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

F. Pantigny
`fpantigny@wanadoo.fr`

March 4, 2021

Abstract

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

In this document, we describe the LaTeX extension **witharrows** (however, **witharrows** can also be used with plain-TeX: see p. 23). 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 arrows are drawn with Tikz and that's why several compilations may be necessary.

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

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

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

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

The environment `{WithArrows}` bears similarities with the environment `{aligned}` of amsmath (and `mathtools`). The extension **witharrows** also provides an environment `{DispWithArrows}` which is similar to the environment `{align}` of amsmath: cf. p. 17.

1 Options for the shape of the arrows

The command `\Arrow` has several options. These options can be put between square brackets, before, or after the mandatory argument.
The option `jump` gives the number[1] of rows the arrow must jump (the default value is, of course, 1).

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

---

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

$$A = \bigl((a+b)+1\bigr)^2$$
$$= (a+b)^2 + 2(a+b) + 1 \quad \Big\rangle \text{ we expand}$$
$$= a^2 + 2ab + b^2 + 2a + 2b + 1$$

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

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

$$A = \bigl((a+b)+1\bigr)^2$$
$$= (a+b)^2 + 2(a+b) + 1$$
$$= a^2 + 2ab + b^2 + 2a + 2b + 1$$

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.

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

$$A = \bigl((a+b)+1\bigr)^2$$
$$= (a+b)^2 + 2(a+b) + 1 \quad \Big\rangle \text{ with } \mathtt{xoffset=1cm}$$

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.

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

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

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

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

$$A = (a+1)^2$$
$$= a^2 + 2a + 1 \quad \Big\rangle \text{ we factorize}$$

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

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

$$A = (a+1)^2$$
$$\quad = a^2 + 2a + 1 \qquad \Big) \; \textit{very classical}$$

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

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

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

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

It's possible to use the Tikz option "`text width`" to control the width of the text associated to the arrow.[2]

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

$$A = \big((a+b)+1\big)^2$$
$$\quad = (a+b)^2 + 2(a+b) + 1 \qquad \Big) \; \textit{We have done a two-stages expansion}$$
$$\quad = a^2 + 2ab + b^2 + 2a + 2b + 1 \qquad \textit{but it would have been clever to expand with the multinomial theorem.}$$

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. .

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

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

$$A = (a+1)^2$$
$$\quad = a^2 + 2a + 1 \qquad \Big) \; \textit{\textbf{we expand}}$$

It's possible to put commands `\\` in the text to force new lines[3]. 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.

---

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

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

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

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

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

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

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

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

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

Without the option `displaystyle`:

```
$\begin{WithArrows}
\int_0^1 (x+1)^2 dx
& = \int_0^1 (x^2+2x+1) dx
\Arrow{linearity of integration}     \\
& = \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \\
& = \frac13 + 2\frac12 + 1 \\
& = \frac73
\end{WithArrows}$
```

$$\int_0^1 (x+1)^2 dx = \int_0^1 (x^2 + 2x + 1)dx$$
$$\quad = \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \quad \Big) \textit{linearity of integration}$$
$$\quad = \tfrac{1}{3} + 2\tfrac{1}{2} + 1$$
$$\quad = \tfrac{7}{3}$$

The same example with the option `displaystyle`:

$$\int_0^1 (x+1)^2 dx = \int_0^1 (x^2 + 2x + 1)dx$$
$$= \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx \quad \Big) \textit{linearity of integration}$$
$$= \frac{1}{3} + 2\frac{1}{2} + 1$$
$$= \frac{7}{3}$$

---

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

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

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

$$\sum_{i=1}^{n}(x_i+1)^2 = \sum_{i=1}^{n}(x_i^2 + 2x_i + 1)$$
$$= \sum_{i=1}^{n}x_i^2 + 2\sum_{i=1}^{n}x_i + n \quad \Big\rangle \textit{by linearity}$$

The command `\Arrow` is recognized only in the environments `{WithArrows}`. If we have a command `\Arrow` previously defined, it's possible to go on using it outside the environments `{WithArrows}`. However, a previouly 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}
$\begin{WithArrows}[command-name=Explanation]
f & = \bigl(x \Arrow (x+1)^2\bigr)
\Explanation{we work directly on fonctions}\\
& = \bigl(x \Arrow x^2+2x+1\bigr)
\end{WithArrows}$
```

$$f = \bigl(x \longmapsto (x+1)^2\bigr)$$
$$= \bigl(x \longmapsto x^2 + 2x + 1\bigr) \quad \Big\rangle \textit{we work directly on fonctions}$$

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

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

$$A = (a+b)^2$$
$$= a^2 + 2ab + b^2 \quad \Big\rangle \textit{we expand}$$

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.

---

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

2 Numbers of columns

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

New 2.6 The letters `R`, `C` and `L` add empty groups `{}` which provide correct spaces when these columns contain symbols with the type `\mathrel` (such as $=$, $\leq$, etc.) or `\mathbin` (such as $+$, $\times$, 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`.

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

$$f(x) \geq g(x)$$
$$f(x)^2 \geq g(x)^2 \qquad \rangle \text{ by squaring both sides}$$
$$f(x)^2 - g(x)^2 \geq 0 \qquad \rangle \text{ by moving to left side}$$

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

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

$$k \quad \leq \quad t \quad \leq \quad k+1$$
$$\frac{1}{k+1} \quad \leq \quad \frac{1}{t} \quad \leq \quad \frac{1}{k}$$
$$\int\limits_k^{k+1} \frac{dt}{k+1} \leq \quad \int\limits_k^{k+1} \frac{dt}{t} \quad \leq \int\limits_k^{k+1} \frac{dt}{k} \qquad \rangle \text{ we can integrate the inequalities since } k \leq k+1$$
$$\frac{1}{k+1} \quad \leq \ln(k+1) - \ln(k) \leq \quad \frac{1}{k}$$

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]

---

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

$$I = \int_{\frac{\pi}{4}}^{0} \ln\left(1 + \tan\left(\tfrac{\pi}{4} - u\right)\right)(-du)$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(1 + \tan\left(\tfrac{\pi}{4} - u\right)\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(1 + \frac{1 - \tan u}{1 + \tan u}\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(\frac{1 + \tan u + 1 - \tan u}{1 + \tan u}\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(\frac{2}{1 + \tan u}\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \left(\ln 2 - \ln(1 + \tan u)\right) du$$

$$= \frac{\pi}{4} \ln 2 - \int_{0}^{\frac{\pi}{4}} \ln(1 + \tan u)\, du$$

$$= \frac{\pi}{4} \ln 2 - I$$

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.

$$\text{Therefore } I = \int_{\frac{\pi}{4}}^{0} \ln\left(1 + \tan\left(\tfrac{\pi}{4} - u\right)\right)(-du)$$

This arrow uses the `lr` *option.*

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(1 + \tan\left(\tfrac{\pi}{4} - u\right)\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(1 + \frac{1 - \tan u}{1 + \tan u}\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(\frac{1 + \tan u + 1 - \tan u}{1 + \tan u}\right) du$$

$$= \int_{0}^{\frac{\pi}{4}} \ln\left(\frac{2}{1 + \tan u}\right) du$$

This arrow uses a `ll` *option and a* `jump` *equal to 2*

$$= \int_{0}^{\frac{\pi}{4}} \left(\ln 2 - \ln(1 + \tan u)\right) du$$

$$= \frac{\pi}{4} \ln 2 - \int_{0}^{\frac{\pi}{4}} \ln(1 + \tan u)\, du$$

$$= \frac{\pi}{4} \ln 2 - I$$

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

```
$\begin{WithArrows}
(a+b)(a+ib)(a-b)(a-ib)
& = (a+b)(a-b)\cdot(a+ib)(a-ib) \\
& = (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}$
```

$$(a+b)(a+ib)(a-b)(a-ib) = (a+b)(a-b)\cdot(a+ib)(a-ib)$$
$$= (a^2-b^2)(a^2+b^2)$$
$$= a^4 - b^4$$

$because\ (x-y)(x+y) = x^2 - y^2$

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.

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

$$2xy' - 3y = \sqrt{x} \Longleftrightarrow 2x(K'y_0 + Ky_0') - 3Ky_0 = \sqrt{x}$$
$$\Longleftrightarrow 2xK'y_0 + K(2xy_0' - 3y_0) = \sqrt{x}$$
$$\Longleftrightarrow 2xK'y_0 = \sqrt{x}$$
$$\Longleftrightarrow 2xK'x^{\frac{3}{2}} = x^{\frac{1}{2}}$$
$$\Longleftrightarrow K' = \frac{1}{2x^2}$$
$$\Longleftrightarrow K = -\frac{1}{2x}$$

we replace y_0 by its value

simplification of the x

antiderivation

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

$$A = B$$
$$= C + D$$
$$= D'$$
$$= E + F + G + H + I$$
$$= K + L + M$$
$$= N$$
$$= O$$

one

two

three

four

In an environment which uses the option `group` or the option `groups`, it's still possible to give an option of position (`ll`, `lr`, `rl`, `rr` or `i`) to an individual arrow[8]. Such arrow will be drawn irrespective of the groups. It's also possible to start a new group by applying the option `new-group` to an given arrow.

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

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

---

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

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

$$\sum_{k=0}^{n} \frac{\cos kx}{\cos^k x} = \sum_{k=0}^{n} \frac{\Re(e^{ikx})}{(\cos x)^k}$$

$$= \sum_{k=0}^{n} \Re\left(\frac{e^{ikx}}{(\cos x)^k}\right)$$

$$= \Re\left(\sum_{k=0}^{n} \left(\frac{e^{ix}}{\cos x}\right)^k\right)$$

$$= \Re\left(\frac{1-\left(\frac{e^{ix}}{\cos x}\right)^{n+1}}{1-\frac{e^{ix}}{\cos x}}\right)$$

$$= \Re\left(\frac{1-\frac{e^{i(n+1)x}}{\cos^{n+1} x}}{1-\frac{e^{ix}}{\cos x}}\right)$$

$$= \Re\left(\frac{\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos^{n+1} x}}{\frac{\cos x - e^{ix}}{\cos x}}\right)$$

$$= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - e^{i(n+1)x}}{\cos x - e^{ix}}\right)$$

$$= \frac{1}{\cos^n x} \Re\left(\frac{\cos^{n+1} x - (\cos(n+1)x + i\sin(n+1)x)}{\cos x - (\cos x + i\sin x)}\right)$$

$$= \frac{1}{\cos^n x} \Re\left(\frac{(\cos^{n+1} x - \cos(n+1)x) - i\sin(n+1)x}{-i\sin x}\right)$$

$$= \frac{1}{\cos^n x} \cdot \frac{\sin(n+1)x}{\sin x}$$

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

4 The 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.

$$A = B$$
$$= C$$
$$= D$$
$$= E$$

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]

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

$$A = B$$
$$= C + C \quad \text{one}$$
$$= D + D + D \quad \text{two} \quad \text{direct}$$
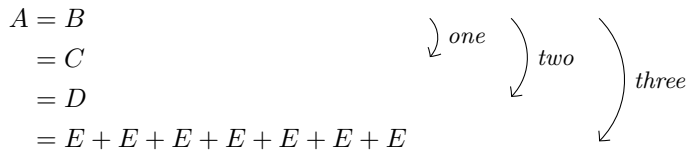$$= E + E \quad \text{three}$$
$$= F + F$$

---

[9] Among 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`.

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

$$A = B$$
$$= C$$
$$= D$$
$$= E + E + E + E + E + E + E$$

one *two* *three*

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:

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

$$A = B$$
an arrow of type `up`
$$= C + C + C + C + C + C + C + C$$
$$= C + C + C + C + C + C + C + C$$
$$= E + E$$
an arrow of type `down`
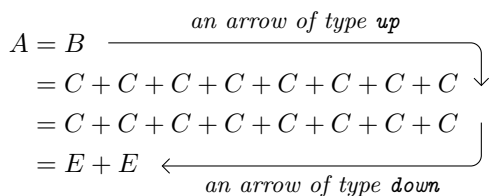
The options `up` and `down` require the Tikz library `calc`. It it has not been previously loaded by the user, an error will be raised.

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 ajusted 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);

---

[10]The 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.

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

$$A = B$$
$$\qquad \xrightarrow{\text{we try}}$$
$$= C + C + C + C + C + C + C + C \downarrow$$

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

$$A = B \qquad \xrightarrow{\text{we try}}$$
$$= C + C + C + C + C + C + C + C \downarrow$$

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.

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

---

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

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

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

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]

```
$\begin{WithArrows}[displaystyle,jot=2ex]
F & = \frac12G      \Arrow{we expand}\\
  & = H + \frac12K \Arrow{we go on}\\
  & = K
\end{WithArrows}$
```

$$F = \frac{1}{2}G$$

$$= H + \frac{1}{2}K \quad \text{we expand}$$

$$= K \quad \text{we go on}$$

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

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

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

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

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.

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

---

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

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

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

$x\ and\ y\ are\ real$

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

So $A = (a + 1)^2$
$\qquad = a^2 + 2a + 1$ $we\ expand$

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

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

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

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}:

$$\begin{cases} \displaystyle\sum_{i=1}^{n}(x_i + 1)^2 = \sum_{i=1}^{n}(x_i^2 + 2x_i + 1) \\ \displaystyle\qquad\qquad = \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n} x_i + n \end{cases}$$

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

$$\begin{cases} \displaystyle\sum_{i=1}^{n}(x_i + 1)^2 = \sum_{i=1}^{n}(x_i^2 + 2x_i + 1) \\ \displaystyle\qquad\qquad = \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n} x_i + n \end{cases}$$

---

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

7 Arrows in nested environments

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

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

$$\varphi(x, y) = 0 \Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0$$
$$\Leftrightarrow \begin{cases} x + 2y = 0 \\ 2x + 4y = 0 \end{cases} \quad \text{the numbers are real}$$
$$\Leftrightarrow \begin{cases} x + 2y = 0 \\ x + 2y = 0 \end{cases} \text{the same equation}$$
$$\Leftrightarrow x + 2y = 0$$

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 :

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

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}.

$$\varphi(x,y) = 0 \Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ 2x+4y = 0 \end{cases} \qquad \textit{environment number 1}$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ x+2y = 0 \end{cases} \qquad \textit{environment number 2}$$

$$\Leftrightarrow x+2y = 0$$

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}$
```

$$\varphi(x,y) = 0 \Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ 2x+4y = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ x+2y = 0 \end{cases} \;\; \textit{division by 2}$$

$$\Leftrightarrow x+2y = 0$$

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}$
```

$$\varphi(x,y) = 0 \Leftrightarrow (x+2y)^2 + (2x+4y)^2 = 0$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ 2x+4y = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} x+2y = 0 \\ x+2y = 0 \end{cases} \;\; \textit{division by 2}$$
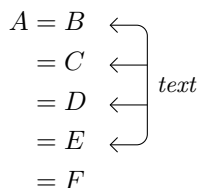
$$\Leftrightarrow x+2y = 0$$

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 \\
```

```
   & = D \\
   & = E \\
   & = F
\end{WithArrows}$
```

$$A = B \quad \leftarrow$$
$$\quad = C \quad \leftarrow$$
$$\quad = D \quad \leftarrow \quad text$$
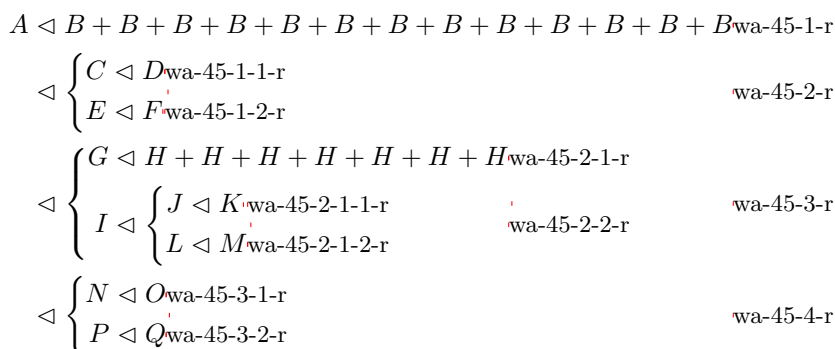$$\quad = E \quad \leftarrow$$
$$\quad = F$$

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

8   Arrows from outside environments {WithArrows}

If someone wants to draw arrows from outside the environments `{WithArrows}`, he can use the Tikz nodes created in the environments.

