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

F. Pantigny
fpantigny@wanadoo.fr

October 19, 2024

Abstract

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

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

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

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

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

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

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

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

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 two environments {DispWithArrows} and {DispWithArrows*} which are similar to the environments {align} and {align*} of the package amsmath: cf. p. 17.

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

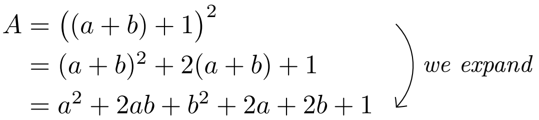
[1] If you use Overleaf, Overleaf will do automatically a number compilations sufficient (by using latexmk).

1 Options for the shape of the arrows

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

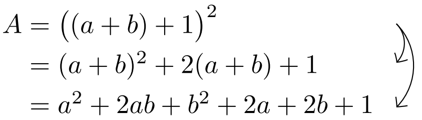
The option `jump` gives the number[2] of rows the arrow must jump (the default value is, of course, 1).

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


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

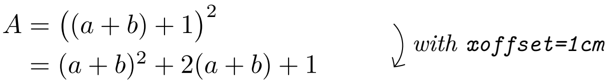
It's possible to put several arrows starting from the same row.

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


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

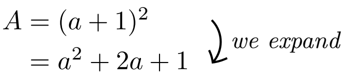
The option `xoffset` shifts the arrow to the right (we usually don't want the arrows to be stucked on the text). The initial value of `xoffset` is 3 mm.

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



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.

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



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

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

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

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

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

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

$$\begin{aligned} A &= (a+1)^2 \\ &= a^2+2a+1 \end{aligned} \qquad \textit{very classical}$$

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

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

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

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

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

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

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

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

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

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

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

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 provides an option `font` to modify the font of the whole text. Nevertheless, if we use the option `tikz={font={\bfseries}}`, the default specification of `\small` and `\itshape` will be overwritten.

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

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

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]

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

$$\begin{aligned} A &= \bigl((a+b)+1\bigr)^2 && \Big) \textit{first expansion.} \\ &= (a+b)^2+2(a+b)+1 && \Big) \textit{second expansion.} \\ &= a^2+2ab+b^2+2a+2b+1 \end{aligned}$$

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

Without the option `displaystyle`:

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

$$\begin{aligned} \textstyle\int_0^1 (x+1)^2 dx &= \textstyle\int_0^1 (x^2+2x+1) dx && \Big) \textit{linearity of integration} \\ &= \textstyle\int_0^1 x^2 dx + 2\int_0^1 x dx + \int_0^1 dx \\ &= \tfrac{1}{3} + 2\tfrac{1}{2} + 1 \\ &= \tfrac{7}{3} \end{aligned}$$

[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.

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

The same example with the option `displaystyle`:

$$\begin{aligned}\int_0^1 (x+1)^2dx &= \int_0^1 (x^2+2x+1)dx \\ &= \int_0^1 x^2dx + 2\int_0^1 xdx + \int_0^1 dx \quad \Big\downarrow \textit{ linearity of integration} \\ &= \frac{1}{3} + 2\frac{1}{2} + 1 \\ &= \frac{7}{3}\end{aligned}$$

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

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

$$\begin{aligned}\sum_{i=1}^n (x_i+1)^2 &= \sum_{i=1}^n (x_i^2+2x_i+1) \\ &= \sum_{i=1}^n x_i^2 + 2\sum_{i=1}^n x_i + n \quad \Big\downarrow \textit{ by linearity}\end{aligned}$$

The command `\Arrow` is recognized only in the environments `{WithArrows}`. If we have a command `\Arrow` previously defined, it's possible to go on using it outside the environments `{WithArrows}`.

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

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

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

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

[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 = {}}}`.

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

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`: the latter two commands are described in the section concerning the nested environments, p. 14.

2 Numbers of columns

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

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

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

$$\begin{array}{l} f(x) \ge g(x) \\ f(x)^2 \ge g(x)^2 \\ f(x)^2 - g(x)^2 \ge 0 \end{array} \quad \begin{array}{l} \big\downarrow \textit{ by squaring both sides} \\ \big\downarrow \textit{ by moving to left side} \end{array}$$

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

```
\begin{DispWithArrows*}[format = cCcCc,
                        wrap-lines,
                        interline=1mm]
k & \;\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*}
```

$$\begin{array}{ccccc} k & \le & t & \le & k+1 \\ \frac{1}{k+1} & \le & \frac{1}{t} & \le & \frac{1}{k} \\ \int\limits_k^{k+1} \frac{dt}{k+1} & \le & \int\limits_k^{k+1} \frac{dt}{t} & \le & \int\limits_k^{k+1} \frac{dt}{k} \\ \frac{1}{k+1} & \le & \ln(k+1)-\ln(k) & \le & \frac{1}{k} \end{array} \quad \Big\downarrow \begin{array}{l} \textit{we can integrate the} \\ \textit{inequalities since } k \leq k+1 \end{array}$$

3 Precise positioning of the arrows

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

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

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

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

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

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

This arrow uses the `lr` *option.*

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

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

[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.

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

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

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

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

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

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

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

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

[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 arrow will be called *independent* in the technical documentation

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

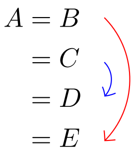
$$\begin{aligned}
\sum_{k=0}^{n} \frac{\cos kx}{\cos^k x} &= \sum_{k=0}^{n} \frac{\Re(e^{ikx})}{(\cos x)^k} \\
&\qquad\qquad \big\downarrow\ (\cos x)^k \textit{ is real} \\
&= \sum_{k=0}^{n} \Re\left(\frac{e^{ikx}}{(\cos x)^k}\right) \\
&\qquad\qquad \big\downarrow\ \Re(z+z') = \Re(z) + \Re(z') \\
&=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

4 The option "o" for individual arrows

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

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

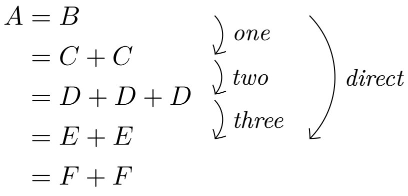
In the exemple on the right, the red arrow covers the blue one.



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]

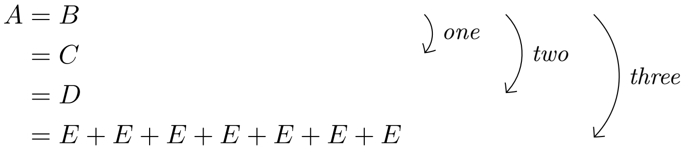
```
$\begin{WithArrows}[groups]
A & = B      \Arrow{one}\Arrow[o,jump=3]{direct} \\
  & = C + C \Arrow{two} \\
  & = D + D + D \Arrow{three} \\
  & = E + E \\
  & = F + F
\end{WithArrows}$
```



[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`.

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

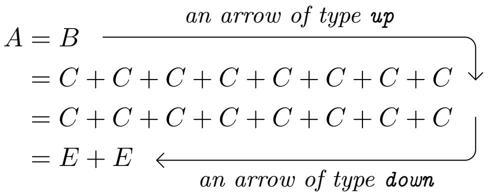


The horizontal space between an arrow of type `o` and the arrows immediately covered is fixed by the dimension `xoffset-for-o-arrows` which can be set with the command `\WithArrowsOptions` (initial value: 2 mm).

5 The options "up" and "down" for individual arrows

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

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


an arrow of type up
$A = B$
$= C + C + C + C + C + C + C + C$
$= C + C + C + C + C + C + C + C$
$= E + E$
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.

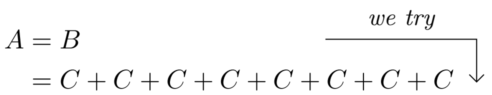
In fact, the options `up` and `down` may be used with a value which is a list of couples key-value.

- 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:

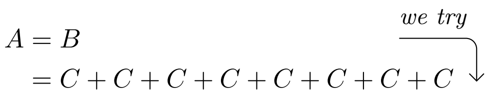
[10] The initial value of this parameter is 4 pt, which is the default value of the "`rounded corners`" of Tikz.

- with the value `max`, the width of the arrow is 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);
- 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.

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



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



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.

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

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

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

$$\begin{aligned} F &= \frac{1}{2}G \\ & \qquad\qquad\quad \Big\downarrow \text{ we expand} \\ &= H+\frac{1}{2}K \\ & \qquad\qquad\quad \Big\downarrow \text{ we go on} \\ &= K \end{aligned}$$

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}$
```

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

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.

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

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

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 $\left\{ \begin{aligned} f(x) &= 3x^3+2x^2-x+4 \\ g(x) &= 5x^2-5x+6 \end{aligned} \right. \quad \textit{both are polynoms}$

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

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

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

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

7 Arrows in nested environments

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

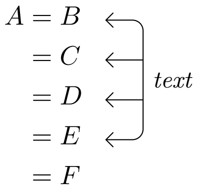
The options allowed for a command `\Arrow` in `code-after` are: `ll`, `lr`, `rl`, `rr`, `v`, `xoffset`, `tikz` and `tikz-code`. Except `v`, which is specific to `\Arrow` in `code-after`, all these options have their usual meaning.
With the option `v`, the arrow drawn is vertical to an abscissa computed with the start row and the end row only : the intermediate lines are not taken into account unlike with the option `i`. Currently, the option `i` is not available for the command `\Arrow` in `code-after`. However, it's always possible to translate an arrow with `xoffset` (or `xshift` of Tikz).

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

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

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

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

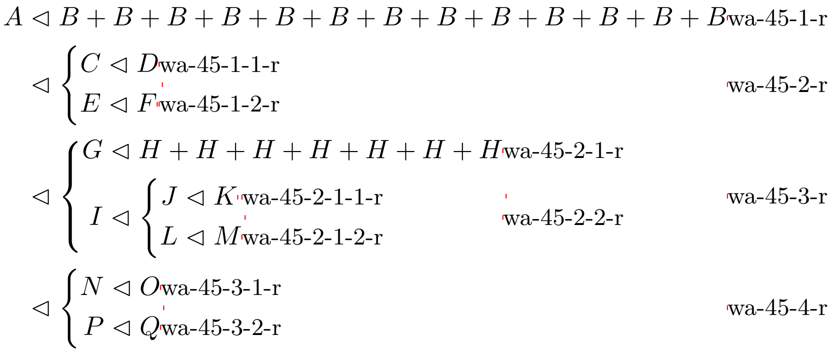


As of now, there is no option available for the command `\MultiArrow` (maybe in a future release).

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.[13]



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

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

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

[14] 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}
```

$$\begin{aligned}
A &\lhd B+B+B+B+B+B+B+B+B+B+B+B+B \\
&\lhd \begin{cases} C \lhd D \\ E \lhd F \end{cases} \\
&\lhd \begin{cases} G \lhd H+H+H+H+H+H+H \\ I \lhd \begin{cases} J \lhd K \\ L \lhd M \end{cases} \end{cases} \\
&\lhd \begin{cases} N \lhd O \\ P \lhd Q \end{cases}
\end{aligned}$$

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

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

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

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

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

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

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

9 The environment {DispWithArrows}

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

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

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

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

It's possible to use the command `\notag` (or `\nonumber`) to suppress a tag.
It's possible to use the command `\tag` to put a special tag (e.g. $\star$).
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}
```

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

A link to the equation ($\star$).[15]

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.[16]

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

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

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.[17]

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

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

[15] In this document, the references have been customized with `\labelformat{equation}{(#1)}` in the preamble.
[16] Notice that the environment `{DispWithArrows}` is compatible with the command `\qedhere` of amsthm.
[17] Even in this case, it's possible to put a "manual tag" with the command `\tag`.

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

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

$$\begin{aligned} A &= A_1 \\ &= A_2 \\ &= A_3 \end{aligned} \qquad \begin{aligned} &\curvearrowright \textit{first stage} \\ &\curvearrowright \textit{second stage} \end{aligned} \tag{3}$$

With the option `fleqn`, the environment is composed flush left (in a way similar to the option `fleqn` of the standard classes of LaTeX). In this case, the left margin can be controlled with the option `mathindent` (with a name inspired by the parameter `\mathindent` of standard LaTeX. The initial value of this parameter is 25 pt. It's possible to use as value for that key a *skip* (=*glue*).

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

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

$$\curvearrowright \textit{we expand}$$

$$= a^2 + 2a + 1 \tag{5}$$

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*}`.[18]

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.
```

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

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

First environment.

$$\begin{aligned} A &= B && \text{(6a)} \\ &= C && \text{(6b)} \end{aligned}$$

Second environment.

$$\begin{aligned} D &= E && \text{(7a)} \\ &= F && \text{(7b)} \end{aligned}$$

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}
```

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

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

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

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

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.[19]

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

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

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

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.

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

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

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

9.1 The option <...> of DispWithArrows

The environment `{DispWithArrows}` provides an option `left-brace`. When present, the value of this option is composed on the left, followed by a curly brace (hence the name) and the body of the environment.[21]
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.[22]

```
\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}
```

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

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}
```

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

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

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

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

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 a variant of the command `\ref` which does not create interactive link (even when hyperref is loaded).

