

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

F. Pantigny
fpantigny@wanadoo.fr

February 14, 2023

Abstract

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

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

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

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

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

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

```


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


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


```

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

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. 17.

*This document corresponds to the version 2.8a of `witharrows`, at the date of 2023/02/14.

¹If you use Overleaf, Overleaf will do automatically the right number of compilations.

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 number² of rows the arrow must jump (the default value is, of course, 1).

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

In fact, it's possible to change more drastically the shape or the arrows with the option `tikz-code` (presented p. 23).

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

```

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

```

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

In the environments `{DispWithArrows}` and `{DispWithArrows*}`, there is an option `wrap-lines`. With this option, the lines of the labels are automatically wrapped on the right: see p. 20.

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

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

Without the option `displaystyle`:

```

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

```

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

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

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

The same example with the option `displaystyle`:

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

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

```

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

```

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

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

```

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

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).

```

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

```

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

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

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

2 Numbers of columns

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

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

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

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

```

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

```

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

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

```

 $\begin{DispWithArrows*}[format = cCcCc,
wrap-lines,
interline=1mm]
k & \leq & t & \leq & k+1 \\
\frac{1}{k+1} & \leq & \frac{1}{t} & \leq & \frac{1}{k} \\
\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*}$ 

```

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

3 Precise positioning of the arrows

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

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

The nodes of the left are at the end of each line of text. These nodes will be called *left nodes*. The nodes of the right side are aligned vertically on the right side of the array. These nodes will be called *right nodes*.

By default, the arrows use the right nodes. We will say that they are in **rr** mode (*r* for *right*). These arrows are vertical (we will say that an arrow is *vertical* when its two ends have the same abscissa).

However, it's possible to use the left nodes, or a combination of left and right nodes, with one of the options **lr**, **rl** and **ll** (*l* for *left*). Those arrows are, usually, not vertical.

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

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

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

```

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

```

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

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

```

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

```

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

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

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

In an environment which uses the option `group` or the option `groups`, it’s still possible to give an option of position (`ll`, `lr`, `rl`, `rr` or `i`) to an individual arrow⁸. 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]`

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

⁸Such arrow will be called *independent* in the technical documentation

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).

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

4 The option “o” for individual arrows

Let's consider, in a given environment, two arrows called a and b . We will note i_a and i_b the numbers of the initial lines of a et b dans f_a and f_b the numbers of the final lines. Of course, we have $i_a \leq f_a$ and $i_b \leq f_b$

We will say that the arrow a covers the arrow b when $i_a \leq i_b \leq f_b \leq f_a$. We will also say that the arrow a is over the arrow b .

In the exemple on the right, the red arrow covers the blue one.

$$\begin{aligned}
 A &= B \\
 &= C \\
 &= D \\
 &= E
 \end{aligned}$$

On the local level, there exists a key **o**. This key is available only when the option **group** or the option **groups** is in force (cf. p. 8).

An arrow of type **o** is drawn with an horizontal shift (such as those set by **xoffset**) automatically computed by taking into account the arrows covered by our arrow.⁹

```


$$\begin{aligned}
 &\begin{array}{l}
 \text{\$}\backslash\begin{array}{l}
 A \& = B \quad \backslash\text{Arrow}\{one\}\backslash\text{Arrow}[o, \text{jump}=3]\{\text{direct}\} \ \backslash\ \\
 \& = C + C \ \backslash\text{Arrow}\{two\} \ \backslash\ \\
 \& = D + D + D \ \backslash\text{Arrow}\{three\} \ \backslash\ \\
 \& = E + E \ \backslash\ \\
 \& = F + F \\
 \end{array} \\
 \text{\$}\backslash\end{array}
 \end{array}$$


```

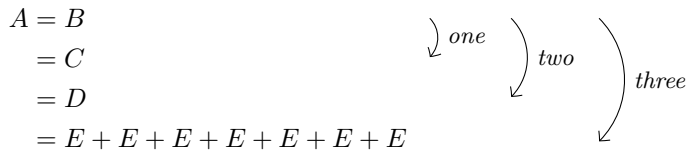
⁹Among the covered arrows, the independent ones (that is to say with an explicit key **rr**, **ll**, **lr**, **rl**, **i**, **up** or **down**) are not taken into account in the computation of the value of **xoffset**.

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

```

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

```



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

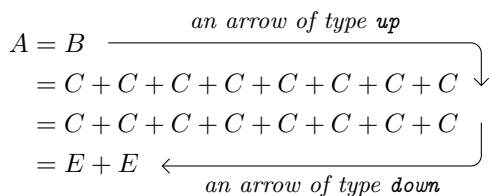
5 The options “up” and “down” for individual arrows

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

```

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

```



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

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

- The key `radius` is the radius of the rounded corner of the arrow.¹⁰
- 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);

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

- with a numerical value, the width of the arrow is directly fixed to that numerical value;
- with the value `min`, the width of the arrow is adjusted with respect to the contents of the label of the arrow.

```

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

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

```

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

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

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

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

6 Comparison with the environment `{aligned}`

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

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

```

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

¹¹In 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.¹²

```

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

```

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

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

```

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

```

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

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.

```

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

```

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

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

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

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

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

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

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

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

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

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

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.¹³

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

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

7 Arrows in nested environments

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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,
code-after = {\MultiArrow{1,...,4}{text}} ]
A & = B \\
& = C \\
```



```

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

```

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

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

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

```

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

```

```

\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}

```

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

9 The environment `{DispWithArrows}`

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

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

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

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

It's possible to use the command `\notag` (or `\nonumber`) to suppress a tag.
 It's possible to use the command `\tag` to put a special tag (e.g. `*`).
 It's also possible to put a label to the line of an equation with the command `\label`.
 These commands must be in the second column of the environment.

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

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

A link to the equation [\(*\)](#).¹⁶

If `amsmath` (or `mathtools`) is loaded, it's also possible to use `\tag*` which, as in `amsmath`, typesets the tag without the parentheses. For example, it's possible to use it to put the symbol `\square` of `amssymb`. This symbol is often used to mark the end of a proof.¹⁷

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

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

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

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

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

In fact, there is also another option `tagged-lines` which can be used to control the lines that will be tagged. The value of this option is a list of the numbers of the lines that must 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}
```

¹⁶In this document, the references have been customized with `\labelformat{equation}{\#1}` in the preamble.

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

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

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

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

```

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

$$\tag{5}$$

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*}`.¹⁹

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

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

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

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

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

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

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

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

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

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

$$= 2p^2 + p - 4p^2 - 4p - 1 + 4p^2 + 8p + 4 \quad (11)$$

$$= 2p^2 + 5p + 3 \quad (12)$$

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

```
\begin{DispWithArrows*}[displaystyle,wrap-lines]
S_n
& = \frac{1}{n} \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*}
```

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

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

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

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

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

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

be used directly after a `\item` of a LaTeX list. In this case, no vertical space is added before the environment.²⁰

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

```

\begin{enumerate}
\item
\begin{DispWithArrows}%
[displaystyle, wrap-lines, tagged-lines = last, fleqn, mathindent = 0 pt]
S_n
& = \frac{1}{n} \Re \left( \sum_{k=0}^{n-1} \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 $e^{i \frac{2\pi}{n}}$}
& = \frac{1}{n} \Re \left( \frac{1 - \bigl( e^{i \frac{\pi}{2n}} \bigr)^n}{1 - e^{i \frac{\pi}{2n}}} \right)
\Arrow{$\bigl( e^{i \frac{\pi}{2n}} \bigr)^n = e^{i \frac{\pi}{2}} = i$}
& = \frac{1}{n} \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{DispWithArrows}
\end{enumerate}

```

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

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

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

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

Concerning the references, the package `witharrows` is compatible with the extensions `autonum`, `cleveref`, `fancyref`, `hyperref`, `listbls`, `prettyref`, `refcheck`, `refstyle`, `showlabels`, `smartref`, `typedref` and `varioref`, and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.²¹

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

²⁰It's possible to disable this feature with the option `standard-behaviour-with-items`.

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

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

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

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

The following code is an example of multi-case equations.²³

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

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

(15)

(16)

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

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

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

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

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

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

²³The environment `{cases}` of `amsmath` is a way to compose such multi-cases equations. However, it's not possible to use the automatic numbering of equations with this environment. The environment `{numcases}` of the extension `cases` (written by Donald Arseneau) provides this possibility but, of course, it's not possible to draw arrows with this extension.

10 Advanced features

10.1 Use with plain-TeX

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

```
\input{witharrows}
```

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

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

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

10.2 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.²⁴

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

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

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}
```

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

$$\begin{array}{r}
3(2x + 4) = 6 \\
2x + 4 = 2 \\
2x = -2 \\
x = -1
\end{array}
\begin{array}{l}
\leftarrow \textcircled{\div 3} \\
\leftarrow \textcircled{-4} \\
\leftarrow \textcircled{\div 2}
\end{array}$$

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. 29.

10.3 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. 29.

10.4 The key `right-overlap`

New 2.8

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

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

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

```

 $\left\{ \begin{array}{l} 2x + 3y = 5 \\ -2x - 5y = 2 \end{array} \right. \begin{array}{l} \text{\textcircled{\div 3}} \\ \text{\textcircled{-4}} \\ \text{\textcircled{\div 2}} \end{array}$ 

```

$$\left\{ \begin{array}{l} 2x + 3y = 5 \\ -2x - 5y = 2 \end{array} \right. \begin{array}{l} \text{\textcircled{\div 3}} \\ \text{\textcircled{-4}} \\ \text{\textcircled{\div 2}} \end{array} \quad \left. \begin{array}{l} 2x + 3y = 5 \\ -2y = 7 \end{array} \right\}$$

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

10.5 Vertical positioning of the arrows

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

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

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

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

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

Here is for example the behaviour without the mechanism of `start-adjust` and `end-adjust`:

```

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

```

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

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

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

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

Since the version 2.1 of `witharrows`, an arrow of `jump` equal to 1 has a maximal length²⁵ 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{
$L_2 \ \gets L_2-L_1$
}$ 

```

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

```

$L_3 \gets L_3-L_1$ \\
$L_4 \gets L_4-L_1$ \\
$L_5 \gets L_5-L_1$ % 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}\}

```

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

10.6 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 preferentially. The package `footnote` has some drawbacks, in particular: it must be loaded after the package `xcolor` and it is not perfectly compatible with `hyperref`.

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

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

10.7 Option no-arrows

The option `no-arrows` is a convenience given to the user. With this option the arrows are not drawn. However, an analysis 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.

²⁶A footnote.

10.8 Note for the users of AUCTeX

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

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

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

10.9 Note for developpers

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}
```

11 Examples

11.1 `\MoveEqLeft`

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

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

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

11.2 A command `\DoubleArrow`

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

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

Example of use:

```

\begin{WithArrows}[groups]
A & = (a+b)^2 \DoubleArrow[tikz={font=\bfseries}]{expansion}{factorization} \\
& = a^2 + 2ab + b^2
\end{WithArrows}
```

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

11.3 Modifying the shape of the nodes

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

```

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

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

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

11.4 Examples with the option tikz-code

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

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.

11.4.1 Example 1

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

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

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

11.4.2 Example 2

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

²⁷If 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 \leftarrow \text{\Arrow{\$ \cos x = \Re(e^{ix})\$}} \\
 \dots\dots\dots

```

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

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

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

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

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

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

11.4.3 Example 3

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

```

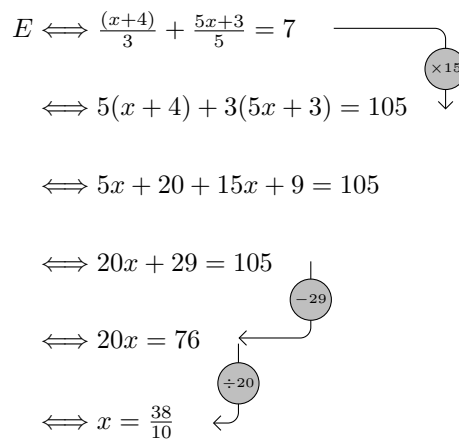
\begin{WithArrows}[ll,interline=5mm,xoffset=5mm,
 tikz-code = {\draw[rounded corners,
                  every node/.style = {circle,
                                        draw,
                                        auto = false,
                                        inner sep = 1pt,
          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

```



11.5 Automatic numbered loop

Assume we want to draw a loop of numbered arrows. In this purpose, it's possible to write a dedicated command `\NumberedLoop` which will do the job when used in `code-after`. In the following example, we write this command with `\NewDocumentCommand` (of L3) and `\foreach` of `pgffor` (which is loaded when `witharrows` is loaded).

```

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

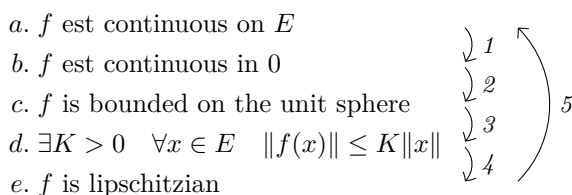
```

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

```

\begin{WithArrows}[code-after = \NumberedLoop]
a.\;& f \text{ est continuous on } E \
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}

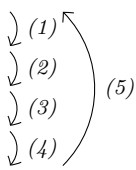
```



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



12 Implementation

12.1 Declaration of the package and extensions loaded

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

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

<@@=witharrows>

First, `tikz` and some Tikz libraries are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.²⁸

```

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

```

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

```

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

16 \RequirePackage { varwidth }
17 </LaTeX>

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

```

²⁸cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

12.2 The packages footnote and footnotehyper

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 2.8a), there are two such options: `footnote` and `footnotehyper`. With the option `footnote`, `witharrows` loads `footnote` and uses it to extract the footnotes from the environments `{WithArrows}`. Idem for the option `footnotehyper`.

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

```
22 <*LaTeX>
23 \bool_new:N \c_@@_footnotehyper_bool
```

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

```
24 \bool_new:N \c_@@_footnote_bool
25 </LaTeX>
```

```
26 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
27 \cs_new_protected:Npn \@@_msg_new:nnn #1 #2 #3
28 {
29   \bool_if:NTF \c_@@_messages_for_Overleaf_bool
30     { \msg_new:nnn { witharrows } { #1 } { #2 \ \ #3 } }
31     { \msg_new:nnnn { witharrows } { #1 } { #2 } { #3 } }
32 }
33 \cs_new_protected:Npn \@@_msg_redirect_name:nn
34 { \msg_redirect_name:nnn { witharrows } }
35 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
36 \cs_new_protected:Npn \@@_warning:n { \msg_warning:nn { witharrows } }
37 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { witharrows } }
38 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { witharrows } }
39 \cs_generate_variant:Nn \@@_error:nn { n x }
```

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

```
40 \cs_new_protected:Npn \@@_error_or_warning:n
41 { \bool_if:NTF \c_@@_messages_for_Overleaf_bool \@@_warning:n \@@_error:n }
```

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

```
42 \bool_set:Nn \c_@@_messages_for_Overleaf_bool
43 {
44   \str_if_eq_p:Vn \c_sys_jobname_str { _region_ } % for Emacs
45   || \str_if_eq_p:Vn \c_sys_jobname_str { output } % for Overleaf
46 }
```

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

```
47 <*LaTeX>
48 \keys_define:nn { WithArrows / package }
49 {
50   footnote .bool_set:N = \c_@@_footnote_bool ,
51   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
52   unknown .code:n =
53     \@@_fatal:n { Option~unknown~for~package }
54 }
55 \@@_msg_new:nn { Option~unknown~for~package }
56 {
57   You~can't~use~the~option~'\l_keys_key_str'~when~loading~the~
58   package~witharrows.~Try~to~use~the~command~
59   \token_to_str:N\WithArrowsOptions.
60 }
```

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

```
61 \ProcessKeysOptions { WithArrows / package }
```

```

62 \@@_msg_new:n { footnote-with-footnotehyper-package }
63 {
64   Footnote-forbidden.\
65   You~can't~use~the~option~'footnote'~because~the~package~
66   footnotehyper~has~already~been~loaded.~
67   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
68   within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
69   of~the~package~footnotehyper.\
70   If~you~go~on,~the~package~footnote~won't~be~loaded.
71 }
72 \@@_msg_new:n { footnotehyper-with-footnote-package }
73 {
74   You~can't~use~the~option~'footnotehyper'~because~the~package~
75   footnote~has~already~been~loaded.~
76   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
77   within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
78   of~the~package~footnote.\
79   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
80 }

81 \bool_if:NT \c_@@_footnote_bool
82 {