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

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

$$A \lhd B + B + B + B + B + B + B + B + B + B + B + B + B \quad \text{wa-45-1-r}$$
$$\lhd \begin{cases} C \lhd D \text{wa-45-1-1-r} \\ E \lhd F \text{wa-45-1-2-r} \end{cases} \qquad\qquad \text{wa-45-2-r}$$
$$\lhd \begin{cases} G \lhd H + H + H + H + H + H + H \text{wa-45-2-1-r} \\ I \lhd \begin{cases} J \lhd K \text{wa-45-2-1-1-r} \\ L \lhd M \text{wa-45-2-1-2-r} \end{cases} \quad \text{wa-45-2-2-r} \end{cases} \quad \text{wa-45-3-r}$$
$$\lhd \begin{cases} N \lhd O \text{wa-45-3-1-r} \\ P \lhd Q \text{wa-45-3-2-r} \end{cases} \qquad\qquad \text{wa-45-4-r}$$

The package `witharrows` provides some tools facilitating the use of these nodes:

- the command `\WithArrowsLastEnv` gives the number of the last environment of level 0 (*i.e.* which is not included in another environment of the package `witharrows`);

- a name can be given to a given environment with the option `name` and, in this case, the nodes created in the environment will have aliases constructed with this name;

- the Tikz style `WithArrows/arrow` is the style used by `witharrows` when drawing an arrow[15];

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

For example, we can draw an arrow from `wa-45-2-1-2-r.south` to `wa-45-3-2-r.north` with the following Tikz command.

---

[14]There is an option `show-node-names` to show the names of these nodes.

[15]More precisely, this style is given to the Tikz option "`every path`" before drawing the arrow with the code of the option `tikz-code`. This style is modified (in TeX scopes) by the option `tikz` of `witharrows`.

```
\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}
```

$$A \triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B$$

$$\triangleleft \begin{cases} C \triangleleft D \\ E \triangleleft F \end{cases}$$

$$\triangleleft \begin{cases} G \triangleleft H + H + H + H + H + H + H \\ I \triangleleft \begin{cases} J \triangleleft K \\ L \triangleleft M \end{cases} \end{cases}$$

$$\triangleleft \begin{cases} N \triangleleft O \\ P \triangleleft Q \end{cases}$$

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}
```

$$A = B$$
$$\quad = C$$

$$A' = B'$$
$$\quad = C'$$

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}
```

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

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

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

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

A link to the equation $(\star)$.[16]

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

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

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

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

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

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

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

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

---

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

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

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

$$A = A_1$$
$$= A_2 \quad \left.\right) \textit{first stage}$$
$$= A_3 \quad \left.\right) \textit{second stage} \tag{3}$$

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[19]). The initial value of this parameter is 25 pt.

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

$$A = (a+1)^2 \tag{4}$$
$$= a^2 + 2a + 1 \quad \left.\right) \textit{we expand} \tag{5}$$

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

If the option `leqno` is used as a class option, the labels will be composed on the left also for the environments {DispWithArrows} and {DispWithArrows*}.[20]

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.

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

---

[19]In LaTeX, `mathindent` is a dimension (`dim`) and not a glue (`skip`) but should become a skip in a future version of LaTeX. As of now, the parameter `mathindent` of witharrows is a dimension.

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

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

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

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

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

```
\begin{DispWithArrows*}[displaystyle,wrap-lines]
S_n
& = \frac1n \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 $e^{i\frac{2\pi}n}$}\\
& = \frac1n \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.} \\
& = \frac1n \Re \left(\frac{1-i}{1-e^{i\frac{\pi}{2n}}}\right)
\end{DispWithArrows*}
```

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

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.[21]

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
& = \frac1n \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 $e^{i\frac{2\pi}n}$}\\
& = \frac1n \Re \left( \frac{1-\bigl(e^{i\frac{\pi}{2n}}\bigr)^n}
                            {1-e^{i\frac{\pi}{2n}}}\right)
\Arrow{$\bigl(e^{i\frac{\pi}{2n}}\bigr)^n = e^{i\frac{\pi}2}=i$} \\
& = \frac1n \Re \left(\frac{1-i}{1-e^{i\frac{\pi}{2n}}}\right)
\end{DispWithArrows}
\end{enumerate}
```

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

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, listlbls, prettyref, refcheck, refstyle, showlabels, smartref, typedref and varioref, and with the options `showonlyrefs` and `showmanualtags` of mathtools.[22]

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

---

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

[22] We 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.[23]

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.[24]

```
\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}
```

$$
\binom{n}{p} = \begin{cases} 0 & \text{if } p > n \\ \dfrac{n(n-1)\cdots(n-p+1)}{p!} & \text{if } 0 \leq p \leq n \\ 0 & \text{if } p < 0 \end{cases}
$$

$$(14)$$
$$(15)$$
$$(16)$$

if fact, it's a special case of the following one

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

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

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

$$(17a)$$
$$(17b)$$
$$(17c)$$

3 equations

The whole system is the equation (17) (this reference has been coded by `\ref{system}`) whereas the last equation is the equation (17c) (this reference has been coded by `\ref{last-equation}`). The command `\ref*` used in the code above is provided by hyperref. It's a variant of `\ref` which doesn't create interactive link.

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

---

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

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

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.[25]

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}
```

---

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

$$3(2x + 4) = 6$$
$$2x + 4 = 2$$
$$2x = -2$$
$$x = -1$$

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. .

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. .

10.4   Vertical positioning of the arrows

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

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

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

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

$$(\cos x + \sin x)^2 = \cos^2 x + 2\cos x \sin x + \sin^2 x \qquad \text{ystart}$$
$$= \cos^2 x + \sin^2 x + 2\sin x \cos x \qquad \text{ygap}$$
$$= 1 + \sin(2x)$$

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

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

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

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

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

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

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

Since the version 2.1 of witharrows, an arrow of `jump` equal to 1 has a maximal length[26] equal to the parameter `max-length-of-arrow`. The initial value of this parameter is 2 cm.
In the following example, the value of `max-length-of-arrow` has been fixed to 1.5 cm.

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

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

---

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

10.5   Footnotes in the environments of witharrows

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

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

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

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

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

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

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

10.6   Option no-arrows

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

10.7   Note for the users of AUCTeX

In a editor of text with a LaTeX-oriented mode, the environments `{DispWithArrows}` and `{DispWithArrows*}` should be formated like the environment `equation` of LaTeX, that is to say with a formating 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]`

10.8   Note for developpers

If you want to construct an environment upon an environment of witharrows, we recommand 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:

---

^{27}A footnote.

```
\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 \frac45 + \arcsin \frac5{13}
\Arrow{because both are in $[-\frac{\pi}2,\frac{\pi}2]$} \\
& \Leftrightarrow x = \sin\left(\arcsin\frac45 + \arcsin\frac5{13}\right) \\
& \Leftrightarrow x = \frac45\cos\arcsin\frac5{13} + \frac5{13} \cos\arcsin\frac45
\Arrow{$\forall x \in [-1,1], \cos(\arcsin x) = \sqrt{1-x^2}$} \\
& \Leftrightarrow x = \frac45\sqrt{1-\bigl(\frac5{13}\bigr)^2}
+ \frac5{13}\sqrt{1-\bigl(\frac45\bigr)^2}
\end{WithArrows}$
```

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

11.2 A command \DoubleArrow

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

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

Example of use:

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

$$
\begin{aligned}
A &= (a+b)^2 \\
&= a^2 + 2ab + b^2
\end{aligned}
\qquad \text{\textbf{expansion}} \qquad \text{\textbf{factorization}}
$$

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}
```
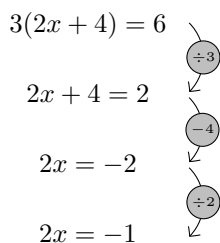

11.4   Examples with the option tikz-code

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

The value by defaut 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
& = \frac1n \sum_{k=0}^{n-1}\cos\bigl(\tfrac{\pi}2\cdot\tfrac kn\bigr)
...........
```

---

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

$$S_n = \frac{1}{n}\sum_{k=0}^{n-1}\cos\Bigl(\tfrac{\pi}{2}\cdot\tfrac{k}{n}\Bigr)$$

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

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

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

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

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

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

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

$\exp$ *is a morphism for* $\times$ *and* $+$

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

11.4.2   Example 2

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

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

```
begin{DispWithArrows}[MyStyle]
  S_n
  & = \frac1n \sum_{k=0}^{n-1}\cos\bigl(\tfrac{\pi}2\cdot\tfrac kn\bigr)
  \Arrow{$\cos x = \Re(e^{ix})$}\\
...........
```

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

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

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

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

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

$$\exp \text{ is a morphism for } \times \text{ and } +$$

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

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

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

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

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

$$E \iff \frac{(x+4)}{3} + \frac{5x+3}{5} = 7$$

$$\iff 5(x+4) + 3(5x+3) = 105$$

$$\iff 5x + 20 + 15x + 9 = 105$$

$$\iff 20x + 29 = 105$$

$$\iff 20x = 76$$

$$\iff x = \frac{38}{10}$$

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 xparse) and `\foreach` of pgffor (both packages are loaded when witharrows is loaded).

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

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

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

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

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

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

a. f est continuous on E
b. f est continuous in 0
c. f is bounded on the unit sphere
d. $\exists K > 0 \quad \forall x \in E \quad \|f(x)\| \le K\|x\|$
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.[29]

```
 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 `expl3`:

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

We use `\seq_set_map_x:NNn` which has been introduced the 2020/07/16.

```
16 \msg_new:nnn { witharrows } { expl3~too~old }
17   {
18     Your~version~of~LaTeX~(especially~expl3)~is~too~old.~
19     You~can~go~on~but~you~will~probably~have~other~errors~
20     if~you~use~the~functionalities~of~witharrows.
21   }
22 \cs_if_exist:NF \seq_set_map_x:NNn
23   { \msg_error:nn { witharrows } { expl3~too~old } }
```

The package `xparse` is still loaded for use on Overleaf.

```
24 \RequirePackage { xparse }
25 \RequirePackage { varwidth }
26 ⟨/LaTeX⟩
27 ⟨*plain-TeX⟩
28 \ExplSyntaxOn
29 \catcode `\@ = 11
30 ⟨/plain-TeX⟩
```

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 2.6c), 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.

```
31 ⟨*LaTeX⟩
32 \bool_new:N \c_@@_footnotehyper_bool
```

---

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

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

```
33 \bool_new:N \c_@@_footnote_bool
34 ⟨/LaTeX⟩
```

```
35 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
36 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { witharrows } }
37 \cs_new_protected:Npn \@@_msg_redirect_name:nn
38   { \msg_redirect_name:nnn { witharrows } }
39 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
40 \cs_new_protected:Npn \@@_warning:n { \msg_warning:nn { witharrows } }
41 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { witharrows } }
42 \cs_new_protected:Npn \@@_error:nn  { \msg_error:nnn { witharrows } }
43 \cs_generate_variant:Nn \@@_error:nn { n x }
```

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

```
44 ⟨*LaTeX⟩
45 \keys_define:nn { WithArrows / package }
46   {
47     footnote .bool_set:N = \c_@@_footnote_bool ,
48     footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
49     unknown .code:n =
50       \@@_fatal:n { Option~unknown~for~package }
51   }
52 \@@_msg_new:nn { Option~unknown~for~package }
53   {
54     You~can't~use~the~option~'\l_keys_key_str'~when~loading~the~
55     package~witharrows.~Try~to~use~the~command~
56     \token_to_str:N\WithArrowsOptions.
57   }
```

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

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

```
59 \@@_msg_new:nn { footnote~with~footnotehyper~package }
60   {
61     You~can't~use~the~option~'footnote'~because~the~package~
62     footnotehyper~has~already~been~loaded.~
63     If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
64     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
65     of~the~package~footnotehyper.\\
66     If~you~go~on,~the~package~footnote~won't~be~loaded.
67   }
68 \@@_msg_new:nn { footnotehyper~with~footnote~package }
69   {
70     You~can't~use~the~option~'footnotehyper'~because~the~package~
71     footnote~has~already~been~loaded.~
72     If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
73     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
74     of~the~package~footnote.\\
75     If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
76   }
```

```
77 \bool_if:NT \c_@@_footnote_bool
78   {
```

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

```
79     \@ifclassloaded { beamer }
80       { \bool_set_false:N \c_@@_footnote_bool }
```

```
81        {
82          \@ifpackageloaded { footnotehyper }
83            { \@@_error:n { footnote~with~footnotehyper~package } }
84            { \usepackage { footnote } }
85        }
86    }
87  \bool_if:NT \c_@@_footnotehyper_bool
88    {
```

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

```
89      \@ifclassloaded { beamer }
90        { \bool_set_false:N \c_@@_footnote_bool }
91        {
92          \@ifpackageloaded { footnote }
93            { \@@_error:n { footnotehyper~with~footnote~package } }
94            { \usepackage { footnotehyper } }
95          \bool_set_true:N \c_@@_footnote_bool
96        }
97    }
```

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 programmation, `leqno` *can't* be given as an option of `witharrows` (by design).

```
 98  \bool_new:N \c_@@_leqno_bool
 99  \DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
100  \DeclareOption* { }
101  \ProcessOptions*
102  ⟨/LaTeX⟩
```

12.4   Some technical definitions

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

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

```
104  \AtBeginDocument
105    {
106      \clist_map_inline:nn
107        {
108          amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
109          typedref, unicode-math
110        }
111        {
112          \bool_new:c { c_@@_#1_loaded_bool }
113  ⟨*LaTeX⟩
```

---

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

34

```
114          \@ifpackageloaded { #1 }
115            { \bool_set_true:c { c_@@_#1_loaded_bool } }
116            { }
117  ⟨/LaTeX⟩
118  ⟨*plain-TeX⟩
119            \bool_set_false:c { c_@@_#1_loaded_bool }
120  ⟨/plain-TeX⟩
121          }
122    }
```

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

```
123  \cs_new_protected:Npn \@@_sort_seq:N #1
124    {
125      \seq_sort:Nn #1
126        {
127          \int_compare:nNnTF
128            {
129              \tex_strcmp:D
130                { \str_lower_case:n { ##1 } }
131                { \str_lower_case:n { ##2 } }
132            }
133            > 0
134            \sort_return_swapped:
135            \sort_return_same:
136        }
137    }
```

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

```
138  \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
139    {
140      \seq_set_from_clist:Nn #1 { #2 }
141      \seq_set_map_x:NNn #1 #1 { \tl_to_str:n { ##1 } }
142    }
```

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

```
143  \cs_new_protected:Npn \@@_save:N #1
144    {
145      \seq_set_split:Nxx \l_tmpa_seq
146        { \char_generate:nn { `_ } { 12 } }
147        { \cs_to_str:N #1 }
148      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
```

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

```
149      \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
150      \use:c { \l_tmpa_str _if_exist:cF }
151        { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
152        {
153          \use:c { \l_tmpa_str _new:c }
154            { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
155        }
156      \use:c { \l_tmpa_str _gset_eq:cN }
157        { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
158    }
```

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

```
159  \cs_new_protected:Npn \@@_restore:N #1
160    {
161      \seq_set_split:Nxx \l_tmpa_seq
162        { \char_generate:nn { `_ } { 12 } }
```

```
163        { \cs_to_str:N #1 }
164      \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
165      \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
166      \use:c { \l_tmpa_str _set_eq:Nc }
167        #1 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ }
168    }
```

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.

```
169  \tikzset
170    {
171      @@_node_style / .style =
172        {
173          above = \l_@@_ystart_dim ,
174          inner~sep = \c_zero_dim ,
175          minimum~width = \c_zero_dim ,
176          minimum~height = \l_@@_ygap_dim
177        }
178    }
```

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

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

```
179  \tikzset
180    {
181      @@_standard / .style =
182        {
183          remember~picture ,
184          overlay ,
185          name~prefix = wa - \l_@@_prefix_str -
186        }
187    }
```

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.

```
188  \tikzset
189    {
190      WithArrows / arrow / tips / .style =
191        { > = { Straight~Barb [ scale = 1.2 , bend ] } }
192    }
```

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.

```
193  \tikzset
194    {
195      WithArrows / arrow / .style  =
196        {
197          align = left ,
```

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

```
198          auto = left ,
199  ⟨*LaTeX⟩
```

---

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

```
200        font = \small \itshape ,
201 ⟨/LaTeX⟩
202        WithArrows / arrow / tips ,
203        bend~left = 45 ,
204        ->
205      }
206   }
```

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}`.

```
207 ⟨*LaTeX⟩
208 \AtBeginDocument
209   {
210     \bool_if:NTF \c_@@_amsmath_loaded_bool
211       {
212         \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
213         \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
214       }
```

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

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

```
215      {
216 ⟨/LaTeX⟩
217        \cs_new_protected:Npn \spread@equation
218          {
219            \openup \jot
220            \cs_set_eq:NN \spread@equation \prg_do_nothing:
221          }
222 ⟨*LaTeX⟩
223      }
224   }
225 ⟨/LaTeX⟩
```

```
226 \tl_new:N \l_@@_left_brace_tl
227 \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`).

```
228 \bool_new:N \l_@@_in_WithArrows_bool
229 \bool_new:N \l_@@_in_DispWithArrows_bool
230 \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}`.

