF \g_@@_col_int = \g_@@_static_col_int
990 { \@@_error:n { omit~probably~used } }
991 \prg_replicate:nn { \l_@@_nb_cols_int - \g_@@_static_col_int } { & { } }
992 \group_align_safe_begin:
993 \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
994 }

```

<sup>34</sup>As 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.

```

995 \cs_new_protected:Npn \@@_cr_i:
996   { \peek_meaning:NTF [ \@@_cr_ii: { \@@_cr_ii: [ \c_zero_dim ] } }

```

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

```

997 <*LaTeX>
998 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
999   {
1000     \peek_meaning_ignore_spaces:NTF \end
1001     {
1002       \@@_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`.

```

1003       \@@_analyze_end:Nn
1004       }
1005       { \@@_cr_iii:n { #1 } }
1006     }
1007 \cs_new_protected:Npn \@@_cr_iii:n #1
1008 </LaTeX>
1009 <*plain-TeX>
1010 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1011 </plain-TeX>
1012   {
1013     \group_align_safe_end:

```

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

```

1014     \bool_if:NT \l_@@_in_DispWithArrows_bool

```

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

```

1015     {
1016 <*LaTeX>
1017       \clist_if_in:NnTF \l_@@_tags_clist { all }
1018       {

```

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).

```

1019       \tl_if_empty:NT \l_@@_tag_tl { \int_gincr:N \c@equation }

```

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

```

1020       \cs_gset:Npx \g_tmpa_tl
1021       { \tl_if_empty:NTF \l_@@_tag_tl \theequation \l_@@_tag_tl }

```

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

```

1022       \seq_if_empty:NF \l_@@_labels_seq
1023       {

```

---

<sup>35</sup>The 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`.

```
1024 \cs_set:Npx \c_currentlabel { \p@equation \g_tmpa_tl }
```

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

```
1025 \bool_if:NT \c_@@_hyperref_loaded_bool
1026 {
1027   \str_set:Nn \This@name { equation }
1028   \hyper@refstepcounter { equation }
1029 }
```

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 `\c_currentlabel` but with more informations.

```
1030 \bool_if:NT \c_@@_cleveref_loaded_bool
1031 {
1032   \cref@constructprefix { equation } \cref@result
1033   \protected@edef \cref@currentlabel
1034   {
1035     [
1036       \cs_if_exist:NTF \cref@equation@alias
1037       \cref@equation@alias
1038       { equation }
1039     ]
1040     [ \arabic { equation } ] [ \cref@result ]
1041     \p@equation \g_tmpa_tl
1042   }
1043 }
```

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`).

```
1044 \seq_map_function:NN \l_@@_labels_seq \@@_old_label
1045 }
```

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.

```
1046 \@@_save:N \l_@@_tag_star_bool
1047 \@@_save:N \l_@@_qedhere_bool
1048 \bool_if:NT \l_@@_tag_next_line_bool
1049 {
1050   \openup -\jot
1051   \bool_set_false:N \l_@@_tag_next_line_bool
1052   \notag \&
1053 }
1054 &
1055 \@@_restore:N \l_@@_tag_star_bool
1056 \@@_restore:N \l_@@_qedhere_bool
1057 \bool_if:NT \l_@@_qedhere_bool
1058 { \hbox_overlap_left:n \@@_qedhere_i: }
1059 \cs_set_eq:NN \theequation \g_tmpa_tl
1060 \bool_if:NT \l_@@_tag_star_bool
1061 { \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`.

```
1062 \hbox_overlap_left:n
1063 {
1064   \bool_if:NF \c_@@_leqno_bool
1065   {
1066     \tikz [ @@@_standard ]
1067       \coordinate ( \int_use:N \g_@@_line_int - v ) ;
```

```

1068         }
1069         \quad
1070         \@eqnnum
1071     }
1072     \bool_if:NT \c_@@_leqno_bool
1073     {
1074         \tikz [ @@_standard ]
1075             \coordinate ( \int_use:N \g_@@_line_int - v ) ;
1076     }
1077 }
1078 {
1079     \@@_save:N \l_@@_qedhere_bool
1080 </LaTeX>
1081 &
1082 <*LaTeX>
1083     \@@_restore:N \l_@@_qedhere_bool
1084     \bool_if:NT \l_@@_qedhere_bool
1085     { \hbox_overlap_left:n \@@_qedhere_i: }
1086 </LaTeX>
1087     \tikz [ @@_standard ]
1088         \coordinate ( \int_use:N \g_@@_line_int - v ) ;
1089 <*LaTeX>
1090 }
1091 </LaTeX>
1092 }
1093 \dim_compare:nNnT { #1 } < \c_zero_dim
1094 { \@@_error:n { option~of~cr~negative } }
1095
1096 \cr
1097 \noalign
1098 {
1099     \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1100     \skip_vertical:n { \l_tmpa_dim + \l_@@_interline_skip }
1101     \scan_stop:
1102 }
1103 }

```

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.

```

1104 <*LaTeX>
1105 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1106 {
1107     \exp_args:NV \str_if_eq:nnT \l_@@_type_env_str { #2 }
1108     {
1109         \@@_error:n { newline~at~the~end~of~env }
1110         \group_begin:
1111         \globaldefs = 1
1112         \@@_msg_redirect_name:nn { newline~at~the~end~of~env } { none }
1113         \group_end:
1114     }

```

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

```

1115     \end { #2 }
1116 }
1117 </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}`.

```

1118 <*LaTeX>
1119 \bool_new:N \l_@@_in_label_or_minipage_bool
1120 </LaTeX>

1121 <*LaTeX>
1122 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1123 </LaTeX>
1124 <*plain-TeX>
1125 \cs_new_protected:Npn \DispWithArrows
1126 {
1127   \group_begin:
1128   \peek_meaning:NTF <
1129     { \DispWithArrows_i }
1130     { \DispWithArrows_i < \c_novalue_tl > }
1131 }
1132 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1133 {
1134   \peek_meaning:NTF [
1135     { \DispWithArrows_ii < #1 > }
1136     { \DispWithArrows_ii < #1 > [ ] }
1137 }
1138 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1139 </plain-TeX>
1140 {
1141   \bool_set_true:N \l_@@_in_DispWithArrows_bool
1142 <*plain-TeX>
1143   \str_clear_new:N \l_@@_type_env_str
1144   \str_set:Nn \l_@@_type_env_str { DispWithArrows }
1145 </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).

```

1146 <*LaTeX>
1147 \bool_if:nT \c_@@_mathtools_loaded_bool
1148 {
1149   \MH_if_boolean:nT { show_only_refs }
1150   {
1151     \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`.

```

1152     \MH_set_boolean:T:n { show_only_refs }
1153   }
1154 }

```

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`.

```

1155 \bool_if:NT \c_@@_typedref_loaded_bool { \str_set:Nn \sr@name { equation } }

```

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

```

1156 \bool_if:NT \c_@@_amsmath_loaded_bool \intertext@
1157 </LaTeX>
1158 \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l_@@_left_brace_tl { #1 } }

```

```
1159 \@@_pre_halign:n { #2 }
```

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

```
1160 (*LaTeX)
1161 \bool_if:NT \l_@@_subequations_bool { \begin { subequations } }
1162 </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`.

```
1163 (*LaTeX)
1164 \bool_if:NF \l_@@_sbwi_bool
1165 {
1166   \if@inlabel
1167     \bool_set_true:N \l_@@_in_label_or_minipage_bool
1168   \fi
1169   \if@minipage
1170     \bool_set_true:N \l_@@_in_label_or_minipage_bool
1171   \fi
1172 }
1173 </LaTeX>

1174 \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
1175 {
```

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

```
1176 \hbox_set:Nn \l_tmpa_box
1177 {
```

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

```
1178 \group_begin:
1179 \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1180 \c_math_toggle_token
1181 \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1182 \c_math_toggle_token
1183 \group_end:
1184 }
1185 \dim_zero_new:N \l_@@_delim_wd_dim
1186 \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
1187 \box_clear_new:N \l_@@_left_brace_box
1188 \hbox_set:Nn \l_@@_left_brace_box
1189 {
1190   \group_begin:
1191     \cs_set_eq:NN \label \@@_old_label
1192     \c_math_toggle_token
1193     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
1194     \l_@@_left_brace_tl
1195     { }
1196     \c_math_toggle_token
1197   \group_end:
1198 }
1199 }
```

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

```
1200 (*LaTeX)
1201 \tl_clear_new:N \l_@@_tag_tl

1202 \bool_set_false:N \l_@@_qedhere_bool
```

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

```
1203 \bool_set_false:N \l_@@_tag_star_bool
1204 </LaTeX>
```

```

1205 \if_mode_math:
1206 \@@_fatal:n { DispWithArrows~in~math~mode }
1207 \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`.

```

1208 <*plain-TeX>
1209 \dim_zero_new:N \linewidth
1210 \dim_set_eq:NN \linewidth \displaywidth
1211 </plain-TeX>
1212 <*LaTeX>
1213 \bool_if:NTF \l_@@_in_label_or_minipage_bool
1214 { \c_math_toggle_token }
1215 {
1216 </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.

```

1217 \if_mode_vertical:
1218 \nointerlineskip
1219 \hbox_to_wd:nn { .6 \linewidth } { }
1220 \fi:
1221 \c_math_toggle_token \c_math_toggle_token
1222 <*LaTeX>
1223 }
1224 </LaTeX>

1225 \dim_zero_new:N \l_@@_linewidth_dim
1226 <*LaTeX>
1227 \bool_if:NTF \l_@@_in_label_or_minipage_bool
1228 { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
1229 { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
1230 </LaTeX>
1231 <*plain-TeX>
1232 \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
1233 </plain-TeX>

1234 \box_clear_new:N \l_@@_halign_box
1235 \setbox \l_@@_halign_box \vtop \bgroup
1236 \tabskip =
1237 \bool_if:NTF \l_@@_fleqn_bool
1238 \c_zero_skip
1239 { 0 pt plus 1000 pt minus 1000 pt }

1240 \@@_construct_halign:
1241 \tabskip = 0 pt plus 1000 pt minus 1000 pt
1242 &

```

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.

```

1243 $ ## $
1244 \tabskip = \c_zero_skip
1245 &&
1246 \@@_fatal:n { Too~much~columns~in~DispWithArrows }
1247 \bool_if:nT \c_false_bool { ## }
1248 \cr
1249 }

```

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

```

1250 <*plain-TeX>
1251 \cs_new_protected:Npn \endDispWithArrows

```

```

1252 </plain-TeX>
1253 {
1254 (*LaTeX)
1255   \clist_if_in:NnT \l_@@_tags_clist { last }
1256   { \clist_set:Nn \l_@@_tags_clist { all } }
1257 </LaTeX>
1258 \}

```

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

```

1259 \egroup
1260 \unskip \unpenalty \unskip \unpenalty
1261 \box_set_to_last:N \l_tmpa_box
1262 \nointerlineskip
1263 \box_use:N \l_tmpa_box
1264 \dim_gzero_new:N \g_@@_alignment_dim
1265 \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
1266 \box_clear_new:N \l_@@_new_box
1267 \hbox_set:Nn \l_@@_new_box { \hbox_unpack_clear:N \l_tmpa_box }
1268 \dim_compare:nNnT
1269   { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
1270   { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }

```

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

```

1271 \egroup
1272 \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
1273   { \box_use_drop:N \l_@@_halign_box }
1274   {
1275     \hbox_to_wd:nn \l_@@_linewidth_dim
1276     {
1277       \bool_if:NTF \l_@@_fleqn_bool
1278       { \skip_horizontal:n \l_@@_mathindent_dim }
1279       \hfil
1280       \hbox_to_wd:nn \g_@@_alignment_dim
1281       {
1282         \box_use_drop:N \l_@@_left_brace_box
1283         \dim_set:Nn \l_tmpa_dim
1284         {
1285           \box_ht:N \l_@@_halign_box
1286           + \box_dp:N \l_@@_halign_box
1287         }
1288         \group_begin:
1289         \dim_set_eq:NN \nullldelimiterspace \c_zero_dim
1290         \c_math_toggle_token
1291         \left \l_@@_replace_left_brace_by_tl
1292         \vcenter to \l_tmpa_dim { \vfil }
1293         \right.
1294         \c_math_toggle_token
1295         \group_end:
1296         \hfil
1297       }
1298       \hfil
1299     }
1300     \skip_horizontal:n { - \l_@@_linewidth_dim }
1301     \vcenter { \box_use_drop:N \l_@@_halign_box }
1302   }

```

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.

```

1303 \dim_gzero_new:N \g_@@_right_x_dim
1304 \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
1305 (*LaTeX)

```

```

1306 \begin { tikzpicture } [ @@_standard ]
1307 </LaTeX>
1308 <*plain-TeX>
1309 \tikzpicture [ @@_standard ]
1310 </plain-TeX>
1311 \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
1312 {
1313 \cs_if_free:cTF
1314 { pgf@sh@ns@wa - \l_@@_prefix_str - \l_tmpa_int - v }
1315 { \@@_fatal:n { Inexistent~v-node } }
1316 {
1317 \tikz@parse@node\pgfutil@firstofone ( \l_tmpa_int - v )
1318 \dim_set:Nn \l_tmpa_dim \pgf@x
1319 \dim_compare:nNnT \l_tmpa_dim < \g_@@_right_x_dim
1320 { \dim_gset:Nn \g_@@_right_x_dim \l_tmpa_dim }
1321 }
1322 }
1323 <*LaTeX>
1324 \end { tikzpicture }
1325 </LaTeX>
1326 <*plain-TeX>
1327 \endtikzpicture
1328 </plain-TeX>

```

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

```

1329 \@@_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).

```

1330 <*LaTeX>
1331 \bool_if:nT \c_@@_mathtools_loaded_bool
1332 { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1333 \bool_if:NTF \l_@@_in_label_or_minipage_bool
1334 {
1335 \c_math_toggle_token
1336 \skip_vertical:N \belowdisplayskip
1337 }
1338 { \c_math_toggle_token \c_math_toggle_token }
1339 </LaTeX>
1340 <*plain-TeX>
1341 \c_math_toggle_token \c_math_toggle_token
1342 </plain-TeX>
1343 <*LaTeX>
1344 \bool_if:NT \l_@@_subequations_bool { \end { subequations } }

```

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

```

1345 \bool_if:NT \g_@@_footnote_bool { \end { savenotes } }
1346 </LaTeX>
1347 <*plain-TeX>
1348 \group_end:
1349 </plain-TeX>
1350 <*LaTeX>
1351 \ignorespacesafterend
1352 </LaTeX>
1353 }

```

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



```

1354 <*LaTeX>
1355 \NewDocumentEnvironment { DispWithArrows* } { }
1356 {
1357     \WithArrowsOptions { notag }
1358     \DispWithArrows
1359 }
1360 \endDispWithArrows
1361 </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.