```

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

```

83 \ifclassloaded { beamer }
84 { \bool_set_false:N \c_@@_footnote_bool }
85 {
86   \ifpackageloaded { footnotehyper }
87     { \@@_error:n { footnote-with-footnotehyper-package } }
88     { \usepackage { footnote } }
89   }
90 }

91 \bool_if:NT \c_@@_footnotehyper_bool
92 {

```

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

```

93 \ifclassloaded { beamer }
94 { \bool_set_false:N \c_@@_footnote_bool }
95 {
96   \ifpackageloaded { footnote }
97     { \@@_error:n { footnotehyper-with-footnote-package } }
98     { \usepackage { footnotehyper } }
99   \bool_set_true:N \c_@@_footnote_bool
100 }
101 }

```

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

12.3 The class option `leqno`

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

```

102 \bool_new:N \c_@@_leqno_bool
103 \DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
104 \DeclareOption* { }
105 \ProcessOptions*
106 </LaTeX>

```

12.4 Some technical definitions

```

107 \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`.²⁹

```

108 \AtBeginDocument
109 {
110   \clist_map_inline:nn
111     {
112       amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
113       typedref, unicode-math
114     }
115     {
116       \bool_new:c { c_@@_#1_loaded_bool }
117 <*LaTeX>
118       \ifpackageloaded { #1 }
119         { \bool_set_true:c { c_@@_#1_loaded_bool } }
120         { }
121 </LaTeX>
122 <*plain-TeX>
123       \bool_set_false:c { c_@@_#1_loaded_bool }
124 </plain-TeX>
125     }
126 }

```

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

```

127 \cs_new_protected:Npn \@@_sort_seq:N #1
128 {
129   \seq_sort:Nn #1
130   {
131     \int_compare:nNnTF
132       {
133         \tex_strcmp:D
134           { \str_lower_case:n { ##1 } }
135           { \str_lower_case:n { ##2 } }
136       }
137       > 0
138       \sort_return_swapped:
139       \sort_return_same:
140   }
141 }

```

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

```

142 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
143 {
144   \seq_set_from_clist:Nn #1 { #2 }
145   \seq_set_map_x:NNn #1 #1 { \tl_to_str:n { ##1 } }
146 }

```

The command `\@@_save:N` saves a L3 variable by creating a global version of the variable. For a variable named `\l_name_type`, the corresponding global variable will be named `\g_name_type`. The

²⁹It's not possible to use `\ifpackageloaded` in the core of the functions because `\ifpackageloaded` is available only in the preamble.

type of the variable is determined by the suffix *type* and is used to apply the corresponding L3 commands.

```

147 \cs_new_protected:Npn \@@_save:N #1
148   {
149     \seq_set_split:Nxx \l_tmpa_seq
150       { \char_generate:nn { ` _ } { 12 } }
151     { \cs_to_str:N #1 }
152     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl

```

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

```

153     \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
154     \use:c { \l_tmpa_str _if_exist:cF }
155     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
156     {
157       \use:c { \l_tmpa_str _new:c }
158       { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
159     }
160     \use:c { \l_tmpa_str _gset_eq:cN }
161     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
162   }

```

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

```

163 \cs_new_protected:Npn \@@_restore:N #1
164   {
165     \seq_set_split:Nxx \l_tmpa_seq
166       { \char_generate:nn { ` _ } { 12 } }
167     { \cs_to_str:N #1 }
168     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
169     \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
170     \use:c { \l_tmpa_str _set_eq:Nc }
171     #1 { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
172   }

```

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.

```

173 \tikzset
174   {
175     @@_node_style / .style =
176     {
177       above = \l_@@_ystart_dim ,
178       inner~sep = \c_zero_dim ,
179       minimum~width = \c_zero_dim ,
180       minimum~height = \l_@@_ygap_dim
181     }
182   }

```

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.³⁰

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.

```

183 \tikzset
184   {
185     @@_standard / .style =
186     {
187       remember~picture ,
188       overlay ,

```

³⁰The *v*-nodes, created near the end of line in `{DispWithArrows}` and `{DispWithArrows*}` are not shown with the option `show-nodes`.

```

189     name~prefix = wa - \l_@@_prefix_str -
190   }
191 }

```

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`.

```

192 \tikzset
193 {
194   WithArrows / arrow / tips / .style =
195     { > = { Straight~Barb [ scale = 1.2 , bend ] } }
196 }

```

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`.

```

197 \tikzset
198 {
199   WithArrows / arrow / .style =
200     {
201       align = flush~left ,
202       auto = left ,
203       < *LaTeX >
204       font = \small \itshape ,
205       < /LaTeX >
206       WithArrows / arrow / tips ,
207       bend~left = 45 ,
208       ->
209     }
210 }

```

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

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}`.

```

211 < *LaTeX >
212 \AtBeginDocument
213 {
214   \bool_if:NTF \c_@@_amsmath_loaded_bool
215     {
216       \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
217       \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
218     }

```

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 the hook `begindocument` because the flag `\c_@@_amsmath_loaded_bool` is itself set in the hook `begindocument`).

```

219   {
220 < /LaTeX >
221     \cs_new_protected:Npn \spread@equation
222       {
223         \openup \jot
224         \cs_set_eq:NN \spread@equation \prg_do_nothing:
225       }
226 < *LaTeX >
227   }
228 }

```

229 `</LaTeX>`

```
230 \tl_new:N \l_@@_left_brace_tl
231 \tl_set_eq:NN \l_@@_left_brace_tl \c_novalue_tl
```

12.5 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`).

```
232 \bool_new:N \l_@@_in_WithArrows_bool
233 \bool_new:N \l_@@_in_DispWithArrows_bool
234 \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}`.

```
235 \seq_new:N \g_@@_position_in_the_tree_seq
236 \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`.

```
237 \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`).

```
238 \int_new:N \l_@@_pos_env_int
```

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

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

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

```
239 \int_new:N \l_@@_pos_arrow_int
240 \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.

```
241 \seq_new:N \g_@@_arrow_int_seq
242 \int_new:N \g_@@_arrow_int
243 \seq_new:N \g_@@_line_int_seq
244 \int_new:N \g_@@_line_int
245 \seq_new:N \g_@@_col_int_seq
246 \int_new:N \g_@@_col_int
```

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

```
247 \seq_new:N \g_@@_static_col_int_seq
248 \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`.

```
249 <*LaTeX>
250 \clist_new:N \l_@@_tags_clist
251 \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`.

```
252 \cs_new_protected:Npn \@@_test_if_to_tag:
253 {
254   \clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int
255   { \clist_set:Nn \l_@@_tags_clist { all } }
256 }
257 </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`.

```
258 \str_new:N \l_@@_command_name_str
259 \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`”.

```
260 \str_new:N \l_@@_string_Arrow_for_msg_str
261 \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`.

```
262 \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.

```
263 <*LaTeX>
264 \bool_new:N \l_@@_sbwi_bool
265 </LaTeX>
```

```
266 <*LaTeX>
267 \bool_new:N \l_@@_tag_star_bool
268 \bool_new:N \l_@@_tag_next_line_bool
269 \bool_new:N \l_@@_qedhere_bool
270 </LaTeX>
271 \bool_new:N \l_@@_in_first_columns_bool
272 \bool_new:N \l_@@_new_group_bool
```

```

273 \bool_new:N \l_@@_initial_r_bool
274 \bool_new:N \l_@@_final_r_bool
275 \tl_new:N \l_@@_initial_tl
276 \tl_new:N \l_@@_final_tl
277 \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}`).

```

278 \str_new:N \l_@@_format_str

```

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

```

279 <*LaTeX>
280 \bool_new:N \l_@@_subequations_bool
281 </LaTeX>

```

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

```

282 \dim_new:N \l_@@_arrow_width_dim
283 \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim

```

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

```

284 \dim_new:N \l_@@_up_and_down_radius_dim
285 \dim_set:Nn \l_@@_up_and_down_radius_dim { 4 pt }

```

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

```

286 \seq_new:N \l_@@_o_arrows_seq

```

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

```

287 \dim_new:N \l_@@_xoffset_for_o_arrows_dim
288 \dim_set:Nn \l_@@_xoffset_for_o_arrows_dim { 2 mm }

```

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

```

289 \bool_new:N \l_@@_right_overlap_bool
290 \bool_set_true:N \l_@@_right_overlap_bool

```

12.6 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* level³¹;
- with `\begin{WithArrows}[...]`: this level will be called *environment* level;
- with `\Arrow[...]` (included in `code-after`): this level will be called *local* level.

³¹This 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 (ll, rl, i, etc.) of the arrows are present. That's why we keep the first option of position in a variable called `\l_@@_previous_key_str`. The following function `\@@_eval_if_allowed:n` will execute its argument only if a first key of position has not been set (and raise an error elsewhere).

```

291 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
292   {
293     \str_if_empty:NTF \l_@@_previous_key_str
294       {
295         \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
296         #1
297       }
298     { \@@_error:n { Incompatible~options } }
299   }

300 \cs_new_protected:Npn \@@_fix_pos_option:n #1
301   { \@@_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.

```

302 \keys_define:nn { WithArrows / Global }
303   {
304     max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
305     max-length-of-arrow .value_required:n = true ,
306     max-length-of-arrow .initial:n = 2 cm ,
307     ygap .dim_set:N = \l_@@_ygap_dim ,
308     ygap .initial:n = 0.4 ex ,
309     ygap .value_required:n = true ,
310     ystart .dim_set:N = \l_@@_ystart_dim ,
311     ystart .value_required:n = true ,
312     ystart .initial:n = 0.4 ex ,
313     more-columns .code:n =
314       \@@_msg_redirect_name:nn { Too-much-columns-in-WithArrows } { none } ,
315     more-columns .value_forbidden:n = true ,
316     command-name .code:n =
317       \str_set:Nn \l_@@_command_name_str { #1 }
318       \str_set:Nx \l_@@_string_Arrow_for_msg_str
319         { \c_backslash_str Arrow-alias-\c_backslash_str #1 } ,
320     command-name .value_required:n = true ,
321     tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
322     tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
323     tikz-code .value_required:n = true ,
324     displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
325     displaystyle .default:n = true ,
326     show-nodes .code:n =
327       \tikzset { @@_node_style / .append~style = { draw , red } } ,
328     show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
329     show-node-names .default:n = true ,
330     group .code:n =
331       \str_if_empty:NTF \l_@@_previous_key_str
332         {
333           \str_set:Nn \l_@@_previous_key_str { group }
334           \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
335           \int_set:Nn \l_@@_pos_arrow_int 7
336         }
337         { \@@_error:n { Incompatible~options } } ,
338     group .value_forbidden:n = true ,
339     groups .code:n =
340       \str_if_empty:NTF \l_@@_previous_key_str
341         {
342           \str_set:Nn \l_@@_previous_key_str { groups }
343           \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
344             { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
345           \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
346           \int_set:Nn \l_@@_pos_arrow_int 6

```

```

347     }
348     { \@@_error:n { Incompatible~options } } ,
349 groups .value_forbidden:n = true ,
350 tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
351 tikz .initial:n = \c_empty_tl ,
352 tikz .value_required:n = true ,
353 rr .code:n = \@@_fix_pos_option:n 3 ,
354 rr .value_forbidden:n = true ,
355 ll .code:n = \@@_fix_pos_option:n 1 ,
356 ll .value_forbidden:n = true ,
357 rl .code:n = \@@_fix_pos_option:n 2 ,
358 rl .value_forbidden:n = true ,
359 lr .code:n = \@@_fix_pos_option:n 0 ,
360 lr .value_forbidden:n = true ,
361 i .code:n = \@@_fix_pos_option:n 5 ,
362 i .value_forbidden:n = true ,
363 xoffset .dim_set:N = \l_@@_xoffset_dim ,
364 xoffset .value_required:n = true ,
365 xoffset .initial:n = 3 mm ,
366 jot .dim_set:N = \jot ,
367 jot .value_required:n = true ,
368 interline .skip_set:N = \l_@@_interline_skip ,
369 start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
370 start-adjust .initial:n = 0.4 ex ,
371 start-adjust .value_required:n = true ,
372 end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
373 end-adjust .initial:n = 0.4 ex ,
374 end-adjust .value_required:n = true ,
375 adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
376 adjust .value_required:n = true ,
377 up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,
378 up-and-down .value_required:n = true ,

```

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

```

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

```

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

```

384 \keys_define:nn { WithArrows / WithArrowsSpecific }
385 {
386     t .code:n = \int_set:Nn \l_@@_pos_env_int 0 ,
387     t .value_forbidden:n = true ,
388     c .code:n = \int_set:Nn \l_@@_pos_env_int 1 ,
389     c .value_forbidden:n = true ,
390     b .code:n = \int_set:Nn \l_@@_pos_env_int 2 ,
391     b .value_forbidden:n = true ,
392     right-overlap .bool_set:N = \l_@@_right_overlap_bool ,
393     right-overlap .value_required:n = true
394 }

```

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

```

395 \clist_new:N \c_@@_extensible_delimiters_clist
396 \clist_set:Nn \c_@@_extensible_delimiters_clist
397 {
398     ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor

```

```

399   }
400 <*LaTeX>
401 \AtBeginDocument
402   {
403     \bool_lazy_or:nnT
404       \c_@@_amsmath_loaded_bool
405       { \use:c { c_@@_unicode-math_loaded_bool } }
406       {
407         \clist_put_right:Nn \c_@@_extensible_delimiters_clist { \lvert, \lVert }
408       }
409   }
410 </LaTeX>

```

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

```

411 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
412   {
413     fleqn .bool_set:N = \l_@@_fleqn_bool ,
414     fleqn .default:n = true ,
415     mathindent .skip_set:N = \l_@@_mathindent_skip ,
416     mathindent .initial:n = 25 pt ,
417     mathindent .value_required:n = true ,
418 <*LaTeX>
419     notag .code:n =
420       \str_if_eq:nnTF { #1 } { true }
421       { \clist_clear:N \l_@@_tags_clist }
422       { \clist_set:Nn \l_@@_tags_clist { all } } ,
423     notag .default:n = true ,

```

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

```

424     subequations .code:n =
425       \bool_if:NTF \c_@@_amsmath_loaded_bool
426       { \bool_set_true:N \l_@@_subequations_bool }
427       {
428         \@@_error:n { amsmath-not~loaded }
429         \group_begin:
430         \globaldefs = 1
431         \@@_msg_redirect_name:nn { amsmath-not~loaded } { info }
432         \group_end:
433       } ,
434     subequations .default:n = true ,
435     subequations .value_forbidden:n = true ,
436     nonumber .meta:n = notag ,
437     allow-multiple-labels .code:n =
438       \@@_msg_redirect_name:nn { Multiple~labels } { none } ,
439     allow-multiple-labels .value_forbidden:n = true ,
440     tagged-lines .code:n =
441       \clist_set:Nn \l_@@_tags_clist { #1 }
442       \clist_if_in:NnT \l_@@_tags_clist { first }
443       {
444         \clist_remove_all:Nn \l_@@_tags_clist { first }
445         \clist_put_left:Nn \l_@@_tags_clist 1
446       } ,
447     tagged-lines .value_required:n = true ,
448 </LaTeX>
449     wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
450     wrap-lines .default:n = true ,
451     replace-left-brace-by .code:n =
452     {
453       \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
454       \clist_if_in:NVTF
455         \c_@@_extensible_delimiters_clist

```

```

456     \l_tmpa_tl
457     { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }
458     { \@@_error:n { Bad-value~for~replace-brace-by } }
459   } ,
460   replace-left-brace-by .initial:n = \lbrace ,

```

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

```

461 {*LaTeX}
462   standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
463   standard-behaviour-with-items .default:n = true
464 {/LaTeX}
465 }

```

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

```

466 \keys_define:nn { WithArrows / Env }
467 {
468   name .code:n =

```

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