```
231 \seq_new:N \g_@@_position_in_the_tree_seq
232 \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`.

```
233 \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).

234 `\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
`\l_@@_pos_arrow_int`	0	1	2	3	4	5	6	7

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

235 `\int_new:N \l_@@_pos_arrow_int`
236 `\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.

237 `\seq_new:N \g_@@_arrow_int_seq`
238 `\int_new:N \g_@@_arrow_int`
239 `\seq_new:N \g_@@_line_int_seq`
240 `\int_new:N \g_@@_line_int`
241 `\seq_new:N \g_@@_col_int_seq`
242 `\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 programmation is to try to detect some use of `\omit` (which should be forbidden) in the cells of the `\halign`.

243 `\seq_new:N \g_@@_static_col_int_seq`
244 `\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`.

245 ⟨*LaTeX⟩
246 `\clist_new:N \l_@@_tags_clist`
247 `\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`.

248 `\cs_new_protected:Npn \@@_test_if_to_tag:`
249   `{`
250   `\clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int`
251   `{ \clist_set:Nn \l_@@_tags_clist { all } }`
252   `}`
253 ⟨/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`.

```
254 \str_new:N \l_@@_command_name_str
255 \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`".

```
256 \str_new:N \l_@@_string_Arrow_for_msg_str
257 \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`.

```
258 \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.

```
259 ⟨*LaTeX⟩
260 \bool_new:N \l_@@_sbwi_bool
261 ⟨/LaTeX⟩
```

```
262 ⟨*LaTeX⟩
263 \bool_new:N \l_@@_tag_star_bool
264 \bool_new:N \l_@@_tag_next_line_bool
265 \bool_new:N \l_@@_qedhere_bool
266 ⟨/LaTeX⟩
267 \bool_new:N \l_@@_in_first_columns_bool
268 \bool_new:N \l_@@_new_group_bool
269 \bool_new:N \l_@@_initial_r_bool
270 \bool_new:N \l_@@_final_r_bool
271 \tl_new:N \l_@@_initial_tl
272 \tl_new:N \l_@@_final_tl
273 \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}`).

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

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

```
275 ⟨*LaTeX⟩
276 \bool_new:N \l_@@_subequations_bool
277 ⟨/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 ajusted to the content of the node.

```
278 \dim_new:N \l_@@_arrow_width_dim
279 \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`.

```
280 \dim_new:N \l_@@_up_and_down_radius_dim
281 \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).

```
282 \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*).

```
283 \dim_new:N \l_@@_xoffset_for_o_arrows_dim
284 \dim_set:Nn \l_@@_xoffset_for_o_arrows_dim { 2 mm }
```

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* level[32];

- with `\begin{WithArrows}[...]`: this level will be called *environment* level;

- with `\Arrow[...]` (included in `code-after`): this level will be called *local* level.

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

```
285 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
286   {
287     \str_if_empty:NTF \l_@@_previous_key_str
288       {
289         \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
290         #1
291       }
292     { \@@_error:n { Incompatible~options } }
293   }
294 \cs_new_protected:Npn \@@_fix_pos_option:n #1
295   { \@@_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.

```
296 \keys_define:nn { WithArrows / Global }
297   {
298     max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
299     max-length-of-arrow .value_required:n = true ,
300     max-length-of-arrow .initial:n = 2 cm ,
301     ygap .dim_set:N = \l_@@_ygap_dim ,
302     ygap .initial:n = 0.4 ex ,
303     ygap .value_required:n = true ,
304     ystart .dim_set:N = \l_@@_ystart_dim ,
305     ystart .value_required:n = true ,
306     ystart .initial:n = 0.4 ex ,
307     more-columns .code:n =
```

---

[32]This level is called *global level* but the settings done by `\WithArrowsOptions` are local in the TeX sense: their scope corresponds to the current TeX group.

```
308        \@@_msg_redirect_name:nn { Too~much~columns~in~WithArrows } { none } ,
309      more-columns .value_forbidden:n = true ,
310      command-name .code:n =
311        \str_set:Nn \l_@@_command_name_str { #1 }
312        \str_set:Nx \l_@@_string_Arrow_for_msg_str
313          { \c_backslash_str Arrow~alias~\c_backslash_str #1 } ,
314      command-name .value_required:n = true ,
315      tikz-code .tl_set:N = \l_@@_tikz_code_tl,
316      tikz-code .initial:n = \draw~(#1)~to~node{#3}~(#2)~; ,
317      tikz-code .value_required:n = true ,
318      displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
319      displaystyle .default:n = true ,
320      show-nodes .code:n =
321        \tikzset { @@_node_style / .append~style = { draw , red } } ,
322      show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
323      show-node-names .default:n = true ,
324      group .code:n =
325        \str_if_empty:NTF \l_@@_previous_key_str
326          {
327            \str_set:Nn \l_@@_previous_key_str { group }
328            \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
329            \int_set:Nn \l_@@_pos_arrow_int 7
330          }
331          { \@@_error:n { Incompatible~options } } ,
332      group  .value_forbidden:n = true ,
333      groups .code:n =
334        \str_if_empty:NTF \l_@@_previous_key_str
335          {
336            \str_set:Nn \l_@@_previous_key_str { groups }
337            \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
338              { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
339            \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
340            \int_set:Nn \l_@@_pos_arrow_int 6
341          }
342          { \@@_error:n { Incompatible~options } } ,
343      groups .value_forbidden:n = true ,
344      tikz   .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
345      tikz   .initial:n       = \c_empty_tl ,
346      tikz   .value_required:n = true ,
347      rr     .code:n          = \@@_fix_pos_option:n 3 ,
348      rr     .value_forbidden:n = true ,
349      ll     .code:n          = \@@_fix_pos_option:n 1 ,
350      ll     .value_forbidden:n = true ,
351      rl     .code:n          = \@@_fix_pos_option:n 2 ,
352      rl     .value_forbidden:n = true ,
353      lr     .code:n          = \@@_fix_pos_option:n 0 ,
354      lr     .value_forbidden:n = true ,
355      i      .code:n          = \@@_fix_pos_option:n 5 ,
356      i      .value_forbidden:n = true ,
357      xoffset .dim_set:N  = \l_@@_xoffset_dim ,
358      xoffset .value_required:n  = true ,
359      xoffset .initial:n = 3 mm ,
360      jot .dim_set:N = \jot ,
361      jot .value_required:n = true ,
362      interline .skip_set:N = \l_@@_interline_skip ,
363      start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
364      start-adjust .initial:n = 0.4 ex ,
365      start-adjust .value_required:n = true ,
366      end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
367      end-adjust .initial:n = 0.4 ex ,
368      end-adjust .value_required:n = true ,
369      adjust .meta:n  = { start-adjust = #1 , end-adjust = #1 } ,
370      adjust .value_required:n = true ,
```

```
371    up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,
372    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`".

```
373    no-arrows .code:n =
374      \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn
375      \cs_set_eq:NN \@@_draw_arrow:nnn \use_none:nnn    ,
376    no-arrows .value_forbidden:n = true
377  }
```

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`.

```
378 \keys_define:nn { WithArrows / WithArrowsSpecific }
379   {
380     t   .code:n            = \int_set:Nn \l_@@_pos_env_int 0 ,
381     t   .value_forbidden:n = true ,
382     c   .code:n            = \int_set:Nn \l_@@_pos_env_int 1 ,
383     c   .value_forbidden:n = true ,
384     b   .code:n            = \int_set:Nn \l_@@_pos_env_int 2 ,
385     b   .value_forbidden:n = true
386   }
```

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

```
387 \clist_new:N \c_@@_extensible_delimiters_clist
388 \clist_set:Nn \c_@@_extensible_delimiters_clist
389   {
390     ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
391   }
392 ⟨*LaTeX⟩
393 \AtBeginDocument
394   {
395     \bool_lazy_or:nnT
396       \c_@@_amsmath_loaded_bool
397       { \use:c { c_@@_unicode-math_loaded_bool } }
398       {
399         \clist_put_right:Nn \c_@@_extensible_delimiters_clist { \lvert, \lVert }
400       }
401   }
402 ⟨/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`.

```
403 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
404   {
405     fleqn .bool_set:N = \l_@@_fleqn_bool ,
406     fleqn .default:n = true ,
407     mathindent .dim_set:N = \l_@@_mathindent_dim ,
408     mathindent .initial:n = 25 pt ,
409     mathindent .value_required:n = true ,
410 ⟨*LaTeX⟩
411     notag .code:n =
412       \str_if_eq:nnTF { #1 } { true }
413         { \clist_clear:N \l_@@_tags_clist }
414         { \clist_set:Nn \l_@@_tags_clist { all } } ,
415     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.

```
416    subequations .code:n =
417      \bool_if:NTF \c_@@_amsmath_loaded_bool
418        { \bool_set_true:N \l_@@_subequations_bool }
419        {
420          \@@_error:n { amsmath~not~loaded }
421          \group_begin:
422          \globaldefs = 1
423          \@@_msg_redirect_name:nn { amsmath~not~loaded } { info }
424          \group_end:
425        } ,
426    subequations .default:n = true ,
427    subequations .value_forbidden:n = true ,
428    nonumber .meta:n = notag ,
429    allow-multiple-labels .code:n =
430      \@@_msg_redirect_name:nn { Multiple~labels } { none } ,
431    allow-multiple-labels .value_forbidden:n = true ,
432    tagged-lines .code:n =
433      \clist_set:Nn \l_@@_tags_clist { #1 }
434      \clist_if_in:NnT \l_@@_tags_clist { first }
435        {
436          \clist_remove_all:Nn \l_@@_tags_clist { first }
437          \clist_put_left:Nn \l_@@_tags_clist 1
438        } ,
439    tagged-lines .value_required:n = true ,
```
440 ⟨/LaTeX⟩
```
441    wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
442    wrap-lines .default:n = true ,
443    replace-left-brace-by .code:n =
444      {
445        \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
446        \clist_if_in:NVTF
447          \c_@@_extensible_delimiters_clist
448          \l_tmpa_tl
449          { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }
450          { \@@_error:n { Bad~value~for~replace~brace~by } }
451      } ,
452    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`).

453 ⟨*LaTeX⟩
```
454    standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
455    standard-behaviour-with-items .default:n = true
```
456 ⟨/LaTeX⟩
```
457  }
```

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

```
458  \keys_define:nn { WithArrows / Env }
459    {
460      name .code:n =
```
First, we convert the value in a `str` because the list of the names will be a list of `str`.

```
461        \str_set:Nn \l_tmpa_str { #1 }
462        \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
463          { \@@_error:n { Duplicate~name } }
464          { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
465        \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
466      name .value_required:n = true ,
467      code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
```

```
468  code-before .value_required:n = true,
469  CodeBefore .meta:n = { code-before = #1 } ,
470  code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
471  code-after .value_required:n = true ,
472  CodeAfter .meta:n = { code-after = #1 } ,
473  format .code:n =
474    \tl_if_empty:nTF { #1 }
475      { \@@_error:n { Invalid~option~format } }
476      {
477        \regex_match:nnTF { \A[rclRCL]*\Z } { #1 }
478          { \tl_set:Nn \l_@@_format_str { #1 } }
479          { \@@_error:n { Invalid~option~format } }
480      } ,
481  format .value_required:n = true
482  }
```

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.

```
483  \keys_define:nn { WithArrows }
484    {
485      WithArrows .inherit:n =
486        {
487          WithArrows / Global ,
488          WithArrows / WithArrowsSpecific ,
489          WithArrows / Env
490        } ,
491      WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
492      DispWithArrows .inherit:n =
493        {
494          WithArrows / DispWithArrowsSpecific ,
495          WithArrows / Global ,
496          WithArrows / Env ,
497        } ,
498      DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
499      WithArrowsOptions .inherit:n =
500        {
501          WithArrows / Global ,
502          WithArrows / WithArrowsSpecific ,
503          WithArrows / DispWithArrowsSpecific ,
504        } ,
505      WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
506    }
```

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

```
507  \seq_new:N \l_@@_options_WithArrows_seq
508  \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrows_seq
509    {
510      adjust, b, c, code-after, code-before, command-name,
511      displaystyle, end-adjust,
512      format, group, groups, i,
513      interline, jot, ll,
514      lr, max-length-of-arrow, more-columns, name,
515      no-arrows, rl, rr, up-and-down,
516      show-node-names, show-nodes, start-adjust,
517      t, tikz, tikz-code,
518      xoffset, ygap, ystart
519    }
```

```
520  \keys_define:nn { WithArrows / WithArrows }
521    {
```

```
522    unknown .code:n  =
523      \@@_sort_seq:N \l_@@_options_WithArrows_seq
524      \@@_error:n { Unknown~option~WithArrows }
525    }


526 \keys_define:nn { WithArrows / DispWithArrows }
527   {
528    left-brace .tl_set:N = \l_@@_left_brace_tl ,
529    unknown .code:n  =
530      \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
531      \@@_error:n { Unknown~option~DispWithArrows } ,
532    }
```

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

```
533 \seq_new:N \l_@@_options_DispWithArrows_seq
534 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
535   {
536    code-after, code-before, command-name, tikz-code, adjust,
537    displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
538    left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
539    up-and-down, replace-left-brace-by, rl, rr, show-node-names,
540    show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
541 ⟨*LaTeX⟩
542    allow-multiple-labels, tagged-lines, nonumber, notag
543 ⟨/LaTeX⟩
544   }
545 \keys_define:nn { WithArrows / WithArrowsOptions }
546   {
547    allow-duplicate-names .code:n =
548      \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
549    allow-duplicate-names .value_forbidden:n = true ,
550    xoffset-for-o-arrows .dim_set:N = \l_@@_xoffset_for_o_arrows_dim ,
551    xoffset-for-o-arrows .value_required:n = true ,
552    unknown .code:n  =
553      \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
554      \@@_error:n { Unknown~option~WithArrowsOptions }
555    }
```

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

```
556 \seq_new:N \l_@@_options_WithArrowsOptions_seq
557 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrowsOptions_seq
558   {
559    allow-duplicate-names, b, c, command-name, more-columns, tikz-code, adjust,
560    displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
561    mathindent, max-length-of-arrow, no-arrows, up-and-down, rl, rr,
562    show-node-names, show-nodes, start-adjust, t, tikz, wrap-lines, xoffset,
563    xoffset-for-o-arrows, ygap, ystart,
564 ⟨*LaTeX⟩
565    allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
566    tagged-lines
567 ⟨/LaTeX⟩
568   }
```

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".

```
569 \cs_new_protected:Npn \@@_set_independent:
570   {
571     \str_if_eq:VnF \l_keys_value_tl { NoValue }
572       { \@@_error:n { Value~for~a~key } }
573     \@@_set_independent_bis:
574   }
```

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

```
575 \cs_new_protected:Npn \@@_set_independent_bis:
576   {
577     \str_if_empty:NTF \l_@@_previous_key_str
578       {
579         \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
580         \str_set:Nn \l_@@_status_arrow_str { independent }
581       }
582       { \@@_error:n { Incompatible~options~in~Arrow } }
583   }
```

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".

```
584 \keys_define:nn { WithArrows / Arrow / FirstPass }
585   {
586     jump .code:n =
587       \int_compare:nTF { #1 > 0 }
588         { \int_set:Nn \l_@@_jump_int { #1 } }
589         { \@@_error:n { Negative~jump } } ,
590     jump .value_required:n  = true,
591     rr .code:n = \@@_set_independent: ,
592     ll .code:n = \@@_set_independent: ,
593     rl .code:n = \@@_set_independent: ,
594     lr .code:n = \@@_set_independent: ,
595     i  .code:n = \@@_set_independent: ,
596     rr .default:n = NoValue ,
597     ll .default:n = NoValue ,
598     rl .default:n = NoValue ,
599     lr .default:n = NoValue ,
600     i  .default:n = NoValue ,
601     new-group .value_forbidden:n = true ,
602     new-group .code:n =
603       \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
604         { \str_set:Nn \l_@@_status_arrow_str { new-group } }
605         { \@@_error:n { new-group~without~groups } } ,
606     o .code:n =
607       \str_if_empty:NTF \l_@@_previous_key_str
608         {
609           \int_compare:nNnTF \l_@@_pos_arrow_int < 6
610             { \@@_error:n { invalid~key~o } }
611             {
612               \str_set:Nn \l_@@_status_arrow_str { over }
613               \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
614             }
615         }
616         { \@@_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.

```
617    tikz-code .code:n = \prg_do_nothing: ,
618    tikz-code .value_required:n = true ,
619    tikz .code:n = \prg_do_nothing: ,
620    tikz .value_required:n = true ,
621    start-adjust .code:n = \prg_do_nothing: ,
622    start-adjust .value_required:n = true ,
623    end-adjust .code:n = \prg_do_nothing: ,
624    end-adjust .value_required:n = true ,
625    adjust .code:n  = \prg_do_nothing: ,
626    adjust .value_required:n = true ,
627    xoffset .code:n = ,
628    unknown .code:n =
629      \@@_sort_seq:N \l_@@_options_Arrow_seq
630      \seq_if_in:NVTF \l_@@_options_WithArrows_seq \l_keys_key_str
631        {
632          \str_set:Nn \l_tmpa_str
633            { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
634        }
635        { \str_clear:N \l_tmpa_str }
636      \@@_error:n { Unknown~option~in~Arrow }
637    }
```

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

```
638 \seq_new:N \l_@@_options_Arrow_seq
639 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_seq
640   {
641     adjust, end-adjust, i, jump, ll, lr, o , rl, rr, start-adjust, tikz,
642     tikz-code, xoffset
643   }
```

```
644 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1
645   {
646     \str_if_empty:NT \l_@@_previous_key_str
647       {
648         \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
649         \int_set:Nn \l_@@_pos_arrow_int { #1 }
650       }
651   }
```

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.