```

1362 \cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
1363 {
1364     \bool_if:NTF \l_@@_in-WithArrows_bool
1365     { \@@_error:nn { Not~allowed~in~WithArrows } { #1 } }
1366     {
1367         \int_compare:nNnTF \g_@@_col_int < \l_@@_nb_cols_int
1368         { \@@_error:nn { Not~allowed~in~DispWithArrows } { #1 } }
1369         { #2 }
1370     }
1371 }

```

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

```

1372 <*LaTeX>
1373 \cs_new_protected:Npn \@@_notag:
1374 { \@@_if_in_last_col_of_disp:Nn \notag { \clist_clear:N \l_@@_tags_clist } }

```

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

```

1375 \cs_new_protected:Npn \@@_nonumber:
1376 { \@@_if_in_last_col_of_disp:Nn \nonumber { \clist_clear:N \l_@@_tags_clist } }

```

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

```

1377 \NewDocumentCommand \@@_tag { s m }
1378 {
1379     \@@_if_in_last_col_of_disp:Nn \tag
1380     {
1381         \tl_if_empty:NF \l_@@_tag_tl
1382         { \@@_error:nn { Multiple~tags } { #2 } }
1383         \clist_set:Nn \l_@@_tags_clist { all }
1384         \bool_if:nT \c_@@_mathtools_loaded_bool
1385         {
1386             \MH_if_boolean:nT { show_only_refs }
1387             {
1388                 \MH_if_boolean:nF { show_manual_tags }
1389                 { \clist_clear:N \l_@@_tags_clist }
1390             }
1391         }
1392         \tl_set:Nn \l_@@_tag_tl { #2 }
1393         \bool_set:Nn \l_@@_tag_star_bool { #1 }

```

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

```

1394     \bool_if:nT { #1 && ! \bool_if_p:N \c_@@_amsmath_loaded_bool }
1395     { \@@_error:n { tag*~without~amsmath } }
1396   }
1397 }

```

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

```

1398 \cs_new_protected:Npn \@@_label:n #1
1399 {
1400   \@@_if_in_last_col_of_disp:Nn \label
1401   {
1402     \seq_if_empty:NF \l_@@_labels_seq
1403     {
1404       \bool_if:NTF \c_@@_cleveref_loaded_bool
1405       { \@@_error:n { Multiple~labels~with~cleveref } }
1406       { \@@_error:n { Multiple~labels } }
1407     }
1408     \seq_put_right:Nn \l_@@_labels_seq { #1 }
1409     \bool_if:nT \c_@@_mathtools_loaded_bool
1410     {
1411       \MH_if_boolean:nT { show_only_refs }
1412       {
1413         \cs_if_exist:cTF { MT_r_#1 }
1414         { \clist_set:Nn \l_@@_tags_clist { all } }
1415         { \clist_clear:N \l_@@_tags_clist }
1416       }
1417     }
1418     \bool_if:nT \c_@@_autonum_loaded_bool
1419     {
1420       \cs_if_exist:cTF { autonum@#1Referenced }
1421       { \clist_set:Nn \l_@@_tags_clist { all } }
1422       { \clist_clear:N \l_@@_tags_clist }
1423     }
1424   }
1425 }

```

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

```

1426 \cs_new_protected:Npn \@@_tagnextline:
1427 {
1428   \@@_if_in_last_col_of_disp:Nn \tagnextline
1429   { \bool_set_true:N \l_@@_tag_next_line_bool }
1430 }

```

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

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

```

1431 \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }
1432 \cs_new_protected:Npn \@@_set_qedhere: { \cs_set_eq:NN \qedhere \@@_qedhere: }

```

---

<sup>36</sup>There 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`.

```

1433 \cs_new_protected:Npn \@@qedhere_i:
1434 {
1435   \group_begin:
1436   \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.

```

1437   \cs_set_eq:NN \qed@elt \setQED@elt
1438   \QED@stack \relax \relax
1439   \group_end:
1440 }
1441 </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.

```

1442 \cs_new_protected:Npn \@@scan_arrows:
1443 {
1444   \group_begin:
1445   \int_compare:nNnT \l_@@_pos_arrow_int = 7
1446   {
1447     \@@scan_arrows_i:
1448     \int_set:Nn \l_@@_pos_arrow_int 8
1449   }
1450   \@@scan_arrows_i:
1451   \group_end:
1452 }

1453 \cs_new_protected:Npn \@@scan_arrows_i:
1454 {

```

`\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).

```

1455   \int_zero_new:N \l_@@_first_arrow_of_group_int
1456   \int_zero_new:N \l_@@_first_line_of_group_int
1457   \int_zero_new:N \l_@@_last_line_of_group_int
1458   \seq_clear_new:N \l_@@_first_arrows_seq
1459   \seq_clear_new:N \l_@@_last_arrows_seq

```

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

```
1460 \bool_set_true:N \l_@@_new_group_bool
```

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

```
1461 \int_set:Nn \l_@@_arrow_int \c_one_int
1462 \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1463 {
```

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.

```
1464 \prop_get:cnN
1465 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1466 { initial } \l_tmpa_tl
1467 \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1468 \prop_get:cnN
1469 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1470 { final } \l_tmpa_tl
1471 \int_set:Nn \l_@@_final_int \l_tmpa_tl
1472 \prop_get:cnN
1473 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1474 { status } \l_@@_status_arrow_str
1475 \prop_get:cnN
1476 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1477 { input-line } \l_@@_input_line_str
```

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

```
1478 \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1479 {
1480 \int_compare:nNnF \l_@@_pos_arrow_int = 8
1481 { \@@_error:n { Too-few-lines-for-an-arrow } }
1482 }
1483 \@@_code_for_possible_arrow:
```

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

```
1484 \int_incr:N \l_@@_arrow_int
1485 }
```

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).

```
1486 \bool_if:nT
1487 {
1488 \int_compare_p:n { \l_@@_pos_arrow_int != 7 }
1489 &&
1490 \int_compare_p:nNn \l_@@_first_arrow_of_group_int > 0
1491 }
1492 { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1493 }
```

```
1494 \cs_new_protected:Npn \@@_code_for_possible_arrow:
1495 {
```

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`).

```

1496   \bool_if:nT
1497   {
1498     \int_compare_p:nNn \l_@@_arrow_int > \c_one_int
1499     &&
1500     ( \int_compare_p:n { \l_@@_initial_int > \l_@@_last_line_of_group_int }
1501     &&
1502     \int_compare_p:n { \l_@@_pos_arrow_int != 7 }
1503     ||
1504     \str_if_eq_p:Vn \l_@@_status_arrow_str { new-group }
1505   )
1506   }
1507   {
1508     \int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
1509     {
1510       \@@_draw_arrows:nn
1511       \l_@@_first_arrow_of_group_int
1512       { \l_@@_arrow_int - 1 }
1513     }
1514     \bool_set_true:N \l_@@_new_group_bool
1515   }

```

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`.

```

1516   \bool_if:nTF \l_@@_new_group_bool
1517   {
1518     \bool_set_false:N \l_@@_new_group_bool
1519     \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1520     \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1521     \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1522     \seq_clear:N \l_@@_first_arrows_seq
1523     \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
1524     \seq_clear:N \l_@@_last_arrows_seq
1525     \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int

```

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

```

1526     \int_compare:nT { \l_@@_pos_arrow_int != 8 }
1527     { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1528   }

```

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

```

1529   {

```

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

```

1530     \bool_if:nF
1531     { \str_if_eq_p:Vn \l_@@_status_arrow_str { independent } }
1532     {

```

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

```

1533         \int_compare:nT
1534         { \l_@@_initial_int = \l_@@_first_line_of_group_int }

```

```

1535         { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1536     \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1537     {
1538         \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1539         \seq_clear:N \l_@@_last_arrows_seq
1540         \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
1541     }
1542     {
1543         \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1544         { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1545     }
1546 }
1547 }

```

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

```

1548     \bool_if:nF { \str_if_eq_p:Vn \l_@@_status_arrow_str { independent } }
1549     {
1550         \int_compare:nT { \l_@@_pos_arrow_int != 8 }
1551         { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1552     }
1553 }

```

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

```

1554 \cs_generate_variant:Nn \keys_set:nn { n o }
1555 \cs_new_protected:Npn \@@_keys_set:
1556 { \keys_set_known:no { WithArrows / Arrow / SecondPass } }

```

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

```

1557 \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1558 {
1559     \group_begin:
1560     \int_zero_new:N \l_@@_first_arrow_int
1561     \int_set:Nn \l_@@_first_arrow_int { #1 }
1562     \int_zero_new:N \l_@@_last_arrow_int
1563     \int_set:Nn \l_@@_last_arrow_int { #2 }

```

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

```

1564     \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1565     \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1566     {

```

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.

```

1567         \prop_get:cnN
1568         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1569         { initial } \l_tmpa_tl
1570     \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1571     \prop_get:cnN
1572     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1573     { final } \l_tmpa_tl
1574     \int_set:Nn \l_@@_final_int \l_tmpa_tl

```

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

```

1575     \int_compare:nT { \l_@@_final_int <= \g_@@_line_int } \@@_draw_arrows_i:
1576     \int_incr:N \l_@@_arrow_int
1577   }
1578   \group_end:
1579 }

```

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).

```

1580 \cs_new_protected:Npn \@@_draw_arrows_i:
1581 {
1582   \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.

```

1583   \prop_get:cnN
1584     { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1585     { options } \l_tmpa_tl
1586   \str_clear_new:N \l_@@_previous_key_str
1587   \exp_args:NNo \exp_args:No
1588   \@@_keys_set: { \l_tmpa_tl , tikz = { xshift = \l_@@_xoffset_dim } }

```

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

```

1589   \bool_set_false:N \l_@@_initial_r_bool
1590   \bool_set_false:N \l_@@_final_r_bool
1591   \int_case:nn \l_@@_pos_arrow_int
1592   {
1593     0 { \bool_set_true:N \l_@@_final_r_bool }
1594     2 { \bool_set_true:N \l_@@_initial_r_bool }
1595     3
1596     {
1597       \bool_set_true:N \l_@@_initial_r_bool
1598       \bool_set_true:N \l_@@_final_r_bool
1599     }
1600   }

```

option	lr	ll	rl	rr	v	i	groups	group
\l_@@_pos_arrow_int	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).

```

1601   \int_compare:nNnT \l_@@_pos_arrow_int = 5
1602   {
1603     \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1604     \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1605   }

```

`\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`.

```

1606 \tl_set:Nx \l_@@_initial_tl
1607 {
1608     \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl
1609     .south
1610 }
1611 \tl_set:Nx \l_@@_final_tl
1612 { \int_use:N \l_@@_final_int - \bool_if:NTF \l_@@_final_r_bool rl .north }

```

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

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

```

1613 \prop_get:cnN
1614 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1615 { label }
1616 \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.

```

1617 \seq_if_in:NxTF \l_@@_first_arrows_seq
1618 { \int_use:N \l_@@_arrow_int }
1619 { \bool_set_true:N \l_tmpa_bool }
1620 { \bool_set_false:N \l_tmpa_bool }
1621 \seq_if_in:NxTF \l_@@_last_arrows_seq
1622 { \int_use:N \l_@@_arrow_int }
1623 { \bool_set_true:N \l_tmpb_bool }
1624 { \bool_set_false:N \l_tmpb_bool }
1625 \int_compare:nNnT \l_@@_pos_arrow_int = 5
1626 {
1627     \bool_set_true:N \l_tmpa_bool
1628     \bool_set_true:N \l_tmpb_bool
1629 }

```

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

- Concerning the  $x$ -values, the abscissa computed in `\l_@@_x_dim` will be used if the option of position is `i`, `group` or `groups`.
- Concerning the  $y$ -values, an adjustment is done for each arrow starting at the first line of the group and each arrow ending at the last line of the group (with the values of `\l_@@_start_adjust_dim` and `\l_@@_end_adjust_dim`).