```

469     \str_set:Nn \l_tmpa_str { #1 }
470     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
471       { \@@_error:n { Duplicate-name } }
472       { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
473     \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
474   name .value_required:n = true ,
475   code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
476   code-before .value_required:n = true ,
477   CodeBefore .meta:n = { code-before = #1 } ,
478   code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
479   code-after .value_required:n = true ,
480   CodeAfter .meta:n = { code-after = #1 } ,
481   format .code:n =
482     \tl_if_empty:nTF { #1 }
483       { \@@_error:n { Invalid-option-format } }
484       {
485         \regex_match:nnTF { \A[rclRCL]*\Z } { #1 }
486         { \tl_set:Nn \l_@@_format_str { #1 } }
487         { \@@_error:n { Invalid-option-format } }
488       } ,
489   format .value_required:n = true
490 }

```

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

```

491 \keys_define:nn { WithArrows }
492 {
493   WithArrows .inherit:n =
494     {
495       WithArrows / Global ,
496       WithArrows / WithArrowsSpecific ,
497       WithArrows / Env
498     } ,
499   WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
500   DispWithArrows .inherit:n =
501     {
502       WithArrows / DispWithArrowsSpecific ,
503       WithArrows / Global ,
504       WithArrows / Env ,

```

```

505     } ,
506     DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
507     WithArrowsOptions .inherit:n =
508     {
509         WithArrows / Global ,
510         WithArrows / WithArrowsSpecific ,
511         WithArrows / DispWithArrowsSpecific ,
512     } ,
513     WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
514 }

```

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

```

515 \seq_new:N \l_@@_options_WithArrows_seq
516 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrows_seq
517 {
518     adjust, b, c, code-after, code-before, command-name,
519     right-overlap, displaystyle, end-adjust,
520     format, group, groups, i,
521     interline, jot, ll,
522     lr, max-length-of-arrow, more-columns, name,
523     no-arrows, rl, rr, up-and-down,
524     show-node-names, show-nodes, start-adjust,
525     t, tikz, tikz-code,
526     xoffset, ygap, ystart
527 }

```

```

528 \keys_define:nn { WithArrows / WithArrows }
529 {
530     unknown .code:n =
531         \@@_sort_seq:N \l_@@_options_WithArrows_seq
532         \@@_error:n { Unknown~option~WithArrows }
533 }

```

```

534 \keys_define:nn { WithArrows / DispWithArrows }
535 {
536     left-brace .tl_set:N = \l_@@_left_brace_tl ,
537     unknown .code:n =
538         \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
539         \@@_error:n { Unknown~option~DispWithArrows } ,
540 }

```

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

```

541 \seq_new:N \l_@@_options_DispWithArrows_seq
542 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
543 {
544     code-after, code-before, command-name, tikz-code, adjust,
545     displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
546     left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
547     up-and-down, replace-left-brace-by, rl, rr, show-node-names,
548     show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
549     <LaTeX>
550     allow-multiple-labels, tagged-lines, nonumber, notag
551     </LaTeX>
552 }
553 \keys_define:nn { WithArrows / WithArrowsOptions }
554 {
555     allow-duplicate-names .code:n =
556     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,

```

```

557 allow-duplicate-names .value_forbidden:n = true ,
558 xoffset-for-o-arrows .dim_set:N = \l_@@_xoffset_for_o_arrows_dim ,
559 xoffset-for-o-arrows .value_required:n = true ,
560 unknown .code:n =
561   \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
562   \@@_error:n { Unknown-option-WithArrowsOptions }
563 }

```

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

```

564 \seq_new:N \l_@@_options_WithArrowsOptions_seq
565 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrowsOptions_seq
566 {
567   allow-duplicate-names, b, c, command-name, right_overlap,
568   more-columns, tikz-code, adjust,
569   displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
570   mathindent, max-length-of-arrow, no-arrows, up-and-down, rl, rr,
571   show-node-names, show-nodes, start-adjust, t, tikz, wrap-lines, xoffset,
572   xoffset-for-o-arrows, ygap, ystart,
573 \LaTeX
574   allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
575   tagged-lines
576 \LaTeX
577 }

```

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”.

```

578 \cs_new_protected:Npn \@@_set_independent:
579 {
580   \str_if_eq:VnF \l_keys_value_tl { NoValue }
581   { \@@_error:n { Value-for-a-key } }
582   \@@_set_independent_bis:
583 }

```

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

```

584 \cs_new_protected:Npn \@@_set_independent_bis:
585 {
586   \str_if_empty:NTF \l_@@_previous_key_str
587   {
588     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
589     \str_set:Nn \l_@@_status_arrow_str { independent }
590   }
591   { \@@_error:n { Incompatible-options-in-Arrow } }
592 }

```

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

```

593 \keys_define:nn { WithArrows / Arrow / FirstPass }
594 {
595   jump .code:n =
596     \int_compare:nTF { #1 > 0 }
597     { \int_set:Nn \l_@@_jump_int { #1 } }
598     { \@@_error:n { Negative-jump } } ,
599   jump .value_required:n = true,

```

```

600 rr .code:n = \@@_set_independent: ,
601 ll .code:n = \@@_set_independent: ,
602 rl .code:n = \@@_set_independent: ,
603 lr .code:n = \@@_set_independent: ,
604 i .code:n = \@@_set_independent: ,
605 rr .default:n = NoValue ,
606 ll .default:n = NoValue ,
607 rl .default:n = NoValue ,
608 lr .default:n = NoValue ,
609 i .default:n = NoValue ,
610 new-group .value_forbidden:n = true ,
611 new-group .code:n =
612   \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
613     { \str_set:Nn \l_@@_status_arrow_str { new-group } }
614     { \@@_error:n { new-group-without~groups } } ,
615 o .code:n =
616   \str_if_empty:NTF \l_@@_previous_key_str
617     {
618       \int_compare:nNnTF \l_@@_pos_arrow_int < 6
619         { \@@_error:n { invalid~key~o } }
620         {
621           \str_set:Nn \l_@@_status_arrow_str { over }
622           \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
623         }
624     }
625     { \@@_error:n { Incompatible~options~in~Arrow } } ,

```

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

```

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

```

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

```

647 \seq_new:N \l_@@_options_Arrow_seq
648 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_seq
649 {
650   adjust, end-adjust, i, jump, ll, lr, o , rl, rr, start-adjust, tikz,
651   tikz-code, xoffset
652 }
653 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1

```

```

654 {
655   \str_if_empty:NT \l_@@_previous_key_str
656   {
657     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
658     \int_set:Nn \l_@@_pos_arrow_int { #1 }
659   }
660 }

```

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.

```

661 \keys_define:nn {WithArrows / Arrow / SecondPass }
662 {
663   tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
664   tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
665   tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
666   tikz .initial:n = \c_empty_tl ,
667   rr .code:n = \@@_fix_pos_arrow:n 3 ,
668   ll .code:n = \@@_fix_pos_arrow:n 1 ,
669   rl .code:n = \@@_fix_pos_arrow:n 2 ,
670   lr .code:n = \@@_fix_pos_arrow:n 0 ,
671   i .code:n = \@@_fix_pos_arrow:n 5 ,
672   o .code:n = \str_set:Nn \l_@@_previous_key_str { o } ,

```

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

```

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 }

```


12.7 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 `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 4 possible values: `empty`, `independent`, `new-group` or `over`.

```

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 some error messages).

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

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

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

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

```
731 \prop_gclear_new:c
732 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
733 \prop_gset_eq:cN
734 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
735 \l_tmpa_prop
736 }
```

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.

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

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

12.8.1 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.

```
739 \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.

```
740 {
741 <*LaTeX>
742 \str_clear_new:N \l_@@_type_env_str
743 \str_set:NV \l_@@_type_env_str \@currentenv
744 </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`.

```
745 \cs_if_exist:NT \tikz@library@external@loaded
746 { \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.

```
747 \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.

```
748 \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.

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

The variable `\l_@@_input_line_str` will be used only to store, for each command `\Arrow` the line (in the TeX file) where the command is issued. This information will be stored in the field “input-line” of the arrow. As of now, this information is used only in some error messages.

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

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

```
751 \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
752 \int_gzero:N \g_@@_arrow_int
753 \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
754 \int_gzero:N \g_@@_line_int
755 \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
756 \int_gzero:N \g_@@_col_int
757 \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
758 \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 program-mation is to detect the use of a command `\omit` in a cell of the `\halign` (it should be forbidden). For example, in the part of the preamble concerning the third column (if there is a third column in the environment), we will have the following instructions :

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

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

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

```
759 \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).

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

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

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

These counters will be used later as variables.

```
766 \int_zero_new:N \l_@@_initial_int
767 \int_zero_new:N \l_@@_final_int
768 \int_zero_new:N \l_@@_arrow_int
769 \int_zero_new:N \l_@@_pos_of_arrow_int
770 \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.

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

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

```
772 \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`.

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

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.³²

```
777 \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`.

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

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

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

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

```
785 \bool_if:NF \l_@@_right_overlap_bool
786 {
787   \bool_if:NT \l_@@_in_WithArrows_bool
788   {
789     \dim_gzero_new:N \g_@@_overlap_x_dim
790     \dim_zero_new:N \l_@@_delta_x_dim
791   }
792 }
```

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.

```
793 \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`).

```
794 \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) ;

³²It's recalled that, by design, the option `interline` of an environment doesn't apply in the nested environments.

- 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.

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

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

```
796 \seq_clear_new:N \l_@@_format_seq
797 \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`).

```
798 <*LaTeX>
799 \bool_if:NT \c_@@_footnote_bool { \begin { savenotes } }
800 </LaTeX>
```

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

```
801 \l_@@_code_before_tl
802 <*LaTeX>
803 \cs_set_eq:NN \notag \@@_notag:
804 \cs_set_eq:NN \nonumber \@@_nonumber:
805 \cs_set_eq:NN \tag \@@_tag
806 \cs_set_eq:NN \@@_old_label \label
807 \cs_set_eq:NN \label \@@_label:n
808 \cs_set_eq:NN \tagnextline \@@_tagnextline:
809 </LaTeX>
810 }
```

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

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

```
811 \cs_new_protected:Npn \@@_construct_halign:
812 {
813 \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
814 {
```

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

```
815     \use:e
816     {
```

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

We begin the construction of a generic column.

```
817     \int_gdecr:N \g_@@_col_int
818     \@@_construct_halign:
819     \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
820     {
```

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

```
821     \cs_set_eq:cN { \l_@@_command_name_str } \@@_Arrow
822     <*LaTeX>
823     \bool_if:NT \l_@@_in_DispWithArrows_bool
824     {
```

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).

```
825     \@@_test_if_to_tag:
```

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

```
826     \bool_if:NT \c_@@_amsthm_loaded_bool \@@_set_qedhere:
827     }
828     </LaTeX>
829     }
```

```
830     \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
831     \str_if_eq:VnT \l_@@_type_col_str { C } \hfil
832     \str_if_eq:VnT \l_@@_type_col_str { r } \hfill
833     \str_if_eq:VnT \l_@@_type_col_str { R } \hfill
834     \int_gincr:N \g_@@_col_int
835     \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
836     \c_math_toggle_token
837     \str_if_eq:VnT \l_@@_type_col_str { C } { { } }
838     \str_if_eq:VnT \l_@@_type_col_str { L } { { } }
839     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
840     ##
841     \str_if_eq:VnT \l_@@_type_col_str { C } { { } }
842     \str_if_eq:VnT \l_@@_type_col_str { R } { { } }
843     \c_math_toggle_token
844     \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
845     \@@_construct_nodes:
846     {
```

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:`).

```
847     \str_if_eq:VnT \l_@@_type_col_str { l } \hfil
848     \str_if_eq:VnT \l_@@_type_col_str { L } \hfil
849     \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
850     \str_if_eq:VnT \l_@@_type_col_str { C } \hfil
851     \bool_if:NT \l_@@_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
852     &
853     }
854     }
855 }
```

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

```

856     {
857       \bool_if:NTF \l_@@_in_WithArrows_bool
858       {
859         \ialign
860         \bgroup
861       }
862       {
863         \halign to \l_@@_linewidth_dim
864         \bgroup
865         \bool_if:NT \l_@@_fleqn_bool
866         { \skip_horizontal:N \l_@@_mathindent_skip }
867       }
868       \int_gincr:N \g_@@_line_int
869       \int_gzero:N \g_@@_col_int
870       \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
871       {
872         \skip_horizontal:n
873         { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
874       }
875       \strut
876     }
877 }

```

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.

```

878 \cs_new_protected:Npn \@@_construct_nodes:
879 {

```

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).

```

880   \tikz [ remember-picture , overlay ]
881   \node
882   [
883     node~contents = { } ,
884     @@_node_style ,
885     name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
886   ]
887   ;
888   \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).

```

889   \tikz [ remember-picture , overlay ]
890   \node
891   [
892     node~contents = { } ,
893     @@_node_style ,
894     name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
895   ]
896   ;
897   \str_if_empty:NF \l_@@_name_str
898   {
899     \pgfpicture
900     \pgfnodealias
901     { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
902     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l }
903     \pgfnodealias
904     { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
905     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
906     \endpgfpicture

```

```

907     }
908     \bool_if:NT \l_@@_show_node_names_bool
909     {
910         \hbox_overlap_right:n
911         { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
912     }
913 }

```

12.8.3 The environment `{WithArrows}`

```

914 <*LaTeX>
915 \NewDocumentEnvironment { WithArrows } { ! 0 { } }
916 </LaTeX>
917 <*plain-TeX>
918 \cs_new_protected:Npn \WithArrows
919 {
920     \group_begin:
921     \peek_meaning:NTF [
922     { \WithArrows_i }
923     { \WithArrows_i [ ] }
924 }
925 \cs_new_protected:Npn \WithArrows_i [ #1 ]
926 </plain-TeX>
927 {
928     \bool_set_true:N \l_@@_in_WithArrows_bool
929     \bool_set_false:N \l_@@_in_DispWithArrows_bool
930 <*plain-TeX>
931     \str_clear_new:N \l_@@_type_env_str
932     \str_set:Nn \l_@@_type_env_str { WithArrows }
933 </plain-TeX>
934     \@@_pre_halign:n { #1 }
935     \if_mode_math: \else:
936         \@@_error:n { WithArrows~outside~math~mode }
937     \fi:
938     \box_clear_new:N \l_@@_env_box
939     \hbox_set:Nw \l_@@_env_box

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`³³ 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 mode³⁴ and therefore, we can use `\vcenter`.

```

940     \int_compare:nNnT \l_@@_pos_env_int = 1 \c_math_toggle_token
941     \int_case:nn \l_@@_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
942     \bgroup

```

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

```

943     \spread@equation

```

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

```

944     \@@_construct_halign:

```

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

³³Notice that the use of `\vtop` seems color-safe here...

³⁴An error is raised if the environment is used outside math mode.

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

```

945   &&
946   \@@_error:n { Too~much~columns~in~WithArrows }
947   \c_math_toggle_token
948   \bool_if:NT \l_@@_displaystyle_bool \displaystyle
949   { ## }
950   \c_math_toggle_token
951   \cr
952 }

```

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

```

953 <*plain-TeX>
954 \cs_new_protected:Npn \endWithArrows
955 </plain-TeX>
956 {
957   \
958   \egroup
959   \egroup
960   \int_compare:nNnT \l_@@_pos_env_int = 1 \c_math_toggle_token
961   \hbox_set_end:
962   \@@_post_halign:

```

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

```

963   \bool_if:NF \l_@@_right_overlap_bool
964   {
965     \box_set_wd:Nn \l_@@_env_box
966     { \g_@@_overlap_x_dim + \box_wd:N \l_@@_env_box }
967   }
968   \box_use_drop:N \l_@@_env_box

```

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

```

969 <*LaTeX>
970   \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
971 </LaTeX>
972 <*plain-TeX>
973   \group_end:
974 </plain-TeX>
975 }

```

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

12.8.4 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}`.