```
652 \keys_define:nn {WithArrows / Arrow / SecondPass }
653   {
654     tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
655     tikz-code .initial:n = \draw~(#1)~to~node{#3}~(#2)~; ,
656     tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
657     tikz .initial:n = \c_empty_tl ,
658     rr .code:n = \@@_fix_pos_arrow:n 3 ,
659     ll .code:n = \@@_fix_pos_arrow:n 1 ,
660     rl .code:n = \@@_fix_pos_arrow:n 2 ,
661     lr .code:n = \@@_fix_pos_arrow:n 0 ,
662     i  .code:n = \@@_fix_pos_arrow:n 5 ,
663     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.

```
664    xoffset .code:n  =
665      \bool_if:nTF
666        {
```

```
667        \int_compare_p:nNn \g_@@_arrow_int > 1
668        &&
669        \int_compare_p:nNn \l_@@_pos_arrow_int > 5
670        &&
671        ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
672      }
673      { \@@_error:n { Option~xoffset~forbidden } }
674      { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
675    xoffset .value_required:n = true ,
676    start-adjust .dim_set:N = \l_@@_start_adjust_dim,
677    end-adjust .dim_set:N = \l_@@_end_adjust_dim,
678    adjust .code:n  =
679      \dim_set:Nn \l_@@_start_adjust_dim { #1 }
680      \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
681  }
```

`\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`.

```
682 ⟨*LaTeX⟩
683 \NewDocumentCommand \WithArrowsOptions { m }
684 ⟨/LaTeX⟩
685 ⟨*plain-TeX⟩
686 \cs_set_protected:Npn \WithArrowsOptions #1
687 ⟨/plain-TeX⟩
688   {
689     \str_clear_new:N \l_@@_previous_key_str
690     \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
691   }
```

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}.

```
692 ⟨*LaTeX⟩
693 \NewDocumentCommand \@@_Arrow { O { } m ! O { } }
694 ⟨/LaTeX⟩
695 ⟨*plain-TeX⟩
696 \cs_new_protected:Npn \@@_Arrow
697   {
698     \peek_meaning:NTF [
699       { \@@_Arrow_i }
700       { \@@_Arrow_i [ ] }
701   }
702 \cs_new_protected:Npn \@@_Arrow_i [ #1 ] #2
703   {
704     \peek_meaning:NTF [
705       { \@@_Arrow_ii [ #1 ] { #2 } }
706       { \@@_Arrow_ii [ #1 ] { #2 } [ ] }
707   }
708 \cs_new_protected:Npn \@@_Arrow_ii [ #1 ] #2 [ #3 ]
709 ⟨/plain-TeX⟩
710   {
```

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.

```
711    \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`.

```
712    \str_clear_new:N \l_@@_previous_key_str
713    \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:

```
714        \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`):

```
715        \int_set:Nn \l_tmpa_int { \g_@@_line_int + \l_@@_jump_int }
716        \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`.

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

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

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

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

```
719        \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).

```
720        \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`.

```
721        \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.

```
722    \prop_gclear_new:c
723        { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
724    \prop_gset_eq:cN
725        { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
726        \l_tmpa_prop
727    }
```

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.

```
728 \cs_new_protected:Npn \@@_Arrow_first_columns:
729    { \@@_error:n { Arrow~not~in~last~column } \@@_Arrow }
```

49

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.

```
730 \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.

```
731   {
732 ⟨*LaTeX⟩
733     \str_clear_new:N \l_@@_type_env_str
734     \str_set:NV \l_@@_type_env_str \@currenvir
735 ⟨/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`.

```
736     \cs_if_exist:NT \tikz@library@external@loaded
737       { \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.

```
738     \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.

```
739     \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.

```
740     \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.

```
741     \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.

```
742     \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
743     \int_gzero:N \g_@@_arrow_int
744     \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
745     \int_gzero:N \g_@@_line_int
746     \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
747     \int_gzero:N \g_@@_col_int
748     \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
749     \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 programmation 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.

```
750     \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).

```
751    \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
752    \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
753    \str_clear_new:N \l_@@_prefix_str
754    \str_set:Nx \l_@@_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

```
755    \cs_set_eq:NN \\ \@@_cr:
756    \dim_zero:N \mathsurround
```

These counters will be used later as variables.

```
757    \int_zero_new:N \l_@@_initial_int
758    \int_zero_new:N \l_@@_final_int
759    \int_zero_new:N \l_@@_arrow_int
760    \int_zero_new:N \l_@@_pos_of_arrow_int
761    \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.

```
762    \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.

```
763    \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`.

```
764    ⟨*LaTeX⟩
765    \seq_clear_new:N \l_@@_labels_seq

766    \bool_set_false:N \l_@@_tag_next_line_bool
767    ⟨/LaTeX⟩
```

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

```
768    \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`.

```
769    \tl_clear_new:N \l_@@_code_before_tl
770    \tl_clear_new:N \l_@@_code_after_tl
```

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

```
771    \str_clear_new:N \l_@@_previous_key_str
772    \bool_if:NT \l_@@_in_WithArrows_bool
773      { \keys_set:nn { WithArrows / WithArrows } { #1 } }
774    \bool_if:NT \l_@@_in_DispWithArrows_bool
775      { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

---

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

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.

```
776        \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`).

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

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

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

- once the preamble constructed, the primitive `\halign` is executed, and, in each row of the `\halign`, the counter `\g_@@_col_int` will be increased from column to column.

```
778        \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.

```
779        \seq_clear_new:N \l_@@_format_seq
780        \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).

```
781    ⟨*LaTeX⟩
782        \bool_if:NT \c_@@_footnote_bool { \begin { savenotes } }
783    ⟨/LaTeX⟩
```

We execute the code `\l_@@_code_before_tl` of the option `code-before` of the environment after the potential `\begin{savenotes}` and, symetrically, 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`).

```
784        \l_@@_code_before_tl
785    ⟨*LaTeX⟩
786        \cs_set_eq:NN \notag \@@_notag:
787        \cs_set_eq:NN \nonumber \@@_nonumber:
788        \cs_set_eq:NN \tag \@@_tag
789        \cs_set_eq:NN \@@_old_label \label
790        \cs_set_eq:NN \label \@@_label:n
791        \cs_set_eq:NN \tagnextline \@@_tagnextline:
792    ⟨/LaTeX⟩
793      }
```

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 differents counters but this way saves a counter).

```
794 \cs_new_protected:Npn \@@_construct_halign:
795   {
796     \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
797       {
```

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.

```
798         \use:e
799           {
```

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.

```
800             \int_gdecr:N \g_@@_col_int
801             \@@_construct_halign:
802             \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
803               {
```

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`.

```
804                 \cs_set_eq:cN { \l_@@_command_name_str } \@@_Arrow
805 ⟨*LaTeX⟩
806                 \bool_if:NT \l_@@_in_DispWithArrows_bool
807                   {
```

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).

```
808                     \@@_test_if_to_tag:
```

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

```
809                     \bool_if:NT \c_@@_amsthm_loaded_bool \@@_set_qedhere:
810                   }
811 ⟨/LaTeX⟩
812               }
813             \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
814             \str_if_eq:VnT \l_@@_type_col_str { C } \hfil
815             \str_if_eq:VnT \l_@@_type_col_str { r } \hfill
816             \str_if_eq:VnT \l_@@_type_col_str { R } \hfill
817             \int_gincr:N \g_@@_col_int
818             \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
819             \c_math_toggle_token
820             \str_if_eq:VnT \l_@@_type_col_str { C } { { } }
821             \str_if_eq:VnT \l_@@_type_col_str { L } { { } }
822             \bool_if:NT \l_@@_displaystyle_bool \displaystyle
823             ##
824             \str_if_eq:VnT \l_@@_type_col_str { C } { { } }
825             \str_if_eq:VnT \l_@@_type_col_str { R } { { } }
826             \c_math_toggle_token
```

53

```
827          \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
828             \@@_construct_nodes:
829                {
```

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:`).

```
830                   \str_if_eq:VnT \l_@@_type_col_str { l } \hfil
831                   \str_if_eq:VnT \l_@@_type_col_str { L } \hfil
832                   \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
833                   \str_if_eq:VnT \l_@@_type_col_str { C } \hfil
834                   \bool_if:NT \l_@@_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
835                   &
836                }
837             }
838       }
```

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

```
839       {
840         \bool_if:NTF \l_@@_in_WithArrows_bool
841           {
842             \ialign
843             \bgroup
844           }
845           {
846             \halign to \l_@@_linewidth_dim
847             \bgroup
848             \bool_if:NT \l_@@_fleqn_bool
849               { \skip_horizontal:N \l_@@_mathindent_dim }
850           }
851         \int_gincr:N \g_@@_line_int
852         \int_gzero:N \g_@@_col_int
853         \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
854           {
855             \skip_horizontal:n
856               { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
857           }
858         \strut
859       }
860   }
```

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.

```
861 \cs_new_protected:Npn \@@_construct_nodes:
862   {
```

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).

```
863     \tikz [ remember~picture , overlay ]
864       \node
865         [
866           node~contents = { } ,
867           @@_node_style ,
868           name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
869         ]
870         ;
871     \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).

```
872     \tikz [ remember~picture , overlay ]
873       \node
874         [
```

```
875            node~contents = { } ,
876            @@_node_style ,
877            name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
878          ]
879          ;
880      \str_if_empty:NF \l_@@_name_str
881        {
882          \pgfpicture
883          \pgfnodealias
884            { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
885            { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l }
886          \pgfnodealias
887            { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
888            { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
889          \endpgfpicture
890        }
891      \bool_if:NT \l_@@_show_node_names_bool
892        {
893          \hbox_overlap_right:n
894            { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
895        }
896    }
```

12.8.3   The environment {WithArrows}

```
897  ⟨*LaTeX⟩
898  \NewDocumentEnvironment { WithArrows } { ! O { } }
899  ⟨/LaTeX⟩
900  ⟨*plain-TeX⟩
901  \cs_new_protected:Npn \WithArrows
902    {
903      \group_begin:
904      \peek_meaning:NTF [
905        { \WithArrows_i }
906        { \WithArrows_i [  ] }
907    }
908  \cs_new_protected:Npn \WithArrows_i [ #1 ]
909  ⟨/plain-TeX⟩
910    {
911      \bool_set_true:N \l_@@_in_WithArrows_bool
912      \bool_set_false:N \l_@@_in_DispWithArrows_bool
913  ⟨*plain-TeX⟩
914      \str_clear_new:N \l_@@_type_env_str
915      \str_set:Nn \l_@@_type_env_str { WithArrows }
916  ⟨/plain-TeX⟩
917      \@@_pre_halign:n { #1 }
918      \if_mode_math: \else:
919        \@@_error:n { WithArrows~outside~math~mode }
920      \fi:
```

The environment begins with a \vtop, a \vcenter or a \vbox[34] depending of the value of \l_@@_pos_env_int (fixed by the options t, c or b). The environment {WithArrows} must be used in math mode[35] and therefore, we can use \vcenter.

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

The command \spread@equation is the command used by amsmath in the beginning of an alignment

---

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

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

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).

```
923      \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.

```
924      \@@_construct_halign:
```

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

An environmemnt {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.

```
925      &&
926      \@@_error:n { Too~much~columns~in~WithArrows }
927      \c_math_toggle_token
928      \bool_if:NT \l_@@_displaystyle_bool \displaystyle
929      { ## }
930      \c_math_toggle_token
931      \cr
932    }
```

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

```
933  ⟨*plain-TeX⟩
934  \cs_new_protected:Npn \endWithArrows
935  ⟨/plain-TeX⟩
936    {
937      \\
938      \egroup
939      \egroup
940      \@@_post_halign:
```

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

```
941  ⟨*LaTeX⟩
942      \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
943  ⟨/LaTeX⟩
944  ⟨*plain-TeX⟩
945      \group_end:
946  ⟨/plain-TeX⟩
947    }
```

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}.

```
948  \cs_new_protected:Npn \@@_post_halign:
```

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

```
949    {
950      \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.

```
951      \normalbaselines
```

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

```
952    \int_compare:nNnT \g_@@_arrow_int > 0
953       {
```

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).

```
954       \int_compare:nNnT \g_@@_arrow_int = 1
955          {
956            \int_compare:nNnT \l_@@_pos_arrow_int > 5
957              { \int_set:Nn \l_@@_pos_arrow_int 5 }
958          }
959       \@@_scan_arrows:
960       }
```

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

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

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

```
963       \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.[36]

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

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

```
969    \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
970    \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
971    \seq_gput_right:Nx \g_@@_position_in_the_tree_seq
972       { \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.

```
973    \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
974       { \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.

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

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

---

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

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 `expl3` commands `\group_align_safe_begin:` and `\group_align_safe_end:` are specifically designed for this purpose: test the token that follows in an `\halign` structure.

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

```
984 \cs_new_protected:Npn \@@_cr:
985   {
986     \scan_stop:
```

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

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

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.

```
993 \cs_new_protected:Npn \@@_cr_i:
994   { \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.

```
995 ⟨*LaTeX⟩
996 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
997   {
998     \peek_meaning_ignore_spaces:NTF \end
999       {
1000        \@@_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`.

```
1001       \@@_analyze_end:Nn
1002     }
1003     { \@@_cr_iii:n { #1 } }
1004   }

1005 \cs_new_protected:Npn \@@_cr_iii:n #1
1006 ⟨/LaTeX⟩
1007 ⟨*plain-TeX⟩
1008 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1009 ⟨/plain-TeX⟩
1010   {
1011     \group_align_safe_end:
```

For the environment `{DispWithArrows}`, the behaviour of `\\` is different because we add the last column which is the column for the tag (number of the equation). Even if there is no tag, this column is used for the v-nodes.[37]

```
1012     \bool_if:NT \l_@@_in_DispWithArrows_bool
```

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

---

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

```
1013                    {
1014  ⟨*LaTeX⟩
1015            \clist_if_in:NnTF \l_@@_tags_clist { all }
1016                {
```

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).

```
1017                    \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 `&`).

```
1018                    \cs_gset:Npx \g_tmpa_tl
1019                      { \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 differents labels of a same line are stored in a sequence `\l_@@_labels_seq`.

```
1020                    \seq_if_empty:NF \l_@@_labels_seq
1021                      {
```

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`.

```
1022                        \cs_set:Npx \@currentlabel { \p@equation \g_tmpa_tl }
```

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

```
1023                        \bool_if:NT \c_@@_hyperref_loaded_bool
1024                          {
1025                            \str_set:Nn \This@name { equation }
1026                            \hyper@refstepcounter { equation }
1027                          }
```

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.

```
1028                        \bool_if:NT \c_@@_cleveref_loaded_bool
1029                          {
1030                            \cref@constructprefix { equation } \cref@result
1031                            \protected@edef \cref@currentlabel
1032                              {
1033                                [
1034                                  \cs_if_exist:NTF \cref@equation@alias
1035                                    \cref@equation@alias
1036                                    { equation }
1037                                ]
1038                                [ \arabic { equation } ] [ \cref@result ]
1039                                \p@equation \g_tmpa_tl
1040                              }
1041                          }
```

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`).

```
1042                        \seq_map_function:NN \l_@@_labels_seq \@@_old_label
1043                      }
```

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.