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

10 Advanced features

10.1 Utilisation with Beamer

New 2.9

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

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

10.2 Use with plain-TeX

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

```
\input{witharrows}
```

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

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

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

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

The option `tikz-code` allows the user to change the shape of the arrows.[23]

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

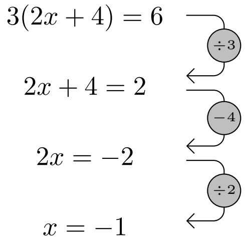
The value of this option must be a valid Tikz drawing instruction (with the final semicolon) with three markers `#1`, `#2` and `#3` for the start point, the end point and the label of the arrow.

The initial value is the following:

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

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

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



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

10.4 The command \WithArrowsNewStyle

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

The command `\WithArrowsNewStyle` takes two mandatory arguments. The first is the name of the style and the second is a list of key-value pairs. The scope of the definition done by `\WithArrowsNewStyle` is the current TeX scope.[24]

[23] 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}`.

[24] We recall that, in particular, every LaTeX environment is a TeX group.

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

For an example of use, see p. 30.

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

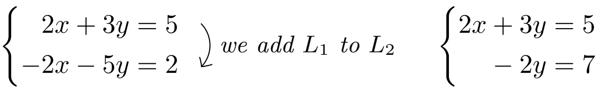
```
\newcommand{\ThickArrow}{\Arrow[tikz=thick]}
```

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

10.5 The key right-overlap

The key `right-overlap` is a boolean key whose initial value is `true`. It deals with the environments `{WithArrows}` only.
When the key `right-overlap` is in force, the arrows (and their labels) are drawn in an overlapping position and are not relevant for the computation of the dimensions of the TeX box containing the environment `{WithArrows}`.
When the key `right-overlap` is set to `false` (with `\WithArrowsOptions` or whithin an individual environment `{WithArrows}`), the overlapping on the right is taken into account in the dimensions of the encompassing box.

```
$\left\{\begin{WithArrows}[c,format = rCrCl,right-overlap=false]
2x & + & 3y & = & 5 \Arrow{we add $L_1$ to $L_2$}\\
-2x & - & 5y & = & 2
\end{WithArrows}\right.$\quad
$\left\{\begin{WithArrows}[c,format = rCrCl]
2x & + & 3y & = & 5 \\
   & - & 2y & = & 7
\end{WithArrows}\right.$
```



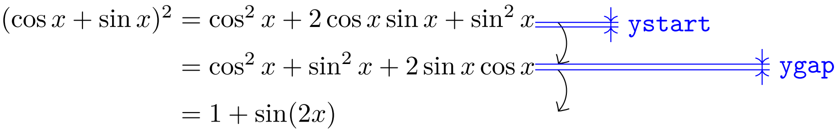
The tuning `right-overlap = false` may also be useful in conjunction with the class `standalone`.

10.6 Vertical positioning of the arrows

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

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

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

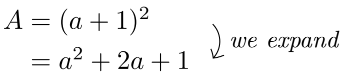


$(\cos x + \sin x)^2 = \cos^2 x + 2\cos x \sin x + \sin^2 x$
$= \cos^2 x + \sin^2 x + 2\sin x \cos x$
$= 1 + \sin(2x)$
ystart
ygap

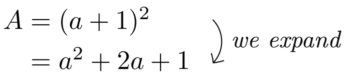
However, for aesthetic reasons, when it's possible, witharrows starts the arrow a bit higher (by an amount `start-adjust`) and ends the arrow a bit lower (by an amount `end-adjust`). By default, both parameters `start-adjust` and `end-adjust` are equal to 0.4 ex.

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

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



Here is the standard behaviour (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.



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

An arrow of `jump` equal to 1 has a maximal length[25] 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.

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

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

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

10.7 Footnotes in the environments of witharrows

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

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

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

If witharrows is loaded with the option `footnotehyper`, the package footnotehyper is loaded (if it is not yet loaded) 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:

$$\begin{aligned} A &= (a+b)^2 \\ &= a^2+b^2+2ab \end{aligned} \quad \textit{We expand}[26]$$

10.8 Option no-arrows

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

10.9 Note for the users of AUCTeX

In a editor of text with a LaTeX-oriented mode, the environments `{DispWithArrows}` and `{DispWithArrows*}` should be 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]`

[26] A footnote.

10.10 Note for the developpers

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

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

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

```
\NewDocumentEnvironment {DWA} {} {\DispWithArrows}{\endDispWithArrows}
```

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

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

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

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

11 Examples

11.1 \MoveEqLeft

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

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

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

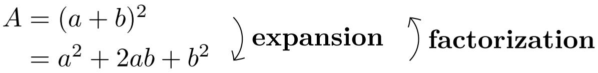
11.2 A command \DoubleArrow

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

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

Example of use:

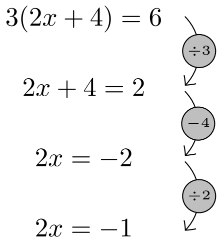
```
$\begin{WithArrows}[groups]
A & = (a+b)^2 \DoubleArrow[tikz={font=\bfseries}]{expansion}{factorization} \\
  & = a^2 + 2ab+b^2
\end{WithArrows}$
```



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.[27]

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.

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

11.4.1 Example 1

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

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

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

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

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

exp *is a morphism for* × *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
  & = \frac{1}{n} \sum_{k=0}^{n-1}\cos\bigl(\tfrac{\pi}{2}\cdot\tfrac{k}{n}\bigr)
  \Arrow{$\cos x = \Re(e^{ix})$}\\
...........
```

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

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

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

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

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

$\exp$ *is a morphism for* × *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 $e^{i\frac{2\pi}{n}}$

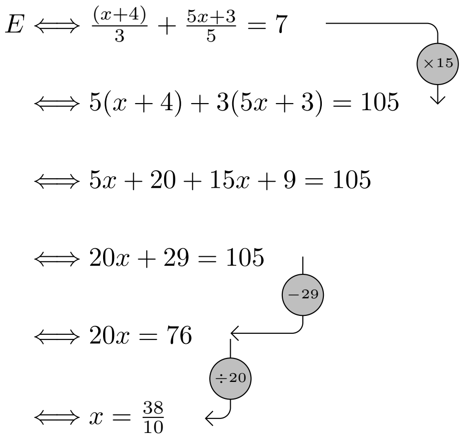
$$= \frac{1}{n}\Re\left(\frac{1-\left(e^{i\frac{\pi}{2n}}\right)^n}{1-e^{i\frac{\pi}{2n}}}\right) \tag{22}$$

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

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}
```


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

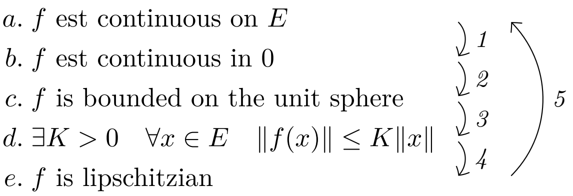
11.5 Automatic numbered loop

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 L3) and `\foreach` of pgffor (which is loaded when witharrows is loaded).

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

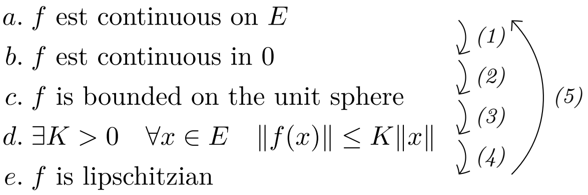
The command `\WithArrowsNbLines` is a command available in `code-after` which gives the total number of lines (=rows) of the current environment (it's a command and not a counter).

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


$a.\; f$ 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
1
2
3
4
5

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

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


$a.\; f$ est continuous on E
$b.\; f$ est continuous in 0
$c.\; f$ is bounded on the unit sphere
$d.\; \exists K > 0 \quad \forall x \in E \quad \|f(x)\| \le K\|x\|$
$e.\; f$ is lipschitzian
(1)
(2)
(3)
(4)
(5)

12 Implementation

The development of the extension witharrows is done on the following GitHub depot:
https://github.com/fpantigny/witharrows

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 L3 code fails.[28]

```
⟨*LaTeX⟩
\RequirePackage{tikz}
⟨/LaTeX⟩
⟨*plain-TeX⟩
\input tikz.tex
\input expl3-generic.tex
⟨/plain-TeX⟩
\usetikzlibrary{arrows.meta}
\usepgfmodule{bending} % https://texnique.fr/osqa/questions/12199
```

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

```
⟨*LaTeX⟩
\RequirePackage{l3keys2e}
\ProvidesExplPackage
  {witharrows}
  {\myfiledate}
  {\myfileversion}
  {Draws arrows for explanations on the right}
```

```
\RequirePackage { varwidth }
⟨/LaTeX⟩
```

```
⟨*plain-TeX⟩
\ExplSyntaxOn
\catcode `\@ = 11
⟨/plain-TeX⟩
```

```
⟨*LaTeX⟩
\ProvideDocumentCommand{\IfPackageLoadedT}{mm}
  {\IfPackageLoadedTF{#1}{#2}{}}
\ProvideDocumentCommand{\IfPackageLoadedF}{mm}
  {\IfPackageLoadedTF{#1}{}{#2}}
⟨/LaTeX⟩
```

12.2 The packages footnote and footnotehyper

A few options can be given to the package witharrows when it is loaded (with \usepackage, \RequirePackage or \PassOptionsToPackage). Currently (version 2.9a), there are two such options: footnote and footnotehyper. With the option footnote, witharrows loads footnote and uses it to extract the footnotes from the environments {WithArrows}. Idem for the option footnotehyper.

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

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

[28] 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.