```

976 \cs_new_protected:Npn \@@_post_halign:

```

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

```

977 {
978   \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.

```

979   \normalbaselines

```

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

```
980 \int_compare:nNnT \g_@@_arrow_int > 0
981 {
```

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).

```
982 \int_compare:nNnT \g_@@_arrow_int = 1
983 {
984 \int_compare:nNnT \l_@@_pos_arrow_int > 5
985 { \int_set:Nn \l_@@_pos_arrow_int 5 }
986 }
987 \@@_scan_arrows:
988 }
```

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

```
989 \group_begin:
990 \tikzset { every~picture / .style = @@_standard }
```

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

```
991 \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.³⁵

```
992 \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
993 \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
994 \bool_set_true:N \l_@@_in_code_after_bool
995 \l_@@_code_after_tl
996 \group_end:
```

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

```
997 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
998 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
999 \seq_gput_right:Nx \g_@@_position_in_the_tree_seq
1000 { \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`.

```
1001 \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
1002 { \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.

```
1003 \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
1004 \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
1005 \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
1006 \int_gset:Nn \g_@@_line_int \l_tmpa_tl
1007 \seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
1008 \int_gset:Nn \g_@@_col_int \l_tmpa_tl
1009 \seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
1010 \int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
1011 }
```

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

³⁵As of now, `\MultiArrow` has no option, and that's why its internal name is a name of L3 with the signature `:nn` whereas `\Arrow` in `code-after` provides options and has the name of a function defined with `\NewDocumentCommand`.

12.8.5 The command of end of row

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

```
1012 \cs_new_protected:Npn \@@_cr:
1013   {
1014     \scan_stop:
```

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

```
1015     \int_compare:nNnF \g_@@_col_int = \g_@@_static_col_int
1016       { \@@_error:n { omit~probably~used } }
1017     \prg_replicate:nn { \l_@@_nb_cols_int - \g_@@_static_col_int } { & { } }
1018     \group_align_safe_begin:
1019     \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
1020   }
```

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.

```
1021 \cs_new_protected:Npn \@@_cr_i:
1022   { \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.

```
1023 <*LaTeX>
1024 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1025   {
1026     \peek_remove_spaces:n
1027     {
1028       \peek_meaning:NTF \end
1029       {
1030         \@@_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`.

```
1031         \@@_analyze_end:Nn
1032       }
1033     { \@@_cr_iii:n { #1 } }
1034   }
1035 }

1036 \cs_new_protected:Npn \@@_cr_iii:n #1
1037 </LaTeX>
1038 <*plain-TeX>
1039 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1040 </plain-TeX>
1041   {
1042     \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.³⁶

```
1043   \bool_if:NT \l_@@_in_DispWithArrows_bool
```

³⁶The v-nodes are used to compute the abscissa of the right margin, used by the option `wrap-lines`.

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`.

```

1044     {
1045     (*LaTeX)
1046     \clist_if_in:NnTF \l_@@_tags_clist { all }
1047     {

```

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).

```

1048     \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 `&`).

```

1049     \cs_gset:Npx \g_tmpa_tl
1050     { \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`.

```

1051     \seq_if_empty:NF \l_@@_labels_seq
1052     {

```

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`.

```

1053     \cs_set:Npx \@currentlabel { \p@equation \g_tmpa_tl }

```

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

```

1054     \bool_if:NT \c_@@_hyperref_loaded_bool
1055     {
1056         % the following line is probably useless (2022/05/16)
1057         % \str_set:Nn \This@name { equation }
1058         \hyper@refstepcounter { equation }
1059     }

```

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

```

1060     \bool_if:NT \c_@@_cleveref_loaded_bool
1061     {
1062         \cref@constructprefix { equation } \cref@result
1063         \protected@edef \cref@currentlabel
1064         {
1065             [
1066                 \cs_if_exist:NTF \cref@equation@alias
1067                 \cref@equation@alias
1068                 { equation }
1069             ]
1070             [ \arabic { equation } ] [ \cref@result ]
1071             \p@equation \g_tmpa_tl
1072         }
1073     }

```

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`).

```

1074     \seq_map_function:NN \l_@@_labels_seq \@@_old_label
1075     }

```

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.

```

1076     \@@_save:N \l_@@_tag_star_bool
1077     \@@_save:N \l_@@_qedhere_bool
1078     \bool_if:NT \l_@@_tag_next_line_bool
1079     {
1080       \openup -\jot
1081       \bool_set_false:N \l_@@_tag_next_line_bool
1082       \notag \&
1083     }
1084     &
1085     \@@_restore:N \l_@@_tag_star_bool
1086     \@@_restore:N \l_@@_qedhere_bool
1087     \bool_if:NT \l_@@_qedhere_bool
1088     { \hbox_overlap_left:n \@@_qedhere_i: }
1089     \cs_set_eq:NN \theequation \g_tmpa_tl
1090     \bool_if:NT \l_@@_tag_star_bool
1091     { \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`.

```

1092     \hbox_overlap_left:n
1093     {
1094       \bool_if:NF \c_@@_leqno_bool
1095       {
1096         \pgfpicture
1097         \pgfrememberpicturepositiononpagetrue
1098         \pgfcoordinate
1099         { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1100         \pgfpointorigin
1101         \endpgfpicture
1102       }
1103       \quad
1104       \@eqnnum
1105     }
1106     \bool_if:NT \c_@@_leqno_bool
1107     {
1108       \pgfpicture
1109       \pgfrememberpicturepositiononpagetrue
1110       \pgfcoordinate
1111       { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1112       \pgfpointorigin
1113       \endpgfpicture
1114     }
1115   }
1116   {
1117     \@@_save:N \l_@@_qedhere_bool
1118   } </LaTeX>
1119   &
1120   { *LaTeX >
1121     \@@_restore:N \l_@@_qedhere_bool
1122     \bool_if:NT \l_@@_qedhere_bool
1123     { \hbox_overlap_left:n \@@_qedhere_i: }
1124   } </LaTeX>
1125     \pgfpicture
1126     \pgfrememberpicturepositiononpagetrue
1127     \pgfcoordinate
1128     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1129     \pgfpointorigin
1130     \endpgfpicture
1131   } *LaTeX >
1132   }

```

```

1133 </LaTeX>
1134 }
1135 \dim_compare:nNnT { #1 } < \c_zero_dim
1136 { \@@_error:n { option~of~cr~negative } }
1137
1138 \cr
1139 \noalign
1140 {
1141   \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1142   \skip_vertical:N \l_tmpa_dim
1143   \skip_vertical:N \l_@@_interline_skip
1144   \scan_stop:
1145 }
1146 }

```

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

The following command will be used when, after a `\` (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.

```

1147 <*LaTeX>
1148 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1149 {
1150   \str_if_eq:VnT \l_@@_type_env_str { #2 }
1151   {
1152     \@@_error:n { newline~at~the~end~of~env }
1153     \group_begin:
1154     \globaldefs = 1
1155     \@@_msg_redirect_name:nn { newline~at~the~end~of~env } { none }
1156     \group_end:
1157   }

```

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

```

1158   \end { #2 }
1159 }
1160 </LaTeX>

```

12.8.6 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}`.

```

1161 <*LaTeX>
1162 \bool_new:N \l_@@_in_label_or_minipage_bool
1163 </LaTeX>
1164 <*LaTeX>
1165 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1166 </LaTeX>
1167 <*plain-TeX>
1168 \cs_new_protected:Npn \DispWithArrows
1169 {
1170   \group_begin:
1171   \peek_meaning:NTF <

```

```

1172     { \DispWithArrows_i }
1173     { \DispWithArrows_i < \c_novalue_tl > }
1174   }
1175 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1176   {
1177     \peek_meaning:NTF [
1178       { \DispWithArrows_ii < #1 > }
1179       { \DispWithArrows_ii < #1 > [ ] }
1180     }
1181 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1182 </plain-TeX>
1183   {
1184     \bool_set_true:N \l_@@_in_DispWithArrows_bool
1185 <*plain-TeX>
1186     \str_clear_new:N \l_@@_type_env_str
1187     \str_set:Nn \l_@@_type_env_str { DispWithArrows }
1188 </plain-TeX>

```

Since the version 1.16 of `witharrows`, no space is added between an `\item` of a LaTeX list and an environment `{DispWithArrows}` except with the option `standard-behaviour-with-items` stored in the boolean `\l_@@_sbwi_bool`. We have to know if we are just after an `\item` and this information will be stored in `\l_@@_in_label_or_minipage_bool`. We have to do this test quickly after the beginning of the environment (in particular, because it must be done before the execution of the `code-before`³⁷).

```

1189 <*LaTeX>
1190   \bool_if:NF \l_@@_sbwi_bool
1191     {
1192       \legacy_if:nT { @inlabel }
1193         { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1194       \legacy_if:nT { @minipage }
1195         { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1196     }
1197 </LaTeX>

```

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

```

1198 <*LaTeX>
1199   \bool_if:nT \c_@@_mathtools_loaded_bool
1200     {
1201       \MH_if_boolean:nT { show_only_refs }
1202         {
1203           \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`.

```

1204           \MH_set_boolean:T:n { show_only_refs }
1205         }
1206     }

```

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`.

```

1207   \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`.

```

1208   \bool_if:NT \c_@@_amsmath_loaded_bool \intertext@
1209 </LaTeX>
1210   \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l_@@_left_brace_tl { #1 } }
1211   \@@_pre_halign:n { #2 }

```

³⁷The `code-before` is not meant to contains typesetting material. However, it may contain, for example, a `{tikzpicture}` with options `overlay` and `remember picture` in order to draw nodes *under* some elements of the environment `{DispWithArrows}`.

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

```

1212 <*LaTeX>
1213   \bool_if:NT \l_@@_subequations_bool { \begin { subequations } }
1214 </LaTeX>

1215   \tl_if_eq:NMF \l_@@_left_brace_tl \c_novalue_tl
1216   {

```

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

```

1217     \hbox_set:Nn \l_tmpa_box
1218     {

```

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

```

1219         \group_begin:
1220         \dim_zero:N \nulldelimiterspace
1221         \c_math_toggle_token
1222         \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1223         \c_math_toggle_token
1224         \group_end:
1225     }
1226     \dim_zero_new:N \l_@@_delim_wd_dim
1227     \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
1228     \box_clear_new:N \l_@@_left_brace_box
1229     \hbox_set:Nn \l_@@_left_brace_box
1230     {
1231         \group_begin:
1232         \cs_set_eq:NN \label \@@_old_label
1233         \c_math_toggle_token
1234         \bool_if:NT \l_@@_displaystyle_bool \displaystyle
1235         \l_@@_left_brace_tl
1236         { }
1237         \c_math_toggle_token
1238         \group_end:
1239     }
1240 }

```

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

```

1241 <*LaTeX>
1242   \tl_clear_new:N \l_@@_tag_tl

1243   \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.

```

1244   \bool_set_false:N \l_@@_tag_star_bool
1245 </LaTeX>

1246   \if_mode_math:
1247     \@@_fatal:n { DispNetArrows~in~math~mode }
1248   \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`.

```

1249 <*plain-TeX>
1250   \dim_zero_new:N \linewidth
1251   \dim_set_eq:NN \linewidth \displaywidth
1252 </plain-TeX>
1253 <*LaTeX>
1254   \bool_if:NTF \l_@@_in_label_or_minipage_bool
1255   {
1256     \noindent % added in v. 2.6d
1257     \c_math_toggle_token

```



```

1258     }
1259     {
1260 </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.

```

1261     \if_mode_vertical:
1262     \nointerlineskip
1263     \hbox_to_wd:nn { .6 \linewidth } { }
1264     \fi:
1265     \c_math_toggle_token \c_math_toggle_token
1266 <*LaTeX>
1267     }
1268 </LaTeX>

1269     \dim_zero_new:N \l_@@_linewidth_dim
1270 <*LaTeX>
1271     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1272     { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
1273     { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
1274 </LaTeX>
1275 <*plain-TeX>
1276     \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
1277 </plain-TeX>

1278     \box_clear_new:N \l_@@_halign_box
1279     \setbox \l_@@_halign_box \vtop \bgroup
1280     \tabskip =
1281     \bool_if:NTF \l_@@_fleqn_bool
1282     \c_zero_skip
1283     { 0 pt plus 1000 pt minus 1000 pt }

```

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

```

1284     \spread@equation
1285     \@@_construct_halign:
1286     \tabskip = 0 pt plus 1000 pt minus 1000 pt
1287     &

```

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.

```

1288     $ ## $
1289     \tabskip = \c_zero_skip
1290     &&
1291     \@@_fatal:n { Too~much~columns~in~DispWithArrows }
1292     \bool_if:nT \c_false_bool { ## }
1293     \cr
1294     }

```

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

```

1295 <*plain-TeX>
1296 \cs_new_protected:Npn \endDispWithArrows
1297 </plain-TeX>
1298 {
1299 <*LaTeX>
1300     \clist_if_in:NnT \l_@@_tags_clist { last }
1301     { \clist_set:Nn \l_@@_tags_clist { all } }
1302 </LaTeX>
1303     \\

```

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

```

1304 \egroup
1305 \unskip \unpenalty \unskip \unpenalty
1306 \box_set_to_last:N \l_tmpa_box
1307 \nointerlineskip
1308 \box_use:N \l_tmpa_box
1309 \dim_gzero_new:N \g_@@_alignment_dim
1310 \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
1311 \box_clear_new:N \l_@@_new_box
1312 \hbox_set:Nn \l_@@_new_box { \hbox_unpack_drop:N \l_tmpa_box }
1313 \dim_compare:nNnT
1314   { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
1315   { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }

```

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

```

1316 \egroup
1317 \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novaluetl
1318   { \box_use_drop:N \l_@@_halign_box }
1319   {
1320     \hbox_to_wd:nn \l_@@_linewidth_dim
1321     {
1322       \bool_if:NTF \l_@@_fleqn_bool
1323         { \skip_horizontal:N \l_@@_mathindent_skip }
1324         \hfil
1325       \hbox_to_wd:nn \g_@@_alignment_dim
1326       {
1327         \box_use_drop:N \l_@@_left_brace_box

```

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

```

1328     \dim_set:Nn \l_tmpa_dim
1329     {
1330       \box_ht:N \l_@@_halign_box
1331       + \box_dp:N \l_@@_halign_box
1332     }
1333     \group_begin:
1334     \dim_zero:N \nulldelimiterspace
1335     \c_math_toggle_token
1336     \left \l_@@_replace_left_brace_by_tl
1337     \vcenter to \l_tmpa_dim { \vfil }
1338     \right.
1339     \c_math_toggle_token
1340     \group_end:
1341     \hfil
1342   }
1343   \hfil
1344 }
1345 \skip_horizontal:N -\l_@@_linewidth_dim
1346 \vcenter { \box_use_drop:N \l_@@_halign_box }
1347 }

```

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.

```

1348 \dim_gzero_new:N \g_@@_right_x_dim
1349 \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
1350 \pgfpicture
1351 \pgfrememberpicturepositiononpagetrue
1352 \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
1353 {
1354   \cs_if_free:cTF
1355     { pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
1356     { \@@_fatal:n { Inexistent-v-node } }

```

```

1357     {
1358         \pgfpointanchor
1359         { wa - \l_@@_prefix_str - \l_tmpa_int - v }
1360         { center }
1361         \dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
1362         { \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x }
1363     }
1364 }
1365 \endpgfpicture

```

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

```

1366 \@@_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).

```

1367 <*LaTeX>
1368 \bool_if:nT \c_@@_mathtools_loaded_bool
1369 { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1370 \bool_if:NTF \l_@@_in_label_or_minipage_bool
1371 {
1372     \c_math_toggle_token
1373     \skip_vertical:N \belowdisplayskip
1374 }
1375 { \c_math_toggle_token \c_math_toggle_token }
1376 </LaTeX>
1377 <*plain-TeX>
1378     \c_math_toggle_token \c_math_toggle_token
1379 </plain-TeX>
1380 <*LaTeX>
1381 \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`).

```

1382 \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
1383 </LaTeX>
1384 <*plain-TeX>
1385 \group_end:
1386 </plain-TeX>
1387 <*LaTeX>
1388 \ignorespacesafterend
1389 </LaTeX>
1390 }

```

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

```

1391 <*LaTeX>
1392 \NewDocumentEnvironment { DispWithArrows* } { }
1393 {
1394     \WithArrowsOptions { notag }
1395     \DispWithArrows
1396 }
1397 \endDispWithArrows
1398 </LaTeX>

```

12.9 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.

```

1399 \cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
1400   {
1401     \bool_if:NTF \l_@@_in-WithArrows_bool
1402       { \@@_error:nn { Not~allowed~in~WithArrows } { #1 } }
1403     {
1404       \int_compare:nNnTF \g_@@_col_int < \l_@@_nb_cols_int
1405         { \@@_error:nn { Not~allowed~in~DispWithArrows } { #1 } }
1406         { #2 }
1407     }
1408   }

```

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

```

1409 ⟨*LaTeX⟩
1410 \cs_new_protected:Npn \@@_notag:
1411   { \@@_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}`.