```
1044                    \@@_save:N \l_@@_tag_star_bool
1045                    \@@_save:N \l_@@_qedhere_bool
1046                    \bool_if:NT \l_@@_tag_next_line_bool
1047                      {
1048                        \openup -\jot
1049                        \bool_set_false:N \l_@@_tag_next_line_bool
1050                        \notag \\ &
```

```
1051                    }
1052                  &
1053              \@@_restore:N \l_@@_tag_star_bool
1054              \@@_restore:N \l_@@_qedhere_bool
1055              \bool_if:NT \l_@@_qedhere_bool
1056                { \hbox_overlap_left:n \@@_qedhere_i: }
1057              \cs_set_eq:NN \theequation \g_tmpa_tl
1058              \bool_if:NT \l_@@_tag_star_bool
1059                { \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`.

```
1060              \hbox_overlap_left:n
1061                {
1062                  \bool_if:NF \c_@@_leqno_bool
1063                    {
1064                      \pgfpicture
1065                      \pgfrememberpicturepositiononpagetrue
1066                      \pgfcoordinate
1067                        { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1068                        \pgfpointorigin
1069                      \endpgfpicture
1070                    }
1071                  \quad
1072                  \@eqnnum
1073                }
1074              \bool_if:NT \c_@@_leqno_bool
1075                {
1076                  \pgfpicture
1077                  \pgfrememberpicturepositiononpagetrue
1078                  \pgfcoordinate
1079                    { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1080                    \pgfpointorigin
1081                  \endpgfpicture
1082                }
1083            }
1084            {
1085              \@@_save:N \l_@@_qedhere_bool
1086 ⟨/LaTeX⟩
1087              &
1088 ⟨*LaTeX⟩
1089              \@@_restore:N \l_@@_qedhere_bool
1090              \bool_if:NT \l_@@_qedhere_bool
1091                { \hbox_overlap_left:n \@@_qedhere_i: }
1092 ⟨/LaTeX⟩
1093              \pgfpicture
1094              \pgfrememberpicturepositiononpagetrue
1095              \pgfcoordinate
1096                { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1097                \pgfpointorigin
1098              \endpgfpicture
1099 ⟨*LaTeX⟩
1100            }
1101 ⟨/LaTeX⟩
1102          }
1103      \dim_compare:nNnT { #1 } < \c_zero_dim
1104        { \@@_error:n { option~of~cr~negative } }
1105
1106      \cr
1107      \noalign
1108        {
1109          \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1110          \skip_vertical:N \l_tmpa_dim
```

60

```
1111        \skip_vertical:N \l_@@_interline_skip
1112        \scan_stop:
1113      }
1114    }
```

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

The following command will be used when, after a \\ (and its optional arguments) there is a \end. You want to known 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.

```
1115  ⟨*LaTeX⟩
1116  \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1117    {
1118      \str_if_eq:VnT \l_@@_type_env_str { #2 }
1119        {
1120          \@@_error:n { newline~at~the~end~of~env }
1121          \group_begin:
1122          \globaldefs = 1
1123          \@@_msg_redirect_name:nn { newline~at~the~end~of~env } { none }
1124          \group_end:
1125        }
```

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

```
1126      \end { #2 }
1127    }
1128  ⟨/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 juste 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}.

```
1129  ⟨*LaTeX⟩
1130  \bool_new:N \l_@@_in_label_or_minipage_bool
1131  ⟨/LaTeX⟩
1132  ⟨*LaTeX⟩
1133  \NewDocumentEnvironment { DispWithArrows } { ! d < > ! O { } }
1134  ⟨/LaTeX⟩
1135  ⟨*plain-TeX⟩
1136  \cs_new_protected:Npn \DispWithArrows
1137    {
1138      \group_begin:
1139      \peek_meaning:NTF <
1140        { \DispWithArrows_i }
1141        { \DispWithArrows_i < \c_novalue_tl > }
1142    }
1143  \cs_new_protected:Npn \DispWithArrows_i < #1 >
1144    {
1145      \peek_meaning:NTF [
1146        { \DispWithArrows_ii < #1 > }
1147        { \DispWithArrows_ii < #1 > [ ] }
1148    }
1149  \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
```

```
1150  ⟨/plain-TeX⟩
1151      {
1152        \bool_set_true:N \l_@@_in_DispWithArrows_bool
1153  ⟨*plain-TeX⟩
1154        \str_clear_new:N \l_@@_type_env_str
1155        \str_set:Nn \l_@@_type_env_str { DispWithArrows }
1156  ⟨/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 quicky after the beginning of the environment (in particular, because it must be done before the execution of the code-before[38]).

```
1157  ⟨*LaTeX⟩
1158        \bool_if:NF \l_@@_sbwi_bool
1159          {
1160            \legacy_if:nT { @inlabel }
1161              { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1162            \legacy_if:nT { @minipage }
1163              { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1164          }
1165  ⟨/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).

```
1166  ⟨*LaTeX⟩
1167        \bool_if:nT \c_@@_mathtools_loaded_bool
1168          {
1169            \MH_if_boolean:nT { show_only_refs }
1170              {
1171                \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.

```
1172                \MH_set_boolean_T:n { show_only_refs }
1173              }
1174          }
```

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.

```
1175        \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.

```
1176        \bool_if:NT \c_@@_amsmath_loaded_bool \intertext@
1177  ⟨/LaTeX⟩
1178      \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l_@@_left_brace_tl { #1 } }
1179      \@@_pre_halign:n { #2 }
```

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

```
1180  ⟨*LaTeX⟩
1181      \bool_if:NT \l_@@_subequations_bool { \begin { subequations } }
1182  ⟨/LaTeX⟩

1183      \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
1184        {
```

---

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

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

```
1185            \hbox_set:Nn \l_tmpa_box
1186              {
```

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

```
1187                \group_begin:
1188                \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1189                \c_math_toggle_token
1190                \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1191                \c_math_toggle_token
1192                \group_end:
1193              }
1194            \dim_zero_new:N \l_@@_delim_wd_dim
1195            \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
1196            \box_clear_new:N \l_@@_left_brace_box
1197            \hbox_set:Nn \l_@@_left_brace_box
1198              {
1199                \group_begin:
1200                  \cs_set_eq:NN \label \@@_old_label
1201                  \c_math_toggle_token
1202                  \bool_if:NT \l_@@_displaystyle_bool \displaystyle
1203                  \l_@@_left_brace_tl
1204                  { }
1205                  \c_math_toggle_token
1206                \group_end:
1207              }
1208          }
```

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

```
1209  ⟨*LaTeX⟩
1210      \tl_clear_new:N \l_@@_tag_tl


1211      \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.

```
1212      \bool_set_false:N \l_@@_tag_star_bool
1213  ⟨/LaTeX⟩

1214      \if_mode_math:
1215        \@@_fatal:n { DispWithArrows~in~math~mode }
1216      \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`.

```
1217  ⟨*plain-TeX⟩
1218      \dim_zero_new:N \linewidth
1219      \dim_set_eq:NN \linewidth \displaywidth
1220  ⟨/plain-TeX⟩
1221  ⟨*LaTeX⟩
1222      \bool_if:NTF \l_@@_in_label_or_minipage_bool
1223        { \c_math_toggle_token }
1224        {
1225  ⟨/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.

```
1226          \if_mode_vertical:
1227          \nointerlineskip
1228          \hbox_to_wd:nn { .6 \linewidth } { }
1229          \fi:
1230          \c_math_toggle_token \c_math_toggle_token
```

```
1231 ⟨*LaTeX⟩
1232        }
1233 ⟨/LaTeX⟩
1234     \dim_zero_new:N \l_@@_linewidth_dim
1235 ⟨*LaTeX⟩
1236     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1237        { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
1238        { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
1239 ⟨/LaTeX⟩
1240 ⟨*plain-TeX⟩
1241     \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
1242 ⟨/plain-TeX⟩
1243     \box_clear_new:N \l_@@_halign_box
1244     \setbox \l_@@_halign_box \vtop \bgroup
1245     \tabskip =
1246       \bool_if:NTF \l_@@_fleqn_bool
1247         \c_zero_skip
1248         { 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).

```
1249     \spread@equation
1250     \@@_construct_halign:
1251     \tabskip = 0 pt plus 1000 pt minus 1000 pt
1252     &
```

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.

```
1253     $ ## $
1254     \tabskip = \c_zero_skip
1255     &&
1256     \@@_fatal:n { Too~much~columns~in~DispWithArrows }
1257     \bool_if:nT \c_false_bool { ## }
1258     \cr
1259  }
```

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

```
1260 ⟨*plain-TeX⟩
1261 \cs_new_protected:Npn \endDispWithArrows
1262 ⟨/plain-TeX⟩
1263    {
1264 ⟨*LaTeX⟩
1265     \clist_if_in:NnT \l_@@_tags_clist { last }
1266        { \clist_set:Nn \l_@@_tags_clist { all } }
1267 ⟨/LaTeX⟩
1268     \\
```

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

```
1269     \egroup
1270     \unskip \unpenalty \unskip \unpenalty
1271     \box_set_to_last:N \l_tmpa_box
1272     \nointerlineskip
1273     \box_use:N \l_tmpa_box
1274     \dim_gzero_new:N \g_@@_alignment_dim
1275     \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
1276     \box_clear_new:N \l_@@_new_box
1277     \hbox_set:Nn \l_@@_new_box { \hbox_unpack_drop:N \l_tmpa_box }
```

```
1278      \dim_compare:nNnT
1279        { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
1280        { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }
```

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

```
1281      \egroup
1282      \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
1283        { \box_use_drop:N \l_@@_halign_box }
1284        {
1285          \hbox_to_wd:nn \l_@@_linewidth_dim
1286            {
1287            \bool_if:NTF \l_@@_fleqn_bool
1288              { \skip_horizontal:N \l_@@_mathindent_dim }
1289              \hfil
1290            \hbox_to_wd:nn \g_@@_alignment_dim
1291              {
1292                \box_use_drop:N \l_@@_left_brace_box
1293                \dim_set:Nn \l_tmpa_dim
1294                  {
1295                    \box_ht:N \l_@@_halign_box
1296                    + \box_dp:N \l_@@_halign_box
1297                  }
1298                \group_begin:
1299                \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1300                \c_math_toggle_token
1301                  \left \l_@@_replace_left_brace_by_tl
1302                    \vcenter to \l_tmpa_dim { \vfil }
1303                  \right.
1304                \c_math_toggle_token
1305                \group_end:
1306                \hfil
1307              }
1308            \hfil
1309            }
1310          \skip_horizontal:N -\l_@@_linewidth_dim
1311          \vcenter { \box_use_drop:N \l_@@_halign_box }
1312        }
```

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.

```
1313      \dim_gzero_new:N \g_@@_right_x_dim
1314      \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
1315      \pgfpicture
1316        \pgfrememberpicturepositiononpagetrue
1317        \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
1318          {
1319          \cs_if_free:cTF
1320            { pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
1321            { \@@_fatal:n { Inexistent~v-node } }
1322            {
1323              \pgfpointanchor
1324                { wa - \l_@@_prefix_str - \l_tmpa_int - v }
1325                { center }
1326              \dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
1327                { \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x }
1328            }
1329          }
1330      \endpgfpicture
```

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

```
1331        \@@_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).

```
1332 ⟨*LaTeX⟩
1333        \bool_if:nT \c_@@_mathtools_loaded_bool
1334          { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1335        \bool_if:NTF \l_@@_in_label_or_minipage_bool
1336          {
1337            \c_math_toggle_token
1338            \skip_vertical:N \belowdisplayskip
1339          }
1340          { \c_math_toggle_token \c_math_toggle_token }
1341 ⟨/LaTeX⟩
1342 ⟨*plain-TeX⟩
1343            \c_math_toggle_token \c_math_toggle_token
1344 ⟨/plain-TeX⟩

1345 ⟨*LaTeX⟩
1346        \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).

```
1347        \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
1348 ⟨/LaTeX⟩
1349 ⟨*plain-TeX⟩
1350        \group_end:
1351 ⟨/plain-TeX⟩
1352 ⟨*LaTeX⟩
1353        \ignorespacesafterend
1354 ⟨/LaTeX⟩
1355      }
```

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

```
1356 ⟨*LaTeX⟩
1357 \NewDocumentEnvironment { DispWithArrows* } { }
1358   {
1359     \WithArrowsOptions { notag }
1360     \DispWithArrows
1361   }
1362   \endDispWithArrows
1363 ⟨/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.

```
1364 \cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
1365   {
1366     \bool_if:NTF \l_@@_in_WithArrows_bool
1367       { \@@_error:nn { Not~allowed~in~WithArrows } { #1 } } }
1368       {
1369         \int_compare:nNnTF \g_@@_col_int < \l_@@_nb_cols_int
```

```
1370            { \@@_error:nn { Not~allowed~in~DispWithArrows } { #1 } } }
1371            { #2 }
1372        }
1373    }
```

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

```
1374 ⟨*LaTeX⟩
1375 \cs_new_protected:Npn \@@_notag:
1376    { \@@_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}.

```
1377 \cs_new_protected:Npn \@@_nonumber:
1378    { \@@_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.

```
1379 \NewDocumentCommand \@@_tag { s m }
1380    {
1381      \@@_if_in_last_col_of_disp:Nn \tag
1382        {
1383          \tl_if_empty:NF \l_@@_tag_tl
1384            { \@@_error:nn  { Multiple~tags } { #2 } }
1385          \clist_set:Nn \l_@@_tags_clist { all }
1386          \bool_if:nT \c_@@_mathtools_loaded_bool
1387            {
1388              \MH_if_boolean:nT { show_only_refs }
1389                {
1390                  \MH_if_boolean:nF { show_manual_tags }
1391                    { \clist_clear:N \l_@@_tags_clist }
1392                }
1393            }
1394          \tl_set:Nn \l_@@_tag_tl { #2 }
1395          \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`.[39]

```
1396          \bool_if:nT { #1 && ! \bool_if_p:N \c_@@_amsmath_loaded_bool }
1397            { \@@_error:n { tag*~without~amsmath } }
1398        }
1399    }
```

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 differents labels of a same line in a sequence `\l_@@_labels_seq`.

```
1400 \cs_new_protected:Npn \@@_label:n #1
1401    {
1402      \@@_if_in_last_col_of_disp:Nn \label
1403        {
1404          \seq_if_empty:NF \l_@@_labels_seq
1405            {
1406              \bool_if:NTF \c_@@_cleveref_loaded_bool
1407                { \@@_error:n { Multiple~labels~with~cleveref } }
1408                { \@@_error:n { Multiple~labels } }
1409            }
```

---

[39]There are two versions of @eqnnum, a standard version and a version for the option leqno.

```
1410          \seq_put_right:Nn \l_@@_labels_seq { #1 }
1411          \bool_if:nT \c_@@_mathtools_loaded_bool
1412            {
1413              \MH_if_boolean:nT { show_only_refs }
1414                {
1415                  \cs_if_exist:cTF { MT_r_#1 }
1416                    { \clist_set:Nn \l_@@_tags_clist { all } }
1417                    { \clist_clear:N \l_@@_tags_clist }
1418                }
1419            }
1420          \bool_if:nT \c_@@_autonum_loaded_bool
1421            {
1422              \cs_if_exist:cTF { autonum@#1Referenced }
1423                { \clist_set:Nn \l_@@_tags_clist { all } }
1424                { \clist_clear:N \l_@@_tags_clist }
1425            }
1426        }
1427    }
```

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

```
1428  \cs_new_protected:Npn \@@_tagnextline:
1429    {
1430      \@@_if_in_last_col_of_disp:Nn \tagnextline
1431        { \bool_set_true:N \l_@@_tag_next_line_bool }
1432    }
```

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`.

```
1433  \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }
1434  \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.

```
1435  \cs_new_protected:Npn \@@_qedhere_i:
1436    {
1437      \group_begin:
1438        \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.

```
1439        \cs_set_eq:NN \qed@elt \setQED@elt
1440        \QED@stack \relax \relax
1441      \group_end:
1442    }
1443  ⟨/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 adjustement) and we actually draw the arrows.

```
1444 \cs_new_protected:Npn \@@_scan_arrows:
1445   {
1446     \group_begin:
1447     \int_compare:nNnT \l_@@_pos_arrow_int = 7
1448       {
1449         \@@_scan_arrows_i:
1450         \int_set:Nn \l_@@_pos_arrow_int 8
1451       }
1452     \@@_scan_arrows_i:
1453     \group_end:
1454   }
```

```
1455 \cs_new_protected:Npn \@@_scan_arrows_i:
1456   {
```

`\l_@@_first_arrow_of_group_int` will be the first arrow of the current group.
`\l_@@_first_line_of_group_int` will be the first line involved in the group of arrows (equal to the initial line of the first arrow of the group because the option `jump` is always positive).
`\l_@@_first_arrows_seq` will be the list the arrows of the group starting at the first line of the group (we may have several arrows starting from the same line). We have to know all these arrows because of the adjustement 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).

```
1457     \int_zero_new:N \l_@@_first_arrow_of_group_int
1458     \int_zero_new:N \l_@@_first_line_of_group_int
1459     \int_zero_new:N \l_@@_last_line_of_group_int
1460     \seq_clear_new:N \l_@@_first_arrows_seq
1461     \seq_clear_new:N \l_@@_last_arrows_seq
```

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

```
1462     \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.