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

```
\cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { witharrows } }
\cs_new_protected:Npn \@@_msg_new:nnn #1 #2 #3
  {
    \bool_if:NTF \c_@@_messages_for_Overleaf_bool
      { \msg_new:nnn { witharrows } { #1 } { #2 \\ #3 } }
      { \msg_new:nnnn { witharrows } { #1 } { #2 } { #3 } }
  }
\cs_new_protected:Npn \@@_msg_redirect_name:nn
  { \msg_redirect_name:nnn { witharrows } }
\cs_new_protected:Npn \@@_error:n { \msg_error:nn { witharrows } }
\cs_new_protected:Npn \@@_warning:n { \msg_warning:nn { witharrows } }
\cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { witharrows } }
\cs_new_protected:Npn \@@_error:nn  { \msg_error:nnn { witharrows } }
\cs_generate_variant:Nn \@@_error:nn { n e }
```

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

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

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

```
\bool_set:Nn \c_@@_messages_for_Overleaf_bool
  {
       \str_if_eq_p:on \c_sys_jobname_str { _region_ }  % for Emacs
    || \str_if_eq_p:on \c_sys_jobname_str { output }   % for Overleaf
  }
```

```
\bool_new:N \g_@@_beamer_bool
```

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

```
⟨*LaTeX⟩
\keys_define:nn { WithArrows / package }
  {
    footnote .bool_set:N = \c_@@_footnote_bool ,
    footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
    footnote .usage:n = load ,
    footnotehyper .usage:n = load ,
    beamer .bool_gset:N = \g_@@_beamer_bool ,
    beamer .default:n = true ,
    beamer .usage:n = load ,
    unknown .code:n = \@@_fatal:n { Option~unknown~for~package }
  }
\@@_msg_new:nn { Option~unknown~for~package }
  {
    You~can't~use~the~option~'\l_keys_key_str'~when~loading~the~
    package~witharrows.~Try~to~use~the~command~
    \token_to_str:N\WithArrowsOptions.
  }
```

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

```
\ProcessKeysOptions { WithArrows / package }
\IfClassLoadedT { beamer } { \bool_gset_true:N \g_@@_beamer_bool }
\IfPackageLoadedT { beamerarticle } { \bool_gset_true:N \g_@@_beamer_bool }
```

```
\@@_msg_new:nn { footnote~with~footnotehyper~package }
  {
    Footnote~forbidden.\\
    You~can't~use~the~option~'footnote'~because~the~package~
    footnotehyper~has~already~been~loaded.~
    If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
    within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
    of~the~package~footnotehyper.\\
    If~you~go~on,~the~package~footnote~won't~be~loaded.
  }
\@@_msg_new:nn { footnotehyper~with~footnote~package }
  {
    You~can't~use~the~option~'footnotehyper'~because~the~package~
    footnote~has~already~been~loaded.~
    If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
    within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
    of~the~package~footnote.\\
    If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
  }
```

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

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

```
    \@ifclassloaded { beamer }
      { \bool_set_false:N \c_@@_footnote_bool }
      {
        \@ifpackageloaded { footnotehyper }
          { \@@_error:n { footnote~with~footnotehyper~package } }
          { \usepackage { footnote } }
      }
  }
\bool_if:NT \c_@@_footnotehyper_bool
  {
```

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

```
    \@ifclassloaded { beamer }
      { \bool_set_false:N \c_@@_footnote_bool }
      {
        \@ifpackageloaded { footnote }
          { \@@_error:n { footnotehyper~with~footnote~package } }
          { \usepackage { footnotehyper } }
        \bool_set_true:N \c_@@_footnote_bool
      }
  }
```

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).

```
\bool_new:N \c_@@_leqno_bool
\DeclareOption { leqno } { \bool_set_true:N \c_@@_leqno_bool }
\DeclareOption* { }
\ProcessOptions*
⟨/LaTeX⟩
```

12.4 Collecting options

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

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

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

```
⟨*LaTeX⟩
\cs_new_protected:Npn \@@_collect_options:n #1
  {
    \peek_meaning:NTF [
      { \@@_collect_options:nw { #1 } }
      { #1 { } }
  }
```

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

```
\NewDocumentCommand \@@_collect_options:nw { m r[] }
  { \@@_collect_options:nn { #1 } { #2 } }

\cs_new_protected:Npn \@@_collect_options:nn #1 #2
  {
    \peek_meaning:NTF [
      { \@@_collect_options:nnw { #1 } { #2 } }
      { #1 { #2 } }
  }

\cs_new_protected:Npn \@@_collect_options:nnw #1#2[#3]
  { \@@_collect_options:nn { #1 } { #2 , #3 } }
⟨/LaTeX⟩
```

12.5 Some technical definitions

```
\cs_generate_variant:Nn \seq_set_split:Nnn { N e e }
\cs_generate_variant:Nn \keys_precompile:nnN { n n c }
\prg_generate_conditional_variant:Nnn \tl_if_novalue:n { o } { F }
\exp_args_generate:n { N N N n o }
```

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

```
\cs_new_protected:Npn \@@_sort_seq:N #1
  {
    \seq_sort:Nn #1
      {
        \str_compare:eNeTF
          { \str_lowercase:n { ##1 } } < { \str_lowercase:n { ##2 } }
          \sort_return_same:
          \sort_return_swapped:
      }
  }
```

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

```
\cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
  {
    \seq_set_from_clist:Nn #1 { #2 }
    \seq_set_map_e:NNn #1 #1 { \tl_to_str:n { ##1 } }
  }
```

The command `\@@_save:N` saves a L3 variable by creating a global version of the variable. For a variable named `\l_`*`name_type`*, the corresponding global variable will be named `\g_`*`name_type`*. The type of the variable is determined by the suffix *`type`* and is used to apply the corresponding L3 commands.

```
\cs_new_protected:Npn \@@_save:N #1
  {
    \seq_set_split:Nee \l_tmpa_seq
       { \char_generate:nn { `_ } { 12 } }
       { \cs_to_str:N #1 }
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
```

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

```
    \str_set:Ne \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
    \use:c { \l_tmpa_str _if_exist:cF }
      { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
      {
        \use:c { \l_tmpa_str _new:c }
          { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
      }
    \use:c { \l_tmpa_str _gset_eq:cN }
      { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
  }
```

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

```
\cs_new_protected:Npn \@@_restore:N #1
  {
    \seq_set_split:Nee \l_tmpa_seq
      { \char_generate:nn { `_ } { 12 } }
      { \cs_to_str:N #1 }
    \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
    \str_set:Ne \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
    \use:c { \l_tmpa_str _set_eq:Nc }
      #1 { g_\seq_use:Nnnn \l_tmpa_seq _ _ _ }
  }
```

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.

```
\tikzset
  {
    @@_node_style / .style =
      {
        above = \l_@@_ystart_dim ,
        inner~sep = \c_zero_dim ,
        minimum~width = \c_zero_dim ,
        minimum~height = \l_@@_ygap_dim
      }
  }
```

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.[29]

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.

```
\tikzset
  {
    @@_standard / .style =
      {
        remember~picture ,
        overlay ,
        name~prefix = wa - \l_@@_prefix_str -
      } ,
    @@_standard_arrow / .style =
      {
        @@_standard ,
        every~path / .style = WithArrows / arrow
      }
  }
```

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

```
\sys_if_engine_xetex:T
  {
    \tikzset
      {
        @@_standard_arrow / .append~style =
          { every~node / .append~style = { text = . } }
      }
  }
```

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.

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

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.

```
\tikzset
 {
   WithArrows / arrow / .style  =
     {
       align = flush~left ,
```

Before the version 2.7, it was align = left.

```
       auto = left ,
⟨*LaTeX⟩
       font = \small \itshape ,
⟨/LaTeX⟩
       WithArrows / arrow / tips ,
       bend~left = 45 ,
       ->
     }
```

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

```
 }
```

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}.

```
⟨*LaTeX⟩
\AtBeginDocument
  {
    \IfPackageLoadedTF { amsmath }
      {
        \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
        \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
      }
```

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.

```
      {
⟨/LaTeX⟩
        \cs_new_protected:Npn \spread@equation
          {
            \openup \jot
            \cs_set_eq:NN \spread@equation \prg_do_nothing:
          }
⟨*LaTeX⟩
      }
  }
⟨/LaTeX⟩
```

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

12.6 Variables

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

```
\bool_new:N \l_@@_in_WithArrows_bool
\bool_new:N \l_@@_in_DispWithArrows_bool
\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}.

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

```
\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).

```
\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).

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

```
\seq_new:N \g_@@_arrow_int_seq
\int_new:N \g_@@_arrow_int
\seq_new:N \g_@@_line_int_seq
\int_new:N \g_@@_line_int
\seq_new:N \g_@@_col_int_seq
\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.

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

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

```
\cs_new_protected:Npn \@@_test_if_to_tag:
  {
    \clist_if_in:NoT \l_@@_tags_clist \g_@@_line_int
      { \clist_set:Nn \l_@@_tags_clist { all } }
  }
⟨/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.

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

```
\str_new:N \l_@@_string_Arrow_for_msg_str
\str_set:Ne \l_@@_string_Arrow_for_msg_str { \token_to_str:N \Arrow }
```

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

```
\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.

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

```
⟨*LaTeX⟩
\bool_new:N \l_@@_tag_star_bool
\bool_new:N \l_@@_tag_next_line_bool
\bool_new:N \l_@@_qedhere_bool
⟨/LaTeX⟩
\bool_new:N \l_@@_in_first_columns_bool
\bool_new:N \l_@@_new_group_bool
\bool_new:N \l_@@_initial_r_bool
\bool_new:N \l_@@_final_r_bool
\tl_new:N \l_@@_initial_tl
\tl_new:N \l_@@_final_tl
\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}).

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

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

```
⟨*LaTeX⟩
\bool_new:N \l_@@_subequations_bool
⟨/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.

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

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

```
\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*).

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

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

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

12.7 The definition of the options

There are four levels where options can be set:

- with `\usepackage[...]{witharrows}`: this level will be called *package* level;
- with `\WithArrowsOptions{...}`: this level will be called *global* level[30];
- 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).

```
\cs_new_protected:Npn \@@_eval_if_allowed:n #1
  {
    \str_if_empty:NTF \l_@@_previous_key_str
      {
        \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
        #1
      }
      { \@@_error:n { Incompatible~options } }
  }
\cs_new_protected:Npn \@@_fix_pos_option:n #1
  { \@@_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.

```
\keys_define:nn { WithArrows / Global }
  {
    max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
    max-length-of-arrow .value_required:n = true ,
    max-length-of-arrow .initial:n = 2 cm ,
    ygap .dim_set:N = \l_@@_ygap_dim ,
    ygap .initial:n = 0.4 ex ,
    ygap .value_required:n = true ,
    ystart .dim_set:N = \l_@@_ystart_dim ,
    ystart .value_required:n = true ,
    ystart .initial:n = 0.4 ex ,
    more-columns .code:n =
      \@@_msg_redirect_name:nn { Too~much~columns~in~WithArrows } { none } ,
    more-columns .value_forbidden:n = true ,
```

[30] 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.

```
    command-name .code:n =
      \str_set:Nn \l_@@_command_name_str { #1 }
      \str_set:Ne \l_@@_string_Arrow_for_msg_str
        { \c_backslash_str Arrow~alias~\c_backslash_str #1 } ,
    command-name .value_required:n = true ,
    tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
    tikz-code .initial:n = \draw~(#1)~to~node{#3}~(#2)~; ,
    tikz-code .value_required:n = true ,
    displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
    displaystyle .default:n = true ,
    show-nodes .code:n =
      \tikzset { @@_node_style / .append~style = { draw , red } } ,
    show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
    show-node-names .default:n = true ,
    group .code:n =
      \str_if_empty:NTF \l_@@_previous_key_str
        {
          \str_set:Nn \l_@@_previous_key_str { group }
          \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
          \int_set:Nn \l_@@_pos_arrow_int 7
        }
        { \@@_error:n { Incompatible~options } } ,
    group .value_forbidden:n = true ,
    groups .code:n =
      \str_if_empty:NTF \l_@@_previous_key_str
        {
          \str_set:Nn \l_@@_previous_key_str { groups }
          \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
            { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
          \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
          \int_set:Nn \l_@@_pos_arrow_int 6
        }
        { \@@_error:n { Incompatible~options } } ,
    groups .value_forbidden:n = true ,
    tikz   .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
    tikz   .value_required:n  = true ,
    rr     .code:n            = \@@_fix_pos_option:n 3 ,
    rr     .value_forbidden:n = true ,
    ll     .code:n            = \@@_fix_pos_option:n 1 ,
    ll     .value_forbidden:n = true ,
    rl     .code:n            = \@@_fix_pos_option:n 2 ,
    rl     .value_forbidden:n = true ,
    lr     .code:n            = \@@_fix_pos_option:n 0 ,
    lr     .value_forbidden:n = true ,
    i      .code:n            = \@@_fix_pos_option:n 5 ,
    i      .value_forbidden:n = true ,
    xoffset .dim_set:N  = \l_@@_xoffset_dim ,
    xoffset .value_required:n  = true ,
    xoffset .initial:n = 3 mm ,
    jot .dim_set:N = \jot ,
    jot .value_required:n = true ,
    interline .skip_set:N = \l_@@_interline_skip ,
    start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
    start-adjust .initial:n = 0.4 ex ,
    start-adjust .value_required:n = true ,
    end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
    end-adjust .initial:n = 0.4 ex ,
    end-adjust .value_required:n = true ,
    adjust .meta:n  = { start-adjust = #1 , end-adjust = #1 } ,
    adjust .value_required:n = true ,
    up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,
    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`”.

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

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`.