```

1412 \cs_new_protected:Npn \@@_nonumber:
1413   { \@@_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.

```

1414 \NewDocumentCommand \@@_tag { s m }
1415   {
1416     \@@_if_in_last_col_of_disp:Nn \tag
1417     {
1418       \tl_if_empty:NF \l_@@_tag_tl
1419         { \@@_error:nn { Multiple~tags } { #2 } }
1420       \clist_set:Nn \l_@@_tags_clist { all }
1421       \bool_if:nT \c_@@_mathtools_loaded_bool
1422         {
1423           \MH_if_boolean:nT { show_only_refs }
1424             {
1425               \MH_if_boolean:nF { show_manual_tags }
1426                 { \clist_clear:N \l_@@_tags_clist }
1427             }
1428         }
1429       \tl_set:Nn \l_@@_tag_tl { #2 }
1430       \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`.³⁸

```

1431     \bool_if:nT { #1 && ! \bool_if_p:N \c_@@_amsmath_loaded_bool }
1432     { \@@_error:n { tag*~without~amsmath } }
1433   }
1434 }

```

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

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`.

```

1435 \cs_new_protected:Npn \@@_label:n #1
1436   {
1437     \@@_if_in_last_col_of_disp:Nn \label
1438     {
1439       \seq_if_empty:NF \l_@@_labels_seq
1440       {
1441         \bool_if:NTF \c_@@_cleveref_loaded_bool
1442         { \@@_error:n { Multiple-labels-with-cleveref } }
1443         { \@@_error:n { Multiple-labels } }
1444       }
1445       \seq_put_right:Nn \l_@@_labels_seq { #1 }
1446       \bool_if:nT \c_@@_mathtools_loaded_bool
1447       {
1448         \MH_if_boolean:nT { show_only_refs }
1449         {
1450           \cs_if_exist:cTF { MT_r_#1 }
1451           { \clist_set:Nn \l_@@_tags_clist { all } }
1452           { \clist_clear:N \l_@@_tags_clist }
1453         }
1454       }
1455       \bool_if:nT \c_@@_autonum_loaded_bool
1456       {
1457         \cs_if_exist:cTF { autonum#1Referenced }
1458         { \clist_set:Nn \l_@@_tags_clist { all } }
1459         { \clist_clear:N \l_@@_tags_clist }
1460       }
1461     }
1462   }

```

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

```

1463 \cs_new_protected:Npn \@@_tagnextline:
1464   {
1465     \@@_if_in_last_col_of_disp:Nn \tagnextline
1466     { \bool_set_true:N \l_@@_tag_next_line_bool }
1467   }

```

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`.

```

1468 \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }
1469 \cs_new_protected:Npn \@@_set_qedhere: { \cs_set_eq:NN \qedhere \@@_qedhere: }

```

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

```

1470 \cs_new_protected:Npn \@@_qedhere_i:
1471   {
1472     \group_begin:
1473     \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.

```

1474     \cs_set_eq:NN \qed@elt \setQED@elt
1475     \QED@stack \relax \relax
1476   \group_end:

```

```

1477   }
1478 </LaTeX>

```

12.10 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.

```

1479 \cs_new_protected:Npn \@@_scan_arrows:
1480   {
1481     \group_begin:
1482     \int_compare:nNnT \l_@@_pos_arrow_int = 7
1483       {
1484         \@@_scan_arrows_i:
1485         \int_set:Nn \l_@@_pos_arrow_int 8
1486       }
1487     \@@_scan_arrows_i:
1488     \group_end:
1489   }

```

```

1490 \cs_new_protected:Npn \@@_scan_arrows_i:
1491   {

```

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

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

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

```

1492     \int_zero_new:N \l_@@_first_arrow_of_group_int
1493     \int_zero_new:N \l_@@_first_line_of_group_int
1494     \int_zero_new:N \l_@@_last_line_of_group_int
1495     \seq_clear_new:N \l_@@_first_arrows_seq
1496     \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.

```

1497     \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.

```

1498     \int_set:Nn \l_@@_arrow_int 1
1499     \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1500     {

```

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.

```

1501     \prop_get:cnN
1502     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1503     { initial } \l_tmpa_tl
1504     \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1505     \prop_get:cnN
1506     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1507     { final } \l_tmpa_tl
1508     \int_set:Nn \l_@@_final_int \l_tmpa_tl
1509     \prop_get:cnN
1510     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1511     { status } \l_@@_status_arrow_str
1512     \prop_get:cnN
1513     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1514     { 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`.

```

1515     \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1516     {
1517         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1518         { \@@_error:n { Too-few-lines-for-an-arrow } }
1519     }
1520     \@@_treat_an_arrow_in_scan:

```

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

```

1521     \int_incr:N \l_@@_arrow_int
1522 }

```

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).

```

1523     \bool_if:nT
1524     {
1525         ! \int_compare_p:nNn \l_@@_pos_arrow_int = 7
1526         &&
1527         \int_compare_p:nNn \l_@@_first_arrow_of_group_int > 0
1528     }
1529     { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1530 }

```

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

```

1531 \cs_new_protected:Npn \@@_treat_an_arrow_in_scan:
1532 {

```

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`).

```

1533     \bool_lazy_and:nnT
1534     { \int_compare_p:nNn \l_@@_arrow_int > 1 }
1535     {
1536         \bool_lazy_or_p:nn
1537         {

```

```

1538     \bool_lazy_and_p:nn
1539     {
1540         \int_compare_p:nNn
1541             \l_@@_initial_int > \l_@@_last_line_of_group_int
1542     }
1543     { \bool_not_p:n { \int_compare_p:nNn \l_@@_pos_arrow_int = 7 } }
1544 }
1545 { \str_if_eq_p:Vn \l_@@_status_arrow_str { new-group } }
1546 }
1547 {
1548     \int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
1549     {
1550         \@@_draw_arrows:nn
1551             \l_@@_first_arrow_of_group_int
1552             { \l_@@_arrow_int - 1 }
1553     }
1554     \bool_set_true:N \l_@@_new_group_bool
1555 }

```

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`.

```

1556     \bool_if:nTF \l_@@_new_group_bool
1557     {
1558         \bool_set_false:N \l_@@_new_group_bool
1559         \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1560         \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1561         \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1562         \seq_clear:N \l_@@_first_arrows_seq
1563         \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
1564         \seq_clear:N \l_@@_last_arrows_seq
1565         \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.

```

1566         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1567         { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1568     }

```

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

```

1569     {

```

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

```

1570         \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1571         {

```

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

```

1572         \int_compare:nT
1573             { \l_@@_initial_int = \l_@@_first_line_of_group_int }
1574             { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1575         \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1576         {
1577             \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1578             \seq_clear:N \l_@@_last_arrows_seq
1579             \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int

```



```

1580     }
1581     {
1582         \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1583         { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1584     }
1585 }
1586 }

```

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.

```

1587     \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1588     {
1589         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1590         { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1591     }
1592 }

```

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

```

1593 \cs_generate_variant:Nn \keys_set:nn { n o }
1594 \cs_new_protected:Npn \@@_keys_set:
1595   { \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`.

```

1596 \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1597   {
1598     \group_begin:
1599     \int_zero_new:N \l_@@_first_arrow_int
1600     \int_set:Nn \l_@@_first_arrow_int { #1 }
1601     \int_zero_new:N \l_@@_last_arrow_int
1602     \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.

```

1603     \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1604     \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1605     {

```

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.

```

1606         \prop_get:cnN
1607         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1608         { initial } \l_tmpa_tl
1609         \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1610         \prop_get:cnN
1611         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1612         { final } \l_tmpa_tl
1613         \int_set:Nn \l_@@_final_int \l_tmpa_tl
1614         \prop_get:cnN
1615         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1616         { status } \l_@@_status_arrow_str

```

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

```

1617         \int_compare:nNnF \l_@@_final_int > \g_@@_line_int

```

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

```

1618     {
1619         \str_if_eq:VnTF \l_@@_status_arrow_str { over }
1620         { \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
1621         \@@_draw_arrow:
1622     }
1623     \int_incr:N \l_@@_arrow_int
1624 }
1625 \@@_draw_o_arrows_of_the_group:
1626 \group_end:
1627 }

```

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).

```

1628 \cs_new_protected:Npn \@@_draw_arrow:
1629 {
1630     \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.

```

1631     \prop_get:cnN
1632     { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1633     { options } \l_tmpa_tl
1634     \str_clear_new:N \l_@@_previous_key_str
1635     \exp_args:NNo \exp_args:No
1636     \@@_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`:

```

1637     \bool_set_false:N \l_@@_initial_r_bool
1638     \bool_set_false:N \l_@@_final_r_bool
1639     \int_case:nn \l_@@_pos_arrow_int
1640     {
1641         0 { \bool_set_true:N \l_@@_final_r_bool }
1642         2 { \bool_set_true:N \l_@@_initial_r_bool }
1643         3
1644         {
1645             \bool_set_true:N \l_@@_initial_r_bool
1646             \bool_set_true:N \l_@@_final_r_bool
1647         }
1648     }

```

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

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

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

```

1649     \int_compare:nNnT \l_@@_pos_arrow_int = 5
1650     {
1651         \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1652         \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1653     }

```

`\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`.

```

1654   \tl_set:Nx \l_@@_initial_tl
1655       { \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl }
1656   \tl_set:Nx \l_@@_final_tl
1657       { \int_use:N \l_@@_final_int - \bool_if:NTF \l_@@_final_r_bool rl }

```

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

```

1658   \prop_get:cnN
1659       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1660       { label }
1661       \l_tmpa_tl

```

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

```

1662   \seq_if_in:NxTF \l_@@_first_arrows_seq
1663       { \int_use:N \l_@@_arrow_int }
1664       { \bool_set_true:N \l_tmpa_bool }
1665       { \bool_set_false:N \l_tmpa_bool }
1666   \seq_if_in:NxTF \l_@@_last_arrows_seq
1667       { \int_use:N \l_@@_arrow_int }
1668       { \bool_set_true:N \l_tmpb_bool }
1669       { \bool_set_false:N \l_tmpb_bool }
1670   \int_compare:nNnT \l_@@_pos_arrow_int = 5
1671       {
1672         \bool_set_true:N \l_tmpa_bool
1673         \bool_set_true:N \l_tmpb_bool
1674       }

```

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`).

```

1675   \dim_gzero_new:N \g_@@_x_initial_dim
1676   \dim_gzero_new:N \g_@@_x_final_dim
1677   \dim_gzero_new:N \g_@@_y_initial_dim
1678   \dim_gzero_new:N \g_@@_y_final_dim
1679   \pgfpicture
1680     \pgfrememberpicturepositiononpagetrue
1681     \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south }
1682     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1683     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1684     \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north }
1685     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1686     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1687   \endpgfpicture
1688   \bool_lazy_and:nnTF
1689       {
1690         \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
1691         > \l_@@_max_length_of_arrow_dim

```

```

1692 }
1693 { \int_compare:nNn { \l_@@_final_int - \l_@@_initial_int } = 1 }
1694 {
1695   \tl_gset:Nx \g_tmpa_tl
1696   {
1697     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1698     { \dim_use:N \g_@@_x_initial_dim }
1699     { \dim_use:N \l_@@_x_dim } ,
1700     \dim_eval:n
1701     {
1702       ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1703       + 0.5 \l_@@_max_length_of_arrow_dim
1704     }
1705   }
1706   \tl_gset:Nx \g_tmpb_tl
1707   {
1708     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1709     { \dim_use:N \g_@@_x_final_dim }
1710     { \dim_use:N \l_@@_x_dim } ,
1711     \dim_eval:n
1712     {
1713       ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1714       - 0.5 \l_@@_max_length_of_arrow_dim
1715     }
1716   }
1717 }
1718 {
1719   \tl_gset:Nx \g_tmpa_tl
1720   {
1721     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1722     { \dim_use:N \g_@@_x_initial_dim }
1723     { \dim_use:N \l_@@_x_dim } ,
1724     \bool_if:NNTF \l_tmpa_bool
1725     { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
1726     { \dim_use:N \g_@@_y_initial_dim }
1727   }
1728   \tl_gset:Nx \g_tmpb_tl
1729   {
1730     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1731     { \dim_use:N \g_@@_x_final_dim }
1732     { \dim_use:N \l_@@_x_dim } ,
1733     \bool_if:NNTF \l_tmpb_bool
1734     { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
1735     { \dim_use:N \g_@@_y_final_dim }
1736   }
1737 }

```

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

```

1738   \bool_if:NF \l_@@_right_overlap_bool
1739   {
1740     \bool_if:NT \l_@@_in_WithArrows_bool
1741     {
1742       \pgfpicture
1743       \pgfrememberpicturepositiononpagetrue
1744       \pgfpointanchor { wa - \l_@@_prefix_str - 1 - r } { south }
1745       \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1746       {
1747         \dim_set:Nn \l_@@_delta_x_dim
1748         {
1749           \pgf@x -
1750           ( \dim_min:nn \g_@@_x_initial_dim \g_@@_x_final_dim )
1751         }
1752       }

```

```

1753         { \dim_set:Nn \l_@@_delta_x_dim { \pgf@x - \l_@@_x_dim } }
1754     \endpgfpicture
1755 }
1756 }

```

Eventually, we can draw the arrow with the code in `\l_@@_tikz_code_tl`. We recall that the value by default for this token list is: “`\draw (#1) to node {#3} (#2) ;`”. This value can be modified with the option `tikz-code`. We use the variant `@@_draw_arrow:nno` of the macro `@@_draw_arrow:nnn` because of the characters *underscore* in the name `\l_tmpa_tl`: if the user uses the Tikz library `babel`, the third argument of the command `@@_draw_arrow:nno` will be rescanned because this third argument will be in the argument of a command `node` of an instruction `\draw` of Tikz... and we will have an error because of the characters *underscore*.³⁹

```

1757     \@@_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).

```

1758     \group_end:
1759 }

```

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`.

```

1760 \cs_new_protected:Npn \@@_def_function_tmpa:n #1
1761 {
1762     \cs_set:Npn \@@_tmpa:nnn ##1 ##2 ##3
1763     {
1764     <*LaTeX>
1765     \begin{tikzpicture}
1766     </LaTeX>
1767     <*plain-TeX>
1768     \tikzpicture
1769     </plain-TeX>
1770     [
1771     @@_standard ,
1772     every-path / .style = WithArrows / arrow
1773     ]

```

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

```

1774     \pgf@relevantforpicturesizetrue
1775     #1
1776     \dim_compare:nNnTF \pgf@picminx = { 16000 pt }
1777     { \dim_zero:N \l_tmpa_dim }
1778     { \dim_set:Nn \l_tmpa_dim { \pgf@picmaxx - \pgf@picminx } }
1779     \dim_add:Nn \l_tmpa_dim \l_@@_xoffset_dim
1780     \prop_gput:cnV
1781     { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1782     { width }
1783     \l_tmpa_dim

```

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

```

1784     \bool_if:NF \l_@@_right_overlap_bool
1785     {
1786     \bool_if:NT \l_@@_in_WithArrows_bool
1787     {
1788     \dim_gset:Nn \g_@@_overlap_x_dim
1789     {
1790     \dim_max:nn

```

³⁹There were other solutions: use another name without *underscore* (like `\ltmpat1`) or use the package `underscore` (with this package, the characters *underscore* will be rescanned without errors, even in text mode).