```
1463     \int_set:Nn \l_@@_arrow_int 1
1464     \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1465       {
```

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.

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

```
1477          \prop_get:cnN
1478            { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1479            { 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.

```
1480          \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1481            {
1482              \int_compare:nNnF \l_@@_pos_arrow_int = 8
1483                { \@@_error:n { Too~few~lines~for~an~arrow } }
1484            }
1485            \@@_treat_an_arrow_in_scan:
```

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

```
1486          \int_incr:N \l_@@_arrow_int
1487        }
```

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).

```
1488      \bool_if:nT
1489        {
1490          ! \int_compare_p:nNn \l_@@_pos_arrow_int = 7
1491          &&
1492          \int_compare_p:nNn \l_@@_first_arrow_of_group_int > 0
1493        }
1494        { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1495    }
```

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)

```
1496  \cs_new_protected:Npn \@@_treat_an_arrow_in_scan:
1497    {
```

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).

```
1498      \bool_lazy_and:nnT
1499        { \int_compare_p:nNn \l_@@_arrow_int > 1 }
1500        {
1501          \bool_lazy_or_p:nn
1502            {
1503              \bool_lazy_and_p:nn
1504                {
1505                  \int_compare_p:nNn
1506                    \l_@@_initial_int > \l_@@_last_line_of_group_int
1507                }
1508                { \bool_not_p:n { \int_compare_p:nNn \l_@@_pos_arrow_int = 7 } }
1509            }
1510            { \str_if_eq_p:Vn \l_@@_status_arrow_str { new-group } }
1511        }
1512        {
1513          \int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
1514            {
1515              \@@_draw_arrows:nn
```

70

```
1516                    \l_@@_first_arrow_of_group_int
1517                    { \l_@@_arrow_int - 1 }
1518                }
1519            \bool_set_true:N \l_@@_new_group_bool
1520        }
```

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 circonstancies: 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`.

```
1521        \bool_if:nTF \l_@@_new_group_bool
1522            {
1523                \bool_set_false:N \l_@@_new_group_bool
1524                \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1525                \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1526                \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1527                \seq_clear:N \l_@@_first_arrows_seq
1528                \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
1529                \seq_clear:N \l_@@_last_arrows_seq
1530                \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.

```
1531                \int_compare:nNnF \l_@@_pos_arrow_int = 8
1532                    { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1533            }
```

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

```
1534            {
```

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

```
1535                \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1536                    {
```

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

```
1537                        \int_compare:nT
1538                            { \l_@@_initial_int = \l_@@_first_line_of_group_int }
1539                            { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1540                        \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1541                            {
1542                                \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1543                                \seq_clear:N \l_@@_last_arrows_seq
1544                                \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
1545                            }
1546                            {
1547                                \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1548                                    { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1549                            }
1550                    }
1551            }
```

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.

```
1552    \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1553      {
1554        \int_compare:nNnF \l_@@_pos_arrow_int = 8
1555          { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1556      }
1557    }
```

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

```
1558  \cs_generate_variant:Nn \keys_set:nn { n o }
1559  \cs_new_protected:Npn \@@_keys_set:
1560    { \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`.

```
1561  \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1562    {
1563      \group_begin:
1564      \int_zero_new:N \l_@@_first_arrow_int
1565      \int_set:Nn \l_@@_first_arrow_int { #1 }
1566      \int_zero_new:N \l_@@_last_arrow_int
1567      \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.

```
1568      \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1569      \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1570        {
```

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.

```
1571          \prop_get:cnN
1572            { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1573            { initial } \l_tmpa_tl
1574          \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1575          \prop_get:cnN
1576            { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1577            { final } \l_tmpa_tl
1578          \int_set:Nn \l_@@_final_int \l_tmpa_tl
1579          \prop_get:cnN
1580            { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1581            { 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).

```
1582          \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).

```
1583            {
1584              \str_if_eq:VnTF \l_@@_status_arrow_str { over }
1585                { \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
1586                \@@_draw_arrow:
1587            }
1588          \int_incr:N \l_@@_arrow_int
1589        }
1590      \@@_draw_o_arrows_of_the_group:
1591      \group_end:
1592    }
```

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).

```
1593 \cs_new_protected:Npn \@@_draw_arrow:
1594   {
1595     \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.

```
1596     \prop_get:cnN
1597       { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1598       { options } \l_tmpa_tl
1599     \str_clear_new:N \l_@@_previous_key_str
1600     \exp_args:NNo \exp_args:No
1601     \@@_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`:

```
1602     \bool_set_false:N \l_@@_initial_r_bool
1603     \bool_set_false:N \l_@@_final_r_bool
1604     \int_case:nn \l_@@_pos_arrow_int
1605       {
1606         0 { \bool_set_true:N \l_@@_final_r_bool }
1607         2 { \bool_set_true:N \l_@@_initial_r_bool }
1608         3
1609           {
1610             \bool_set_true:N \l_@@_initial_r_bool
1611             \bool_set_true:N \l_@@_final_r_bool
1612           }
1613       }
```

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

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

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

```
1614     \int_compare:nNnT \l_@@_pos_arrow_int = 5
1615       {
1616         \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1617         \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1618       }
```

`\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`.

```
1619     \tl_set:Nx \l_@@_initial_tl
1620       { \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl }
1621     \tl_set:Nx \l_@@_final_tl
1622       { \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`.

```
1623     \prop_get:cnN
1624       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1625       { label }
1626       \l_tmpa_tl
```

73

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.

```
1627    \seq_if_in:NxTF \l_@@_first_arrows_seq
1628      { \int_use:N \l_@@_arrow_int }
1629      { \bool_set_true:N \l_tmpa_bool }
1630      { \bool_set_false:N \l_tmpa_bool }
1631    \seq_if_in:NxTF \l_@@_last_arrows_seq
1632      { \int_use:N \l_@@_arrow_int }
1633      { \bool_set_true:N \l_tmpb_bool }
1634      { \bool_set_false:N \l_tmpb_bool }
1635    \int_compare:nNnT \l_@@_pos_arrow_int = 5
1636      {
1637        \bool_set_true:N \l_tmpa_bool
1638        \bool_set_true:N \l_tmpb_bool
1639      }
```

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 adjustement 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).

```
1640    \dim_gzero_new:N \g_@@_x_initial_dim
1641    \dim_gzero_new:N \g_@@_x_final_dim
1642    \dim_gzero_new:N \g_@@_y_initial_dim
1643    \dim_gzero_new:N \g_@@_y_final_dim
1644    \pgfpicture
1645      \pgfrememberpicturepositiononpagetrue
1646      \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south }
1647      \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1648      \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1649      \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north }
1650      \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1651      \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1652    \endpgfpicture
1653    \bool_lazy_and:nnTF
1654      {
1655        \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
1656                          > \l_@@_max_length_of_arrow_dim
1657      }
1658      { \int_compare_p:nNn { \l_@@_final_int - \l_@@_initial_int } = 1 }
1659      {
1660        \tl_gset:Nx \g_tmpa_tl
1661          {
1662            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1663              { \dim_use:N \g_@@_x_initial_dim }
1664              { \dim_use:N \l_@@_x_dim } ,
1665            \dim_eval:n
1666              {
1667                ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1668                + 0.5 \l_@@_max_length_of_arrow_dim
1669              }
1670          }
1671        \tl_gset:Nx \g_tmpb_tl
1672          {
```

```
1673          \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1674            { \dim_use:N \g_@@_x_final_dim }
1675            { \dim_use:N \l_@@_x_dim } ,
1676          \dim_eval:n
1677            {
1678              ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1679              - 0.5 \l_@@_max_length_of_arrow_dim
1680            }
1681        }
1682      }
1683      {
1684        \tl_gset:Nx \g_tmpa_tl
1685          {
1686            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1687              { \dim_use:N \g_@@_x_initial_dim }
1688              { \dim_use:N \l_@@_x_dim } ,
1689            \bool_if:NTF \l_tmpa_bool
1690              { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
1691              { \dim_use:N \g_@@_y_initial_dim }
1692          }
1693        \tl_gset:Nx \g_tmpb_tl
1694          {
1695            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1696              { \dim_use:N \g_@@_x_final_dim }
1697              { \dim_use:N \l_@@_x_dim } ,
1698            \bool_if:NTF \l_tmpb_bool
1699              { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
1700              { \dim_use:N \g_@@_y_final_dim }
1701          }
1702      }
```

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*.[40]

```
1703        \@@_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).

```
1704      \group_end:
1705    }
```

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`.

```
1706  \cs_new_protected:Npn \@@_def_function_tmpa:n #1
1707    {
1708      \cs_set:Npn \@@_tmpa:nnn ##1 ##2 ##3
1709        {
1710  ⟨*LaTeX⟩
1711          \begin{tikzpicture}
1712  ⟨/LaTeX⟩
1713  ⟨*plain-TeX⟩
1714          \tikzpicture
1715  ⟨/plain-TeX⟩
```

---

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

```
1716          [
1717            @@_standard ,
1718            every~path / .style = WithArrows / arrow
1719          ]
```

You keep track of the bouding box because we want to compute the total width of the arrow (with the label) for the arrows of type over.

```
1720          \pgf@relevantforpicturesizetrue
1721          #1
1722          \dim_compare:nNnTF \pgf@picminx = { 16000 pt }
1723            { \dim_set_eq:NN \l_tmpa_dim \c_zero_dim }
1724            { \dim_set:Nn \l_tmpa_dim { \pgf@picmaxx - \pgf@picminx } }
1725          \dim_add:Nn \l_tmpa_dim \l_@@_xoffset_dim
1726          \prop_gput:cnV
1727            { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1728            { width }
1729            \l_tmpa_dim
1730          \pgfresetboundingbox
1731 ⟨*LaTeX⟩
1732          \end{tikzpicture}
1733 ⟨/LaTeX⟩
1734 ⟨*plain-TeX⟩
1735          \endtikzpicture
1736 ⟨/plain-TeX⟩
1737        }
1738    }
```

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

```
1739 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1740    {
```

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`).

```
1741        \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
1742          { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }
```

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

```
1743        \exp_args:NV \@@_def_function_tmpa:n \l_@@_tikz_code_tl
1744        \@@_tmpa:nnn { #1 } { #2 } { #3 }
1745    }
1746 \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`).

```
1747 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1748    {
```

First, we draw the arrow without the label.

```
1749        \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;
```

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

```
1750        \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.

```
1751        \dim_set:Nn \l_tmpa_dim
1752          { \g_@@_right_x_dim - \pgf@x - \pgfkeysvalueof { / pgf / inner~xsep } }
```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter "`text width`".[41]

```
1753        \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
```

---

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

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`).

```
1754        \tl_if_empty:NF \g_tmpa_tl
1755          {
1756            \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1757            \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1758              { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1759          }
```

Now, we can put the label with the right value for "`text width`".

```
1760        \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1761          {
1762            \path ( @@_label.west )
1763  ⟨*LaTeX⟩
1764            node [ anchor = west ]
1765              {
1766                \begin { minipage } { \l_tmpa_dim }
1767                #3
1768                \end { minipage }
1769              } ;
1770  ⟨/LaTeX⟩
1771  ⟨*plain-TeX⟩
1772            node [ anchor = west , text~width = \dim_use:N \l_tmpa_dim ]
1773              { #3 } ;
1774  ⟨/plain-TeX⟩
1775          }
1776      }
```

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`).

```
1777  \cs_new_protected:Npn \@@_update_x:nn #1 #2
1778    {
1779      \dim_gset_eq:NN \g_tmpa_dim \l_@@_x_dim
1780      \pgfpicture
1781      \pgfrememberpicturepositiononpagetrue
1782      \int_step_inline:nnn { #1 } { #2 }
1783        {
1784          \pgfpointanchor { wa - \l_@@_prefix_str - ##1 - l } { center }
1785          \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \pgf@x }
1786        }
1787      \endpgfpicture
1788      \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1789    }
```

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.

```
1790  \cs_new_protected:Npn \@@_draw_o_arrows_of_the_group:
1791    {
```

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 :

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

$$A = B$$
$$= C$$
$$= D$$
$$= E + E$$

```
1792      \seq_sort:Nn \l_@@_o_arrows_seq
1793        {
1794          \prop_get:cnN
1795            { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1796            { 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 expl3 integers because such token lists can be used directly in `\int_compare:nNnTF`.

```
1797          \prop_get:cnN
1798            { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1799            { final } \l_tmpb_tl
1800          \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1801            \sort_return_same:
1802            {
1803              \int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
1804                \sort_return_swapped:
1805                {
1806                  \prop_get:cnN
1807                    { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1808                    { initial } \l_tmpa_tl
1809                  \prop_get:cnN
1810                    { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1811                    { initial } \l_tmpb_tl
1812                  \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1813                    \sort_return_swapped:
1814                    \sort_return_same:
1815                }
1816            }
1817        }
```

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

```
1818      \seq_map_inline:Nn \l_@@_o_arrows_seq
1819        {
```

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

```
1820          \prop_get:cnN
1821            { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1822            { initial } \l_tmpa_tl
1823          \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1824          \prop_get:cnN
1825            { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1826            { final } \l_tmpa_tl
1827          \int_set:Nn \l_@@_final_int \l_tmpa_tl
```

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

```
1828          \prop_get:cnN
1829            { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1830            { 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 global dimension because we will have to exit a \pgfpicture.

```
1831        \dim_gzero:N \g_tmpa_dim
```

We will raise the boolean \g_tmpa_bool if we find an arrow "under" our arrow (we should find at least once since you are drawing an arrow of type o: if not, we will raise an error[42]).

```
1832        \bool_set_false:N \g_tmpa_bool
1833        \pgfpicture
1834        \pgfremberpicturepositiononpagetrue
1835        \int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
1836          {
1837            \prop_get:cnN
1838              { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1839              { initial } \l_tmpa_tl
1840            \prop_get:cnN
1841              { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1842              { final } \l_tmpb_tl
1843            \prop_get:cnN
1844              { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
1845              { status } \l_@@_status_arrow_str
1846            \bool_if:nT
1847              {
1848                ! \int_compare_p:n { ##1 = ####1 }
1849                && \int_compare_p:n { \l_@@_initial_int <= \l_tmpa_tl }
1850                && \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 positionned only relatively to the current group.

```
1851                && ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
1852              }
1853              {
```

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

```
1854                \bool_gset_true:N \g_tmpa_bool
1855                \prop_get:cnN
1856                  { g_@@ _ arrow _ \l_@@_prefix_str _ ####1 _ prop }
1857                  { width }
1858                  \l_tmpa_tl
```

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

```
1859                \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \l_tmpa_tl }
1860              }
1861          }
1862        \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).

```
1863        \bool_if:NTF \g_tmpa_bool
1864          {
1865            \int_set:Nn \l_@@_arrow_int { ##1 }
1866            \dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
1867            \dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim

1868            \@@_draw_arrow:
1869          }
1870          { \@@_error:n { o~arrow~with~no~arrow~under } }
1871      }
1872  }
```

---

[42]Maybe 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.

```
1873 ⟨*LaTeX⟩
1874 \NewExpandableDocumentCommand \WithArrowsLastEnv { }
1875   { \int_use:N \g_@@_last_env_int }
1876 ⟨/LaTeX⟩
1877 ⟨*plain-TeX⟩
1878 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
1879 ⟨/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`.

```
1880 \keys_define:nn { WithArrows / Arrow / code-after }
1881   {
1882     tikz     .code:n =
1883       \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
1884     tikz     .value_required:n  = true ,
1885     rr       .value_forbidden:n = true ,
1886     rr       .code:n            = \@@_fix_pos_option:n 0 ,
1887     ll       .value_forbidden:n = true,
1888     ll       .code:n            = \@@_fix_pos_option:n 1 ,
1889     rl       .value_forbidden:n = true ,
1890     rl       .code:n            = \@@_fix_pos_option:n 2 ,
1891     lr       .value_forbidden:n = true ,
1892     lr       .code:n            = \@@_fix_pos_option:n 3 ,
1893     v        .value_forbidden:n = true ,
1894     v        .code:n            = \@@_fix_pos_option:n 4 ,
1895     tikz-code .tl_set:N          = \l_@@_tikz_code_tl ,
1896     tikz-code .value_required:n  = true ,
1897     xoffset  .dim_set:N          = \l_@@_xoffset_dim ,
1898     xoffset  .value_required:n  = true ,
1899     unknown .code:n  =
1900       \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
1901       \@@_error:n { Unknown~option~Arrow~in~code-after }
1902   }
```

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.

```
1903 \seq_new:N \l_@@_options_Arrow_code_after_seq
1904 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq
1905   { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }
```

```
1906 ⟨*LaTeX⟩
1907 \NewDocumentCommand \@@_Arrow_code_after { O { } m m m ! O { } }
1908 ⟨/LaTeX⟩
1909 ⟨*plain-TeX⟩
1910 \cs_new_protected:Npn \@@_Arrow_code_after
1911   {
1912     \peek_meaning:NTF [
1913       { \@@_Arrow_code_after_i }
1914       { \@@_Arrow_code_after_i [ ] }
1915   }
1916 \cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4
```

```
1917      {
1918        \peek_meaning:NTF [
1919          { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1920          { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
1921      }
1922    \cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
1923  ⟨/plain-TeX⟩
1924      {
1925        \int_set:Nn \l_@@_pos_arrow_int 1
1926        \str_clear_new:N \l_@@_previous_key_str
1927        \group_begin:
1928          \keys_set:nn { WithArrows / Arrow / code-after }
1929            { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
1930          \bool_set_false:N \l_@@_initial_r_bool
1931          \bool_set_false:N \l_@@_final_r_bool
1932          \int_case:nn \l_@@_pos_arrow_int
1933            {
1934              0
1935                {
1936                  \bool_set_true:N \l_@@_initial_r_bool
1937                  \bool_set_true:N \l_@@_final_r_bool
1938                }
1939              2 { \bool_set_true:N \l_@@_initial_r_bool }
1940              3 { \bool_set_true:N \l_@@_final_r_bool }
1941            }
```