```
\keys_define:nn { WithArrows / WithArrowsSpecific }
  {
    t   .code:n            = \int_set:Nn \l_@@_pos_env_int 0 ,
    t   .value_forbidden:n = true ,
    c   .code:n            = \int_set:Nn \l_@@_pos_env_int 1 ,
    c   .value_forbidden:n = true ,
    b   .code:n            = \int_set:Nn \l_@@_pos_env_int 2 ,
    b   .value_forbidden:n = true ,
    right-overlap .bool_set:N       = \l_@@_right_overlap_bool ,
    right-overlap .value_required:n = true
  }
```

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

```
\clist_new:N \c_@@_ext_delimiters_clist
\clist_set:Nn \c_@@_ext_delimiters_clist
  {
    ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
  }
⟨*LaTeX⟩
\AtBeginDocument
  {
    \bool_set_false:N \l_tmpa_bool
    \IfPackageLoadedT { amsmath } { \bool_set_true:N \l_tmpa_bool }
    \IfPackageLoadedT { unicode-math } { \bool_set_true:N \l_tmpa_bool }
    \bool_if:NT \l_tmpa_bool
      { \clist_put_right:Nn \c_@@_ext_delimiters_clist { \lvert, \lVert } }
  }
⟨/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`.

```
\keys_define:nn { WithArrows / DispWithArrowsSpecific }
  {
    fleqn .bool_set:N = \l_@@_fleqn_bool ,
    fleqn .default:n = true ,
    mathindent .skip_set:N = \l_@@_mathindent_skip ,
    mathindent .initial:n = 25 pt ,
    mathindent .value_required:n = true ,
⟨*LaTeX⟩
    notag .code:n =
      \str_if_eq:nnTF { #1 } { true }
        { \clist_clear:N \l_@@_tags_clist }
        { \clist_set:Nn \l_@@_tags_clist { all } } ,
    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.

```
    subequations .code:n =
      \IfPackageLoadedTF { amsmath }
        { \bool_set_true:N \l_@@_subequations_bool }
    {
     \@@_error:n { amsmath~not~loaded }
     \group_begin:
     \globaldefs = 1
     \@@_msg_redirect_name:nn { amsmath~not~loaded } { info }
     \group_end:
    } ,
  subequations .default:n = true ,
  subequations .value_forbidden:n = true ,
  nonumber .meta:n = notag ,
  allow-multiple-labels .code:n =
   \@@_msg_redirect_name:nn { Multiple~labels } { none } ,
  allow-multiple-labels .value_forbidden:n = true ,
  tagged-lines .code:n =
   \clist_set:Nn \l_@@_tags_clist { #1 }
   \clist_if_in:NnT \l_@@_tags_clist { first }
    {
     \clist_remove_all:Nn \l_@@_tags_clist { first }
     \clist_put_left:Nn \l_@@_tags_clist 1
    } ,
  tagged-lines .value_required:n = true ,
⟨/LaTeX⟩
  wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
  wrap-lines .default:n = true ,
  replace-left-brace-by .code:n =
   {
    \tl_set:Ne \l_tmpa_tl { \tl_head:n { #1 } }
    \clist_if_in:NoTF
     \c_@@_ext_delimiters_clist
     \l_tmpa_tl
     { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }
     { \@@_error:n { Bad~value~for~replace~brace~by } }
   } ,
  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).

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

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

```
\keys_define:nn { WithArrows / Env }
 {
  name .code:n =
```

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

```
   \str_set:Nn \l_tmpa_str { #1 }
   \seq_if_in:NoTF \g_@@_names_seq \l_tmpa_str
    { \@@_error:n { Duplicate~name } }
    { \seq_gput_left:No \g_@@_names_seq \l_tmpa_str }
   \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
  name .value_required:n = true ,
  code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
  code-before .value_required:n = true,
  CodeBefore .meta:n = { code-before = #1 } ,
  code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
  code-after .value_required:n = true ,
  CodeAfter .meta:n = { code-after = #1 } ,
    format .code:n =
      \tl_if_empty:nTF { #1 }
        { \@@_error:n { Invalid~option~format } }
        {
          \regex_match:nnTF { \A[rclRCL]*\Z } { #1 }
            { \tl_set:Nn \l_@@_format_str { #1 } }
            { \@@_error:n { Invalid~option~format } }
        } ,
    format .value_required:n = true
  }
```

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.

```
\keys_define:nn { WithArrows }
  {
    WithArrows .inherit:n =
      {
        WithArrows / Global ,
        WithArrows / WithArrowsSpecific ,
        WithArrows / Env
      } ,
    WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
    DispWithArrows .inherit:n =
      {
        WithArrows / DispWithArrowsSpecific ,
        WithArrows / Global ,
        WithArrows / Env ,
      } ,
    DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
    WithArrowsOptions .inherit:n =
      {
        WithArrows / Global ,
        WithArrows / WithArrowsSpecific ,
        WithArrows / DispWithArrowsSpecific ,
      } ,
    WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
  }
```

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

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

```
\keys_define:nn { WithArrows / WithArrows }
  {
    unknown .code:n  =
      \@@_sort_seq:N \l_@@_options_WithArrows_seq
      \@@_error:n { Unknown~option~WithArrows }
  }
```

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

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

```
\seq_new:N \l_@@_options_DispWithArrows_seq
\@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
  {
    code-after, code-before, command-name, tikz-code, adjust,
    displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
    left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
    up-and-down, replace-left-brace-by, rl, rr, show-node-names,
    show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
⟨*LaTeX⟩
    allow-multiple-labels, tagged-lines, nonumber, notag
⟨/LaTeX⟩
  }
```

```
\keys_define:nn { WithArrows / WithArrowsOptions }
  {
    allow-duplicate-names .code:n =
      \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
    allow-duplicate-names .value_forbidden:n = true ,
    xoffset-for-o-arrows .dim_set:N = \l_@@_xoffset_for_o_arrows_dim ,
    xoffset-for-o-arrows .value_required:n = true ,
    unknown .code:n  =
      \@@_sort_seq:N \l_@@_options_WithArrowsOptions_seq
      \@@_error:n { Unknown~option~WithArrowsOptions }
  }
```

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

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

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".

```
\cs_new_protected:Npn \@@_set_independent:
  {
    \str_if_eq:onF \l_keys_value_tl { NoValue }
      { \@@_error:n { Value~for~a~key } }
    \@@_set_independent_bis:
  }
```

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

```
\cs_new_protected:Npn \@@_set_independent_bis:
  {
    \str_if_empty:NTF \l_@@_previous_key_str
      {
        \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
        \str_set:Nn \l_@@_status_arrow_str { independent }
      }
      { \@@_error:n { Incompatible~options~in~Arrow } }
  }
```

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”.

```
\keys_define:nn { WithArrows / Arrow / FirstPass }
  {
    jump .code:n =
      \int_compare:nNnTF { #1 } > \c_zero_int
        { \int_set:Nn \l_@@_jump_int { #1 } }
        { \@@_error:n { Negative~jump } } ,
    jump .value_required:n  = true,
    rr .code:n = \@@_set_independent: ,
    ll .code:n = \@@_set_independent: ,
    rl .code:n = \@@_set_independent: ,
    lr .code:n = \@@_set_independent: ,
    i  .code:n = \@@_set_independent: ,
    rr .default:n = NoValue ,
    ll .default:n = NoValue ,
    rl .default:n = NoValue ,
    lr .default:n = NoValue ,
    i  .default:n = NoValue ,
    new-group .value_forbidden:n = true ,
    new-group .code:n =
      \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
        { \str_set:Nn \l_@@_status_arrow_str { new-group } }
        { \@@_error:n { new-group~without~groups } } ,
    o .code:n =
      \str_if_empty:NTF \l_@@_previous_key_str
        {
          \int_compare:nNnTF \l_@@_pos_arrow_int < 6
            { \@@_error:n { invalid~key~o } }
            {
              \str_set:Nn \l_@@_status_arrow_str { over }
              \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
            }
        }
        { \@@_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.

```
    tikz-code .code:n = \prg_do_nothing: ,
    tikz-code .value_required:n = true ,
    tikz .code:n = \prg_do_nothing: ,
    tikz .value_required:n = true ,
    start-adjust .code:n = \prg_do_nothing: ,
    start-adjust .value_required:n = true ,
    end-adjust .code:n = \prg_do_nothing: ,
    end-adjust .value_required:n = true ,
    adjust .code:n  = \prg_do_nothing: ,
    adjust .value_required:n = true ,
    xoffset .code:n = ,
    unknown .code:n =
      \@@_sort_seq:N \l_@@_options_Arrow_seq
      \seq_if_in:NoTF \l_@@_options_WithArrows_seq \l_keys_key_str
        {
          \str_set:Nn \l_tmpa_str
           { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
        }
        { \str_clear:N \l_tmpa_str }
      \@@_error:n { Unknown~option~in~Arrow }
  }
```

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

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

```
\cs_new_protected:Npn \@@_fix_pos_arrow:n #1
  {
    \str_if_empty:NT \l_@@_previous_key_str
      {
        \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
        \int_set:Nn \l_@@_pos_arrow_int { #1 }
      }
  }
```

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.