```

1791             \g_@@_overlap_x_dim
1792             { \l_tmpa_dim - \l_@@_delta_x_dim }
1793         }
1794     }
1795 }
1796 \pgfresetboundingbox
1797 <*LaTeX>
1798 \end{tikzpicture}
1799 </LaTeX>
1800 <*plain-TeX>
1801 \endtikzpicture
1802 </plain-TeX>
1803 }
1804 }

```

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

```

1805 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1806 {

```

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`).

```

1807     \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
1808     { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }

```

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

```

1809     \exp_args:NV \@@_def_function_tmpa:n \l_@@_tikz_code_tl
1810     \@@_tmpa:nnn { #1 } { #2 } { #3 }
1811 }
1812 \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`).

```

1813 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1814 {
1815     \pgfset { inner~sep = Opt}

```

First, we draw the arrow without the label.

```

1816     \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;

```

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

```

1817     \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }

```

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

```

1818     \dim_set:Nn \l_tmpa_dim { \g_@@_right_x_dim - \pgf@x - 0.3333 ex }

```

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

```

1819     \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`).

```

1820     \tl_if_empty:NF \g_tmpa_tl
1821     {
1822         \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1823         \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1824         { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1825     }

```

⁴⁰In fact, it's not the current value of “`text width`”: it's the value of “`text width`” set in the option `tikz` provided by `witharrows`. These options are given to Tikz in a “`every path`”. That's why we have to retrieve it in a path.

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

```

1826   \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1827   {
1828     \path ( @@_label.west )
1829   (*LaTeX)
1830     node [ anchor = west ]
1831     {
1832       \skip_horizontal:n { 0.3333 ex }
1833       \begin { minipage } { \l_tmpa_dim }
1834       \tikz@text@action
1835       \pgfkeysgetvalue { / tikz / node-halign-header } \l_tmpa_tl
1836       \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@left@header }
1837       { \pgfutil@raggedright }
1838       {
1839         \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@right@header }
1840         { \pgfutil@raggedleft }
1841         {
1842           \tl_if_eq:NnT \l_tmpa_tl { \tikz@align@center@header }
1843           { \centering }
1844         }
1845       }
1846       #3
1847       \end { minipage }
1848     } ;
1849   (/LaTeX)
1850 (*plain-TeX)
1851   node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1852   { #3 } ;
1853 (/plain-TeX)
1854 }
1855 }

```

12.10.1 The command `update_x`

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

```

1856 \cs_new_protected:Npn \@@_update_x:nn #1 #2
1857 {
1858   \dim_gset_eq:NN \g_tmpa_dim \l_@@_x_dim
1859   \pgfpicture
1860   \pgfrememberpicturepositiononpagetrue
1861   \int_step_inline:nnn { #1 } { #2 }
1862   {
1863     \pgfpointanchor { wa - \l_@@_prefix_str - ##1 - 1 } { center }
1864     \dim_gset:NN \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \pgf@x }
1865   }
1866   \endpgfpicture
1867   \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1868 }

```

12.10.2 We draw the arrows of type `o`

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

```

1869 \cs_new_protected:Npn \@@_draw_o_arrows_of_the_group:
1870 {

```

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

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

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

```


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


```

```

1871 \seq_sort:Nn \l_@@_o_arrows_seq
1872 {
1873   \prop_get:cnN
1874   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1875   { final } \l_tmpa_tl

```

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

```

1876 \prop_get:cnN
1877 { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1878 { final } \l_tmpb_tl
1879 \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1880 \sort_return_same:
1881 {
1882   \int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
1883   \sort_return_swapped:
1884   {
1885     \prop_get:cnN
1886     { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1887     { initial } \l_tmpa_tl
1888     \prop_get:cnN
1889     { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1890     { initial } \l_tmpb_tl
1891     \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1892     \sort_return_swapped:
1893     \sort_return_same:
1894   }
1895 }
1896 }

```

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

```

1897 \seq_map_inline:Nn \l_@@_o_arrows_seq
1898 {

```

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

```

1899 \prop_get:cnN
1900 { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1901 { initial } \l_tmpa_tl
1902 \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1903 \prop_get:cnN
1904 { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1905 { final } \l_tmpa_tl
1906 \int_set:Nn \l_@@_final_int \l_tmpa_tl

```


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

```

1907     \prop_get:cnN
1908     { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1909     { input-line } \l_@@_input_line_str

```

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

```

1910     \dim_gzero:N \g_tmpa_dim

```

We will raise the boolean `\g_tmpa_bool` if we find an arrow “under” our arrow (we should find at least once since you are drawing an arrow of type `o`: if not, we will raise an error⁴¹).

```

1911     \bool_set_false:N \g_tmpa_bool
1912     \pgfpicture
1913     \pgfrememberpicturepositiononpagetrue
1914     \int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
1915     {
1916         \prop_get:cnN
1917         { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1918         { initial } \l_tmpa_tl
1919         \prop_get:cnN
1920         { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1921         { final } \l_tmpb_tl
1922         \prop_get:cnN
1923         { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1924         { status } \l_@@_status_arrow_str
1925         \bool_if:nT
1926         {
1927             ! \int_compare_p:n { ##1 = ####1 }
1928             && \int_compare_p:n { \l_@@_initial_int <= \l_tmpa_tl }
1929             && \int_compare_p:n { \l_tmpb_tl <= \l_@@_final_int }

```

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

```

1930         && ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
1931     }
1932     {

```

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

```

1933         \bool_gset_true:N \g_tmpa_bool
1934         \prop_get:cnN
1935         { g_@@ _ arrow _ \l_@@_prefix_str _ ####1 _ prop }
1936         { width }
1937         \l_tmpa_tl

```

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

```

1938         \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \l_tmpa_tl }
1939     }
1940 }
1941 \endpgfpicture

```

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

```

1942     \bool_if:NTF \g_tmpa_bool
1943     {
1944         \int_set:Nn \l_@@_arrow_int { ##1 }
1945         \dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
1946         \dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
1947         \@_draw_arrow:
1948     }
1949     { \@_error:n { o~arrow~with~no~arrow~under } }
1950 }
1951 }

```

⁴¹Maybe we will change that in future versions.

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.

```

1952 <LaTeX>
1953 \NewExpandableDocumentCommand \WithArrowsLastEnv { }
1954   { \int_use:N \g_@@_last_env_int }
1955 </LaTeX>
1956 <plain-TeX>
1957 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
1958 </plain-TeX>

```

12.11 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`.

```

1959 \keys_define:nn { WithArrows / Arrow / code-after }
1960   {
1961     tikz      .code:n =
1962       \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
1963     tikz      .value_required:n = true ,
1964     rr        .value_forbidden:n = true ,
1965     rr        .code:n = \@@_fix_pos_option:n 0 ,
1966     ll        .value_forbidden:n = true ,
1967     ll        .code:n = \@@_fix_pos_option:n 1 ,
1968     rl        .value_forbidden:n = true ,
1969     rl        .code:n = \@@_fix_pos_option:n 2 ,
1970     lr        .value_forbidden:n = true ,
1971     lr        .code:n = \@@_fix_pos_option:n 3 ,
1972     v         .value_forbidden:n = true ,
1973     v         .code:n = \@@_fix_pos_option:n 4 ,
1974     tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
1975     tikz-code .value_required:n = true ,
1976     xoffset   .dim_set:N = \l_@@_xoffset_dim ,
1977     xoffset   .value_required:n = true ,
1978     unknown   .code:n =
1979       \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
1980       \@@_error:n { Unknown~option~Arrow~in~code~after }
1981   }

```

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.

```

1982 \seq_new:N \l_@@_options_Arrow_code_after_seq
1983 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq
1984   { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }

```

```

1985 <LaTeX>
1986 \NewDocumentCommand \@@_Arrow_code_after { 0 { } m m m ! 0 { } }
1987 </LaTeX>
1988 <plain-TeX>
1989 \cs_new_protected:Npn \@@_Arrow_code_after
1990   {
1991     \peek_meaning:NTF [
1992       { \@@_Arrow_code_after_i }
1993       { \@@_Arrow_code_after_i [ ] }
1994     }
1995 \cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4

```

```

1996 {
1997   \peek_meaning:NTF [
1998     { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1999     { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
2000   ]
2001 \cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
2002 </plain-TeX>
2003 {
2004   \int_set:Nn \l_@@_pos_arrow_int 1
2005   \str_clear_new:N \l_@@_previous_key_str
2006   \group_begin:
2007     \keys_set:nn { WithArrows / Arrow / code-after }
2008       { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
2009     \bool_set_false:N \l_@@_initial_r_bool
2010     \bool_set_false:N \l_@@_final_r_bool
2011     \int_case:nn \l_@@_pos_arrow_int
2012       {
2013         0
2014         {
2015           \bool_set_true:N \l_@@_initial_r_bool
2016           \bool_set_true:N \l_@@_final_r_bool
2017         }
2018         2 { \bool_set_true:N \l_@@_initial_r_bool }
2019         3 { \bool_set_true:N \l_@@_final_r_bool }
2020       }

```

We prevent drawing an arrow from a line to itself.

```

2021 \tl_if_eq:nnTF { #2 } { #3 }
2022   { \@@_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.

```

2023 {
2024   \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - 1 }
2025     { \@@_error:nx { Wrong~line~in~Arrow } { #2 } }
2026   {
2027     \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - 1 }
2028       { \@@_error:nx { Wrong~line~in~Arrow } { #3 } }
2029     {
2030       \int_compare:nNnTF \l_@@_pos_arrow_int = 4
2031         {
2032           \pgfpicture
2033             \pgfrememberpicturepositiononpagetrue
2034             \pgfpointanchor { wa - \l_@@_prefix_str - #2 - 1 }
2035               { south }
2036             \dim_set_eq:NN \l_tmpa_dim \pgf@x
2037             \dim_set_eq:NN \l_tmpb_dim \pgf@y
2038             \pgfpointanchor { wa - \l_@@_prefix_str - #3 - 1 }
2039               { north }
2040             \dim_set:Nn \l_tmpa_dim
2041               { \dim_max:nn \l_tmpa_dim \pgf@x }
2042             \tl_gset:Nx \g_tmpa_tl
2043               { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
2044             \tl_gset:Nx \g_tmpb_tl
2045               { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
2046           \endpgfpicture
2047         }
2048       {
2049         \pgfpicture
2050           \pgfrememberpicturepositiononpagetrue
2051           \pgfpointanchor
2052             {
2053               wa - \l_@@_prefix_str -
2054               #2 - \bool_if:NTF \l_@@_initial_r_bool r l

```

```

2055         }
2056         { south }
2057         \tl_gset:Nx \g_tmpa_tl
2058         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
2059         \pgfpicture
2060         {
2061             wa - \l_@@_prefix_str -
2062             #3 - \bool_if:NTF \l_@@_final_r_bool r l
2063         }
2064         { north }
2065         \tl_gset:Nx \g_tmpb_tl
2066         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
2067         \endpgfpicture
2068     }
2069     \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
2070 }
2071 }
2072 }
2073 \group_end:
2074 }

```

12.12 The command `\MultiArrow` in code-after

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

```

2075 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
2076 {

```

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.

```

2077     \exp_args:Nnx
2078     \regex_match:nnTF
2079     { \A \d+ (\, \d+)* ( \, \.\.\. (\, \d+)+ )* \Z }
2080     { #1 }
2081     { \@@_MultiArrow_i:nn { #1 } { #2 } }
2082     { \@@_error:nx { Invalid~specification~for~MultiArrow } { #1 } }
2083 }
2084 \cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
2085 {

```

That’s why we construct a “clist” of L3 from the specification of list given by the user. The construction of the “clist” must be global in order to exit the `\foreach` and that’s why we will construct the list in `\g_tmpa_clist`.

```

2086     \foreach \x in { #1 }
2087     {
2088         \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
2089         { \@@_error:nx { Wrong~line~specification~in~MultiArrow } \x }
2090         { \clist_gput_right:Nx \g_tmpa_clist \x }
2091     }

```

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

```

2092     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
2093     { \@@_error:n { Too~small~specification~for~MultiArrow } }
2094     {
2095         \clist_sort:Nn \g_tmpa_clist
2096         {
2097             \int_compare:nTF { ##1 > ##2 }
2098             \sort_return_swapped:
2099             \sort_return_same:
2100         }

```

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

```
2101 \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...

```
2102 \clist_reverse:N \g_tmpa_clist
2103 \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).

```
2104 \exp_args:NV \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
2105 <*LaTeX>
2106 \begin { tikzpicture }
2107 </LaTeX>
2108 <*plain-TeX>
2109 \tikzpicture
2110 </plain-TeX>
2111 [
2112 \@@_standard ,
2113 every~path / .style = { WithArrows / arrow }
2114 ]
2115 \draw [<->] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
2116 -- ++(5mm,0)
2117 -- node (@@_label) {}
2118 ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
2119 -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south) ;
2120 \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }
2121 \dim_set:Nn \l_tmpa_dim { 20 cm }
2122 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
2123 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
2124 \bool_lazy_and:nNT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
2125 {
2126 \dim_set:Nn \l_tmpb_dim
2127 { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
2128 \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
2129 { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
2130 }
2131 \path (@@_label.west)
2132 node [ anchor = west, text~width = \dim_use:N \l_tmpa_dim ] { #2 } ;
2133 <*LaTeX>
2134 \end { tikzpicture }
2135 </LaTeX>
2136 <*plain-TeX>
2137 \endtikzpicture
2138 </plain-TeX>
2139 }
2140 }

2141 \cs_new_protected:Npn \@@_MultiArrow_i:n #1
2142 {
2143 <*LaTeX>
2144 \begin { tikzpicture }
2145 </LaTeX>
2146 <*plain-TeX>
2147 \tikzpicture
2148 </plain-TeX>
2149 [
2150 \@@_standard ,
2151 every~path / .style = { WithArrows / arrow }
2152 ]
2153 \foreach \k in { #1 }
```

```

2154     {
2155         \draw [ <- ]
2156             ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
2157     } % ;
2158 <*LaTeX>
2159     \end{tikzpicture}
2160 </LaTeX>
2161 <*plain-TeX>
2162     \endtikzpicture
2163 </plain-TeX>
2164 }

```

12.13 The error messages of the package

```

2165 \bool_if:NTF \c_@@_messages_for_Overleaf_bool
2166 { \str_const:Nn \c_@@_available_keys_str { } }
2167 {
2168     \str_const:Nn \c_@@_available_keys_str
2169     { For~a~list~of~the~available~keys,~type-H~<return>. }
2170 }
2171 \str_new:N \l_witharrows_body_str

```

The following commands must *not* be protected since they will be used in error messages.

```

2172 \cs_new:Npn \@@_potential_body_i:
2173 {
2174     \str_if_empty:NF \l_witharrows_body_str
2175     { \ \ If~you~want~to~see~the~body~of~the~environment,~type-H~<return>. }
2176 }
2177 \cs_new:Npn \@@_potential_body_ii:
2178 {
2179     \str_if_empty:NTF \l_witharrows_body_str
2180     { No~further~help~available }
2181     {
2182         The~body~of~your~environment~was:\\
2183         \l_witharrows_body_str
2184     }
2185 }
2186 \str_const:Nn \c_@@_option_ignored_str
2187 { If~you~go~on,~this~option~will~be~ignored. }
2188 \str_const:Nn \c_@@_command_ignored_str
2189 { If~you~go~on,~this~command~will~be~ignored. }
2190 <*LaTeX>
2191 \@@_msg_new:nn { amsmath~not~loaded }
2192 {
2193     amsmath~not~loaded.\\
2194     You~can't~use~the~option~'\l_keys_key_str'~because~the~
2195     package~'amsmath'~has~not~been~loaded.\\
2196     If~you~go~on,~this~option~will~be~ignored~in~the~rest~
2197     of~the~document.
2198 }
2199 </LaTeX>
2200 \@@_msg_new:nn { Bad~value~for~replace~brace~by }
2201 {
2202     Incorrect~value.\\
2203     Bad~value~for~the~option~'\l_keys_key_str'.~The~value~must~begin~
2204     with~an~extensible~left~delimiter.~The~possible~values~are:~.,
2205     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
2206     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
2207     \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
2208     \token_to_str:N \lfloor~and~\token_to_str:N \lceil\
2209     (and~\token_to_str:N \lvert~and~\token_to_str:N \lVert\