```

1630 \dim_gzero_new:N \g_@@_x_initial_dim
1631 \dim_gzero_new:N \g_@@_x_final_dim
1632 \dim_gzero_new:N \g_@@_y_initial_dim
1633 \dim_gzero_new:N \g_@@_y_final_dim
1634 \begin{tikzpicture} [ @@_standard ]
1635 \begin{tikzpicture} [ @@_standard ]
1636 \end{tikzpicture}
1637 \end{tikzpicture}
1638 \begin{tikzpicture} [ @@_standard ]
1639 \end{tikzpicture}
1640 \tikzscan@one@point \pgfutil@firstofone ( \l_@@_initial_tl )
1641 \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1642 \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1643 \tikzscan@one@point \pgfutil@firstofone ( \l_@@_final_tl )
1644 \dim_gset:Nn \g_@@_x_final_dim \pgf@x

```



```

1645 \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1646 \<LaTeX>
1647 \end { tikzpicture }
1648 \</LaTeX>
1649 \<plain-TeX>
1650 \endtikzpicture
1651 \</plain-TeX>
1652 \bool_if:nTF
1653 { \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
1654 > \l_@@_max_length_of_arrow_dim
1655 &&
1656 \int_compare_p:nNn { \l_@@_final_int - \l_@@_initial_int } = 1
1657 }
1658 {
1659 \tl_gset:Nx \g_tmpa_tl
1660 {
1661 \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1662 { \dim_use:N \g_@@_x_initial_dim }
1663 { \dim_use:N \l_@@_x_dim } ,
1664 \dim_eval:n
1665 {
1666 ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1667 + ( \l_@@_max_length_of_arrow_dim / 2 )
1668 }
1669 }
1670 \tl_gset:Nx \g_tmpb_tl
1671 {
1672 \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1673 { \dim_use:N \g_@@_x_final_dim }
1674 { \dim_use:N \l_@@_x_dim } ,
1675 \dim_eval:n
1676 {
1677 ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1678 - ( \l_@@_max_length_of_arrow_dim / 2 )
1679 }
1680 }
1681 }
1682 {
1683 \tl_gset:Nx \g_tmpa_tl
1684 {
1685 \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1686 { \dim_use:N \g_@@_x_initial_dim }
1687 { \dim_use:N \l_@@_x_dim } ,
1688 \bool_if:NTF \l_tmpa_bool
1689 { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
1690 { \dim_use:N \g_@@_y_initial_dim }
1691 }
1692 \tl_gset:Nx \g_tmpb_tl
1693 {
1694 \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1695 { \dim_use:N \g_@@_x_final_dim }
1696 { \dim_use:N \l_@@_x_dim } ,
1697 \bool_if:NTF \l_tmpb_bool
1698 { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
1699 { \dim_use:N \g_@@_y_final_dim }
1700 }
1701 }

```

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

third argument will be in the argument of a command `node` of an instruction `\draw` of Tikz... and we will have an error because of the characters *underscore*.<sup>37</sup>

```
1702 \@@_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).

```
1703 \group_end:
1704 }
```

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`.

```
1705 \cs_new_protected:Npn \@@_def_function_tmpa:n #1
1706 {
1707   \cs_set:Npn \@@_tmpa:nnn ##1 ##2 ##3
1708   {
1709     \LaTeX
1710     \begin{tikzpicture}
1711     \LaTeX
1712     \plain-TeX
1713     \tikzpicture
1714     \plain-TeX
1715     [
1716       @@_standard ,
1717       every~path / .style = WithArrows / arrow
1718     ]
1719     #1
1720     \LaTeX
1721     \end{tikzpicture}
1722     \LaTeX
1723     \plain-TeX
1724     \endtikzpicture
1725     \plain-TeX
1726   }
1727 }
```

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

```
1728 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1729 {
```

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`).

```
1730 \bool_if:nT { \l_@@_wrap_lines_bool && \l_@@_in_DispWithArrows_bool }
1731 { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }
```

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

```
1732 \exp_args:NV \@@_def_function_tmpa:n \l_@@_tikz_code_tl
1733 \@@_tmpa:nnn { #1 } { #2 } { #3 }
1734 }
1735 \cs_generate_variant:Nn \@@_draw_arrow:nnn { n n o }
```

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

```
1736 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1737 {
```

---

<sup>37</sup>There 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.

```
1738 \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;
```

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

```
1739 \tikz@parse@node \pgfutil@firstofone ( @@_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.

```
1740 \dim_set:Nn \l_tmpa_dim
1741 { \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”.<sup>38</sup>

```
1742 \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`).

```
1743 \tl_if_empty:NF \g_tmpa_tl
1744 {
1745   \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1746   \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1747     { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1748 }
```

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

```
1749 \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1750 {
1751   \path ( @@_label.west )
1752     node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1753       { #3 } ;
1754 }
1755 }
```

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`).

```
1756 \cs_new_protected:Npn \@@_update_x:nn #1 #2
1757 {
1758   \int_step_inline:nnn { #1 } { #2 }
1759   {
1760     \*LaTeX
1761     \begin { tikzpicture } [ @@_standard ]
1762     \*LaTeX
1763     \*plain-TeX
1764     \tikzpicture [ @@_standard ]
1765     \*plain-TeX
1766     \tikz@scan@one@point \pgfutil@firstofone ( ##1 - 1 )
1767     \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \l_@@_x_dim \pgf@x }
1768     \*LaTeX
1769     \end { tikzpicture }
1770     \*LaTeX
1771     \*plain-TeX
1772     \endtikzpicture
1773     \*plain-TeX
1774     \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1775   }
1776 }
```

---

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

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.

```
1777 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_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`.

```
1778 \keys_define:nn { WithArrows / Arrow / code-after }
1779 {
1780   tikz      .code:n =
1781     \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
1782   tikz      .value_required:n = true ,
1783   rr        .value_forbidden:n = true ,
1784   rr        .code:n = \@@_fix_pos_option:n 0 ,
1785   ll        .value_forbidden:n = true ,
1786   ll        .code:n = \@@_fix_pos_option:n 1 ,
1787   rl        .value_forbidden:n = true ,
1788   rl        .code:n = \@@_fix_pos_option:n 2 ,
1789   lr        .value_forbidden:n = true ,
1790   lr        .code:n = \@@_fix_pos_option:n 3 ,
1791   v         .value_forbidden:n = true ,
1792   v         .code:n = \@@_fix_pos_option:n 4 ,
1793   tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
1794   tikz-code .value_required:n = true ,
1795   xoffset   .dim_set:N = \l_@@_xoffset_dim ,
1796   xoffset   .value_required:n = true ,
1797   unknown   .code:n =
1798     \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
1799     \@@_error:n { Unknown~option~Arrow~in~code-after }
1800 }
```

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.

```
1801 \seq_new:N \l_@@_options_Arrow_code_after_seq
1802 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq
1803 { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }
```

```
1804 <*LaTeX>
1805 \NewDocumentCommand \@@_Arrow_code_after { 0 { } m m m ! 0 { } }
1806 </LaTeX>
1807 <*plain-TeX>
1808 \cs_new_protected:Npn \@@_Arrow_code_after
1809 {
1810   \peek_meaning:NTF [
1811     { \@@_Arrow_code_after_i }
1812     { \@@_Arrow_code_after_i [ ] }
1813   }
1814   \cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4
1815   {
1816     \peek_meaning:NTF [
1817       { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1818       { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
1819     }
1820   \cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
```

```

1821 </plain-TeX>
1822 {
1823   \int_set:Nn \l_@@_pos_arrow_int 1
1824   \str_clear_new:N \l_@@_previous_key_str
1825   \group_begin:
1826     \keys_set:nn { WithArrows / Arrow / code-after }
1827     { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
1828     \bool_set_false:N \l_@@_initial_r_bool
1829     \bool_set_false:N \l_@@_final_r_bool
1830     \int_case:nn \l_@@_pos_arrow_int
1831     {
1832       0
1833       {
1834         \bool_set_true:N \l_@@_initial_r_bool
1835         \bool_set_true:N \l_@@_final_r_bool
1836       }
1837       2 { \bool_set_true:N \l_@@_initial_r_bool }
1838       3 { \bool_set_true:N \l_@@_final_r_bool }
1839     }

```

We prevent drawing an arrow from a line to itself.

```

1840   \tl_if_eq:nnTF { #2 } { #3 }
1841   { \@@_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.

```

1842   {
1843     \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - 1 }
1844     { \@@_error:nx { Wrong~line~in~Arrow } { #2 } }
1845     {
1846       \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - 1 }
1847       { \@@_error:nx { Wrong~line~in~Arrow } { #3 } }
1848       {
1849         \int_compare:nNnTF \l_@@_pos_arrow_int = 4
1850         {
1851           <*LaTeX>
1852           \begin { tikzpicture } [ @@_standard ]
1853           </LaTeX>
1854           <*plain-TeX>
1855           \tikzpicture [ @@_standard ]
1856           </plain-TeX>
1857           \tikz@scan@one@point \pgfutil@firstofone (#2-1.south)
1858           \dim_set_eq:NN \l_tmpa_dim \pgf@x
1859           \dim_set_eq:NN \l_tmpb_dim \pgf@y
1860           \tikz@scan@one@point \pgfutil@firstofone (#3-1.north)
1861           \dim_set:Nn \l_tmpa_dim
1862           { \dim_max:nn \l_tmpa_dim \pgf@x }
1863           \tl_gset:Nx \g_tmpa_tl
1864           { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1865           \tl_gset:Nx \g_tmpb_tl
1866           { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1867           <*LaTeX>
1868           \end { tikzpicture }
1869           </LaTeX>
1870           <*plain-TeX>
1871           \endtikzpicture
1872           </plain-TeX>
1873         }
1874         {
1875           <*LaTeX>
1876           \begin { tikzpicture } [ @@_standard ]
1877           </LaTeX>
1878           <*plain-TeX>
1879           \tikzpicture [ @@_standard ]

```

```

1880 </plain-TeX>
1881
1882 \tikz@scan@one@point \pgfutil@firstofone
1883 ( #2-\bool_if:NTF\l_@@_initial_r_bool rl .south )
1884 \tl_gset:Nx \g_tmpa_tl
1885 { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1886 \tikz@scan@one@point \pgfutil@firstofone
1887 ( #3-\bool_if:NTF\l_@@_final_r_bool rl .north )
1888 \tl_gset:Nx \g_tmpb_tl
1889 { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1890 <*LaTeX>
1891 \end { tikzpicture }
1892 </LaTeX>
1893 <*plain-TeX>
1894 \endtikzpicture
1895 </plain-TeX>
1896 }
1897 \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
1898 }
1899 }
1900 \group_end:
1901 }

```

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

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

```

1902 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
1903 {

```

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.

```

1904 \exp_args:Nnx
1905 \regex_match:nnTF
1906 { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
1907 { #1 }
1908 { \@@_MultiArrow_i:nn { #1 } { #2 } }
1909 { \@@_error:nx { Invalid-specification-for-MultiArrow } { #1 } }
1910 }
1911 \cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
1912 {

```

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`.

```

1913 \foreach \x in { #1 }
1914 {
1915 \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
1916 { \@@_error:nx { Wrong-line-specification-in-MultiArrow } \x }
1917 { \clist_gput_right:Nx \g_tmpa_clist \x }
1918 }

```

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

```

1919 \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
1920 { \@@_error:n { Too-small-specification-for-MultiArrow } }
1921 {
1922 \clist_sort:Nn \g_tmpa_clist
1923 {
1924 \int_compare:nTF { ##1 > ##2 }
1925 \sort_return_swapped:
1926 \sort_return_same:
1927 }

```

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

```
1928 \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...

```
1929 \clist_reverse:N \g_tmpa_clist
1930 \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).

```
1931 \exp_args:NV \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
1932 \*LaTeX>
1933 \begin { tikzpicture }
1934 \*LaTeX>
1935 \*plain-TeX>
1936 \tikzpicture
1937 \*plain-TeX>
1938 [
1939 \@@_standard ,
1940 every~path / .style = { WithArrows / arrow }
1941 ]
1942 \draw [ <-> ] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
1943 -- ++(5mm,0)
1944 -- node (@@_label) {
1945 ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
1946 -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south) ;
1947 \tikz@parse@node \pgfutil@firstofone (@@_label.west)
1948 \dim_set:Nn \l_tmpa_dim { 20 cm }
1949 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
1950 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
1951 \bool_if:nT { \l_@@_wrap_lines_bool && \l_@@_in_DisWithArrows_bool }
1952 {
1953 \dim_set:Nn \l_tmpb_dim
1954 { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
1955 \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1956 { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1957 }
1958 \path (@@_label.west)
1959 node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
1960 \*LaTeX>
1961 \end{tikzpicture}
1962 \*LaTeX>
1963 \*plain-TeX>
1964 \endtikzpicture
1965 \*plain-TeX>
1966 }
1967 }

1968 \cs_new_protected:Npn \@@_MultiArrow_i:n #1
1969 {
1970 \*LaTeX>
1971 \begin { tikzpicture }
1972 \*LaTeX>
1973 \*plain-TeX>
1974 \tikzpicture
1975 \*plain-TeX>
1976 [
1977 \@@_standard ,
1978 every~path / .style = { WithArrows / arrow }
1979 ]
```

```

1980     \foreach \k in { #1 }
1981     {
1982         \draw [ <- ]
1983             ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
1984     } ;
1985 \LaTeX
1986 \end{tikzpicture}
1987 \LaTeX
1988 \plain-TeX
1989 \endtikzpicture
1990 \plain-TeX
1991 }

```

### 11.13 The error messages of the package

```

1992 \str_const:Nn \c_@@_option_ignored_str
1993 { If-you-go-on,~this~option~will~be~ignored. }
1994 \str_const:Nn \c_@@_command_ignored_str
1995 { If-you-go-on,~this~command~will~be~ignored. }
1996 \LaTeX
1997 \@@_msg_new:nn { amsmath~not~loaded }
1998 {
1999     You~can't~use~the~option~'\l_keys_key_tl'~because~the~
2000     package~'amsmath'~has~not~been~loaded.\\
2001     If~you~go~on,~this~option~will~be~ignored~in~the~rest~
2002     of~the~document.
2003 }
2004 \LaTeX
2005 \@@_msg_new:nn { Bad~value~for~replace~brace~by }
2006 {
2007     Bad~value~for~the~option~'\l_keys_key_tl'.~The~value~must~begin~
2008     with~an~extensible~left~delimiter.~The~possible~values~are:~.,
2009     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
2010     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
2011     \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
2012     \token_to_str:N \lfloor~and~\token_to_str:N \lceil~
2013     (and~\token_to_str:N \lvert~and~\token_to_str:N \lVert~
2014     if~amsmath~or~unicode-math~is~loaded~in~LaTeX).\\
2015     \c_@@_option_ignored_str
2016 }
2017 \@@_msg_new:nn { option~of~cr~negative }
2018 {
2019     The~argument~of~the~command~\token_to_str:N~~
2020     should~be~positive~in~the~row~\int_use:N \g_@@_line_int~
2021     of~your~environment~\{\l_@@_type_env_str\}.\\
2022     \c_@@_option_ignored_str
2023 }
2024 \@@_msg_new:nn { omit~probably~used }
2025 {
2026     There~is~a~problem.~Maybe~you~have~used~a~command~
2027     \token_to_str:N\omit~ in~the~line~\int_use:N \g_@@_line_int~
2028     (or~another~line)~of~your~environment~\{\l_@@_type_env_str\}.\\
2029     You~can~go~on~but~you~may~have~others~errors.
2030 }
2031 \LaTeX
2032 \@@_msg_new:nn { newline~at~the~end~of~env }
2033 {
2034     The~environments~of~witharrows~(\{WithArrows\}~and~
2035     \{DispWithArrows\})~should~not~end~by~\token_to_str:N \\\\.
2036     However,~you~can~go~on~for~this~time.~No~similar~error~will~be~

```