```
\keys_define:nn {WithArrows / Arrow / SecondPass }
  {
    tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
    tikz-code .initial:n = \draw~(#1)~to~node{#3}~(#2)~; ,
    tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
    rr .code:n = \@@_fix_pos_arrow:n 3 ,
    ll .code:n = \@@_fix_pos_arrow:n 1 ,
    rl .code:n = \@@_fix_pos_arrow:n 2 ,
    lr .code:n = \@@_fix_pos_arrow:n 0 ,
    i  .code:n = \@@_fix_pos_arrow:n 5 ,
    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.

```
    xoffset .code:n  =
      \bool_lazy_all:nTF
        {
          { \int_compare_p:nNn \g_@@_arrow_int > 1 }
          { \int_compare_p:nNn \l_@@_pos_arrow_int > 5 }
          { ! \str_if_eq_p:on \l_@@_status_arrow_str { independent } }
        }
        { \@@_error:n { Option~xoffset~forbidden } }
        { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
    xoffset .value_required:n = true ,
    start-adjust .dim_set:N = \l_@@_start_adjust_dim,
    end-adjust .dim_set:N = \l_@@_end_adjust_dim,
    adjust .code:n  =
      \dim_set:Nn \l_@@_start_adjust_dim { #1 }
      \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
  }
```

\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.

```
⟨*LaTeX⟩
\NewDocumentCommand \WithArrowsOptions { m }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_set_protected:Npn \WithArrowsOptions #1
⟨/plain-TeX⟩
  {
    \str_clear_new:N \l_@@_previous_key_str
    \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
  }
```

12.8 The command \Arrow

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

```
⟨*LaTeX⟩
\cs_new_protected:Npn \@@_Arrow
  { \@@_collect_options:n { \@@_Arrow_iii } }
\bool_if:NTF \g_@@_beamer_bool
  {
    \NewDocumentCommand \@@_Arrow_iii { m d < > m ! O { } }
      {
        \tl_if_novalue:nTF { #2 }
          { \@@_Arrow_ii { #1 } { #3 } [ #4 ] }
          { \only <#2> { \@@_Arrow_ii { #1 } { #3 } [ #4 ] } }
      }
  }
  {
    \NewDocumentCommand \@@_Arrow_iii { m m ! O { } }
      { \@@_Arrow_ii { #1 } { #2 } [ #3 ] }
  }
\NewDocumentCommand \@@_Arrow_ii { m m ! O { } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \@@_Arrow
  {
    \peek_meaning:NTF [
      { \@@_Arrow_i }
      { \@@_Arrow_i [ ] }
  }
\cs_new_protected:Npn \@@_Arrow_i [ #1 ] #2
  {
    \peek_meaning:NTF [
      { \@@_Arrow_ii [ #1 ] { #2 } }
      { \@@_Arrow_ii [ #1 ] { #2 } [ ] }
  }
\cs_new_protected:Npn \@@_Arrow_ii [ #1 ] #2 [ #3 ]
⟨/plain-TeX⟩
  {
```

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.

```
    \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`.

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

```
        \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`):

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

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

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

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

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

```
        \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).

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

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

```
        \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.

```
    \prop_gclear_new:c
      { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
    \prop_gset_eq:cN
      { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
      \l_tmpa_prop
  }
```

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.

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

12.9 The environments {WithArrows} and {DispWithArrows}

12.9.1 Code before the \halign

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

```
\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.

```
  {
⟨*LaTeX⟩
    \str_clear_new:N \l_@@_type_env_str
    \str_set:NV \l_@@_type_env_str \@currenvir
⟨/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`.

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

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

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

```
    \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.

```
    \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.

```
    \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.

```
    \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.

```
    \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
    \int_gzero:N \g_@@_arrow_int
    \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
    \int_gzero:N \g_@@_line_int
    \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
    \int_gzero:N \g_@@_col_int
    \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
    \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.

```
    \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).

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

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

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

These counters will be used later as variables.

```
    \int_zero_new:N \l_@@_initial_int
    \int_zero_new:N \l_@@_final_int
    \int_zero_new:N \l_@@_arrow_int
    \int_zero_new:N \l_@@_pos_of_arrow_int
    \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.

```
    \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.

```
    \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`.

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

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

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.[31]

```
    \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`.

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

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

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

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

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

```
    \bool_if:NF \l_@@_right_overlap_bool
      {
        \bool_if:NT \l_@@_in_WithArrows_bool
          {
            \dim_gzero_new:N \g_@@_overlap_x_dim
            \dim_zero_new:N \l_@@_delta_x_dim
          }
      }
```

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.

```
    \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).

```
    \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.

```
    \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.

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

```
⟨*LaTeX⟩
    \bool_if:NT \c_@@_footnote_bool { \begin { savenotes } }
⟨/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).

```
    \l_@@_code_before_tl
```

```
⟨*LaTeX⟩
    \cs_set_eq:NN \notag \@@_notag:
    \cs_set_eq:NN \nonumber \@@_nonumber:
    \cs_set_eq:NN \tag \@@_tag
    \cs_set_eq:NN \@@_old_label \label
    \cs_set_eq:NN \label \@@_label:n
    \cs_set_eq:NN \tagnextline \@@_tagnextline:
⟨/LaTeX⟩
  }
```

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

12.9.2 The construction of the preamble of the \halign

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

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

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

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

```
\cs_new_protected:Npn \@@_construct_halign:
  {
    \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
      {
```

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.

```
        \use:e
          {
```

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.

```
            \int_gdecr:N \g_@@_col_int
            \@@_construct_halign:
            \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
              {
```

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`.

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

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).

```
                    \@@_test_if_to_tag:
```

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

```
                    \IfPackageLoadedT { amsmath } { \@@_set_qedhere: }
                  }
⟨/LaTeX⟩
              }
            \str_if_eq:onT \l_@@_type_col_str { c } \hfil
            \str_if_eq:onT \l_@@_type_col_str { C } \hfil
            \str_if_eq:onT \l_@@_type_col_str { r } \hfill
            \str_if_eq:onT \l_@@_type_col_str { R } \hfill
            \int_gincr:N \g_@@_col_int
            \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
            \c_math_toggle_token
            \str_if_eq:onT \l_@@_type_col_str { C } { { } }
            \str_if_eq:onT \l_@@_type_col_str { L } { { } }
            \bool_if:NT \l_@@_displaystyle_bool \displaystyle
            ##
            \str_if_eq:onT \l_@@_type_col_str { C } { { } }
            \str_if_eq:onT \l_@@_type_col_str { R } { { } }
            \c_math_toggle_token
            \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
              \@@_construct_nodes:
              {
```

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:).

```
                \str_if_eq:onT \l_@@_type_col_str { l } \hfil
                \str_if_eq:onT \l_@@_type_col_str { L } \hfil
                \str_if_eq:onT \l_@@_type_col_str { c } \hfil
                \str_if_eq:onT \l_@@_type_col_str { C } \hfil
                \bool_if:NT \l_@@_in_DispWithArrows_bool { \tabskip = \c_zero_skip }
                &
              }
          }
      }
```

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

```
      {
        \bool_if:NTF \l_@@_in_WithArrows_bool
          {
            \ialign
            \bgroup
          }
          {
            \halign to \l_@@_linewidth_dim
            \bgroup
            \bool_if:NT \l_@@_fleqn_bool
              { \skip_horizontal:N \l_@@_mathindent_skip }
          }
        \int_gincr:N \g_@@_line_int
        \int_gzero:N \g_@@_col_int
        \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
          {
            \skip_horizontal:n
              { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
          }
        \strut
      }
  }
```

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.

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

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).

```
    \tikz [ remember~picture , overlay ]
      \node
        [
          node~contents = { } ,
          @@_node_style ,
          name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
        ]
        ;
    \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).

```
    \tikz [ remember~picture , overlay ]
      \node
        [
          node~contents = { } ,
          @@_node_style ,
          name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
        ]
        ;
    \str_if_empty:NF \l_@@_name_str
      {
        \pgfpicture
        \pgfnodealias
          { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
          { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l }
        \pgfnodealias
          { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
          { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
        \endpgfpicture
      }
    \bool_if:NT \l_@@_show_node_names_bool
      {
        \hbox_overlap_right:n
          { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
      }
  }
```

12.9.3 The environment {WithArrows}

```
⟨*LaTeX⟩
\NewDocumentEnvironment { WithArrows } { ! O { } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \WithArrows
  {
    \group_begin:
    \peek_meaning:NTF [
     { \WithArrows_i }
     { \WithArrows_i [  ] }
  }
\cs_new_protected:Npn \WithArrows_i [ #1 ]
⟨/plain-TeX⟩
  {
    \bool_set_true:N \l_@@_in_WithArrows_bool
    \bool_set_false:N \l_@@_in_DispWithArrows_bool
⟨*plain-TeX⟩
    \str_clear_new:N \l_@@_type_env_str
    \str_set:Nn \l_@@_type_env_str { WithArrows }
⟨/plain-TeX⟩
    \@@_pre_halign:n { #1 }
    \if_mode_math: \else:
      \@@_error:n { WithArrows~outside~math~mode }
    \fi:

    \box_clear_new:N \l_@@_env_box
    \hbox_set:Nw \l_@@_env_box
```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`[32] 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[33] and therefore, we can use `\vcenter`.

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

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

```
    \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.

```
    \@@_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`.

```
    &&
    \@@_error:n { Too~much~columns~in~WithArrows }
    \c_math_toggle_token
    \bool_if:NT \l_@@_displaystyle_bool \displaystyle
    { ## }
    \c_math_toggle_token
    \cr
  }
```

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

```
⟨*plain-TeX⟩
\cs_new_protected:Npn \endWithArrows
⟨/plain-TeX⟩
  {
    \\
    \egroup
    \egroup
    \int_compare:nNnT \l_@@_pos_env_int = 1 \c_math_toggle_token
    \hbox_set_end:
    \@@_post_halign:
```

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

```
    \bool_if:NF \l_@@_right_overlap_bool
```

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

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

```
      {
        \box_set_wd:Nn \l_@@_env_box
          { \g_@@_overlap_x_dim + \box_wd:N \l_@@_env_box }
      }
    \box_use_drop:N \l_@@_env_box
```

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

```
⟨*LaTeX⟩
    \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
    \group_end:
⟨/plain-TeX⟩
  }
```

This is the end of the environment {WithArrows}.

12.9.4 After the construction of the \halign

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

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

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

```
  {
    \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.

```
    \normalbaselines
```

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

```
    \int_if_zero:nF \g_@@_arrow_int
      {
```

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).

```
        \int_compare:nNnT \g_@@_arrow_int = 1
          {
            \int_compare:nNnT \l_@@_pos_arrow_int > 5
              { \int_set:Nn \l_@@_pos_arrow_int 5 }
          }
        \@@_scan_arrows:
      }
```

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

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

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

```
      \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.[34]

```
      \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
      \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
      \bool_set_true:N \l_@@_in_code_after_bool
      \l_@@_code_after_tl
    \group_end:
```

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

```
    \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
    \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
    \seq_gput_right:Ne \g_@@_position_in_the_tree_seq
      { \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.

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

```
    \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
    \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
    \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
    \int_gset:Nn \g_@@_line_int \l_tmpa_tl
    \seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
    \int_gset:Nn \g_@@_col_int \l_tmpa_tl
    \seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
    \int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
  }
```

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

12.9.5 The command of end of row

We give now the definition of \@@_cr: which is the definition of \\ in an environment {WithArrows}. The two commands \group_align_safe_begin: and \group_align_safe_end: are specifically designed for this purpose: test the token that follows in an \halign structure.
First, we remove an eventual token * (just after the \\: there should not be space between the two) since the commands \\ and \\* are equivalent in an environment {WithArrows} (an environment {WithArrows}, like an environment {aligned} of amsmath, is always unbreakable).

```
\cs_new_protected:Npn \@@_cr:
  {
    \scan_stop:
```

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

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

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

Then, we peek the next token to see if it's a [. In this case, the command \\ has an optional argument which is the vertical skip (=glue) to put.

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

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

```
⟨*LaTeX⟩
\cs_new_protected:Npn \@@_cr_ii: [ #1 ]
  {
    \peek_remove_spaces:n
      {
        \peek_meaning:NTF \end
          {
            \@@_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.

```
            \@@_analyze_end:Nn
          }
          { \@@_cr_iii:n { #1 } }
      }
  }
\cs_new_protected:Npn \@@_cr_iii:n #1
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \@@_cr_ii: [ #1 ]
⟨/plain-TeX⟩
  {
    \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.[35]

```
    \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.