```

```

2210   if-amsmath-or-unicode-math-is-loaded-in-LaTeX).\
2211   \c_@@_option_ignored_str
2212   }
2213 \@@_msg_new:nn { option-of-cr-negative }
2214   {
2215   Bad-value.\
2216   The-argument-of-the-command~\token_to_str:N\
2217   should-be-positive-in-the-row~\int_use:N \g_@@_line_int\
2218   of-your-environment~\{\l_@@_type_env_str\}.\
2219   \c_@@_option_ignored_str
2220   }
2221 \@@_msg_new:nn { omit-probably-used }
2222   {
2223   Strange-problem.\
2224   Maybe-you-have-used-a-command~
2225   \token_to_str:N\omit\ in-the-line~\int_use:N \g_@@_line_int\
2226   (or-another-line)~of-your-environment~\{\l_@@_type_env_str\}.\
2227   You-can-go-on-but-you-may-have-others-errors.
2228   }
2229 <*LaTeX>
2230 \@@_msg_new:nnn { newline-at-the-end-of-env }
2231   {
2232   Incorrect-end.\
2233   The-environments-of-witharrows~(\{WithArrows\})~and~
2234   \{DispWithArrows\}~should-not-end-by~\token_to_str:N \.\
2235   However,~you-can-go-on-for-this-time.~No-similar-error-will-be-
2236   raised-in-this-document.
2237   \@@_potential_body_i:
2238   }
2239   { \@@_potential_body_ii: }
2240 </LaTeX>
2241 \@@_msg_new:nnn { Invalid-option-format }
2242   {
2243   Invalide-value.\
2244   The-key~'format'~should-contain-only-letters~r,~c~and~l~and~
2245   must-not-be-empty.\
2246   \c_@@_option_ignored_str
2247   \@@_potential_body_i:
2248   }
2249   { \@@_potential_body_ii: }
2250 \@@_msg_new:nnn { invalid-key-o }
2251   {
2252   Invalid-use-of-a-key.\
2253   The-key~'o'~for-individual-arrows-can-be-used-only-in-mode~
2254   'group'~or-in-mode~'groups'.\
2255   \c_@@_option_ignored_str
2256   \@@_potential_body_i:
2257   }
2258   { \@@_potential_body_ii: }
2259 \@@_msg_new:nnn { Value-for-a-key }
2260   {
2261   Misuse-of-a-key.\
2262   The-key~'\l_keys_key_str'~should-be-used-without-value. \
2263   However,~you-can-go-on-for-this-time.
2264   \@@_potential_body_i:
2265   }
2266   { \@@_potential_body_ii: }
2267 \@@_msg_new:nnn { Unknown-option-in-Arrow }
2268   {
2269   Unknown-option.\
2270   The-key~'\l_keys_key_str'~is-unknown-for-the-command~

```

```

2271 \l_@@_string_Arrow_for_msg_str\ in~the~row~
2272 \int_use:N \g_@@_line_int\ of~your~environment~
2273 \{\l_@@_type_env_str\}. \l_tmpa_str \\  

2274 \c_@@_option_ignored_str \\  

2275 \c_@@_available_keys_str  

2276 }  

2277 {  

2278 The~available~keys~are~(in~alphabetic~order):~  

2279 \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.  

2280 }  

2281 \@@_msg_new:nnn { Unknown~option~WithArrows }  

2282 {  

2283 Unknown~option.\  

2284 The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\  

2285 \c_@@_option_ignored_str \\  

2286 \c_@@_available_keys_str  

2287 }  

2288 {  

2289 The~available~keys~are~(in~alphabetic~order):~  

2290 \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.  

2291 }  

2292 \@@_msg_new:nnn { Unknown~option~DispWithArrows }  

2293 {  

2294 Unknown~option.\  

2295 The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\  

2296 \c_@@_option_ignored_str \\  

2297 \c_@@_available_keys_str  

2298 }  

2299 {  

2300 The~available~keys~are~(in~alphabetic~order):~  

2301 \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.  

2302 }  

2303 \@@_msg_new:nnn { Unknown~option~WithArrowsOptions }  

2304 {  

2305 Unknown~option.\  

2306 The~key~'\l_keys_key_str'~is~unknown~in~  

2307 \token_to_str:N \WithArrowsOptions. \\  

2308 \c_@@_option_ignored_str \\  

2309 \c_@@_available_keys_str  

2310 }  

2311 {  

2312 The~available~keys~are~(in~alphabetic~order):~  

2313 \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.  

2314 }  

2315 \@@_msg_new:nnn { Unknown~option~Arrow~in~code~after }  

2316 {  

2317 Unknown~option.\  

2318 The~key~'\l_keys_key_str'~is~unknown~in~  

2319 \token_to_str:N \Arrow\ in~code~after. \\  

2320 \c_@@_option_ignored_str \\  

2321 \c_@@_available_keys_str  

2322 }  

2323 {  

2324 The~available~keys~are~(in~alphabetic~order):~  

2325 \seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.  

2326 }  

2327 \@@_msg_new:nnn { Too~much~columns~in~WithArrows }  

2328 {  

2329 Too~much~columns.\  

2330 Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N  

2331 \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~  

2332 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~

```



```

2333 If-you-really-want-to-use-more-columns-(after-the-arrows)-you-should-use-
2334 the-option-'more-columns'~at-a-global-level-or-for-an-environment. \\
2335 However,~you-can-go-one-for~this~time.
2336 \@@_potential_body_i:
2337 }
2338 { \@@_potential_body_ii: }
2339 \@@_msg_new:nnn { Too-much-columns-in-DispWithArrows }
2340 {
2341 Too-much-columns.\\
2342 Your-environment~\{\l_@@_type_env_str\}~has~\int_use:N
2343 \l_@@_nb_cols_int\ columns~and-you-try-to-use-one-more.~
2344 Maybe-you-have-forgotten-a~\c_backslash_str\c_backslash_str\
2345 at-the-end-of~row~\int_use:N \g_@@_line_int. \\
2346 This-error-is-fatal.
2347 \@@_potential_body_i:
2348 }
2349 { \@@_potential_body_ii: }
2350 \@@_msg_new:nn { Negative~jump }
2351 {
2352 Incorrect-value.\\
2353 You-can't-use-a-negative-value~for~the-option~'jump'~of~command~
2354 \l_@@_string_Arrow_for_msg_str\
2355 in~the~row~\int_use:N \g_@@_line_int\
2356 of~your~environment~\{\l_@@_type_env_str\}.~
2357 You-can-create-an-arrow-going-backwards~with~the-option~'<'~of~Tikz. \\
2358 \c_@@_option_ignored_str
2359 }
2360 \@@_msg_new:nn { new-group-without-groups }
2361 {
2362 Misuse-of-a-key.\\
2363 You-can't-use-the-option-'new-group'~for~the~command~
2364 \l_@@_string_Arrow_for_msg_str\
2365 because-you-are-not-in~'groups'~mode.~Try-to-use-the-option~
2366 'groups'~in~your~environment~\{\l_@@_type_env_str\}. \\
2367 \c_@@_option_ignored_str
2368 }
2369 \@@_msg_new:nnn
2370 { Too-few-lines-for-an-arrow }
2371 {
2372 Impossible-arrow.\\
2373 Line~\l_@@_input_line_str\
2374 :~an-arrow-specified-in-the-row~\int_use:N \l_@@_initial_int\
2375 of~your~environment~\{\l_@@_type_env_str\}~can't-be-drawn~
2376 because-it-arrives-after-the-last-row-of-the-environment. \\
2377 If-you-go-on,~this~arrow~will~be~ignored.
2378 \@@_potential_body_i:
2379 }
2380 { \@@_potential_body_ii: }
2381 \@@_msg_new:nn { o~arrow~with~no~arrow~under }
2382 {
2383 Problem~with~the~key~'o'.\\
2384 Line~\l_@@_input_line_str\
2385 :~there~is~no~arrow~'under'~your~arrow~of~type~'o'.\\
2386 If~you~go~on,~this~arrow~won't~be~drawn.
2387 }
2388 \@@_msg_new:nnn { WithArrows-outside-math-mode }
2389 {
2390 You-are~outside~math~mode.\\
2391 The~environment~\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
2392 like~the~environment~\{aligned\}~of~amsmath. \\
2393 Nevertheless,~you~can~go~on.

```

```

2394 \@@_potential_body_i:
2395 }
2396 { \@@_potential_body_ii: }
2397 \@@_msg_new:nn { DispWithArrows-in-math-mode }
2398 {
2399   You~are~in~math~mode.\\
2400   The~environment~\{\l_@@_type_env_str\}~should~be~used~only~outside~math~
2401   mode~like~the~environments~\{align\}~and~\{align*\}~of~amsmath. \\
2402   This~error~is~fatal.
2403   \@@_potential_body_i:
2404 }
2405 { \@@_potential_body_ii: }
2406 \@@_msg_new:nn { Incompatible~options~in~Arrow }
2407 {
2408   Incompatible~options.\\
2409   You~try~to~use~the~option~'\l_keys_key_str'~but~
2410   this~option~is~incompatible~or~redundant~with~the~option~
2411   '\l_@@_previous_key_str'~set~in~the~same~command~
2412   \l_@@_string_Arrow_for_msg_str. \\
2413   \c_@@_option_ignored_str
2414 }
2415 \@@_msg_new:nn { Incompatible~options }
2416 {
2417   Incompatible~options.
2418   You~try~to~use~the~option~'\l_keys_key_str'~but~
2419   this~option~is~incompatible~or~redundant~with~the~option~
2420   '\l_@@_previous_key_str'~set~in~the~same~command~
2421   \bool_if:NT \l_@@_in_code_after_bool
2422   {
2423     \l_@@_string_Arrow_for_msg_str\
2424     in~the~code~after~of~your~environment~\{\l_@@_type_env_str\}
2425   }. \\
2426   \c_@@_option_ignored_str
2427 }
2428 \@@_msg_new:nn { Arrow~not~in~last~column }
2429 {
2430   Bad~use~of~\l_@@_string_Arrow_for_msg_str.
2431   You~should~use~the~command~\l_@@_string_Arrow_for_msg_str\
2432   only~in~the~last~column~(column~\int_use:N\l_@@_nb_cols_int)~
2433   in~the~row~\int_use:N \g_@@_line_int\
2434   of~your~environment~\{\l_@@_type_env_str\}. \\
2435   However~you~can~go~on~for~this~time.
2436   \@@_potential_body_i:
2437 }
2438 { \@@_potential_body_ii: }
2439 \@@_msg_new:nn { Wrong~line~in~Arrow }
2440 {
2441   Wrong~line.\\
2442   The~specification~of~line~'#1'~you~use~in~the~command~
2443   \l_@@_string_Arrow_for_msg_str\
2444   in~the~'code~after'~of~\{\l_@@_type_env_str\}~doesn't~exist. \\
2445   \c_@@_option_ignored_str
2446 }
2447 \@@_msg_new:nn { Both~lines~are~equal }
2448 {
2449   Both~lines~are~equal.\\
2450   In~the~'code~after'~of~\{\l_@@_type_env_str\}~you~try~to~
2451   draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\
2452   \c_@@_option_ignored_str
2453 }
2454 \@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }

```

```

2455 {
2456 Wrong-line-specification.\\
2457 The-specification-of-line-#1'-doesn't-exist. \\
2458 If-you-go-on,~it-will-be-ignored-for-~\token_to_str:N \MultiArrow.
2459 }
2460 \@@_msg_new:nn { Too-small-specification-for-MultiArrow }
2461 {
2462 Too-small-specification.\\
2463 The-specification-of-lines-you-gave-to-~\token_to_str:N \MultiArrow\
2464 is-too-small:~you-need-at-least-two-lines. \\
2465 \c_@@_command_ignored_str
2466 }
2467 \@@_msg_new:nn { Not-allowed-in-DispWithArrows }
2468 {
2469 Forbidden-command.\\
2470 The-command-~\token_to_str:N #1
2471 is-allowed-only-in-the-last-column-
2472 (column-~\int_use:N\l_@@_nb_cols_int)-of-~\{\l_@@_type_env_str\}. \\
2473 \c_@@_option_ignored_str
2474 }
2475 \@@_msg_new:nn { Not-allowed-in-WithArrows }
2476 {
2477 Forbidden-command.\\
2478 The-command-~\token_to_str:N #1 is-not-allowed-in-~\{\l_@@_type_env_str\}-
2479 (it's-allowed-in-the-last-column-of-~\{DispWithArrows\}). \\
2480 \c_@@_option_ignored_str
2481 }
2482 (*LaTeX)
2483 \@@_msg_new:nn { tag*-without-amsmath }
2484 {
2485 amsmath-not-loaded.\\
2486 We-can't-use-~\token_to_str:N\tag*-because-you-haven't-loaded-amsmath-
2487 (or-mathtools). \\
2488 If-you-go-on,~the-command-~\token_to_str:N\tag\
2489 will-be-used-instead.
2490 }
2491 \@@_msg_new:nn { Multiple-tags }
2492 {
2493 Multiple-tags.\\
2494 You-can't-use-twice-the-command-~\token_to_str:N\tag\
2495 in-a-line-of-the-environment-~\{\l_@@_type_env_str\}. \\
2496 If-you-go-on,~the-tag-#1'-will-be-used.
2497 }
2498 \@@_msg_new:nn { Multiple-labels }
2499 {
2500 Multiple-labels.\\
2501 Normally,~we-can't-use-the-command-~\token_to_str:N\label\
2502 twice-in-a-line-of-the-environment-~\{\l_@@_type_env_str\}. \\
2503 However,~you-can-go-on.~
2504 \bool_if:NT \c_@@_showlabels_loaded_bool
2505 { However,~only-the-last-label-will-be-shown-by-showlabels.~ }
2506 If-you-don't-want-to-see-this-message-again,~you-can-use-the-option-
2507 'allow-multiple-labels'-at-the-global-or-environment-level.
2508 }
2509 \@@_msg_new:nn { Multiple-labels-with-cleveref }
2510 {
2511 Multiple-labels.\\
2512 Since-you-use-cleveref,~you-can't-use-the-command-~\token_to_str:N\label\
2513 twice-in-a-line-of-the-environment-~\{\l_@@_type_env_str\}. \\
2514 If-you-go-on,~you-may-have-undefined-references.
2515 }

```

```

2516 </LaTeX>
2517 \@@_msg_new:nn { Inexistent-v-node }
2518 {
2519   There-is-a-problem.\!
2520   Maybe-you-have-put-a-command-\token_to_str:N\cr\
2521   instead-of-a-command-\token_to_str:N\~at-the-end-of~
2522   the-row~\l_tmpa_int\
2523   of-your-environment-\{\l_@@_type_env_str\}. \!
2524   This-error-is-fatal.
2525 }

```

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`).

```

2526 \@@_msg_new:nn { Option-xoffset-forbidden }
2527 {
2528   Incorrect-key.\!
2529   You-can't-use-the-option-'xoffset'~in-the-command~
2530   \l_@@_string_Arrow_for_msg_str\ in-the-row~\int_use:N \g_@@_line_int\
2531   of-your-environment-\{\l_@@_type_env_str\}~
2532   because-you-are-using-the-option~
2533   ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
2534     { group }
2535     { groups } '~It's-possible-for-an-independent-arrow-or-if-there-is-
2536     only-one-arrow. \!
2537   \c_@@_option_ignored_str
2538 }
2539 \@@_msg_new:nnn { Duplicate-name }
2540 {
2541   Duplicate-name.\!
2542   The-name~'\l_keys_value_tl'~is-already-used-and-you-shouldn't-use~
2543   the-same-environment-name-twice.~You-can-go-on,~but,~
2544   maybe,~you-will-have-incorrect-results. \!
2545   For-a-list-of-the-names-already-used,~type-H<return>. \!
2546   If-you-don't-want-to-see-this-message-again,~use-the-option~
2547   'allow-duplicate-names'.
2548 }
2549 {
2550   The-names-already-defined-in-this-document-are:~
2551   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2552 }
2553 \@@_msg_new:nn { Invalid-specification-for-MultiArrow }
2554 {
2555   Invalid-specification.\!
2556   The-specification-of-rows-for-\token_to_str:N\MultiArrow\
2557   (i.e.~#1)~is-invalid. \!
2558   \c_@@_command_ignored_str
2559 }