```

2037     raised-in-this-document.
2038   }
2039 </LaTeX>
2040 \@@_msg_new:nn { Invalid-option-format }
2041   {
2042     The-key~'format'~should-contain-only~letters~r,~c~and~l~and~
2043     must-not-be-empty.\\
2044     \c_@@_option_ignored_str
2045   }
2046 \@@_msg_new:nn { Value-for-a-key }
2047   {
2048     The-key~'\l_keys_key_tl'~should-be-used-without-value. \\
2049     However,~you-can-go-on-for~this-time.
2050   }
2051 \@@_msg_new:nnn { Unknown-option-in-Arrow }
2052   {
2053     The-key~'\l_keys_key_tl'~is-unknown-for~the-command~
2054     \l_@@_string_Arrow_for_msg_str\ in~the-row~
2055     \int_use:N \g_@@_line_int\ of~your~environment~
2056     \{\l_@@_type_env_str\}. \l_tmpa_str \\
2057     \c_@@_option_ignored_str \\
2058     For~a~list~of~the~available~keys,~type-H~<return>.
2059   }
2060   {
2061     The~available~keys~are~(in~alphabetic~order):~
2062     \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
2063   }
2064 \@@_msg_new:nnn { Unknown-option-WithArrows }
2065   {
2066     The-key~'\l_keys_key_tl'~is-unknown-in~\{\l_@@_type_env_str\}. \\
2067     \c_@@_option_ignored_str \\
2068     For~a~list~of~the~available~keys,~type-H~<return>.
2069   }
2070   {
2071     The~available~keys~are~(in~alphabetic~order):~
2072     \seq_use:Nnnn \l_@@_options-WithArrows_seq {~and~} {,~} {~and~}.
2073   }
2074 \@@_msg_new:nnn { Unknown-option-DispWithArrows }
2075   {
2076     The-key~'\l_keys_key_tl'~is-unknown-in~\{\l_@@_type_env_str\}. \\
2077     \c_@@_option_ignored_str \\
2078     For~a~list~of~the~available~keys,~type-H~<return>.
2079   }
2080   {
2081     The~available~keys~are~(in~alphabetic~order):~
2082     \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.
2083   }
2084 \@@_msg_new:nnn { Unknown-option-WithArrowsOptions }
2085   {
2086     The-key~'\l_keys_key_tl'~is-unknown-in~
2087     \token_to_str:N \WithArrowsOptions. \\
2088     \c_@@_option_ignored_str \\
2089     For~a~list~of~the~available~keys,~type-H~<return>.
2090   }
2091   {
2092     The~available~keys~are~(in~alphabetic~order):~
2093     \seq_use:Nnnn \l_@@_options-WithArrowsOptions_seq {~and~} {,~} {~and~}.
2094   }
2095 \@@_msg_new:nnn { Unknown-option-Arrow-in-code-after }
2096   {
2097     The-key~'\l_keys_key_tl'~is-unknown-in~

```

```

2098 \token_to_str:N \Arrow\ in~code~after. \\
2099 \c_@@_option_ignored_str \\
2100 For~a~list~of~the~available~keys,~type~H~<return>.
2101 }
2102 {
2103 The~available~keys~are~(in~alphabetic~order):~
2104 \seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2105 }
2106 \@@_msg_new:nn { Too~much~columns~in~WithArrows }
2107 {
2108 Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
2109 \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2110 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2111 If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2112 the~option~'more~columns'~at~a~global~level~or~for~an~environment. \\
2113 However,~you~can~go~one~for~this~time.
2114 }
2115 \@@_msg_new:nn { Too~much~columns~in~DispWithArrows }
2116 {
2117 Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
2118 \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2119 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\
2120 at~the~end~of~row~\int_use:N \g_@@_line_int. \\
2121 This~error~is~fatal.
2122 }
2123 \@@_msg_new:nn { Negative~jump }
2124 {
2125 You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
2126 \l_@@_string_Arrow_for_msg_str\
2127 in~the~row~\int_use:N \g_@@_line_int\
2128 of~your~environment~\{\l_@@_type_env_str\}.~
2129 You~can~create~an~arrow~going~backwards~with~the~option~'<~'~of~Tikz. \\
2130 \c_@@_option_ignored_str
2131 }
2132 \@@_msg_new:nn { new~group~without~groups }
2133 {
2134 You~can't~use~the~option~'new~group'~for~the~command~
2135 \l_@@_string_Arrow_for_msg_str\
2136 because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
2137 'groups'~in~your~environment~\{\l_@@_type_env_str\}. \\
2138 \c_@@_option_ignored_str
2139 }
2140 \@@_msg_new:nn
2141 { Too~few~lines~for~an~arrow }
2142 {
2143 Line~\l_@@_input_line_str\
2144 :~an~arrow~specified~in~the~row~\int_use:N \l_@@_initial_int\
2145 of~your~environment~\{\l_@@_type_env_str\}~can't~be~drawn~
2146 because~it~arrives~after~the~last~row~of~the~environment. \\
2147 If~you~go~on,~this~arrow~will~be~ignored.
2148 }
2149 \@@_msg_new:nn { WithArrows~outside~math~mode }
2150 {
2151 The~environment~\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
2152 like~the~environment~\{aligned\}~of~amsmath. \\
2153 Nevertheless,~you~can~go~on.
2154 }
2155 \@@_msg_new:nn { DispWithArrows~in~math~mode }
2156 {
2157 The~environment~\{\l_@@_type_env_str\}~should~be~used~only~outside~math~
2158 mode~like~the~environment~\{align\}~of~amsmath. \\

```

```

2159     This-error-is-fatal.
2160 }

2161 \@@_msg_new:nn { Incompatible-options-in-Arrow }
2162 {
2163     You-try-to-use-the-option~'\l_keys_key_tl'~but~
2164     this-option-is-incompatible-or-redundant-with-the-option~
2165     '\l_@@_previous_key_str'~set-in-the-same-command~
2166     \l_@@_string_Arrow_for_msg_str. \\
2167     \c_@@_option_ignored_str
2168 }

2169 \@@_msg_new:nn { Incompatible-options }
2170 { You-try-to-use-the-option~'\l_keys_key_tl'~but~
2171     this-option-is-incompatible-or-redundant-with-the-option~
2172     '\l_@@_previous_key_str'~set-in-the-same-command~
2173     \bool_if:NT \l_@@_in_code_after_bool
2174     {
2175         \l_@@_string_Arrow_for_msg_str\
2176         in-the-code-after-of-your-environment~\{\l_@@_type_env_str\}
2177     }. \\
2178     \c_@@_option_ignored_str
2179 }

2180 \@@_msg_new:nn { Arrow-not-in-last-column }
2181 {
2182     You-should-use-the-command~\l_@@_string_Arrow_for_msg_str\
2183     only-in-the-last-column~(column~\int_use:N\l_@@_nb_cols_int)~
2184     of-your-environment~\{\l_@@_type_env_str\}. \\
2185     However-you-can-go-on-for-this-time.
2186 }

2187 \@@_msg_new:nn { Wrong-line-in-Arrow }
2188 {
2189     The-specification-of-line~'#1'~you-use-in-the-command~
2190     \l_@@_string_Arrow_for_msg_str\
2191     in-the~'code-after'~of~\{\l_@@_type_env_str\}~doesn't-exist. \\
2192     \c_@@_option_ignored_str
2193 }

2194 \@@_msg_new:nn { Both-lines-are-equal }
2195 {
2196     In-the~'code-after'~of~\{\l_@@_type_env_str\}~you-try-to~
2197     draw-an-arrow-going-to-itself-from-the-line~'#1'.~This-is-not-possible. \\
2198     \c_@@_option_ignored_str
2199 }

2200 \@@_msg_new:nn { Wrong-line-specification-in-MultiArrow }
2201 {
2202     The-specification-of-line~'#1'~doesn't-exist. \\
2203     If-you-go-on,~it~will-be-ignored-for~\token_to_str:N \MultiArrow.
2204 }

2205 \@@_msg_new:nn { Too-small-specification-for-MultiArrow }
2206 {
2207     The-specification-of-lines-you-gave-to~\token_to_str:N \MultiArrow\
2208     is-too-small:~you-need-at-least-two-lines. \\
2209     \c_@@_command_ignored_str
2210 }

2211 \@@_msg_new:nn { Not-allowed-in-DispWithArrows }
2212 {
2213     The-command~\token_to_str:N #1
2214     is-allowed-only-in-the-last-column~
2215     (column~\int_use:N\l_@@_nb_cols_int)~of~\{\l_@@_type_env_str\}. \\
2216     \c_@@_option_ignored_str
2217 }

2218 \@@_msg_new:nn { Not-allowed-in-WithArrows }

```

```

2219 {
2220   The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
2221   (it's~allowed~in~the~last~column~of~\{DispWithArrows\}). \\
2222   \c_@@_option_ignored_str
2223 }
2224 \*LaTeX\
2225 \@@_msg_new:nn { tag*~without~amsmath }
2226 {
2227   We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2228   (or~mathtools). \\
2229   If~you~go~on,~the~command~\token_to_str:N\tag\
2230   will~be~used~instead.
2231 }
2232 \@@_msg_new:nn { Multiple~tags }
2233 {
2234   You~can't~use~twice~the~command~\token_to_str:N\tag\
2235   in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2236   If~you~go~on,~the~tag~'#1'~will~be~used.
2237 }
2238 \@@_msg_new:nn { Multiple~labels }
2239 {
2240   Normally,~we~can't~use~the~command~\token_to_str:N\label\
2241   twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2242   However,~you~can~go~on.~
2243   \bool_if:NT \c_@@_showlabels_loaded_bool
2244     { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
2245   If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
2246   'allow~multiple~labels'~at~the~global~or~environment~level.
2247 }
2248 \@@_msg_new:nn { Multiple~labels~with~cleveref }
2249 {
2250   Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
2251   twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2252   If~you~go~on,~you~may~have~undefined~references.
2253 }
2254 \*LaTeX\
2255 \@@_msg_new:nn { Inexistent~v-node }
2256 {
2257   There~is~a~problem.~Maybe~you~have~put~a~command~\token_to_str:N\cr\
2258   instead~of~a~command~\token_to_str:N\\~at~the~end~of~
2259   the~row~\l_tmpa_int\
2260   of~your~environment~\{\l_@@_type_env_str\}. \\
2261   This~error~is~fatal.
2262 }

```

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`).

```

2263 \@@_msg_new:nn { Option~xoffset~forbidden }
2264 {
2265   You~can't~use~the~option~'xoffset'~in~the~command~
2266   \l_@@_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g_@@_line_int\
2267   of~your~environment~\{\l_@@_type_env_str\}~
2268   because~you~are~using~the~option~
2269   ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
2270     { group }
2271     { groups } ' .~It's~possible~for~an~independent~arrow~or~if~there~is~
2272   only~one~arrow. \\
2273   \c_@@_option_ignored_str
2274 }
2275 \@@_msg_new:nnn { Duplicate~name }

```

```

2276 {
2277   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2278   the~same~environment~name~twice.~You~can~go~on,~but,~
2279   maybe,~you~will~have~incorrect~results. \\
2280   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2281   If~you~don't~want~to~see~this~message~again,~use~the~option~
2282   'allow~duplicate~names'.
2283 }
2284 {
2285   The~names~already~defined~in~this~document~are:~
2286   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2287 }
2288 \@@_msg_new:nn { Invalid~specification~for~MultiArrow }
2289 {
2290   The~specification~of~rows~for~\token_to_str:N\MultiArrow\
2291   (i.e.~#1)~is~invalid. \\
2292   \c_@@_command_ignored_str
2293 }

```

## 11.14 The command `\WithArrowsNewStyle`

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

```

2294 <*LaTeX>
2295 \NewDocumentCommand \WithArrowsNewStyle { m m }
2296 </LaTeX>
2297 <*plain-TeX>
2298 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2299 </plain-TeX>
2300 {
2301   \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2302   { \@@_error:nn { Key~already~defined } { #1 } }
2303   {
2304     \keys_define:nn { WithArrows / Global }
2305     {
2306       #1 .code:n =
2307       { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
2308     }
2309     \seq_put_right:Nx \l_@@_options_WithArrows_seq { \tl_to_str:n { #1 } }
2310     \seq_put_right:Nx \l_@@_options_DispWithArrows_seq
2311     { \tl_to_str:n { #1 } }
2312     \seq_put_right:Nx \l_@@_options_WithArrowsOptions_seq
2313     { \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.

```

2314   \group_begin:
2315   \msg_set:nnn { witharrows } { Unknown~option~WithArrowsOptions }
2316   {
2317     The~key~'\l_keys_key_tl'~can't~be~set~in~the~
2318     definition~of~a~style.~You~can~go~on~for~this~time~
2319     but~you~should~suppress~this~key.
2320   }
2321   \WithArrowsOptions { #2 }
2322   \group_end:
2323 }
2324 }
2325 \@@_msg_new:nn { Key~already~defined }
2326 {
2327   The~key~'#1'~is~already~defined. \\
2328   If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2329   will~be~ignored.
2330 }

```

## 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 by default 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 keys `up` and `down` can be used with a value. This value is a list of pairs key-value specific to the options `up` and `down`.