```
      {
⟨*LaTeX⟩
        \clist_if_in:NnTF \l_@@_tags_clist { all }
          {
```

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).

```
            \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 &).

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

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

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

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

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`.

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

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

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

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.

```
    \IfPackageLoadedT { cleveref }
      {
        \cref@constructprefix { equation } \cref@result
        \protected@edef \cref@currentlabel
          {
            [
              \cs_if_exist:NTF \cref@equation@alias
                \cref@equation@alias
                { equation }
            ]
            [ \arabic { equation } ] [ \cref@result ]
            \p@equation \g_tmpa_tl
          }
      }
```

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`).

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

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.

```
\@@_save:N \l_@@_tag_star_bool
\@@_save:N \l_@@_qedhere_bool
\bool_if:NT \l_@@_tag_next_line_bool
  {
    \openup -\jot
    \bool_set_false:N \l_@@_tag_next_line_bool
    \notag \\ &
  }
&
\@@_restore:N \l_@@_tag_star_bool
\@@_restore:N \l_@@_qedhere_bool
\bool_if:NT \l_@@_qedhere_bool
  { \hbox_overlap_left:n \@@_qedhere_i: }
\cs_set_eq:NN \theequation \g_tmpa_tl
\bool_if:NT \l_@@_tag_star_bool
  { \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`.

```
\hbox_overlap_left:n
```

```
              {
                \bool_if:NF \c_@@_leqno_bool
                  {
                    \pgfpicture
                    \pgfrememberpicturepositiononpagetrue
                    \pgfcoordinate
                      { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
                      \pgfpointorigin
                    \endpgfpicture
                  }
                \quad
                \@eqnnum
              }
            \bool_if:NT \c_@@_leqno_bool
              {
                \pgfpicture
                \pgfrememberpicturepositiononpagetrue
                \pgfcoordinate
                  { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
                  \pgfpointorigin
                \endpgfpicture
              }
          }
          {
            \@@_save:N \l_@@_qedhere_bool
⟨/LaTeX⟩
            &
⟨*LaTeX⟩
            \@@_restore:N \l_@@_qedhere_bool
            \bool_if:NT \l_@@_qedhere_bool
              { \hbox_overlap_left:n \@@_qedhere_i: }
⟨/LaTeX⟩
            \pgfpicture
            \pgfrememberpicturepositiononpagetrue
            \pgfcoordinate
              { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
              \pgfpointorigin
            \endpgfpicture
⟨*LaTeX⟩
          }
⟨/LaTeX⟩
      }
    \dim_compare:nNnT { #1 } < \c_zero_dim
       { \@@_error:n { option~of~cr~negative } }

    \cr
    \noalign
      {
        \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
        \skip_vertical:N \l_tmpa_dim
        \skip_vertical:N \l_@@_interline_skip
        \scan_stop:
      }
  }
```

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

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

```
⟨*LaTeX⟩
```

```
\cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
  {
    \str_if_eq:onT \l_@@_type_env_str { #2 }
      {
        \@@_error:n { newline~at~the~end~of~env }
        \group_begin:
        \globaldefs = 1
        \@@_msg_redirect_name:nn { newline~at~the~end~of~env } { none }
        \group_end:
      }
```

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

```
    \end { #2 }
  }
⟨/LaTeX⟩
```

12.9.6 The environment {DispWithArrows}

For the environment `{DispWithArrows}`, the general form of the construction is of the type:
`\[\vtop{\halign to \displaywidth {...}}\]`
The purpose of the `\vtop` is to have an environment unbreakable.
However, if we are 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}`.

```
⟨*LaTeX⟩
\bool_new:N \l_@@_in_label_or_minipage_bool
⟨/LaTeX⟩
```

```
⟨*LaTeX⟩
\NewDocumentEnvironment { DispWithArrows } { ! d < > ! O { } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \DispWithArrows
  {
    \group_begin:
    \peek_meaning:NTF <
      { \DispWithArrows_i }
      { \DispWithArrows_i < \c_novalue_tl > }
  }
\cs_new_protected:Npn \DispWithArrows_i < #1 >
  {
    \peek_meaning:NTF [
      { \DispWithArrows_ii < #1 > }
      { \DispWithArrows_ii < #1 > [ ] }
  }
\cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
⟨/plain-TeX⟩
  {
    \bool_set_true:N \l_@@_in_DispWithArrows_bool
⟨*plain-TeX⟩
    \str_clear_new:N \l_@@_type_env_str
    \str_set:Nn \l_@@_type_env_str { DispWithArrows }
⟨/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[36]).

```
⟨*LaTeX⟩
    \bool_if:NF \l_@@_sbwi_bool
      {
        \legacy_if:nT { @inlabel }
          { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
        \legacy_if:nT { @minipage }
          { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
      }
⟨/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).

```
⟨*LaTeX⟩
    \IfPackageLoadedT { mathtools }
      {
        \MH_if_boolean:nT { show_only_refs }
          {
            \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.

```
            \MH_set_boolean_T:n { show_only_refs }
          }
      }
```

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.

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

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

```
    \IfPackageLoadedT { amsmath } { \intertext@ }
⟨/LaTeX⟩
    \tl_if_novalue:oF { #1 }
      { \cs_set_nopar:Npn \l_@@_left_brace_tl { #1 } }
    \@@_pre_halign:n { #2 }
```

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

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

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

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

```
        \hbox_set:Nn \l_tmpa_box
          {
```

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

```
            \group_begin:
            \dim_zero:N \nulldelimiterspace
            \c_math_toggle_token
            \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
            \c_math_toggle_token
            \group_end:
          }
```

[36] 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}.

```
        \dim_zero_new:N \l_@@_delim_wd_dim
        \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
        \box_clear_new:N \l_@@_left_brace_box
        \hbox_set:Nn \l_@@_left_brace_box
          {
            \group_begin:
              \cs_set_eq:NN \label \@@_old_label
              \c_math_toggle_token
              \bool_if:NT \l_@@_displaystyle_bool \displaystyle
              \l_@@_left_brace_tl
              { }
              \c_math_toggle_token
            \group_end:
          }
      }
```

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

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

```
    \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.

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

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

```
⟨*plain-TeX⟩
     \dim_zero_new:N \linewidth
     \dim_set_eq:NN \linewidth \displaywidth
⟨/plain-TeX⟩
⟨*LaTeX⟩
    \bool_if:NTF \l_@@_in_label_or_minipage_bool
      {
        \noindent
        \c_math_toggle_token
      }
      {
⟨/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.

```
        \if_mode_vertical:
        \nointerlineskip
        \hbox_to_wd:nn { .6 \linewidth } { }
        \fi:
        \c_math_toggle_token \c_math_toggle_token
⟨*LaTeX⟩
      }
⟨/LaTeX⟩
```

```
    \dim_zero_new:N \l_@@_linewidth_dim
⟨*LaTeX⟩
    \bool_if:NTF \l_@@_in_label_or_minipage_bool
      { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
      { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
⟨/LaTeX⟩
```

```
⟨*plain-TeX⟩
  \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
⟨/plain-TeX⟩
  \box_clear_new:N \l_@@_halign_box
  \setbox \l_@@_halign_box \vtop \bgroup
  \tabskip =
   \bool_if:NTF \l_@@_fleqn_bool
    \c_zero_skip
    { 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).

```
  \spread@equation
  \@@_construct_halign:
  \tabskip = 0 pt plus 1000 pt minus 1000 pt
  &
```

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.

```
  $ ## $
  \tabskip = \c_zero_skip
  &&
  \@@_fatal:n { Too~much~columns~in~DispWithArrows }
  \bool_if:nT \c_false_bool { ## }
  \cr
 }
```

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

```
⟨*plain-TeX⟩
\cs_new_protected:Npn \endDispWithArrows
⟨/plain-TeX⟩
 {
⟨*LaTeX⟩
  \clist_if_in:NnT \l_@@_tags_clist { last }
   { \clist_set:Nn \l_@@_tags_clist { all } }
⟨/LaTeX⟩
  \\
```

The following \egroup is for the \halign.

```
  \egroup
  \unskip \unpenalty \unskip \unpenalty
  \box_set_to_last:N \l_tmpa_box
  \nointerlineskip
  \box_use:N \l_tmpa_box
  \dim_gzero_new:N \g_@@_alignment_dim
  \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
  \box_clear_new:N \l_@@_new_box
  \hbox_set:Nn \l_@@_new_box { \hbox_unpack_drop:N \l_tmpa_box }
  \dim_compare:nNnT
   { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
   { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }
```

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

```
  \egroup
   \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
    { \box_use_drop:N \l_@@_halign_box }
    {
     \hbox_to_wd:nn \l_@@_linewidth_dim
```

```
           {
            \bool_if:NTF \l_@@_fleqn_bool
              { \skip_horizontal:N \l_@@_mathindent_skip }
              \hfil
            \hbox_to_wd:nn \g_@@_alignment_dim
              {
                \box_use_drop:N \l_@@_left_brace_box
```

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

```
                \dim_set:Nn \l_tmpa_dim
                  {
                    \box_ht:N \l_@@_halign_box
                    + \box_dp:N \l_@@_halign_box
                  }
                \group_begin:
                \dim_zero:N \nulldelimiterspace
                \c_math_toggle_token
                  \left \l_@@_replace_left_brace_by_tl
                    \vcenter to \l_tmpa_dim { \vfil }
                  \right.
                \c_math_toggle_token
                \group_end:
                \hfil
              }
            \hfil
           }
         \skip_horizontal:N -\l_@@_linewidth_dim
         \vcenter { \box_use_drop:N \l_@@_halign_box }
       }
```

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.

```
    \dim_gzero_new:N \g_@@_right_x_dim
    \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
    \pgfpicture
      \pgfrememberpicturepositiononpagetrue
      \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
        {
          \cs_if_free:cTF
            { pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
            { \@@_fatal:n { Inexistent~v-node } }
            {
              \pgfpointanchor
                { wa - \l_@@_prefix_str - \l_tmpa_int - v }
                { center }
              \dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
                { \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x }
            }
        }
    \endpgfpicture
```

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

```
    \@@_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).

```
⟨*LaTeX⟩
    \IfPackageLoadedT { mathtools }
      { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
    \bool_if:NTF \l_@@_in_label_or_minipage_bool
      {
        \c_math_toggle_token
        \skip_vertical:N \belowdisplayskip
      }
      { \c_math_toggle_token \c_math_toggle_token }
⟨/LaTeX⟩
⟨*plain-TeX⟩
      \c_math_toggle_token \c_math_toggle_token
⟨/plain-TeX⟩
```

```
⟨*LaTeX⟩
    \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).

```
    \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
    \group_end:
⟨/plain-TeX⟩
⟨*LaTeX⟩
    \ignorespacesafterend
⟨/LaTeX⟩
  }
```

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.