We prevent drawing an arrow from a line to itself.

```
1942          \tl_if_eq:nnTF { #2 } { #3 }
1943            { \@@_error:nn { Both~lines~are~equal } { #2 } }
```

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

```
1944            {
1945              \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - l }
1946                { \@@_error:nx { Wrong~line~in~Arrow } { #2 } }
1947                {
1948                  \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - l }
1949                    { \@@_error:nx { Wrong~line~in~Arrow } { #3 }  }
1950                    {
1951                      \int_compare:nNnTF \l_@@_pos_arrow_int = 4
1952                        {
1953                          \pgfpicture
1954                            \pgfrememberpicturepositiononpagetrue
1955                            \pgfpointanchor { wa - \l_@@_prefix_str - #2 - l }
1956                              { south }
1957                            \dim_set_eq:NN \l_tmpa_dim \pgf@x
1958                            \dim_set_eq:NN \l_tmpb_dim \pgf@y
1959                            \pgfpointanchor { wa - \l_@@_prefix_str - #3 - l }
1960                              { north }
1961                            \dim_set:Nn \l_tmpa_dim
1962                              { \dim_max:nn \l_tmpa_dim \pgf@x }
1963                            \tl_gset:Nx \g_tmpa_tl
1964                              { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1965                            \tl_gset:Nx \g_tmpb_tl
1966                              { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1967                          \endpgfpicture
1968                        }
1969                        {
1970                          \pgfpicture
1971                            \pgfrememberpicturepositiononpagetrue
1972                            \pgfpointanchor
1973                              {
1974                                wa - \l_@@_prefix_str -
1975                                #2 - \bool_if:NTF \l_@@_initial_r_bool r l
```

```
1976                        }
1977                      { south }
1978                    \tl_gset:Nx \g_tmpa_tl
1979                      { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1980                    \pgfpointanchor
1981                      {
1982                        wa - \l_@@_prefix_str -
1983                        #3 - \bool_if:NTF \l_@@_final_r_bool r l
1984                      }
1985                      { north }
1986                    \tl_gset:Nx \g_tmpb_tl
1987                      { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1988                  \endpgfpicture
1989                }
1990              \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
1991            }
1992        }
1993      }
1994    \group_end:
1995  }
```

12.12 The command \MultiArrow in code-after

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

```
1996 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
1997   {
```

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.

```
1998     \exp_args:Nnx
1999       \regex_match:nnTF
2000       { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
2001       { #1 }
2002       { \@@_MultiArrow_i:nn { #1 } { #2 } }
2003       { \@@_error:nx { Invalid~specification~for~MultiArrow } { #1 } }
2004   }
2005 \cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
2006   {
```

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

```
2007     \foreach \x in { #1 }
2008       {
2009         \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - l }
2010           { \@@_error:nx { Wrong~line~specification~in~MultiArrow } \x }
2011           { \clist_gput_right:Nx \g_tmpa_clist \x }
2012       }
```

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

```
2013     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
2014       { \@@_error:n { Too~small~specification~for~MultiArrow } }
2015       {
2016         \clist_sort:Nn \g_tmpa_clist
2017           {
2018             \int_compare:nTF { ##1 > ##2 }
2019               \sort_return_swapped:
2020               \sort_return_same:
2021           }
```

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

```
2022          \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...

```
2023          \clist_reverse:N \g_tmpa_clist
2024          \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 fonction 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).

```
2025          \exp_args:NV \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
2026    ⟨*LaTeX⟩
2027          \begin { tikzpicture }
2028    ⟨/LaTeX⟩
2029    ⟨*plain-TeX⟩
2030          \tikzpicture
2031    ⟨/plain-TeX⟩
2032            [
2033              @@_standard ,
2034              every~path / .style = { WithArrows / arrow }
2035            ]
2036          \draw [<->] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
2037                    -- ++(5mm,0)
2038                    -- node (@@_label) {}
2039                        ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
2040                    -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south)  ;
2041
2042          \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }
2043          \dim_set:Nn \l_tmpa_dim { 20 cm }
2044          \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
2045          \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
2046          \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
2047            {
2048              \dim_set:Nn \l_tmpb_dim
2049                { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
2050              \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
2051                { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
2052            }
2053          \path (@@_label.west)
2054            node [ anchor = west, text~width = \dim_use:N \l_tmpa_dim ] { #2 } ;
2055    ⟨*LaTeX⟩
2056          \end { tikzpicture }
2057    ⟨/LaTeX⟩
2058    ⟨*plain-TeX⟩
2059          \endtikzpicture
2060    ⟨/plain-TeX⟩
2061        }
2062    }
2063  \cs_new_protected:Npn \@@_MultiArrow_i:n #1
2064    {
2065    ⟨*LaTeX⟩
2066          \begin { tikzpicture }
2067    ⟨/LaTeX⟩
2068    ⟨*plain-TeX⟩
2069          \tikzpicture
2070    ⟨/plain-TeX⟩
2071        [
2072          @@_standard ,
2073          every~path / .style = { WithArrows / arrow }
```

```
2074          ]
2075       \foreach \k in { #1 }
2076          {
2077            \draw [ <- ]
2078               ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
2079          } ;
2080  ⟨*LaTeX⟩
2081          \end{tikzpicture}
2082  ⟨/LaTeX⟩
2083  ⟨*plain-TeX⟩
2084          \endtikzpicture
2085  ⟨/plain-TeX⟩
2086     }
```

12.13   The error messages of the package

```
2087  \str_new:N \l_witharrows_body_str
```

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

```
2088  \cs_new:Npn \@@_potential_body_i:
2089     {
2090       \str_if_empty:NF \l_witharrows_body_str
2091         { \\ If~you~want~to~see~the~body~of~the~environment,~type~H~<return>. }
2092     }
2093  \cs_new:Npn \@@_potential_body_ii:
2094     {
2095       \str_if_empty:NTF \l_nicematrix_body_str
2096         { No~further~help~available }
2097         {
2098           The~body~of~your~environment~was:\\
2099           \l_witharrows_body_str
2100         }
2101     }
2102  \str_const:Nn \c_@@_option_ignored_str
2103     { If~you~go~on,~this~option~will~be~ignored. }
2104  \str_const:Nn \c_@@_command_ignored_str
2105     { If~you~go~on,~this~command~will~be~ignored. }
2106  ⟨*LaTeX⟩
2107  \@@_msg_new:nn { amsmath~not~loaded }
2108     {
2109       You~can't~use~the~option~'\l_keys_key_str'~because~the~
2110       package~'amsmath'~has~not~been~loaded.\\
2111       If~you~go~on,~this~option~will~be~ignored~in~the~rest~
2112       of~the~document.
2113     }
2114  ⟨/LaTeX⟩
2115  \@@_msg_new:nn { Bad~value~for~replace~brace~by }
2116     {
2117       Bad~value~for~the~option~'\l_keys_key_str'.~The~value~must~begin~
2118       with~an~extensible~left~delimiter.~The~possible~values~are:~.,
2119       \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
2120       \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
2121       \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
2122       \token_to_str:N \lfloor\ and~\token_to_str:N \lceil\
2123       (and~\token_to_str:N \lvert\ and~\token_to_str:N \lVert\
2124       if~amsmath~or~unicode-math~is~loaded~in~LaTeX).\\
2125       \c_@@_option_ignored_str
2126     }
2127  \@@_msg_new:nn { option~of~cr~negative }
2128     {
2129       The~argument~of~the~command~\token_to_str:N\\~
```

```
2130      should~be~positive~in~the~row~\int_use:N \g_@@_line_int\
2131      of~your~environment~\{\l_@@_type_env_str\}.\\
2132      \c_@@_option_ignored_str
2133    }
2134  \@@_msg_new:nn { omit~probably~used }
2135    {
2136      There~is~a~problem.~Maybe~you~have~used~a~command~
2137      \token_to_str:N\omit\ in~the~line~\int_use:N \g_@@_line_int\
2138      (or~another~line)~of~your~environment~\{\l_@@_type_env_str\}.\\
2139      You~can~go~on~but~you~may~have~others~errors.
2140    }
2141  ⟨*LaTeX⟩
2142  \@@_msg_new:nnn { newline~at~the~end~of~env }
2143    {
2144      The~environments~of~witharrows~(\{WithArrows\}~and~
2145      \{DispWithArrows\})~should~not~end~by~\token_to_str:N \\.\\
2146      However,~you~can~go~on~for~this~time.~No~similar~error~will~be~
2147      raised~in~this~document.
2148      \@@_potential_body_i:
2149    }
2150    { \@@_potential_body_ii: }
2151  ⟨/LaTeX⟩
2152  \@@_msg_new:nnn { Invalid~option~format }
2153    {
2154      The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
2155      must~not~be~empty.\\
2156      \c_@@_option_ignored_str
2157      \@@_potential_body_i:
2158    }
2159    { \@@_potential_body_ii: }
2160  \@@_msg_new:nnn { invalid~key~o }
2161    {
2162      The~key~'o'~for~individual~arrows~can~be~used~only~in~mode~
2163      'group'~or~in~mode~'groups'.\\
2164      \c_@@_option_ignored_str
2165      \@@_potential_body_i:
2166    }
2167    { \@@_potential_body_ii: }
2168  \@@_msg_new:nnn { Value~for~a~key }
2169    {
2170      The~key~'\l_keys_key_str'~should~be~used~without~value. \\
2171      However,~you~can~go~on~for~this~time.
2172      \@@_potential_body_i:
2173    }
2174    { \@@_potential_body_ii: }
2175  \@@_msg_new:nnn { Unknown~option~in~Arrow }
2176    {
2177      The~key~'\l_keys_key_str'~is~unknown~for~the~command~
2178      \l_@@_string_Arrow_for_msg_str\ in~the~row~
2179      \int_use:N \g_@@_line_int\ of~your~environment~
2180      \{\l_@@_type_env_str\}. \l_tmpa_str \\
2181      \c_@@_option_ignored_str \\
2182      For~a~list~of~the~available~keys,~type~H~<return>.
2183    }
2184    {
2185      The~available~keys~are~(in~alphabetic~order):~
2186      \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
2187    }
2188  \@@_msg_new:nnn { Unknown~option~WithArrows }
2189    {
2190      The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
```

```
2191      \c_@@_option_ignored_str \\
2192      For~a~list~of~the~available~keys,~type~H~<return>.
2193    }
2194    {
2195      The~available~keys~are~(in~alphabetic~order):~
2196      \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.
2197    }
2198  \@@_msg_new:nnn { Unknown~option~DispWithArrows }
2199    {
2200      The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
2201      \c_@@_option_ignored_str \\
2202      For~a~list~of~the~available~keys,~type~H~<return>.
2203    }
2204    {
2205      The~available~keys~are~(in~alphabetic~order):~
2206      \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.
2207    }
2208  \@@_msg_new:nnn { Unknown~option~WithArrowsOptions }
2209    {
2210      The~key~'\l_keys_key_str'~is~unknown~in~
2211      \token_to_str:N \WithArrowsOptions. \\
2212      \c_@@_option_ignored_str \\
2213      For~a~list~of~the~available~keys,~type~H~<return>.
2214    }
2215    {
2216      The~available~keys~are~(in~alphabetic~order):~
2217      \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
2218    }
2219  \@@_msg_new:nnn { Unknown~option~Arrow~in~code-after }
2220    {
2221      The~key~'\l_keys_key_str'~is~unknown~in~
2222      \token_to_str:N \Arrow\ in~code-after. \\
2223      \c_@@_option_ignored_str \\
2224      For~a~list~of~the~available~keys,~type~H~<return>.
2225    }
2226    {
2227      The~available~keys~are~(in~alphabetic~order):~
2228      \seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2229    }
2230  \@@_msg_new:nnn { Too~much~columns~in~WithArrows }
2231    {
2232      Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
2233      \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2234      Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2235      If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2236      the~option~'more-columns'~at~a~global~level~or~for~an~environment. \\
2237      However,~you~can~go~one~for~this~time.
2238      \@@_potential_body_i:
2239    }
2240    { \@@_potential_body_ii: }
2241  \@@_msg_new:nnn { Too~much~columns~in~DispWithArrows }
2242    {
2243      Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
2244      \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2245      Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\
2246      at~the~end~of~row~\int_use:N \g_@@_line_int. \\
2247      This~error~is~fatal.
2248      \@@_potential_body_i:
2249    }
2250    { \@@_potential_body_ii: }
2251  \@@_msg_new:nn { Negative~jump }
```

```
2252      {
2253        You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
2254        \l_@@_string_Arrow_for_msg_str\
2255        in~the~row~\int_use:N \g_@@_line_int\
2256        of~your~environment~\{\l_@@_type_env_str\}.~
2257        You~can~create~an~arrow~going~backwards~with~the~option~'<-'~of~Tikz. \\
2258        \c_@@_option_ignored_str
2259      }
2260  \@@_msg_new:nn { new-group~without~groups }
2261      {
2262        You~can't~use~the~option~'new-group'~for~the~command~
2263        \l_@@_string_Arrow_for_msg_str\
2264        because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
2265        'groups'~in~your~environment~\{\l_@@_type_env_str\}. \\
2266        \c_@@_option_ignored_str
2267      }
2268  \@@_msg_new:nnn
2269      { Too~few~lines~for~an~arrow }
2270      {
2271        Line~\l_@@_input_line_str\
2272        :~an~arrow~specifiead~in~the~row~\int_use:N \l_@@_initial_int\
2273        of~your~environment~\{\l_@@_type_env_str\}~can't~be~drawn~
2274        because~it~arrives~after~the~last~row~of~the~environment. \\
2275        If~you~go~on,~this~arrow~will~be~ignored.
2276        \@@_potential_body_i:
2277      }
2278      { \@@_potential_body_ii: }
2279  \@@_msg_new:nn { o~arrow~with~no~arrow~under }
2280      {
2281        Line~\l_@@_input_line_str\
2282        :~there~is~no~arrow~'under'~your~arrow~of~type~'o'.\\
2283        If~you~go~on,~this~arrow~won't~be~drawn.
2284      }
2285  \@@_msg_new:nnn { WithArrows~outside~math~mode }
2286      {
2287        The~environment~\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
2288        like~the~environment~\{aligned\}~of~amsmath. \\
2289        Nevertheless,~you~can~go~on.
2290        \@@_potential_body_i:
2291      }
2292      { \@@_potential_body_ii: }
2293  \@@_msg_new:nnn { DispWithArrows~in~math~mode }
2294      {
2295        The~environment~\{\l_@@_type_env_str\}~should~be~used~only~outside~math~
2296        mode~like~the~environments~\{align\}~and~\{align*\}~of~amsmath. \\
2297        This~error~is~fatal.
2298        \@@_potential_body_i:
2299      }
2300      { \@@_potential_body_ii: }
2301  \@@_msg_new:nn { Incompatible~options~in~Arrow }
2302      {
2303        You~try~to~use~the~option~'\l_keys_key_str'~but~
2304        this~option~is~incompatible~or~redundant~with~the~option~
2305        '\l_@@_previous_key_str'~set~in~the~same~command~
2306        \l_@@_string_Arrow_for_msg_str. \\
2307        \c_@@_option_ignored_str
2308      }
2309  \@@_msg_new:nn { Incompatible~options }
2310      { You~try~to~use~the~option~'\l_keys_key_str'~but~
2311        this~option~is~incompatible~or~redundant~with~the~option~
2312        '\l_@@_previous_key_str'~set~in~the~same~command~
```

```
2313    \bool_if:NT \l_@@_in_code_after_bool
2314      {
2315        \l_@@_string_Arrow_for_msg_str\
2316        in~the~code-after~of~your~environment~\{\l_@@_type_env_str\}
2317      }. \\
2318    \c_@@_option_ignored_str
2319    }
2320  \@@_msg_new:nnn { Arrow~not~in~last~column }
2321    {
2322      You~should~use~the~command~\l_@@_string_Arrow_for_msg_str\
2323      only~in~the~last~column~(column~\int_use:N\l_@@_nb_cols_int)~
2324      of~your~environment~\{\l_@@_type_env_str\}.\\
2325      However~you~can~go~on~for~this~time.
2326      \@@_potential_body_i:
2327    }
2328    { \@@_potential_body_ii: }
2329  \@@_msg_new:nn { Wrong~line~in~Arrow }
2330    {
2331      The~specification~of~line~'#1'~you~use~in~the~command~
2332      \l_@@_string_Arrow_for_msg_str\
2333      in~the~'code-after'~of~\{\l_@@_type_env_str\}~doesn't~exist. \\
2334      \c_@@_option_ignored_str
2335    }
2336  \@@_msg_new:nn { Both~lines~are~equal }
2337    {
2338      In~the~'code-after'~of~\{\l_@@_type_env_str\}~you~try~to~
2339      draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\
2340      \c_@@_option_ignored_str
2341    }
2342  \@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }
2343    {
2344      The~specification~of~line~'#1'~doesn't~exist. \\
2345      If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
2346    }
2347  \@@_msg_new:nn { Too~small~specification~for~MultiArrow }
2348    {
2349      The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
2350      is~too~small:~you~need~at~least~two~lines. \\
2351      \c_@@_command_ignored_str
2352    }
2353  \@@_msg_new:nn { Not~allowed~in~DispWithArrows }
2354    {
2355      The~command~\token_to_str:N #1
2356      is~allowed~only~in~the~last~column~
2357      (column~\int_use:N\l_@@_nb_cols_int)~of~\{\l_@@_type_env_str\}. \\
2358      \c_@@_option_ignored_str
2359    }
2360  \@@_msg_new:nn { Not~allowed~in~WithArrows }
2361    {
2362      The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
2363      (it's~allowed~in~the~last~column~of~\{DispWithArrows\}). \\
2364      \c_@@_option_ignored_str
2365    }
2366  ⟨*LaTeX⟩
2367  \@@_msg_new:nn { tag*~without~amsmath }
2368    {
2369      We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2370      (or~mathtools). \\
2371      If~you~go~on,~the~command~\token_to_str:N\tag\
2372      will~be~used~instead.
```