- The key `radius` is the radius of the rounded corner of the arrow.
- The key `width` is the width of the horizontal part of the arrow. The corresponding dimension is `\l_@@_arrow_width_dim`. By convention, a value of 0 pt for `\l_@@_arrow_width_dim` means that the option `width` has been used with the special value `min` and a value of `\c_max_dim` means that it has been used with the value `max`.

```

2331 \keys_define:nn { WithArrows / up-and-down }
2332 {
2333   radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
2334   radius .value_required:n = true ,
2335   width .code:n =
2336     \str_case:nnF { #1 }
2337     {
2338       { min } { \dim_zero:N \l_@@_arrow_width_dim }
2339       { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
2340     }
2341     { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
2342   width .value_required:n = true ,
2343   unknown .code:n = \@@_error:n { Option-unknown~for~up-and-down }
2344 }
2345 \@@_msg_new:nn { Option-unknown~for~up-and-down }
2346 {
2347   The~option~'\l_keys_key_tl'~is~unknown.~\c_@@_option_ignored_str
2348 }

```

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

```

2349 (*LaTeX)
2350 \tl_const:Nn \c_@@_tikz_code_up_tl
2351 {

```

First the case when the key `up` is used with `width=max` (that's the default behaviour).

```

2352   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2353   {
2354     \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2355       let \p1 = ( #1 ) , \p2 = ( #2 )
2356       in (\p1) -- node
2357       {
2358         \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2359         \begin { varwidth } \l_tmpa_dim

```

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

```

2360         \narrowragged
2361         #3
2362       \end { varwidth }
2363     }
2364     (\x2,\y1) -- (\p2) ;
2365   }

```

Now the case where the key `up` is used with `width=value` with `value` equal to `min` or a numeric value. The instruction `\path` doesn't draw anything: its aim is to compute the natural width of the label of the arrow. We can't use `\pgfextra` here because of the `\hbox_gset:Nn`.

```

2366     {
2367       \path
2368       let \p1 = ( #1 ) , \p2 = ( #2 )
2369       in node
2370       {

```

The length `\l_tmpa_dim` will be the maximal width of the box composed by the environment `{varwidth}`.

```

2371         \dim_set:Nn \l_tmpa_dim
2372         { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
2373       \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
2374       {
2375         \dim_set:Nn \l_tmpa_dim
2376         { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
2377       }

```

Now, the length `\l_tmpa_dim` is computed. We can compose the label in the box `\g_tmpa_box`. We have to do a global affectation to be able to exit the node.

```

2378         \hbox_gset:Nn \g_tmpa_box
2379         {
2380           \begin { varwidth } \l_tmpa_dim
2381             \narrowragged
2382             #3
2383           \end { varwidth }
2384         }

```

The length `\g_tmpa_dim` will be the width of the arrow (+ the radius of the corner).

```

2385         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2386         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2387         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2388       \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2389     } ;
2390   \draw
2391   let \p1 = ( #1 ) , \p2 = ( #2 )
2392   in (\x2-\g_tmpa_dim,\y1)
2393   -- node { \box_use:N \g_tmpa_box }
2394   (\x2-\l_@@_up_and_down_radius_dim,\y1)
2395   [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2396   -| (\p2) ;
2397 }
2398 }
2399 </LaTeX>
2400 <*plain-TeX>
2401 \tl_const:Nn \c_@@_tikz_code_up_tl
2402 {
2403   \dim_case:nnF \l_@@_arrow_width_dim
2404   {
2405     \c_max_dim
2406     {
2407       \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2408       let \p1 = ( #1 ) , \p2 = ( #2 )
2409       in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
2410     }
2411   \c_zero_dim
2412   {
2413     \path node
2414     {
2415       \hbox_gset:Nn \g_tmpa_box { #3 }
2416       \dim_gset:Nn \g_tmpa_dim
2417       { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2418     } ;

```

```

2419         \draw
2420         let \p1 = ( #1 ) , \p2 = ( #2 )
2421         in (\x2-\g_tmpa_dim,\y1)
2422         -- node { \box_use:N \g_tmpa_box }
2423         (\x2-\l_@@_up_and_down_radius_dim,\y1)
2424         [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2425         -| (\p2) ;
2426     }
2427 }
2428 {
2429     \draw
2430     let \p1 = ( #1 ) , \p2 = ( #2 )
2431     in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
2432     -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
2433     [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2434     -| (\p2) ;
2435 }
2436 }
2437 \end{plain-TeX}

```

The code for a arrow of type down is similar to the previous code (for an arrow of type up).

```

2438 \begin{LaTeX}
2439 \tl_const:Nn \c_@@_tikz_code_down_tl
2440 {
2441     \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2442     {
2443         \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2444         let \p1 = ( #1 ) , \p2 = ( #2 )
2445         in (\p1) -- (\x1,\y2) -- node
2446         {
2447             \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2448             \begin { varwidth } \l_tmpa_dim
2449                 \narrowragged
2450                 #3
2451             \end { varwidth }
2452         }
2453         (\p2) ;
2454     }
2455     {
2456         \path
2457         let \p1 = ( #1 ) , \p2 = ( #2 )
2458         in node
2459         {
2460             \hbox_gset:Nn \g_tmpa_box
2461             {
2462                 \dim_set:Nn \l_tmpa_dim

```

The 2 mm are for the tip of the arrow. We don't want the label of the arrow too close to the tip of arrow (we assume that to the tip of the arrow has its standard position, that is at the end of the arrow.).

```

2463             { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
2464             \begin { varwidth } \l_tmpa_dim
2465                 \narrowragged
2466                 #3
2467             \end { varwidth }
2468         }
2469         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2470         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2471         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2472         \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2473     } ;
2474
2475     \draw

```



```

2476     let \p1 = ( #1 ) , \p2 = ( #2 )
2477     in (\p1)
2478     { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2479     -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2480     -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2481     -- ++ (-2mm,0) ;
2482   }
2483 }
2484 \end{LaTeX}
2485 %
2486 \begin{plain-TeX}
2487 \tl_const:Nn \c_@@_tikz_code_down_tl
2488 {
2489   \dim_case:nnF \l_@@_arrow_width_dim
2490   {
2491     \c_max_dim
2492     {
2493       \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2494         let \p1 = ( #1 ) , \p2 = ( #2 )
2495         in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
2496     }
2497     \c_zero_dim
2498     {
2499       \path node
2500       {
2501         \hbox_gset:Nn \g_tmpa_box { #3 }
2502         \dim_gset:Nn \g_tmpa_dim
2503         { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2504       } ;
2505       \draw
2506         let \p1 = ( #1 ) , \p2 = ( #2 )
2507         in (\p1)
2508         { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2509         -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2510         -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2511         -- ++ (-2mm,0) ;
2512     }
2513   }
2514 }
2515 \draw
2516   let \p1 = ( #1 ) , \p2 = ( #2 )
2517   in (\p1)
2518   { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2519   -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2520   -- node { #3 }
2521     (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
2522   -- ++ (-2mm,0) ;
2523 }
2524 \end{plain-TeX}
2525 \end{plain-TeX}

```

We recall that the options of the individual arrows are scanned twice. First, when are scanned when the command `\Arrow` occurs (we try to know whether the arrow is “individual”, etc.). That’s the first pass.

```

2526 \keys_define:nn { WithArrows / Arrow / FirstPass }
2527 {
2528   up .code:n = \@@_set_independent_bis: ,
2529   down .code:n = \@@_set_independent_bis: ,
2530   up .default:n = NoValue ,
2531   down .default:n = NoValue
2532 }

```

The options are scanned a second time when the arrow is actually drawn. That’s the second pass.

```

2533 \keys_define:nn { WithArrows / Arrow / SecondPass }
2534 {
2535   up .code:n =
2536     \str_if_empty:NT \l_@@_previous_key_str
2537     {
2538       \str_set:Nn \l_@@_previous_key_str { up }
2539     }
2540     \bool_if:NTF \c_@@_varwidth_loaded_bool
2541     {
2542     }
2543     \cs_if_exist:cTF { tikz@library@calc@loaded }
2544     {
2545       \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2546       \int_set:Nn \l_@@_pos_arrow_int \c_one_int

```

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

```

2547       \bool_set_false:N \l_@@_wrap_lines_bool

```

The main action occurs now. We change the value of the `tikz-code`.

```

2548       \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
2549     }
2550     { \@@_error:n { calc~not~loaded } }
2551   }
2552   { \@@_error:n { varwidth~not~loaded } }
2553 }
2554 \end{tikzpicture}
2555 } ,
2556 down .code:n =
2557   \str_if_empty:NT \l_@@_previous_key_str
2558   {
2559     \str_set:Nn \l_@@_previous_key_str { down }
2560   }
2561   \bool_if:NTF \c_@@_varwidth_loaded_bool
2562   {
2563   }
2564   \cs_if_exist:cTF { tikz@library@calc@loaded }
2565   {
2566     \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2567     \int_set:Nn \l_@@_pos_arrow_int \c_one_int
2568     \bool_set_false:N \l_@@_wrap_lines_bool
2569     \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
2570   }
2571   { \@@_error:n { calc~not~loaded } }
2572 }
2573 { \@@_error:n { varwidth~not~loaded } }
2574 }
2575 \end{tikzpicture}
2576 }
2577 }

```

```

2578 \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
2579 \seq_put_right:Nn \l_@@_options_Arrow_seq { up }
2580 \@@_msg_new:nn { varwidth~not~loaded }
2581 {
2582   You~can't~use~the~option~'\l_keys_key_tl'~because~
2583   you~don't~have~loaded~the~package~'varwidth'. \\
2584   \c_@@_option_ignored_str
2585 }
2586 \@@_msg_new:nn { calc~not~loaded }
2587 {
2588   You~can't~use~the~option~'\l_keys_key_tl'~because~you~don't~have~loaded~the~
2589   Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~