```
⟨*LaTeX⟩
\NewDocumentEnvironment { DispWithArrows* } { }
  {
    \WithArrowsOptions { notag }
    \DispWithArrows
  }
  \endDispWithArrows
⟨/LaTeX⟩
```

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

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

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

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

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

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

```
\NewDocumentCommand \@@_tag { s m }
  {
    \@@_if_in_last_col_of_disp:Nn \tag
      {
        \tl_if_empty:NF \l_@@_tag_tl
          { \@@_error:nn  { Multiple~tags } { #2 } }
        \clist_set:Nn \l_@@_tags_clist { all }
        \IfPackageLoadedT { mathtools }
          {
            \MH_if_boolean:nT { show_only_refs }
              {
                \MH_if_boolean:nF { show_manual_tags }
                  { \clist_clear:N \l_@@_tags_clist }
              }
          }
        \tl_set:Nn \l_@@_tag_tl { #2 }
        \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`.[37]

```
        \bool_if:nT { #1 }
          {
            \IfPackageLoadedF { amsmath }
              { \@@_error:n { tag*~without~amsmath } }
          }
      }
  }
```

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`.

```
\cs_new_protected:Npn \@@_label:n #1
  {
    \@@_if_in_last_col_of_disp:Nn \label
      {
        \seq_if_empty:NF \l_@@_labels_seq
          {
            \IfPackageLoadedTF { cleveref }
              { \@@_error:n { Multiple~labels~with~cleveref } }
              { \@@_error:n { Multiple~labels } }
          }
        \seq_put_right:Nn \l_@@_labels_seq { #1 }
        \IfPackageLoadedT { mathtools }
          {
            \MH_if_boolean:nT { show_only_refs }
```

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

```
       {
         \cs_if_exist:cTF { MT_r_#1 }
           { \clist_set:Nn \l_@@_tags_clist { all } }
           { \clist_clear:N \l_@@_tags_clist }
       }
     }
    \IfPackageLoadedT { autonum }
     {
      \cs_if_exist:cTF { autonum@#1Referenced }
       { \clist_set:Nn \l_@@_tags_clist { all } }
       { \clist_clear:N \l_@@_tags_clist }
     }
   }
 }
```

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

```
\cs_new_protected:Npn \@@_tagnextline:
 {
  \@@_if_in_last_col_of_disp:Nn \tagnextline
   { \bool_set_true:N \l_@@_tag_next_line_bool }
 }
```

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`.

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

```
\cs_new_protected:Npn \@@_qedhere_i:
 {
  \group_begin:
   \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.

```
   \cs_set_eq:NN \qed@elt \setQED@elt
   \QED@stack \relax \relax
  \group_end:
 }
⟨/LaTeX⟩
```

12.11 We draw the arrows

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

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

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

```
\cs_new_protected:Npn \@@_scan_arrows:
  {
    \group_begin:
    \int_compare:nNnT \l_@@_pos_arrow_int = 7
      {
        \@@_scan_arrows_i:
        \int_set:Nn \l_@@_pos_arrow_int 8
      }
    \@@_scan_arrows_i:
    \group_end:
  }
```

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

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

```
    \int_zero_new:N \l_@@_first_arrow_of_group_int
    \int_zero_new:N \l_@@_first_line_of_group_int
    \int_zero_new:N \l_@@_last_line_of_group_int
    \seq_clear_new:N \l_@@_first_arrows_seq
    \seq_clear_new:N \l_@@_last_arrows_seq
```

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

```
    \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.

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

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.

```
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { initial } \l_tmpa_tl
        \int_set:Nn \l_@@_initial_int \l_tmpa_tl
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { final } \l_tmpa_tl
        \int_set:Nn \l_@@_final_int \l_tmpa_tl
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { status } \l_@@_status_arrow_str
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { 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.

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

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

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

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).

```
      \bool_lazy_and:nnT
        { ! \int_compare_p:nNn \l_@@_pos_arrow_int = 7 }
        { \int_compare_p:nNn \l_@@_first_arrow_of_group_int > \c_zero_int }
        { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
    }
```

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)

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

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).

```
    \bool_lazy_and:nnT
      { \int_compare_p:nNn \l_@@_arrow_int > 1 }
      {
        \bool_lazy_or_p:nn
          {
            \bool_lazy_and_p:nn
              {
                \int_compare_p:nNn
                  \l_@@_initial_int > \l_@@_last_line_of_group_int
              }
              { \bool_not_p:n { \int_compare_p:nNn \l_@@_pos_arrow_int = 7 } }
          }
          { \str_if_eq_p:on \l_@@_status_arrow_str { new-group } }
      }
      {
        \int_if_zero:nF \l_@@_first_arrow_of_group_int
          {
            \@@_draw_arrows:nn
              \l_@@_first_arrow_of_group_int
              { \l_@@_arrow_int - 1 }
          }
        \bool_set_true:N \l_@@_new_group_bool
      }
```

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.

```
    \bool_if:nTF \l_@@_new_group_bool
      {
        \bool_set_false:N \l_@@_new_group_bool
        \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
        \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
        \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
        \seq_clear:N \l_@@_first_arrows_seq
        \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
        \seq_clear:N \l_@@_last_arrows_seq
        \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.

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

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

```
      {
```

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

```
        \str_if_eq:onF \l_@@_status_arrow_str { independent }
          {
```

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

```
            \int_compare:nNnT \l_@@_initial_int = \l_@@_first_line_of_group_int
              { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
            \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
              {
                \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
                \seq_clear:N \l_@@_last_arrows_seq
                \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
              }
              {
                \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
                  { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
              }
          }
      }
```

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.

```
    \str_if_eq:onF \l_@@_status_arrow_str { independent }
      {
        \int_compare:nNnF \l_@@_pos_arrow_int = 8
          { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
      }
  }
```

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.

```
\cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
  {
    \group_begin:
    \int_zero_new:N \l_@@_first_arrow_int
    \int_set:Nn \l_@@_first_arrow_int { #1 }
    \int_zero_new:N \l_@@_last_arrow_int
    \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.

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

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.

```
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { initial } \l_tmpa_tl
        \int_set:Nn \l_@@_initial_int \l_tmpa_tl
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { final } \l_tmpa_tl
        \int_set:Nn \l_@@_final_int \l_tmpa_tl
        \prop_get:cnN
          { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
          { 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).

```
        \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).

```
          {
            \str_if_eq:onTF \l_@@_status_arrow_str { over }
              { \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
              \@@_draw_arrow:
          }
        \int_incr:N \l_@@_arrow_int
      }
    \@@_draw_o_arrows_of_the_group:
    \group_end:
  }
```

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).

```
\cs_new_protected:Npn \@@_draw_arrow:
  {
    \group_begin:
```

We process the options of the current arrow.

```
    \prop_get:cnN
      { g_@@_arrow _\l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
      { options }
      \l_tmpa_tl
    \str_clear_new:N \l_@@_previous_key_str
```

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

```
    \exp_args:NNNno \exp_args:NNno \exp_args:Nno
    \keys_set_known:nn
      { WithArrows / Arrow / SecondPass }
      { \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`:

```
    \bool_set_false:N \l_@@_initial_r_bool
    \bool_set_false:N \l_@@_final_r_bool
    \int_case:nn \l_@@_pos_arrow_int
      {
        0 { \bool_set_true:N \l_@@_final_r_bool }
        2 { \bool_set_true:N \l_@@_initial_r_bool }
        3
          {
            \bool_set_true:N \l_@@_initial_r_bool
            \bool_set_true:N \l_@@_final_r_bool
          }
      }
```

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).

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

`\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`.

```
    \tl_set:Ne \l_@@_initial_tl
      { \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl }
    \tl_set:Ne \l_@@_final_tl
      { \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`.

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

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

```
    \seq_if_in:NeTF \l_@@_first_arrows_seq
      { \int_use:N \l_@@_arrow_int }
      { \bool_set_true:N \l_tmpa_bool }
      { \bool_set_false:N \l_tmpa_bool }
    \seq_if_in:NeTF \l_@@_last_arrows_seq
      { \int_use:N \l_@@_arrow_int }
      { \bool_set_true:N \l_tmpb_bool }
      { \bool_set_false:N \l_tmpb_bool }
    \int_compare:nNnT \l_@@_pos_arrow_int = 5
      {
        \bool_set_true:N \l_tmpa_bool
        \bool_set_true:N \l_tmpb_bool
      }
```

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`).

```
    \dim_gzero_new:N \g_@@_x_initial_dim
    \dim_gzero_new:N \g_@@_x_final_dim
    \dim_gzero_new:N \g_@@_y_initial_dim
    \dim_gzero_new:N \g_@@_y_final_dim
    \pgfpicture
      \pgfrememberpicturepositiononpagetrue
      \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south }
      \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
      \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
      \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north }
      \dim_gset:Nn \g_@@_x_final_dim \pgf@x
      \dim_gset:Nn \g_@@_y_final_dim \pgf@y
    \endpgfpicture
    \bool_lazy_and:nnTF
      {
        \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
                             > \l_@@_max_length_of_arrow_dim
      }
      { \int_compare_p:nNn { \l_@@_final_int - \l_@@_initial_int } = 1 }
      {
        \tl_gset:Ne \g_tmpa_tl
          {
            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
              { \dim_use:N \g_@@_x_initial_dim }
              { \dim_use:N \l_@@_x_dim } ,
            \dim_eval:n
              {
                ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
                + 0.5 \l_@@_max_length_of_arrow_dim
              }
          }
        \tl_gset:Ne \g_tmpb_tl
          {
            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
              { \dim_use:N \g_@@_x_final_dim }
              { \dim_use:N \l_@@_x_dim } ,
            \dim_eval:n
              {
                ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
                - 0.5 \l_@@_max_length_of_arrow_dim
              }
          }
      }
      {
        \tl_gset:Ne \g_tmpa_tl
          {
            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
              { \dim_use:N \g_@@_x_initial_dim }
              { \dim_use:N \l_@@_x_dim } ,
            \bool_if:NTF \l_tmpa_bool
              { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
              { \dim_use:N \g_@@_y_initial_dim }
          }
        \tl_gset:Ne \g_tmpb_tl
          {
            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
              { \dim_use:N \g_@@_x_final_dim }
              { \dim_use:N \l_@@_x_dim } ,
            \bool_if:NTF \l_tmpb_bool
              { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
              { \dim_use:N \g_@@_y_final_dim }
          }
      }
```

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

```
    \bool_if:NF \l_@@_right_overlap_bool
      {
        \bool_if:NT \l_@@_in_WithArrows_bool
          {
            \pgfpicture
            \pgfrememberpicturepositiononpagetrue
            \pgfpointanchor { wa - \l_@@_prefix_str - 1 - r } { south }
            \int_compare:nNnTF \l_@@_pos_arrow_int < 5
              {
                \dim_set:Nn \l_@@_delta_x_dim
                  {
                    \pgf@x -
                    ( \dim_min:nn \g_@@_x_initial_dim \g_@@_x_final_dim )
                  }
              }
              { \dim_set:Nn \l_@@_delta_x_dim { \pgf@x - \l_@@_x_dim } }
            \endpgfpicture
          }
      }
```

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*.[38]

[38] There were other solutions: use another name without *underscore* (like `\ltmpatl`) or use the package underscore

```
    \@@_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).

```
    \group_end:
  }
```

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

```
\cs_generate_variant:Nn \@@_def_function_arrow:n { o }
\cs_new_protected:Npn \@@_def_function_arrow:n #1
  {
    \cs_set:Npn \@@_arrow:nnn ##1 ##2 ##3
      {
⟨*LaTeX⟩
        \begin{tikzpicture}
⟨/LaTeX⟩
⟨*plain-TeX⟩
        \tikzpicture
⟨/plain-TeX⟩
        [ @@_standard_arrow ]
```

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

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

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

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

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

```
        \bool_if:NF \l_@@_right_overlap_bool
          {
            \bool_if:NT \l_@@_in_WithArrows_bool
              {
                \dim_gset:Nn \g_@@_overlap_x_dim
                  {
                    \dim_max:nn
                      \g_@@_overlap_x_dim
                      { \l_tmpa_dim - \l_@@_delta_x_dim }
                  }
              }
          }
        \pgfresetboundingbox
⟨*LaTeX⟩
        \end{tikzpicture}
⟨/LaTeX⟩
⟨*plain-TeX⟩
        \endtikzpicture
⟨/plain-TeX⟩
```

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

```
      }
  }
```

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

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

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`).