```

12.14 The command `\WithArrowsNewStyle`

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

```

2560 <*LaTeX>
2561 \NewDocumentCommand \WithArrowsNewStyle { m m }
2562 </LaTeX>
2563 <*plain-TeX>
2564 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2565 </plain-TeX>
2566 {
2567   \keys_if_exist:nNTF { WithArrows / Global } { #1 }
2568   { \@@_error:nn { Key-already-defined } { #1 } }
2569   {

```

First, we detect whether there is unknown keys in #2 by storing in `\l_tmpa_seq` the list of the unknown keys.

```

2570     \seq_clear:N \l_tmpa_seq
2571     \keyval_parse:NNn \@@_valid_key:n \@@_valid_key:nn { #2 }
2572     \seq_if_empty:NTF \l_tmpa_seq
2573     {
2574         \seq_put_right:Nx \l_@@_options_WithArrows_seq
2575         { \tl_to_str:n { #1 } }
2576         \seq_put_right:Nx \l_@@_options_DispWithArrows_seq
2577         { \tl_to_str:n { #1 } }
2578         \seq_put_right:Nx \l_@@_options_WithArrowsOptions_seq
2579         { \tl_to_str:N { #1 } }

```

When we will consider that `\keys_precompile:nnN` (introduced in LaTeX on 2022-03-09) is widely available, we will delete that test and keep only the first version.

```

2580     \cs_if_exist:NTF \keys_precompile:nnN
2581     {
2582         \keys_precompile:nnN
2583         { WithArrows / WithArrowsOptions }
2584         { #2 }
2585         \l_tmpa_tl
2586         \@@_key_define:nV { #1 } \l_tmpa_tl
2587     }
2588     {
2589         \keys_define:nn { WithArrows / Global }
2590         {
2591             #1 .code:n =
2592             { \keys_set:nn { WithArrows / WithArrowsOptions } { #2 } }
2593         }
2594     }
2595 }
2596 { \@@_error:nn { Impossible~style } { #1 } }
2597 }
2598 }
2599 \@@_msg_new:nn { Impossible~style }
2600 {
2601     Impossible~style.\\
2602     It's~impossible~to~define~the~style~'#1'~
2603     because~it~contains~unknown~keys:~'
2604     \seq_use:Nnnn \l_tmpa_seq { '~and~' } { ',~' } { ',~and~'}.
2605 }
2606 \cs_new_protected:Npn \@@_valid_key:n #1
2607 {
2608     \keys_if_exist:nnF { WithArrows / Global } { #1 }
2609     { \seq_put_right:Nn \l_tmpa_seq { #1 } }
2610 }
2611 \cs_new_protected:Npn \@@_valid_key:nn #1 #2
2612 {
2613     \keys_if_exist:nnF { WithArrows / Global } { #1 }
2614     { \seq_put_right:Nn \l_tmpa_seq { #1 } }
2615 }
2616 \cs_new_protected:Npn \@@_key_define:nn #1 #2
2617 { \keys_define:nn { WithArrows / Global } { #1 .code:n = #2 } }
2618 \cs_generate_variant:Nn \@@_key_define:nn { n V }
2619 \@@_msg_new:nn { Key~already~defined }
2620 {
2621     Key~already~define.\\
2622     The~key~'#1'~is~already~defined. \\
2623     If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2624     will~be~ignored.
2625 }

```

12.15 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 the Tikz library `calc`. That's why we have decided not to load by default this library. If it is not loaded, the user will have an error only when using the option `up` or the option `down`.

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

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

```
2626 \keys_define:nn { WithArrows / up-and-down }
2627   {
2628     radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
2629     radius .value_required:n = true ,
2630     width .code:n =
2631       \str_case:nnF { #1 }
2632         {
2633           { min } { \dim_zero:N \l_@@_arrow_width_dim }
2634           { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
2635         }
2636         { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
2637     width .value_required:n = true ,
2638     unknown .code:n = \@@_error:n { Option-unknown~for~up-and-down }
2639   }

2640 \@@_msg_new:nn { Option-unknown~for~up-and-down }
2641   {
2642     Unknown~option.\@
2643     The~option~'\l_keys_key_str'~is~unknown.~\c_@@_option_ignored_str
2644   }
```

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

```
2645 (*LaTeX)
2646 \tl_const:Nn \c_@@_tikz_code_up_tl
2647   {
```

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

```
2648     \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2649       {
2650         \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2651           let \p1 = ( #1 ) , \p2 = ( #2 )
2652           in (\p1) -- node
2653             {
2654               \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2655               \begin { varwidth } \l_tmpa_dim
```

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

```
2656               \narrowragged
2657               #3
2658             \end { varwidth }
2659           }
2660         (\x2,\y1) -- (\p2) ;
2661       }
```

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`.

```

2662     {
2663     \path
2664     let \p1 = ( #1 ) , \p2 = ( #2 )
2665     in node
2666     {

```

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

```

2667         \dim_set:Nn \l_tmpa_dim
2668         { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
2669     \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
2670     {
2671         \dim_set:Nn \l_tmpa_dim
2672         { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
2673     }

```

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.

```

2674         \hbox_gset:Nn \g_tmpa_box
2675         {
2676         \begin { varwidth } \l_tmpa_dim
2677         \narrowragged
2678         #3
2679         \end { varwidth }
2680     }

```

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

```

2681         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2682         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2683         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2684     \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2685     } ;
2686 \draw
2687 let \p1 = ( #1 ) , \p2 = ( #2 )
2688 in (\x2-\g_tmpa_dim,\y1)
2689 -- node { \box_use:N \g_tmpa_box }
2690 (\x2-\l_@@_up_and_down_radius_dim,\y1)
2691 [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2692 -| (\p2) ;
2693 }
2694 }
2695 </LaTeX>
2696 <*plain-TeX>
2697 \tl_const:Nn \c_@@_tikz_code_up_tl
2698 {
2699     \dim_case:nnF \l_@@_arrow_width_dim
2700     {
2701         \c_max_dim
2702         {
2703             \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2704             let \p1 = ( #1 ) , \p2 = ( #2 )
2705             in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
2706         }
2707     \c_zero_dim
2708     {
2709         \path node
2710         {
2711             \hbox_gset:Nn \g_tmpa_box { #3 }
2712             \dim_gset:Nn \g_tmpa_dim
2713             { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2714         } ;

```

```

2715     \draw
2716     let \p1 = ( #1 ) , \p2 = ( #2 )
2717     in (\x2-\g_tmpa_dim,\y1)
2718     -- node { \box_use:N \g_tmpa_box }
2719     (\x2-\l_@@_up_and_down_radius_dim,\y1)
2720     [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2721     -| (\p2) ;
2722   }
2723 }
2724 {
2725   \draw
2726   let \p1 = ( #1 ) , \p2 = ( #2 )
2727   in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
2728   -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
2729   [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2730   -| (\p2) ;
2731 }
2732 }
2733 \end{plain-TeX}

```

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

```

2734 \begin{LaTeX}
2735 \tl_const:Nn \c_@@_tikz_code_down_tl
2736 {
2737   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2738   {
2739     \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2740     let \p1 = ( #1 ) , \p2 = ( #2 )
2741     in (\p1) -- (\x1,\y2) -- node
2742     {
2743       \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2744       \begin { varwidth } \l_tmpa_dim
2745         \narrowragged
2746         #3
2747       \end { varwidth }
2748     }
2749     (\p2) ;
2750   }
2751   {
2752     \path
2753     let \p1 = ( #1 ) , \p2 = ( #2 )
2754     in node
2755     {
2756       \hbox_gset:Nn \g_tmpa_box
2757       {
2758         \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.).

```

2759         { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
2760         \begin { varwidth } \l_tmpa_dim
2761         \narrowragged
2762         #3
2763       \end { varwidth }
2764     }
2765     \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2766     { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2767     { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2768     \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2769   } ;
2770
2771   \draw

```



```

2772     let \p1 = ( #1 ) , \p2 = ( #2 )
2773     in (\p1)
2774       { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2775       -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2776       -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2777       -- ++ (-2mm,0) ;
2778   }
2779 }
2780 </LaTeX>
2781 %
2782 <*plain-TeX>
2783 \tl_const:Nn \c_@@_tikz_code_down_tl
2784 {
2785   \dim_case:nnF \l_@@_arrow_width_dim
2786   {
2787     \c_max_dim
2788     {
2789       \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2790         let \p1 = ( #1 ) , \p2 = ( #2 )
2791         in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
2792     }
2793     \c_zero_dim
2794     {
2795       \path node
2796       {
2797         \hbox_gset:Nn \g_tmpa_box { #3 }
2798         \dim_gset:Nn \g_tmpa_dim
2799         { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2800       } ;
2801       \draw
2802         let \p1 = ( #1 ) , \p2 = ( #2 )
2803         in (\p1)
2804           { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2805           -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2806           -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2807           -- ++ (-2mm,0) ;
2808     }
2809   }
2810 }
2811 \draw
2812   let \p1 = ( #1 ) , \p2 = ( #2 )
2813   in (\p1)
2814     { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2815     -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2816     -- node { #3 }
2817       (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
2818     -- ++ (-2mm,0) ;
2819 }
2820 }
2821 </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.

```

2822 \keys_define:nn { WithArrows / Arrow / FirstPass }
2823 {
2824   up .code:n = \@@_set_independent_bis: ,
2825   down .code:n = \@@_set_independent_bis: ,
2826   up .default:n = NoValue ,
2827   down .default:n = NoValue
2828 }

```

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

```

2829 \keys_define:nn { WithArrows / Arrow / SecondPass }
2830 {
2831   up .code:n =
2832     \str_if_empty:NT \l_@@_previous_key_str
2833     {
2834       \str_set:Nn \l_@@_previous_key_str { up }
2835       \cs_if_exist:cTF { tikz@library@calc@loaded }
2836       {
2837         \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2838         \int_set:Nn \l_@@_pos_arrow_int 1

```

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

```

2839         \bool_set_false:N \l_@@_wrap_lines_bool

```

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

```

2840         \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
2841     }
2842     { \@@_error:n { calc-not~loaded } }
2843   } ,
2844   down .code:n =
2845     \str_if_empty:NT \l_@@_previous_key_str
2846     {
2847       \str_set:Nn \l_@@_previous_key_str { down }
2848       \cs_if_exist:cTF { tikz@library@calc@loaded }
2849       {
2850         \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2851         \int_set:Nn \l_@@_pos_arrow_int 1
2852         \bool_set_false:N \l_@@_wrap_lines_bool
2853         \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
2854       }
2855       { \@@_error:n { calc-not~loaded } }
2856     }
2857   }
2858 \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
2859 \seq_put_right:Nn \l_@@_options_Arrow_seq { up }
2860 \@@_msg_new:nn { calc-not~loaded }
2861 {
2862   calc-not~loaded.\\
2863   You~can't~use~the~option~'\l_keys_key_str'~because~you~don't~have~loaded~the~
2864   Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
2865   ~in~the~preamble~of~your~document. \\
2866   \c_@@_option_ignored_str
2867 }
2868 <*plain-TeX>
2869 \catcode ` \@ = 12
2870 \ExplSyntaxOff
2871 </plain-TeX>

```

13 History

Changes between 2.7 and 2.8

New key `right-overlap`

Changes between 2.6b and 2.7

Correction of a bug: when the key `wrap_lines` was in force, the content of the annotations was not “flush left” by default as it should be (but justified).

Changes between 2.6 and 2.6a (and 2.6b)

Replacement of `\hbox_unpack_clear:N` by `\hbox_unpack_drop:N` since `\hbox_unpack_clear:N` is now deprecated in L3.

Version 2.6d: correction of a bug (cf. question 628461 on TeX StackExchange).

Changes between 2.5 and 2.5.1

Correction of the erroneous programming of the nodes aliases.

Changes between 2.4 and 2.5

Arrows of type `o` which are *over* other arrows.

`witharrows` now requires and loads `varwidth`

Changes between 2.3 and 2.4

Correction of a bug with `{DispWithArrows}` : cf. question 535989 on TeX StackExchange.

Changes between 2.2 and 2.3

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

Changes between 2.1 and 2.2

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

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

Changes between 2.0 and 2.1

Option `max-length-of-arrow`.

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

Changes between 1.18 and 2.0

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

Changes between 1.17 and 1.18

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

Option `subequations`.

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

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

Changes between 1.16 and 1.17

Option `format`.

Changes between 1.15 and 1.16

Option `no-arrows`

The behaviour of `{DispWithArrows}` after an `\item` of a LaTeX list has been changed : no vertical is added. The previous behaviour can be restored with the option `standard-behaviour-with-items`.

A given name can no longer be used for two distinct environments. However, it's possible to deactivate this control with the option `allow-duplicate-names`.

Changes between 1.14 and 1.15

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

Tikz externalization is now deactivated in the environments of the extension `witharrows`.⁴²

Changes between 1.13 and 1.14

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

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

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

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

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

Changes between 1.12 and 1.13

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

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

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

Changes between 1.11 and 1.12

New command `\tagnextline`.

New option `tagged-lines`.

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

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

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

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

Changes between 1.10 and 1.11

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

Changes between 1.9 and 1.10

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

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

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

Changes between 1.8 and 1.9

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

⁴²Before 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 1.7 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 1.6 and 1.7

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

Changes between versions 1.5 and 1.6

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

Changes between versions 1.4 and 1.5

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

Changes between versions 1.3 and 1.4

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

Changes between versions 1.2 and 1.3

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

Changes between versions 1.1 and 1.2

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

Changes between versions 1.0 and 1.1

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

Contents

1	Options for the shape of the arrows	2
2	Numbers of columns	6
3	Precise positioning of the arrows	7
4	The option 'o' for individual arrows	9
5	The options 'up' and 'down' for individual arrows	10
6	Comparison with the environment <code>{aligned}</code>	11
7	Arrows in nested environments	14
8	Arrows from outside environments <code>{WithArrows}</code>	16

9	The environment <code>{DispWithArrows}</code>	17
9.1	The option <code><...></code> of <code>DispWithArrows</code>	22
10	Advanced features	23
10.1	Use with plain-TeX	23
10.2	The option <code>tikz-code</code> : how to change the shape of the arrows	23
10.3	The command <code>\WithArrowsNewStyle</code>	24
10.4	The key <code>right-overlap</code>	24
10.5	Vertical positioning of the arrows	25
10.6	Footnotes in the environments of <code>witharrows</code>	26
10.7	Option <code>no-arrows</code>	26
10.8	Note for the users of AUCTeX	27
10.9	Note for developpers	27
11	Examples	27
11.1	<code>\MoveEqLeft</code>	27
11.2	A command <code>\DoubleArrow</code>	28
11.3	Modifying the shape of the nodes	28
11.4	Examples with the option <code>tikz-code</code>	29
11.4.1	Example 1	29
11.4.2	Example 2	29
11.4.3	Example 3	30
11.5	Automatic numbered loop	31
12	Implementation	32
12.1	Declaration of the package and extensions loaded	32
12.2	The packages <code>footnote</code> and <code>footnotehyper</code>	33
12.3	The class option <code>leqno</code>	34
12.4	Some technical definitions	35
12.5	Variables	38
12.6	The definition of the options	40
12.7	The command <code>\Arrow</code>	49
12.8	The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code>	50
12.8.1	Code before the <code>\halign</code>	50
12.8.2	The construction of the preamble of the <code>\halign</code>	53
12.8.3	The environment <code>{WithArrows}</code>	56
12.8.4	After the construction of the <code>\halign</code>	57
12.8.5	The command of end of row	59
12.8.6	The environment <code>{DispWithArrows}</code>	62
12.9	The commands <code>\tag</code> , <code>\notag</code> , <code>\label</code> , <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code>	68
12.10	We draw the arrows	70
12.10.1	The command <code>update_x</code>	79
12.10.2	We draw the arrows of type <code>o</code>	79
12.11	The command <code>\Arrow</code> in code-after	82
12.12	The command <code>\MultiArrow</code> in code-after	84
12.13	The error messages of the package	86
12.14	The command <code>\WithArrowsNewStyle</code>	92
12.15	The options <code>up</code> and <code>down</code>	94
13	History	98