```

```

2590 ~in~the~preamble~of~your~document. \\
2591 \c_@@_option_ignored_str
2592 }
2593 <*plain-TeX>
2594 \catcode \@ = 12
2595 \ExplSyntaxOff
2596 </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`.<sup>39</sup>

---

<sup>39</sup>Before this version, there was an error when using `witharrows` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `witharrows` because they use the options `overlay` and `remember picture`.

## Changes between 1.15 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`.

## Changes between 2.1 and 2.2

Addition of `\normalbaselines` at the beginning of `\@@_post_halign:`.

The warning for an environment ending by `\\` has been transformed in `error`.

## Changes between 2.2 and 2.3

Two options for the arrows of type up and down: `width` and `radius`.

# 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> .....	1906
<code>\.</code> .....	1906
@@ commands:	
<code>\@@_Arrow</code> .....	702, 705, 737, 813
<code>\@@_Arrow_code_after</code> .....	967, 1805, 1808
<code>\@@_Arrow_code_after_i</code> ...	1811, 1812, 1814
<code>\@@_Arrow_code_after_ii</code> ..	1817, 1818, 1820
<code>\@@_Arrow_first_columns:</code> .....	736, 784
<code>\@@_Arrow_i</code> .....	708, 709, 711
<code>\@@_Arrow_ii</code> .....	714, 715, 717
<code>\@@_MultiArrow:nn</code> .....	966, 1902
<code>\@@_MultiArrow_i:n</code> .....	1931, 1968
<code>\@@_MultiArrow_i:nn</code> .....	1908, 1911
<code>\g_@@_alignment_dim</code> .....	1264, 1265, 1269, 1270, 1280
<code>\c_@@_amsmath_loaded_bool</code> .....	226, 417, 436, 1156, 1394
<code>\c_@@_amsthm_loaded_bool</code> .....	818
<code>\@@_analyze_end:Nn</code> .....	1003, 1105
<code>\g_@@_arrow_int</code> .....	254, 676, 720, 731, 733, 750, 751, 954, 956, 978, 1462, 1492
<code>\l_@@_arrow_int</code> .....	767, 1461, 1462, 1465, 1469, 1473, 1476, 1484, 1498, 1512, 1519, 1523, 1525, 1535, 1540, 1544, 1564, 1565, 1568, 1572, 1576, 1584, 1614, 1618, 1622
<code>\g_@@_arrow_int_seq</code> .....	253, 750, 977
<code>\l_@@_arrow_width_dim</code> .....	294, 295, 2338, 2339, 2341, 2352, 2373, 2376, 2385, 2386, 2403, 2431, 2441, 2469, 2470, 2489, 2521
<code>\c_@@_autonum_loaded_bool</code> .....	1418
<code>\c_@@_cleveref_loaded_bool</code> ....	1030, 1404
<code>\l_@@_code_after_tl</code> .....	490, 778, 969

<code>\l_@@_code_before_tl</code> .....	488, 777, 792	<code>\l_@@_in_WithArrows_bool</code> .....	244, 780, 844, 914, 1364
<code>\@@_code_for_possible_arrow:</code> ...	1483, 1494	<code>\l_@@_in_code_after_bool</code> ...	246, 968, 2173
<code>\g_@@_col_int</code> .....	258, 754, 755, 786, 809, 811, 824, 825, 833, 856, 982, 989, 1367	<code>\l_@@_in_first_columns_bool</code> .....	283
<code>\g_@@_col_int_seq</code> .....	257, 754, 981	<code>\l_@@_in_label_or_minipage_bool</code> .....	1119, 1167, 1170, 1213, 1227, 1333
<code>\c_@@_command_ignored_str</code> .....	1994, 2209, 2292	<code>\@@_info:n</code> .....	87
<code>\l_@@_command_name_str</code> .....	270, 271, 322, 784, 813, 967	<code>\l_@@_initial_int</code> .....	765, 1467, 1500, 1520, 1534, 1551, 1570, 1604, 1608, 1656, 2144
<code>\@@_construct_halign:</code> ..	803, 810, 926, 1240	<code>\l_@@_initial_r_bool</code> .....	285, 1589, 1594, 1597, 1608, 1828, 1834, 1837, 1882
<code>\@@_construct_nodes:</code> .....	834, 865	<code>\l_@@_initial_tl</code> .....	287, 1606, 1640
<code>\@@_convert_to_str_seq:N</code> ....	145, 157, 542	<code>\l_@@_input_line_str</code> .....	749, 1477, 2143
<code>\@@_cr:</code> .....	763, 986	<code>\l_@@_interline_skip</code> .....	362, 776, 1100
<code>\@@_cr_i:</code> .....	993, 995	<code>\l_@@_jump_int</code> .....	609, 724, 769, 770
<code>\@@_cr_ii:</code> .....	996, 998, 1010	<code>\@@_keys_set:</code> .....	1555, 1588
<code>\@@_cr_iii:n</code> .....	1002, 1005, 1007	<code>\@@_label:n</code> .....	799, 1398
<code>\@@_def_function_tmpa:n</code> .....	1705, 1732	<code>\l_@@_labels_seq</code> .....	773, 1022, 1044, 1402, 1408
<code>\l_@@_delim_wd_dim</code> .....	860, 1185, 1186	<code>\l_@@_last_arrow_int</code> .....	1562, 1563, 1565
<code>\l_@@_displaystyle_bool</code> .....	328, 829, 930, 1193	<code>\l_@@_last_arrows_seq</code> .....	1459, 1524, 1525, 1539, 1540, 1544, 1621
<code>\@@_draw_arrow:nnn</code> .....	371, 1702, 1728, 1735, 1896	<code>\g_@@_last_env_int</code> .....	249, 976, 1777
<code>\@@_draw_arrows:nn</code> ....	370, 1492, 1510, 1557	<code>\l_@@_last_line_of_group_int</code> .....	1457, 1500, 1521, 1536, 1538, 1543
<code>\@@_draw_arrows_i:</code> .....	1575, 1580	<code>\l_@@_left_brace_box</code> ..	860, 1187, 1188, 1282
<code>\l_@@_end_adjust_dim</code> ...	365, 686, 689, 1698	<code>\l_@@_left_brace_tl</code> .....	242, 243, 551, 857, 1158, 1174, 1194, 1272
<code>\@@_error:n</code> .....	31, 80, 90, 305, 341, 351, 439, 466, 485, 494, 498, 547, 554, 575, 593, 603, 610, 626, 646, 682, 737, 922, 928, 990, 1094, 1109, 1395, 1405, 1406, 1481, 1799, 1920, 2343, 2550, 2553, 2571, 2574	<code>\c_@@_leqno_bool</code> .....	95, 96, 1064, 1072
<code>\@@_error:nn</code> .....	34, 35, 1365, 1368, 1382, 1841, 1844, 1847, 1909, 1916, 2302	<code>\g_@@_line_int</code> .....	256, 266, 723, 724, 752, 753, 855, 872, 876, 886, 890, 897, 965, 980, 1067, 1075, 1088, 1311, 1478, 1575, 2020, 2027, 2055, 2120, 2127, 2266
<code>\@@_eval_if_allowed:n</code> .....	298, 308	<code>\g_@@_line_int_seq</code> .....	255, 752, 979
<code>\c_@@_extensible_delimiters_clist</code> ...	408, 409, 419, 463	<code>\l_@@_linewidth_dim</code> .....	850, 1225, 1228, 1229, 1232, 1275, 1300
<code>\@@_fatal:n</code> .....	33, 42, 1206, 1246, 1315	<code>\l_@@_mathindent_dim</code> .....	427, 853, 1278
<code>\l_@@_final_int</code> .....	766, 1471, 1478, 1521, 1536, 1538, 1543, 1551, 1574, 1575, 1604, 1612, 1656	<code>\c_@@_mathtools_loaded_bool</code> .....	1147, 1331, 1384, 1409
<code>\l_@@_final_r_bool</code> .....	286, 1590, 1593, 1598, 1612, 1829, 1835, 1838, 1886	<code>\l_@@_max_length_of_arrow_dim</code> .....	311, 1654, 1667, 1678
<code>\l_@@_final_tl</code> .....	288, 1611, 1643	<code>\@@_msg_new:nn</code> .....	27, 44, 51, 56, 65, 1997, 2005, 2017, 2024, 2032, 2040, 2046, 2106, 2115, 2123, 2132, 2140, 2149, 2155, 2161, 2169, 2180, 2187, 2194, 2200, 2205, 2211, 2218, 2225, 2232, 2238, 2248, 2255, 2263, 2288, 2325, 2345, 2580, 2586
<code>\l_@@_first_arrow_int</code> ....	1560, 1561, 1564	<code>\@@_msg_new:nnn</code> .....	28, 2051, 2064, 2074, 2084, 2095, 2275
<code>\l_@@_first_arrow_of_group_int</code> .....	1455, 1490, 1492, 1508, 1511, 1519	<code>\@@_msg_redirect_name:nn</code> .....	29, 320, 442, 448, 571, 1112
<code>\l_@@_first_arrows_seq</code> .....	1458, 1522, 1523, 1535, 1617	<code>\l_@@_name_str</code> ..	487, 746, 875, 876, 889, 890
<code>\l_@@_first_line_of_group_int</code> .....	1456, 1520, 1534	<code>\g_@@_names_seq</code> .....	274, 484, 486, 2286
<code>\@@_fix_pos_arrow:n</code> .....	654, 668, 669, 670, 671, 672	<code>\l_@@_nb_cols_int</code> .....	289, 785, 786, 811, 833, 991, 1367, 2109, 2118, 2183, 2215
<code>\@@_fix_pos_option:n</code> .....	307, 354, 355, 356, 357, 358, 1784, 1786, 1788, 1790, 1792	<code>\l_@@_new_box</code> .....	1266, 1267, 1269, 1270
<code>\l_@@_fleqn_bool</code> .....	425, 852, 1237, 1277	<code>\l_@@_new_group_bool</code> .....	284, 1460, 1514, 1516, 1518
<code>\g_@@_footnote_bool</code> .....	25, 39, 74, 93, 790, 944, 1345	<code>\@@_nonumber:</code> .....	796, 1375
<code>\g_@@_footnotehyper_bool</code> .....	24, 40, 84	<code>\@@_notag:</code> .....	795, 1373
<code>\l_@@_format_seq</code> .....	787, 788, 805	<code>\@@_old_label</code> .....	798, 1044, 1191
<code>\l_@@_format_str</code> ....	290, 497, 771, 785, 788	<code>\c_@@_option_ignored_str</code> .....	1992, 2015, 2022, 2044, 2057, 2067,
<code>\l_@@_halign_box</code> .....	1234, 1235, 1273, 1285, 1286, 1301		
<code>\c_@@_hyperref_loaded_bool</code> .....	1025		
<code>\@@_if_in_last_col_of_disp:Nn</code> .....	1362, 1374, 1376, 1379, 1400, 1428		
<code>\l_@@_in_DispWithArrows_bool</code> .....	245, 782, 815, 838, 915, 1014, 1141, 1730, 1951		

2077, 2088, 2099, 2130, 2138, 2167, 2178,  
 2192, 2198, 2216, 2222, 2273, 2347, 2584, 2591  
 \l\_@@\_options\_Arrow\_code\_after\_seq ...  
 ..... 1798, 1801, 1802, 2104  
 \l\_@@\_options\_Arrow\_seq ..... 338,  
 346, 347, 348, 639, 648, 649, 2062, 2578, 2579  
 \l\_@@\_options\_DispWithArrows\_seq ....  
 ..... 229, 553, 556, 557, 2082, 2310  
 \l\_@@\_options\_WithArrowsOptions\_seq ..  
 ..... 228, 574, 577, 578, 2093, 2312  
 \l\_@@\_options\_WithArrows\_seq .....  
 ..... 529, 530, 542, 546, 640, 2072, 2309  
 \l\_@@\_pos\_arrow\_int .....  
 .. 251, 252, 308, 339, 349, 624, 659, 678,  
 958, 959, 1445, 1448, 1480, 1488, 1502,  
 1526, 1550, 1591, 1601, 1625, 1661, 1672,  
 1685, 1694, 1823, 1830, 1849, 2269, 2546, 2567  
 \l\_@@\_pos\_env\_int ... 250, 399, 400, 401, 924  
 \l\_@@\_pos\_of\_arrow\_int ..... 768  
 \g\_@@\_position\_in\_the\_tree\_seq .....  
 ..... 247, 248, 758, 759, 971, 972, 973, 975  
 \@@\_post\_halign: ..... 942, 950, 1329  
 \@@\_pre\_halign:n ..... 738, 920, 1159  
 \l\_@@\_prefix\_str .... 201, 731, 733, 761,  
 762, 872, 886, 897, 1314, 1465, 1469, 1473,  
 1476, 1568, 1572, 1584, 1614, 1843, 1846, 1915  
 \l\_@@\_previous\_key\_str 300, 302, 335, 337,  
 343, 345, 598, 600, 656, 658, 698, 721, 779,  
 1586, 1824, 2165, 2172, 2536, 2538, 2557, 2559  
 \@@\_qedhere: ..... 1431, 1432  
 \l\_@@\_qedhere\_bool ..... 281,  
 1047, 1056, 1057, 1079, 1083, 1084, 1202, 1431  
 \@@\_qedhere\_i: ..... 1058, 1085, 1433  
 \l\_@@\_replace\_left\_brace\_by\_tl .....  
 ..... 465, 1181, 1291  
 \@@\_restore:N ..... 175, 1055, 1056, 1083  
 \g\_@@\_right\_x\_dim .....  
 .... 952, 1303, 1304, 1319, 1320, 1741, 1954  
 \@@\_save:N ..... 159, 1046, 1047, 1079  
 \l\_@@\_sbwi\_bool ..... 276, 470, 1164  
 \@@\_scan\_arrows: ..... 961, 1442  
 \@@\_scan\_arrows\_i: ..... 1447, 1450, 1453  
 \@@\_set\_independent: .....  
 ..... 590, 612, 613, 614, 615, 616  
 \@@\_set\_independent\_bis: 594, 596, 2528, 2529  
 \@@\_set\_qedhere: ..... 818, 1432  
 \@@\_set\_seq\_of\_str\_from\_clist:Nn ....  
 ..... 154, 530, 557, 578, 649, 1802  
 \l\_@@\_show\_node\_names\_bool ..... 332, 894  
 \c\_@@\_showlabels\_loaded\_bool ..... 2243  
 \@@\_sort\_seq:N . 130, 546, 553, 574, 639, 1798  
 \l\_@@\_start\_adjust\_dim . 363, 685, 688, 1689  
 \g\_@@\_static\_col\_int .....  
 ..... 260, 756, 757, 825, 984, 989, 991  
 \g\_@@\_static\_col\_int\_seq .... 259, 756, 983  
 \l\_@@\_status\_arrow\_str ..... 601,  
 625, 680, 726, 747, 1474, 1504, 1531, 1548  
 \@@\_strcmp:nn ..... 123, 127, 136  
 \l\_@@\_string\_Arrow\_for\_msg\_str .....  
 ..... 272, 273, 323,  
 2054, 2126, 2135, 2166, 2175, 2182, 2190, 2266  
 \l\_@@\_subequations\_bool 292, 437, 1161, 1344  
 \@@\_tag ..... 797, 1377  
 \l\_@@\_tag\_next\_line\_bool .....  
 ..... 280, 774, 1048, 1051, 1429  
 \l\_@@\_tag\_star\_bool .....  
 ..... 279, 1046, 1055, 1060, 1203, 1393  
 \l\_@@\_tag\_tl ... 1019, 1021, 1201, 1381, 1392  
 \@@\_tagnextline: ..... 800, 1426  
 \l\_@@\_tags\_clist 262, 263, 266, 267, 432,  
 433, 450, 451, 453, 454, 1017, 1255, 1256,  
 1374, 1376, 1383, 1389, 1414, 1415, 1421, 1422  
 \@@\_test\_if\_to\_tag: ..... 264, 817  
 \c\_@@\_tikz\_code\_down\_tl .. 2439, 2487, 2569  
 \l\_@@\_tikz\_code\_tl .....  
 .... 325, 664, 1731, 1732, 1793, 2548, 2569  
 \c\_@@\_tikz\_code\_up\_tl .... 2350, 2401, 2548  
 \c\_@@\_tikz\_code\_wrap\_lines\_tl .. 1731, 1736  