88

```
2373     }
2374  \@@_msg_new:nn { Multiple~tags }
2375    {
2376      You~can't~use~twice~the~command~\token_to_str:N\tag\
2377      in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2378      If~you~go~on,~the~tag~'#1'~will~be~used.
2379    }
2380  \@@_msg_new:nn { Multiple~labels }
2381    {
2382      Normally,~we~can't~use~the~command~\token_to_str:N\label\
2383      twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2384      However,~you~can~go~on.~
2385      \bool_if:NT \c_@@_showlabels_loaded_bool
2386        { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
2387      If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
2388      'allow-multiple-labels'~at~the~global~or~environment~level.
2389    }
2390  \@@_msg_new:nn { Multiple~labels~with~cleveref }
2391    {
2392      Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
2393      twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2394      If~you~go~on,~you~may~have~undefined~references.
2395    }
2396  ⟨/LaTeX⟩
2397  \@@_msg_new:nn { Inexistent~v-node }
2398    {
2399      There~is~a~problem.~Maybe~you~have~put~a~command~\token_to_str:N\cr\
2400      instead~of~a~command~\token_to_str:N\\~at~the~end~of~
2401      the~row~\l_tmpa_int\
2402      of~your~environment~\{\l_@@_type_env_str\}. \\
2403      This~error~is~fatal.
2404    }
```

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).

```
2405  \@@_msg_new:nn { Option~xoffset~forbidden }
2406    {
2407      You~can't~use~the~option~'xoffset'~in~the~command~
2408      \l_@@_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g_@@_line_int\
2409      of~your~environment~\{\l_@@_type_env_str\}~
2410      because~you~are~using~the~option~
2411      ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
2412          { group }
2413          { groups } '.~It's~possible~for~an~independent~arrow~or~if~there~is~
2414      only~one~arrow. \\
2415      \c_@@_option_ignored_str
2416    }
2417  \@@_msg_new:nnn { Duplicate~name }
2418    {
2419      The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2420      the~same~environment~name~twice.~You~can~go~on,~but,~
2421      maybe,~you~will~have~incorrect~results. \\
2422      For~a~list~of~the~names~already~used,~type~H~<return>. \\
2423      If~you~don't~want~to~see~this~message~again,~use~the~option~
2424      'allow-duplicate-names'.
2425    }
2426    {
2427      The~names~already~defined~in~this~document~are:~
2428      \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2429    }
```

```
2430   \@@_msg_new:nn { Invalid~specification~for~MultiArrow }
2431     {
2432       The~specification~of~rows~for~\token_to_str:N\MultiArrow\
2433       (i.e.~#1)~is~invalid. \\
2434       \c_@@_command_ignored_str
2435     }
```

12.14   The command \WithArrowsNewStyle

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

```
2436   ⟨*LaTeX⟩
2437   \NewDocumentCommand \WithArrowsNewStyle { m m }
2438   ⟨/LaTeX⟩
2439   ⟨*plain-TeX⟩
2440   \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2441   ⟨/plain-TeX⟩
2442     {
2443       \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2444         { \@@_error:nn { Key~already~defined } { #1 } }
2445         {
2446           \keys_define:nn { WithArrows / Global }
2447             {
2448               #1 .code:n =
2449                 { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
2450             }
2451           \seq_put_right:Nx \l_@@_options_WithArrows_seq { \tl_to_str:n { #1 } }
2452           \seq_put_right:Nx \l_@@_options_DispWithArrows_seq
2453             { \tl_to_str:n { #1 } }
2454           \seq_put_right:Nx \l_@@_options_WithArrowsOptions_seq
2455             { \tl_to_str:N { #1 } }
```

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

```
2456           \group_begin:
2457             \msg_set:nnn { witharrows } { Unknown~option~WithArrowsOptions }
2458               {
2459                 The~key~'\l_keys_key_str'~can't~be~set~in~the~
2460                 definition~of~a~style.~You~can~go~on~for~this~time~
2461                 but~you~should~suppress~this~key.
2462               }
2463             \WithArrowsOptions { #2 }
2464           \group_end:
2465         }
2466     }
2467   \@@_msg_new:nn { Key~already~defined }
2468     {
2469       The~key~'#1'~is~already~defined. \\
2470       If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2471       will~be~ignored.
2472     }
```

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`.

```
2473  \keys_define:nn { WithArrows / up-and-down }
2474    {
2475      radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
2476      radius .value_required:n = true ,
2477      width .code:n =
2478        \str_case:nnF { #1 }
2479          {
2480            { min } { \dim_zero:N \l_@@_arrow_width_dim }
2481            { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
2482          }
2483          { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
2484      width .value_required:n = true ,
2485      unknown .code:n = \@@_error:n { Option~unknown~for~up-and-down }
2486    }
2487  \@@_msg_new:nn { Option~unknown~for~up-and-down }
2488    {
2489      The~option~'\l_keys_key_str'~is~unknown.~\c_@@_option_ignored_str
2490    }
```

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

```
2491  ⟨*LaTeX⟩
2492  \tl_const:Nn \c_@@_tikz_code_up_tl
2493    {
```

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

```
2494      \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2495        {
2496          \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2497            let \p1 = ( #1 ) , \p2 = ( #2 )
2498            in (\p1) -- node
2499              {
2500                \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2501                \begin { varwidth } \l_tmpa_dim
```

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

```
2502                  \narrowragged
2503                  #3
2504                \end { varwidth }
2505              }
2506            (\x2,\y1) -- (\p2) ;
2507        }
```

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`.

```
2508        {
2509          \path
2510            let \p1 = ( #1 ) , \p2 = ( #2 )
2511            in node
2512              {
```

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

```
2513                \dim_set:Nn \l_tmpa_dim
2514                  { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
2515                \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
```

91

```
2516                    {
2517                      \dim_set:Nn \l_tmpa_dim
2518                        { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
2519                    }
```

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.

```
2520                    \hbox_gset:Nn \g_tmpa_box
2521                      {
2522                        \begin { varwidth } \l_tmpa_dim
2523                          \narrowragged
2524                          #3
2525                        \end { varwidth }
2526                      }
```

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

```
2527                    \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2528                      { \dim_gset_eq:NN  \g_tmpa_dim \l_@@_arrow_width_dim }
2529                      { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2530                    \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2531                  } ;
2532          \draw
2533            let \p1 = ( #1 ) , \p2 = ( #2 )
2534            in (\x2-\g_tmpa_dim,\y1)
2535              -- node { \box_use:N \g_tmpa_box }
2536              (\x2-\l_@@_up_and_down_radius_dim,\y1)
2537              [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2538              -| (\p2) ;
2539        }
2540    }
2541 ⟨/LaTeX⟩
2542 ⟨*plain-TeX⟩
2543 \tl_const:Nn \c_@@_tikz_code_up_tl
2544   {
2545     \dim_case:nnF \l_@@_arrow_width_dim
2546       {
2547         \c_max_dim
2548           {
2549             \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2550               let \p1 = ( #1 ) , \p2 = ( #2 )
2551               in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
2552           }
2553         \c_zero_dim
2554           {
2555             \path node
2556               {
2557                 \hbox_gset:Nn \g_tmpa_box { #3 }
2558                 \dim_gset:Nn \g_tmpa_dim
2559                   { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2560               } ;
2561             \draw
2562               let \p1 = ( #1 ) , \p2 = ( #2 )
2563               in (\x2-\g_tmpa_dim,\y1)
2564                 -- node { \box_use:N \g_tmpa_box }
2565                 (\x2-\l_@@_up_and_down_radius_dim,\y1)
2566                 [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2567                 -| (\p2) ;
2568           }
2569       }
2570       {
2571         \draw
2572           let \p1 = ( #1 ) , \p2 = ( #2 )
2573           in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
2574             -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
```

```
2575            [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2576            -| (\p2) ;
2577        }
2578    }
2579 ⟨/plain-TeX⟩
```

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

```
2580 ⟨*LaTeX⟩
2581 \tl_const:Nn \c_@@_tikz_code_down_tl
2582    {
2583        \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2584            {
2585                \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2586                    let \p1 = ( #1 ) , \p2 = ( #2 )
2587                    in (\p1) -- (\x1,\y2) -- node
2588                        {
2589                            \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2590                            \begin { varwidth } \l_tmpa_dim
2591                                \narrowragged
2592                                #3
2593                            \end { varwidth }
2594                        }
2595                    (\p2) ;
2596            }
2597            {
2598                \path
2599                    let \p1 = ( #1 ) , \p2 = ( #2 )
2600                    in node
2601                        {
2602                            \hbox_gset:Nn \g_tmpa_box
2603                                {
2604                                    \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.).

```
2605                                    { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
2606                                \begin { varwidth } \l_tmpa_dim
2607                                    \narrowragged
2608                                    #3
2609                                \end { varwidth }
2610                            }
2611                        \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2612                            { \dim_gset_eq:NN  \g_tmpa_dim \l_@@_arrow_width_dim }
2613                            { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2614                        \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2615                        } ;
2616
2617            \draw
2618                let \p1 = ( #1 ) , \p2 = ( #2 )
2619                in (\p1)
2620                    { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2621                    -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2622                    -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2623                    -- ++ (-2mm,0) ;
2624        }
2625    }
2626 ⟨/LaTeX⟩
2627 %
2628 ⟨*plain-TeX⟩
2629 \tl_const:Nn \c_@@_tikz_code_down_tl
2630    {
2631        \dim_case:nnF \l_@@_arrow_width_dim
```

```
2632          {
2633            \c_max_dim
2634              {
2635                \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2636                  let \p1 = ( #1 ) , \p2 = ( #2 )
2637                  in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
2638              }
2639            \c_zero_dim
2640              {
2641                \path node
2642                  {
2643                    \hbox_gset:Nn \g_tmpa_box { #3 }
2644                    \dim_gset:Nn \g_tmpa_dim
2645                      { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2646                  } ;
2647                \draw
2648                  let \p1 = ( #1 ) , \p2 = ( #2 )
2649                  in (\p1)
2650                    { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2651                    -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2652                    -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2653                    -- ++ (-2mm,0) ;
2654              }
2655          }
2656          {
2657            \draw
2658              let \p1 = ( #1 ) , \p2 = ( #2 )
2659              in (\p1)
2660                { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2661                -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2662                -- node { #3 }
2663                   (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
2664                -- ++ (-2mm,0) ;
2665          }
2666      }
2667  ⟨/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.

```
2668  \keys_define:nn { WithArrows / Arrow / FirstPass }
2669    {
2670      up   .code:n = \@@_set_independent_bis: ,
2671      down .code:n = \@@_set_independent_bis: ,
2672      up    .default:n = NoValue ,
2673      down .default:n = NoValue
2674    }
```

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

```
2675  \keys_define:nn { WithArrows / Arrow / SecondPass }
2676    {
2677      up .code:n =
2678        \str_if_empty:NT \l_@@_previous_key_str
2679          {
2680            \str_set:Nn \l_@@_previous_key_str { up }
2681            \cs_if_exist:cTF { tikz@library@calc@loaded }
2682              {
2683                \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2684                \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.

```
2685                    \bool_set_false:N \l_@@_wrap_lines_bool
```
The main action occurs now. We change the value of the tikz-code.
```
2686                    \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
2687                  }
2688                  { \@@_error:n { calc~not~loaded } }
2689            } ,
2690        down .code:n =
2691          \str_if_empty:NT \l_@@_previous_key_str
2692            {
2693              \str_set:Nn \l_@@_previous_key_str { down }
2694              \cs_if_exist:cTF { tikz@library@calc@loaded }
2695                {
2696                  \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2697                  \int_set:Nn \l_@@_pos_arrow_int 1
2698                  \bool_set_false:N \l_@@_wrap_lines_bool
2699                  \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
2700                }
2701                { \@@_error:n { calc~not~loaded } }
2702          }
2703      }
2704  \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
2705  \seq_put_right:Nn \l_@@_options_Arrow_seq { up }
2706  \@@_msg_new:nn { calc~not~loaded }
2707    {
2708      You~can't~use~the~option~'\l_keys_key_str'~because~you~don't~have~loaded~the~
2709      Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
2710      ~in~the~preamble~of~your~document. \\
2711      \c_@@_option_ignored_str
2712    }
2713  ⟨*plain-TeX⟩
2714  \catcode `\@ = 12
2715  \ExplSyntaxOff
2716  ⟨/plain-TeX⟩
```

13   History

The successive versions of the file `witharrows.sty` provided by TeXLive are available on the SVN server of TeXLive:

https:www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/witharrows/witharrows.sty

Changes between versions 1.0 and 1.1

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

Changes between versions 1.1 and 1.2

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

Changes between versions 1.2 and 1.3

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

Changes between versions 1.3 and 1.4

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

Changes between versions 1.4 and 1.5

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

Changes between versions 1.5 and 1.6

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

Changes between 1.6 and 1.7

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

Changes between 1.7 and 1.8

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

Changes between 1.8 and 1.9

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

Changes between 1.9 and 1.10

If the option `wrap-lines` is used, the option "`text width`" of Tikz is still active: if the value given to "`text width`" is lower than the width computed by `wrap-lines`, this value is used to wrap the lines.
The option `wrap-lines` is now fully compatible with the class option `leqno`.
Correction of a bug: `\nointerlineskip` and `\makebox[.6\linewidth]{}` should be inserted in `{DispWithArrows}` only in vertical mode.

Changes between 1.10 and 1.11

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

Changes between 1.11 and 1.12

New command `\tagnextline`.
New option `tagged-lines`.
An option of position (`ll`, `lr`, `rl`, `rr` or `i`) is now allowed at the local level even if the option `group` or the option `groups` is used at the global or environment level.
Compatibility of `{DispWithArrows}` with `\qedhere` of amsthm.
Compatibility with the packages refcheck, showlabels and listlbls.
The option `\AllowLineWithoutAmpersand` is deprecated because lines without ampersands are now always allowed.

Changes between 1.12 and 1.13

Options `start-adjust`, `end-adjust` and `adjust`.
This version is not stricty compatible with previous ones. To restore the behaviour of the previous versions, one has to use the option `adjust` with the value 0 pt:

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

Changes between 1.13 and 1.14

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

Replacement of some options `O { }` in commands and environments defined with `xparse` by `! O { }` (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`).

Modification of the code of `\WithArrowsNewStyle` following a correction of a bug in l3keys in the version of l3kernel of 2019/01/28.
New error message `Inexistent~v-node` to avoid a `pgf` error.
The error `Option incompatible with 'group(s)'` was suppressed in the version 1.12 but this was a mistake since this error is used with the option `xoffset` at the local level. The error is put back.

Changes between 1.14 and 1.15

Option `new-group` to start a new group of arrows (only available when the environment is composed with the option `groups`).
Tikz externalization is now deactivated in the environments of the extension `witharrows`.[43]

Changes between 1.15 and 1.16

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

Changes between 1.16 and 1.17

Option `format`.

Changes between 1.17 and 1.18

New option `<...>` for `{DispWithArrows}`.
Option `subequations`.
Warning when `{WithArrows}` or `{DispWithArrows}` ends by `\\`.
No space before an environment `{DispWithArrows}` if we are at the beginning of a `{minipage}`.

Changes between 1.18 and 2.0

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

Changes between 2.0 and 2.1

Option `max-length-of-arrow`.
Validation with regular expression for the first argument of `\MultiArrow`.

Changes between 2.1 and 2.2

Addition of `\normalbaselines` at the beginning of `\@@_post_halign:`.
The warning for an environment ending by `\\` has been transformed in `error`.

Changes between 2.2 and 2.3

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

---

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

Changes between 2.3 and 2.4

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

Changes between 2.4 and 2.5

Arrows of type `o` which are *over* other arrows.
`witharrows` now requires and loads `varwidth`

Changes between 2.5 and 2.5.1

Correction of the erroneous programmation of the nodes aliases.

Changes between 2.5.1 and 2.6

The key `format` now supports the letters R, C and L.

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 `expl3`.

Index

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

103

Contents