 \@@\_tmpa:nnn ..... 1707, 1733  
 \l\_@@\_type\_col\_str .. 805, 822, 823, 836, 837  
 \l\_@@\_type\_env\_str ..... 741,  
 742, 917, 918, 1107, 1143, 1144, 2021,  
 2028, 2056, 2066, 2076, 2108, 2117, 2128,  
 2137, 2145, 2151, 2157, 2176, 2184, 2191,  
 2196, 2215, 2220, 2235, 2241, 2251, 2260, 2267  
 \c\_@@\_typedref\_loaded\_bool ..... 1155  
 \l\_@@\_up\_and\_down\_radius\_dim .....  
 ..... 296, 297, 2333, 2354, 2372,  
 2388, 2394, 2395, 2407, 2417, 2423, 2424,  
 2431, 2432, 2433, 2443, 2463, 2472, 2478,  
 2479, 2493, 2503, 2508, 2509, 2518, 2519, 2521  
 \@@\_update\_x:nn ..... 1551, 1604, 1756  
 \c\_@@\_varwidth\_loaded\_bool .... 2540, 2561  
 \@@\_warning:n ..... 32  
 \l\_@@\_wrap\_lines\_bool .....  
 ..... 457, 1730, 1951, 2547, 2568  
 \l\_@@\_x\_dim ..... 748,  
 1527, 1603, 1663, 1674, 1687, 1696, 1767, 1774  
 \g\_@@\_x\_final\_dim ... 1631, 1644, 1673, 1695  
 \g\_@@\_x\_initial\_dim .. 1630, 1641, 1662, 1686  
 \l\_@@\_xoffset\_dim ..... 359,  
 683, 1588, 1795, 1827, 1942, 1945, 1946, 1983  
 \g\_@@\_y\_final\_dim .....  
 ... 1633, 1645, 1653, 1666, 1677, 1698, 1699  
 \g\_@@\_y\_initial\_dim .....  
 ... 1632, 1642, 1653, 1666, 1677, 1689, 1690  
 \l\_@@\_ygap\_dim ..... 192, 314  
 \l\_@@\_ystart\_dim ..... 189, 316  
 \{ ..... 62, 71, 763,  
 939, 1052, 1258, 2000, 2014, 2019, 2021,  
 2028, 2035, 2043, 2048, 2056, 2057, 2066,  
 2067, 2076, 2077, 2087, 2088, 2098, 2099,  
 2112, 2120, 2129, 2137, 2146, 2152, 2158,  
 2166, 2177, 2184, 2191, 2197, 2202, 2208,  
 2215, 2221, 2228, 2235, 2241, 2251, 2258,  
 2260, 2272, 2279, 2280, 2291, 2327, 2583, 2590  
 \{ 411, 2009, 2021, 2028, 2034, 2035, 2056, 2066,  
 2076, 2108, 2117, 2128, 2137, 2145, 2151,  
 2152, 2157, 2158, 2176, 2184, 2191, 2196,  
 2215, 2220, 2221, 2235, 2241, 2251, 2260, 2267  
 \} ..... 2021, 2028, 2034, 2035, 2056, 2066,  
 2076, 2108, 2117, 2128, 2137, 2145, 2151,  
 2152, 2157, 2158, 2176, 2184, 2191, 2196,  
 2215, 2220, 2221, 2235, 2241, 2251, 2260, 2267

$\sqcup$ . . . . .	53, 2012, 2013, 2020, 2027, 2054, 2055, 2098, 2109, 2118, 2119, 2126, 2127, 2135, 2143, 2144, 2175, 2182, 2190, 2207, 2229, 2234, 2240, 2250, 2257, 2259, 2266, 2290, 2328
<b>A</b>	
$\backslash A$ . . . . .	496, 1906
$\backslash arabic$ . . . . .	1040
$\backslash Arrow$ . . . . .	273, 2098
$\backslash AtBeginDocument$ . . . . .	102, 224, 414
<b>B</b>	
$\backslash begin$ . . . . .	790, 1161, 1306, 1635, 1710, 1761, 1852, 1876, 1933, 1971, 2359, 2380, 2448, 2464
$\backslash belowdisplayskip$ . . . . .	1336
$\backslash bgroup$ . . . . .	847, 851, 925, 1235
bool commands:	
$\backslash bool_gset\_true:N$ . . . . .	93
$\backslash bool\_if:N\TF$ . . . . .	74, 84, 226, 436, 780, 782, 790, 815, 818, 829, 838, 844, 852, 894, 930, 944, 1014, 1025, 1030, 1048, 1057, 1060, 1064, 1072, 1084, 1155, 1156, 1161, 1164, 1193, 1213, 1227, 1237, 1277, 1333, 1344, 1345, 1364, 1404, 1608, 1612, 1688, 1697, 1882, 1886, 2173, 2243, 2540, 2561
$\backslash bool\_if:n\TF$ . . . . .	416, 674, 1147, 1247, 1331, 1384, 1394, 1409, 1418, 1486, 1496, 1516, 1530, 1548, 1652, 1730, 1951
$\backslash bool\_if\_p:N$ . . . . .	1394
$\backslash bool\_new:N$ . . . . .	24, 25, 95, 110, 244, 245, 246, 276, 279, 280, 281, 283, 284, 285, 286, 292, 1119
$\backslash bool\_set:Nn$ . . . . .	1393
$\backslash bool\_set\_false:N$ . . . . .	117, 774, 915, 1051, 1202, 1203, 1518, 1589, 1590, 1620, 1624, 1828, 1829, 2547, 2568
$\backslash bool\_set\_true:N$ . . . . .	96, 113, 437, 914, 968, 1141, 1167, 1170, 1429, 1431, 1460, 1514, 1593, 1594, 1597, 1598, 1619, 1623, 1627, 1628, 1834, 1835, 1837, 1838
$\backslash c\_false\_bool$ . . . . .	1247
$\backslash l\_tmpa\_bool$ . . . . .	1619, 1620, 1627, 1688
$\backslash l\_tmpb\_bool$ . . . . .	1623, 1624, 1628, 1697
box commands:	
$\backslash box\_clear\_new:N$ . . . . .	1187, 1234, 1266
$\backslash box\_dp:N$ . . . . .	1286
$\backslash box\_ht:N$ . . . . .	1285
$\backslash box\_set\_to\_last:N$ . . . . .	1261
$\backslash box\_use:N$ . . . . .	1263, 2393, 2422, 2480, 2510
$\backslash box\_use\_drop:N$ . . . . .	1273, 1282, 1301
$\backslash box\_wd:N$ . . . . .	860, 1186, 1265, 1269, 1270, 2387, 2417, 2471, 2503
$\backslash g\_tmpa\_box$ . . . . .	2378, 2387, 2393, 2415, 2417, 2422, 2460, 2471, 2480, 2501, 2503, 2510
$\backslash l\_tmpa\_box$ . . . . .	1176, 1186, 1261, 1263, 1265, 1267
<b>C</b>	
$\backslash catcode$ . . . . .	21, 2594
char commands:	
$\backslash char\_generate:nn$ . . . . .	162, 178
clist commands:	
$\backslash clist\_clear:N$ . . . . .	432, 1374, 1376, 1389, 1415, 1422
$\backslash clist\_count:N$ . . . . .	1919
$\backslash clist\_gput\_right:Nn$ . . . . .	1917
$\backslash clist\_if\_in:Nn\TF$ . . . . .	266, 451, 462, 1017, 1255
$\backslash clist\_map\_inline:nn$ . . . . .	104
$\backslash clist\_new:N$ . . . . .	262, 408
$\backslash clist\_pop:Nn$ . . . . .	1928, 1930
$\backslash clist\_put\_left:Nn$ . . . . .	454
$\backslash clist\_put\_right:Nn$ . . . . .	419
$\backslash clist\_remove\_all:Nn$ . . . . .	453
$\backslash clist\_reverse:N$ . . . . .	1929
$\backslash clist\_set:Nn$ . . . . .	263, 267, 409, 433, 450, 1256, 1383, 1414, 1421
$\backslash clist\_sort:Nn$ . . . . .	1922
$\backslash g\_tmpa\_clist$ . . . . .	1917, 1919, 1922, 1928, 1929, 1930, 1931
$\backslash coordinate$ . . . . .	1067, 1075, 1088
$\backslash cr$ . . . . .	933, 1096, 1248, 2257
cs commands:	
$\backslash cs\_generate\_variant:Nn$ . . . . .	35, 100, 101, 1554, 1735
$\backslash cs\_gset:Npx$ . . . . .	1020
$\backslash cs\_if\_exist:N\TF$ . . . . .	744, 1036, 1413, 1420, 2543, 2564
$\backslash cs\_if\_free:N\TF$ . . . . .	1313, 1843, 1846, 1915
$\backslash cs\_new:Npn$ . . . . .	1777
$\backslash cs\_new\_protected:Npn$ . . . . .	27, 28, 29, 31, 32, 33, 34, 123, 127, 130, 145, 154, 159, 175, 233, 264, 298, 307, 590, 596, 654, 705, 711, 717, 736, 738, 803, 865, 904, 911, 936, 950, 986, 995, 998, 1007, 1010, 1105, 1125, 1132, 1138, 1251, 1362, 1373, 1375, 1398, 1426, 1431, 1432, 1433, 1442, 1453, 1494, 1555, 1557, 1580, 1705, 1728, 1756, 1808, 1814, 1820, 1902, 1911, 1968, 2298
$\backslash cs\_set:Npn$ . . . . .	952, 965, 1707
$\backslash cs\_set:Npx$ . . . . .	1024
$\backslash cs\_set\_eq:NN$ . . . . .	236, 370, 371, 763, 784, 795, 796, 797, 798, 799, 800, 813, 966, 967, 1059, 1061, 1191, 1432, 1436, 1437
$\backslash cs\_set\_protected:Npn$ . . . . .	695
$\backslash cs\_to\_str:N$ . . . . .	163, 179
<b>D</b>	
$\backslash d$ . . . . .	1906
$\backslash DeclareOption$ . . . . .	96, 97
dim commands:	
$\backslash dim\_case:nn\TF$ . . . . .	2403, 2489
$\backslash dim\_compare:nNn\TF$ . . . . .	1093, 1268, 1319, 1746, 1749, 1955, 2352, 2373, 2385, 2441, 2469
$\backslash dim\_compare\_p:nNn$ . . . . .	1653
$\backslash dim\_eval:n$ . . . . .	1664, 1675, 1689, 1698
$\backslash dim\_gadd:Nn$ . . . . .	2388, 2472
$\backslash dim\_gset:Nn$ . . . . .	1265, 1270, 1320, 1641, 1642, 1644, 1645, 1767, 2387, 2416, 2471, 2502
$\backslash dim\_gset\_eq:NN$ . . . . .	1304, 2386, 2470
$\backslash dim\_gzero\_new:N$ . . . . .	1264, 1303, 1630, 1631, 1632, 1633
$\backslash dim\_max:nn$ . . . . .	1099, 1767, 1862
$\backslash dim\_min:nn$ . . . . .	2376
$\backslash dim\_new:N$ . . . . .	294, 296
$\backslash dim\_set:Nn$ . . . . .	297, 683, 688, 689, 1099, 1186, 1283, 1318, 1527, 1603, 1740, 1745, 1861, 1948, 1950, 1953, 2341, 2358, 2371, 2375, 2447, 2462



<code>\dim_set_eq:NN</code>	295, 1179, 1210, 1228, 1229, 1232, 1289, 1747, 1774, 1858, 1859, 1956, 2339
<code>\dim_use:N</code>	1662, 1663, 1673, 1674, 1686, 1687, 1690, 1695, 1696, 1699, 1752, 1864, 1866, 1884, 1888, 1959
<code>\dim_zero:N</code>	764, 2338
<code>\dim_zero_new:N</code>	748, 1185, 1209, 1225
<code>\c_max_dim</code>	295, 1304, 1527, 1603, 2339, 2352, 2405, 2441, 2491
<code>\g_tmpa_dim</code>	1767, 1774, 2386, 2387, 2388, 2392, 2416, 2421, 2470, 2471, 2472, 2480, 2502, 2510
<code>\l_tmpa_dim</code>	1099, 1100, 1283, 1292, 1318, 1319, 1320, 1740, 1746, 1747, 1749, 1752, 1858, 1861, 1862, 1864, 1866, 1948, 1950, 1955, 1956, 1959, 2358, 2359, 2371, 2375, 2376, 2380, 2447, 2448, 2462, 2464
<code>\l_tmpb_dim</code>	1745, 1746, 1747, 1859, 1864, 1953, 1955, 1956
<code>\c_zero_dim</code>	190, 191, 996, 1093, 1099, 1179, 1289, 1749, 2373, 2385, 2411, 2469, 2497
<code>\displaystyle</code>	829, 930, 1193
<code>\displaywidth</code>	1210, 1229, 1232
<code>\DispWithArrows</code>	1125, 1358
DispWithArrows commands:	
<code>\DispWithArrows_i</code>	1129, 1130, 1132
<code>\DispWithArrows_ii</code>	1135, 1136, 1138
<code>\draw</code>	326, 665, 1738, 1942, 1982, 2354, 2390, 2407, 2419, 2429, 2443, 2475, 2493, 2505, 2515
<b>E</b>	
<code>\egroup</code>	940, 941, 1259, 1271
else commands:	
<code>\else:</code>	921
<code>\end</code>	944, 1000, 1115, 1324, 1344, 1345, 1647, 1721, 1769, 1868, 1890, 1961, 1986, 2362, 2383, 2451, 2467
<code>\endDispWithArrows</code>	1251, 1360
<code>\endtikzpicture</code>	1327, 1650, 1724, 1772, 1871, 1893, 1964, 1989
<code>\endWithArrows</code>	936
exp commands:	
<code>\exp_args:NNo</code>	1587
<code>\exp_args:Nnx</code>	1904
<code>\exp_args:No</code>	1158, 1587
<code>\exp_args:NV</code>	1107, 1732, 1931
<code>\ExplSyntaxOff</code>	2595
<code>\ExplSyntaxOn</code>	20
<b>F</b>	
<code>\fi</code>	1168, 1171
fi commands:	
<code>\fi:</code>	923, 1207, 1220
<code>\foreach</code>	1913, 1980
<b>G</b>	
<code>\globaldefs</code>	441, 1111
group commands:	
<code>\group_align_safe_begin:</code>	992
<code>\group_align_safe_end:</code>	1013
<code>\group_begin:</code>	440, 906, 963, 1110, 1127, 1178, 1190, 1288, 1435, 1444, 1559, 1582, 1825, 2314
<code>\group_end:</code>	443, 947, 970, 1113, 1183, 1197, 1295, 1348, 1439, 1451, 1578, 1703, 1900, 2322
<b>H</b>	
<code>\halign</code>	850
hbox commands:	
<code>\hbox_gset:Nn</code>	2378, 2415, 2460, 2501
<code>\hbox_overlap_left:n</code>	1058, 1062, 1085
<code>\hbox_overlap_right:n</code>	896
<code>\hbox_set:Nn</code>	1176, 1188, 1267
<code>\hbox_to_wd:nn</code>	1219, 1275, 1280
<code>\hbox_unpack_clear:N</code>	1267
<code>\hfil</code>	822, 836, 837, 880, 1279, 1296, 1298
<code>\hfill</code>	823
<b>I</b>	
<code>\ialign</code>	846
if commands:	
<code>\if_mode_math:</code>	921, 1205
<code>\if_mode_vertical:</code>	1217
<code>\ignorespacesafterend</code>	1351
<code>\input</code>	6, 7
int commands:	
<code>\int_case:nn</code>	924, 1591, 1830
<code>\int_compare:nNnTF</code>	134, 811, 833, 954, 956, 958, 975, 989, 1367, 1445, 1478, 1480, 1508, 1536, 1543, 1601, 1625, 1661, 1672, 1685, 1694, 1849, 2269
<code>\int_compare:nTF</code>	608, 624, 1526, 1533, 1550, 1575, 1919, 1924
<code>\int_compare_p:n</code>	1488, 1500, 1502
<code>\int_compare_p:nNn</code>	676, 678, 1490, 1498, 1656
<code>\int_eval:n</code>	974
<code>\int_gdecr:N</code>	809
<code>\int_gincr:N</code>	720, 824, 855, 976, 1019
<code>\int_gset:Nn</code>	825, 978, 980, 982, 984
<code>\int_gset_eq:NN</code>	786
<code>\int_gzero:N</code>	751, 753, 755, 757, 856
<code>\int_incr:N</code>	1484, 1576
<code>\int_new:N</code>	249, 250, 251, 254, 256, 258, 260, 289
<code>\int_set:Nn</code>	252, 308, 339, 349, 399, 400, 401, 609, 659, 724, 770, 785, 959, 1448, 1461, 1467, 1471, 1561, 1563, 1564, 1570, 1574, 1823, 2546, 2567
<code>\int_set_eq:NN</code>	1519, 1520, 1521, 1538
<code>\int_step_inline:nnn</code>	1758
<code>\int_step_variable:nNn</code>	1311
<code>\int_until_do:nNnn</code>	1462, 1565
<code>\int_use:N</code>	731, 733, 825, 872, 876, 886, 890, 897, 965, 1067, 1075, 1088, 1465, 1469, 1473, 1476, 1568, 1572, 1584, 1608, 1612, 1614, 1618, 1622, 1777, 2020, 2027, 2055, 2108, 2117, 2120, 2127, 2144, 2183, 2215, 2266
<code>\int_zero_new:N</code>	765, 766, 767, 768, 769, 1455, 1456, 1457, 1560, 1562
<code>\c_one_int</code>	454, 770, 1461, 1498, 2546, 2567
<code>\l_tmpa_int</code>	724, 725, 1311, 1314, 1317, 2259
<code>\c_zero_int</code>	1508
<code>\itshape</code>	216
<b>J</b>	
<code>\jot</code>	235, 361, 1050
<b>K</b>	
<code>\k</code>	1980, 1983

keys commands:		\nonumber . . . . . 796, 1376
\keys_define:nn . . . . .		\normalbaselines . . . . . 953
. . . 37, 309, 397, 423, 480, 505, 543, 549,		\notag . . . . . 795, 1052, 1374
568, 605, 662, 1778, 2304, 2331, 2526, 2533		\nulldelimiterspace . . . . . 1179, 1289
\keys_if_exist:nnTF . . . . . 2301		
\l_keys_key_tl . . . . 46, 53, 302, 600, 640,		<b>O</b>
658, 1999, 2007, 2048, 2053, 2066, 2076,		\omit . . . . . 2027
2086, 2097, 2163, 2170, 2317, 2347, 2582, 2588		\openup . . . . . 235, 1050
\keys_set:nn . . . . . 368,		
699, 722, 781, 783, 1554, 1826, 2545, 2566		<b>P</b>
\keys_set_known:nn . . . . . 1556, 2307		\p . . . . . 2355, 2356,
\l_keys_value_tl . . . . 592, 2277, 2545, 2566		2364, 2368, 2391, 2396, 2408, 2409, 2420,
		2425, 2430, 2434, 2444, 2445, 2453, 2457,
		2476, 2477, 2494, 2495, 2506, 2507, 2516, 2517
		\path 1742, 1751, 1949, 1958, 2367, 2413, 2456, 2499
<b>L</b>		peek commands:
\label . . . . . 798, 799, 1191, 1400, 2240, 2250		\peek_meaning:NTF . . . . .
\langle . . . . . 411, 2011		. . 707, 713, 907, 996, 1128, 1134, 1810, 1816
\lbrace . . . . . 411, 468, 2009		\peek_meaning_ignore_spaces:NTF . . . . 1000
\lbrack . . . . . 411, 2010		\peek_meaning_remove:NTF . . . . . 993
\lceil . . . . . 411, 2012		\pgfextra . . . . . 1742, 1949
\left . . . . . 1181, 1291		\pgfkeysvalueof . . . . . 1741
\lfloor . . . . . 411, 2012		prg commands:
\lggroup . . . . . 411, 2010		\prg_do_nothing: . . . . .
\linewidth . . . . . 1209, 1210, 1219, 1228		. . . . . 236, 627, 629, 631, 633, 635, 1061
\lmoustache . . . . . 411, 2011		\prg_replicate:nn . . . . . 991
lua commands:		\ProcessKeysOptions . . . . . 50
\lua_now:n . . . . . 124		\ProcessOptions . . . . . 98
\lVert . . . . . 419, 2013		prop commands:
\lvert . . . . . 419, 2013		\prop_gclear_new:N . . . . . 730
		\prop_get:NnN . . . . .
		1464, 1468, 1472, 1475, 1567, 1571, 1583, 1613
		\prop_gset_eq:NN . . . . . 732
		\prop_put:Nnn . . . 723, 725, 726, 727, 728, 729
		\l_tmpa_prop 723, 725, 726, 727, 728, 729, 734
		\ProvidesExplPackage . . . . . 12
<b>M</b>		
math commands:		<b>Q</b>
\c_math_toggle_token . . . . .		\qed . . . . . 1436
. . . 826, 832, 929, 932, 1180, 1182, 1192,		\qedhere . . . . . 1432
1196, 1214, 1221, 1290, 1294, 1335, 1338, 1341		\qedsymbol . . . . . 1436
\mathsurround . . . . . 764		\quad . . . . . 1069
MH commands:		
\MH_if_boolean:nTF . . . . .		
. . . . . 1149, 1332, 1386, 1388, 1411		
\MH_set_boolean_T:n . . . . . 1152		
msg commands:		<b>R</b>
\msg_error:nn . . . . . 31		regex commands:
\msg_error:nnn . . . . . 34		\regex_match:nnTF . . . . . 496, 1905
\msg_fatal:nn . . . . . 33		\relax . . . . . 1438
\msg_info:nn . . . . . 77		\RequirePackage . . . . . 2, 3, 11, 17
\msg_line_number: . . . . . 729		\right . . . . . 1181, 1293
\msg_new:nnn . . . . . 27		
\msg_new:nnnn . . . . . 28		<b>S</b>
\msg_redirect_name:nnn . . . . . 30		scan commands:
\msg_set:nnn . . . . . 2315		\scan_stop: . . . . . 988, 1101
\msg_warning:nn . . . . . 32		seq commands:
MT commands:		\seq_clear:N . . . . . 147, 1522, 1524, 1539
\MT_showonlyrefs_false: . . . . . 1151		\seq_clear_new:N . . . . 773, 787, 1458, 1459
\MT_showonlyrefs_true: . . . . . 1332		\seq_count:N . . . . . 975
\MultiArrow . . . . . 966, 2203, 2207, 2290		\seq_gpop_right:NN 971, 972, 977, 979, 981, 983
\myfiledate . . . . . 14		\seq_gput_left:Nn . . . . . 486
\myfileversion . . . . . 15		\seq_gput_right:Nn . . . . .
		. . . . . 248, 750, 752, 754, 756, 758, 973
<b>N</b>		\seq_if_empty:NTF . . . . . 1022, 1402
\narrowragged . . . . . 2360, 2381, 2449, 2465		\seq_if_in:NnTF . . . 346, 484, 640, 1617, 1621
\NewDocumentCommand . . 692, 702, 1377, 1805, 2295		\seq_item:Nn . . . . . 165, 181
\NewDocumentEnvironment . . . . . 901, 1122, 1355		
\noalign . . . . . 1097		
\node . . . . . 868, 882		
\nointerlineskip . . . . . 1218, 1262		

<code>\seq_map_function:NN</code> .....	1044	<code>\@currentenv</code> .....	742
<code>\seq_map_inline:Nn</code> .....	148	<code>\@eqnnum</code> .....	1070
<code>\seq_new:N</code> .....	247, 253, 255, 257, 259, 274, 529, 556, 577, 648, 1801	<code>\@ifclassloaded</code> .....	76, 86
<code>\seq_pop_left:NN</code> .....	164, 180	<code>\@ifpackageloaded</code> .....	79, 89, 112
<code>\seq_pop_right:NN</code> .....	760	<code>\c@equation</code> .....	1019
<code>\seq_pop_right:NNTF</code> .....	805	<code>\cref@constructprefix</code> .....	1032
<code>\seq_put_left:Nn</code> .....	150, 1523, 1525, 1535, 1540, 1544	<code>\cref@currentlabel</code> .....	1033
<code>\seq_put_right:Nn</code> .....	228, 229, 347, 1408, 2309, 2310, 2312, 2578, 2579	<code>\cref@equation@alias</code> .....	1036, 1037
<code>\seq_remove_all:Nn</code> .....	338, 348	<code>\cref@result</code> .....	1032, 1040
<code>\seq_set_eq:NN</code> .....	152, 759	<code>\hyper@refstepcounter</code> .....	1028
<code>\seq_set_from_clist:Nn</code> .....	156	<code>\if@inlabel</code> .....	1166
<code>\seq_set_split:Nnn</code> .....	101, 161, 177, 788	<code>\if@minipage</code> .....	1169
<code>\seq_sort:Nn</code> .....	132	<code>\intertext@</code> .....	1156
<code>\seq_use:Nnnn</code> .....	167, 170, 173, 183, 762, 2062, 2072, 2082, 2093, 2104, 2286	<code>\p@equation</code> .....	1024, 1041
<code>\l_tmpa_seq</code> ..	147, 150, 152, 161, 164, 165, 167, 170, 173, 177, 180, 181, 183, 759, 760, 762	<code>\pgf@x</code> .....	1318, 1641, 1644, 1741, 1767, 1858, 1862, 1884, 1888, 1954
<code>\setbox</code> .....	1235	<code>\pgf@y</code> .....	1642, 1645, 1859, 1866, 1884, 1888
skip commands:		<code>\pgfutil@firstofone</code> .....	1317, 1640, 1643, 1739, 1766, 1857, 1860, 1881, 1885, 1947
<code>\skip_horizontal:N</code> .....	853	<code>\protected@edef</code> .....	1033
<code>\skip_horizontal:n</code> .....	859, 1278, 1300	<code>\qed@elt</code> .....	1437
<code>\skip_vertical:N</code> .....	1336	<code>\QED@stack</code> .....	1438
<code>\skip_vertical:n</code> .....	1100	<code>\setQED@elt</code> .....	1437
<code>\skip_zero:N</code> .....	776	<code>\spread@equation</code> .....	233, 236, 793
<code>\c_zero_skip</code> .....	838, 1238, 1244	<code>\sr@name</code> .....	1155
<code>\small</code> .....	216, 897	<code>\tagform@</code> .....	1061
sort commands:		<code>\This@name</code> .....	1027
<code>\sort_return_same:</code> .....	142, 1926	<code>\tikz@library@external@loaded</code> .....	744
<code>\sort_return_swapped:</code> .....	141, 1925	<code>\tikz@parse@node</code> .....	1317, 1739, 1947
str commands:		<code>\tikz@scan@one@point</code> .....	1640, 1643, 1766, 1857, 1860, 1881, 1885
<code>\c_backslash_str</code> .....	324, 2110, 2119	<code>\tikz@text@width</code> .....	1742, 1949
<code>\str_case:nnTF</code> .....	2336	tex commands:	
<code>\str_clear:N</code> .....	645	<code>\tex_strcmp:D</code> .....	128
<code>\str_clear_new:N</code> .....	698, 721, 741, 746, 747, 749, 761, 779, 917, 1143, 1586, 1824	<code>\theequation</code> .....	1021, 1059
<code>\str_const:Nn</code> .....	1992, 1994	<code>\tikz</code> .....	867, 881, 1066, 1074, 1087
<code>\str_count:N</code> .....	785	<code>\tikzpicture</code> .....	1309, 1638, 1713, 1764, 1855, 1879, 1936, 1974
<code>\str_if_empty:NNTF</code> .....	300, 335, 343, 598, 656, 875, 889, 2536, 2557	<code>\tikzset</code> .....	185, 195, 204, 209, 331, 352, 666, 745, 964, 1781
<code>\str_if_eq:nnTF</code> .....	431, 592, 822, 823, 836, 837, 1107	tl commands:	
<code>\str_if_eq_p:nn</code> .....	680, 1504, 1531, 1548	<code>\c_empty_tl</code> .....	353, 667
<code>\str_lower_case:n</code> .....	137, 138	<code>\c_novalue_tl</code> .....	243, 857, 1130, 1174, 1272
<code>\str_new:N</code> .....	270, 272, 290	<code>\tl_clear_new:N</code> .....	777, 778, 1201
<code>\str_set:Nn</code> .....	165, 181, 271, 273, 322, 323, 337, 345, 483, 601, 625, 642, 742, 762, 771, 918, 1027, 1144, 1155, 2538, 2559	<code>\tl_const:Nn</code> .....	1736, 2350, 2401, 2439, 2487
<code>\str_set_eq:NN</code> .....	302, 487, 600, 658	<code>\tl_gset:Nn</code> .....	1659, 1670, 1683, 1692, 1742, 1863, 1865, 1883, 1887, 1949
<code>\l_tmpa_str</code> .....	165, 166, 169, 172, 181, 182, 483, 484, 486, 487, 642, 645, 2056	<code>\tl_head:n</code> .....	461
<code>\strut</code> .....	862	<code>\tl_if_empty:NNTF</code> .....	1019, 1021, 1381, 1743, 1950
sys commands:		<code>\tl_if_empty:nTF</code> .....	493
<code>\sys_if_engine luatex:TF</code> .....	121	<code>\tl_if_eq:NNTF</code> .....	857, 1174, 1272
<b>T</b>		<code>\tl_if_eq:nNTF</code> .....	1840
<code>\tabskip</code> .....	838, 1236, 1241, 1244	<code>\tl_if_novalue:nTF</code> .....	1158
<code>\tag</code> .....	797, 1379, 2227, 2229, 2234	<code>\tl_new:N</code> .....	242, 287, 288
<code>\tagnextline</code> .....	800, 1428	<code>\tl_put_right:Nn</code> .....	100, 488, 490
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:		<code>\tl_set:Nn</code> .....	461, 465, 497, 1158, 1392, 1606, 1611
<code>\@</code> .....	21, 2594	<code>\tl_set_eq:NN</code> .....	243, 1731, 2548, 2569
<code>\@currentlabel</code> .....	1024	<code>\tl_to_str:N</code> .....	2313
		<code>\tl_to_str:n</code> .....	150, 2309, 2311
		<code>\g_tmpa_tl</code> .....	1020, 1024, 1041, 1059, 1659, 1683, 1702, 1742, 1743, 1745, 1863, 1883, 1896, 1949, 1950



<b>10</b>	<b>Examples</b>	<b>26</b>
10.1	<code>\MoveEqLeft</code> . . . . .	26
10.2	Modifying the shape of the nodes . . . . .	26
10.3	Examples with the option <code>tikz-code</code> . . . . .	27
10.3.1	Example 1 . . . . .	27
10.3.2	Example 2 . . . . .	27
10.3.3	Example 3 . . . . .	28
10.4	Automatic numbered loop . . . . .	29
<b>11</b>	<b>Implementation</b>	<b>30</b>
11.1	Declaration of the package and extensions loaded . . . . .	30
11.2	The packages <code>footnote</code> and <code>footnotehyper</code> . . . . .	31
11.3	The class option <code>leqno</code> . . . . .	32
11.4	Some technical definitions . . . . .	32
11.5	Variables . . . . .	36
11.6	The definition of the options . . . . .	38
11.7	The command <code>\Arrow</code> . . . . .	47
11.8	The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code> . . . . .	48
11.8.1	Code before the <code>\halign</code> . . . . .	48
11.8.2	The construction of the preamble of the <code>\halign</code> . . . . .	51
11.8.3	The environment <code>{WithArrows}</code> . . . . .	54
11.8.4	After the construction of the <code>\halign</code> . . . . .	55
11.8.5	The command of end of row . . . . .	56
11.8.6	The environment <code>{DispWithArrows}</code> . . . . .	60
11.9	The commands <code>\tag</code> , <code>\notag</code> , <code>\label</code> , <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code> . . . . .	65
11.10	We draw the arrows . . . . .	67
11.11	The command <code>\Arrow</code> in code-after . . . . .	76
11.12	The command <code>\MultiArrow</code> in code-after . . . . .	78
11.13	The error messages of the package . . . . .	80
11.14	The command <code>\WithArrowsNewStyle</code> . . . . .	85
11.15	The options up and down . . . . .	86
<b>12</b>	<b>History</b>	<b>91</b>
	<b>Index</b>	<b>93</b>