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

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

```
    \@@_def_function_arrow:o \l_@@_tikz_code_tl
    \@@_arrow:nnn { #1 } { #2 } { #3 }
  }
```

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`).

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

First, we draw the arrow without the label.

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

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

```
    \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.

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

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “`text width`”.[39]

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

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

```
    \tl_if_empty:NF \g_tmpa_tl
      {
        \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
        \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
          { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
      }
```

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

```
    \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
      {
        \path ( @@_label.west )
⟨*LaTeX⟩
           node [ anchor = west ]
            {
              \skip_horizontal:n { 0.33333 em }
              \begin { minipage } { \l_tmpa_dim }
              \tikz@text@action
              \pgfkeysgetvalue { / tikz / node~halign~header } \l_tmpa_tl
              \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@left@header }
                { \pgfutil@raggedright }
```

[39] 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.

```
                {
                  \tl_if_eq:NnTF \l_tmpa_tl { \tikz@align@right@header }
                    { \pgfutil@raggedleft }
                    {
                      \tl_if_eq:NnT \l_tmpa_tl { \tikz@align@center@header }
                        { \centering }
                    }
                }
              #3
              \end { minipage }
            } ;
⟨/LaTeX⟩
⟨*plain-TeX⟩
           node [ anchor = west , text~width = \dim_use:N \l_tmpa_dim ]
            { \skip_horizontal:n { 0.33333 em } #3 } ;
⟨/plain-TeX⟩
      }
  }
```

12.11.1 The command update_x

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

```
\cs_new_protected:Npn \@@_update_x:nn #1 #2
  {
    \dim_gset_eq:NN \g_tmpa_dim \l_@@_x_dim
    \pgfpicture
    \pgfrememberpicturepositiononpagetrue
    \int_step_inline:nnn { #1 } { #2 }
      {
        \pgfpointanchor { wa - \l_@@_prefix_str - ##1 - l } { center }
        \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \pgf@x }
      }
    \endpgfpicture
    \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
  }
```

12.11.2 We draw the arrows of type o

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

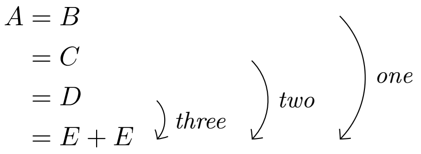
```
\cs_new_protected:Npn \@@_draw_o_arrows_of_the_group:
  {
```

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:

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



```
    \seq_sort:Nn \l_@@_o_arrows_seq
     {
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
       { final } \l_tmpa_tl
```

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

```
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
       { final } \l_tmpb_tl
      \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
       \sort_return_same:
       {
        \int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
         \sort_return_swapped:
         {
          \prop_get:cnN
           { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
           { initial } \l_tmpa_tl
          \prop_get:cnN
           { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
           { initial } \l_tmpb_tl
          \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
           \sort_return_swapped:
           \sort_return_same:
         }
       }
     }
```

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

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

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

```
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
       { initial } \l_tmpa_tl
      \int_set:Nn \l_@@_initial_int \l_tmpa_tl
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
       { final } \l_tmpa_tl
      \int_set:Nn \l_@@_final_int \l_tmpa_tl
```

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

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

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

```
      \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[40]).

```
    \bool_set_false:N \g_tmpa_bool
    \pgfpicture
    \pgfrememberpicturepositiononpagetrue
    \int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
     {
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
       { initial } \l_tmpa_tl
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
       { final } \l_tmpb_tl
      \prop_get:cnN
       { g_@@_arrow _ \l_@@_prefix_str _ ####1 _ prop }
       { status } \l_@@_status_arrow_str
      \bool_lazy_any:nF
       {
        { \int_compare_p:n { ##1 = ####1 } }
        { \int_compare_p:nNn \l_@@_initial_int > \l_tmpa_tl }
        { \int_compare_p:nNn \l_tmpb_tl > \l_@@_final_int }
```

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

```
        { \str_if_eq_p:on \l_@@_status_arrow_str { independent } }
       }
       {
```

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

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

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

```
        \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \l_tmpa_tl }
       }
     }
    \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`).

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

      \@@_draw_arrow:
     }
     { \@@_error:n { o~arrow~with~no~arrow~under } }
   }
 }
```

The command `\WithArrowsLastEnv` is not used by the package withtarrows. 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.

```
⟨*LaTeX⟩
\NewExpandableDocumentCommand \WithArrowsLastEnv { }
```

[40] Maybe we will change that in future versions.

```
  { \int_use:N \g_@@_last_env_int }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
⟨/plain-TeX⟩
```

12.12 The command \Arrow in code-after

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

```
\keys_define:nn { WithArrows / Arrow / code-after }
  {
    tikz     .code:n =
      \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
    tikz      .value_required:n   = true ,
    rr        .value_forbidden:n  = true ,
    rr        .code:n             = \@@_fix_pos_option:n 0 ,
    ll        .value_forbidden:n  = true,
    ll        .code:n             = \@@_fix_pos_option:n 1 ,
    rl        .value_forbidden:n  = true ,
    rl        .code:n             = \@@_fix_pos_option:n 2 ,
    lr        .value_forbidden:n  = true ,
    lr        .code:n             = \@@_fix_pos_option:n 3 ,
    v         .value_forbidden:n  = true ,
    v         .code:n             = \@@_fix_pos_option:n 4 ,
    tikz-code .tl_set:N          = \l_@@_tikz_code_tl ,
    tikz-code .value_required:n  = true ,
    xoffset   .dim_set:N          = \l_@@_xoffset_dim ,
    xoffset   .value_required:n   = true ,
    unknown   .code:n  =
      \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
      \@@_error:n { Unknown~option~Arrow~in~code-after }
  }
```

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.

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

```
⟨*LaTeX⟩
\NewDocumentCommand \@@_Arrow_code_after { O { } m m m ! O { } }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \@@_Arrow_code_after
  {
    \peek_meaning:NTF [
      { \@@_Arrow_code_after_i }
      { \@@_Arrow_code_after_i [ ] }
  }
\cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4
  {
    \peek_meaning:NTF [
      { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
      { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
  }
\cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
⟨/plain-TeX⟩
```

```
{
  \int_set_eq:NN \l_@@_pos_arrow_int \c_one_int
  \str_clear_new:N \l_@@_previous_key_str
  \group_begin:
    \keys_set:nn { WithArrows / Arrow / code-after }
      { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
    \bool_set_false:N \l_@@_initial_r_bool
    \bool_set_false:N \l_@@_final_r_bool
    \int_case:nn \l_@@_pos_arrow_int
      {
        0
          {
            \bool_set_true:N \l_@@_initial_r_bool
            \bool_set_true:N \l_@@_final_r_bool
          }
        2 { \bool_set_true:N \l_@@_initial_r_bool }
        3 { \bool_set_true:N \l_@@_final_r_bool }
      }
```

We prevent drawing an arrow from a line to itself.

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

```
      {
        \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - l }
          { \@@_error:ne { Wrong~line~in~Arrow } { #2 } }
          {
            \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - l }
              { \@@_error:ne { Wrong~line~in~Arrow } { #3 }  }
              {
                \int_compare:nNnTF \l_@@_pos_arrow_int = 4
                  {
                    \pgfpicture
                      \pgfrememberpicturepositiononpagetrue
                      \pgfpointanchor { wa - \l_@@_prefix_str - #2 - l }
                        { south }
                      \dim_set_eq:NN \l_tmpa_dim \pgf@x
                      \dim_set_eq:NN \l_tmpb_dim \pgf@y
                      \pgfpointanchor { wa - \l_@@_prefix_str - #3 - l }
                        { north }
                      \dim_set:Nn \l_tmpa_dim
                        { \dim_max:nn \l_tmpa_dim \pgf@x }
                      \tl_gset:Ne \g_tmpa_tl
                        { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
                      \tl_gset:Ne \g_tmpb_tl
                        { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
                    \endpgfpicture
                  }
                  {
                    \pgfpicture
                      \pgfrememberpicturepositiononpagetrue
                      \pgfpointanchor
                        {
                          wa - \l_@@_prefix_str -
                          #2 - \bool_if:NTF \l_@@_initial_r_bool r l
                        }
                        { south }
                      \tl_gset:Ne \g_tmpa_tl
                        { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
                      \pgfpointanchor
                        {
                          wa - \l_@@_prefix_str -
```

```
                            #3 - \bool_if:NTF \l_@@_final_r_bool r l
                          }
                          { north }
                        \tl_gset:Ne \g_tmpb_tl
                          { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
                      \endpgfpicture
                    }
                  \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
                }
            }
        }
    \group_end:
  }
```

12.13 The command \MultiArrow in code-after

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

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

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.

```
    \exp_args:Nne
      \regex_match:nnTF
      { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
      { #1 }
      { \@@_MultiArrow_i:nn { #1 } { #2 } }
      { \@@_error:ne { Invalid~specification~for~MultiArrow } { #1 } }
  }
```

```
\cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
  {
```

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

```
    \foreach \x in { #1 }
      {
        \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
          { \@@_error:ne { Wrong~line~specification~in~MultiArrow } \x }
          { \clist_gput_right:Ne \g_tmpa_clist \x }
      }
```

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

```
    \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
      { \@@_error:n { Too~small~specification~for~MultiArrow } }
      {
        \clist_sort:Nn \g_tmpa_clist
          {
            \int_compare:nNnTF { ##1 } > { ##2 }
              \sort_return_swapped:
              \sort_return_same:
          }
```

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

```
        \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...

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

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

Now, we draw the rest of the structure.

```
⟨*LaTeX⟩
        \begin { tikzpicture }
⟨/LaTeX⟩
⟨*plain-TeX⟩
        \tikzpicture
⟨/plain-TeX⟩
          [
            @@_standard ,
            every~path / .style = { WithArrows / arrow }
          ]
          \draw [<->] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
                      -- ++(5mm,0)
                      -- node (@@_label) {}
                         ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
                      -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south)  ;
          \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }
          \dim_set:Nn \l_tmpa_dim { 20 cm }
          \path \pgfextra { \tl_gset:Ne \g_tmpa_tl \tikz@text@width } ;
          \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
          \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
            {
              \dim_set:Nn \l_tmpb_dim
                { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
              \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
                { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
            }
          \path (@@_label.west)
           node [ anchor = west, text~width = \dim_use:N \l_tmpa_dim ] { #2 } ;
⟨*LaTeX⟩
        \end { tikzpicture }
⟨/LaTeX⟩
⟨*plain-TeX⟩
        \endtikzpicture
⟨/plain-TeX⟩
      }
  }

\cs_new_protected:Npn \@@_MultiArrow_i:n #1
  {
⟨*LaTeX⟩
        \begin { tikzpicture }
⟨/LaTeX⟩
⟨*plain-TeX⟩
        \tikzpicture
⟨/plain-TeX⟩
      [
        @@_standard ,
        every~path / .style = { WithArrows / arrow }
      ]
      \foreach \k in { #1 }
        {
          \draw [ <- ]
            ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
        } % ;
⟨*LaTeX⟩
        \end{tikzpicture}
⟨/LaTeX⟩
```

```
⟨*plain-TeX⟩
        \endtikzpicture
⟨/plain-TeX⟩
  }
```

12.14 The error messages of the package

```
\bool_if:NTF \c_@@_messages_for_Overleaf_bool
  { \str_const:Nn \c_@@_available_keys_str { } }
  {
    \str_const:Nn \c_@@_available_keys_str
      { For~a~list~of~the~available~keys,~type~H~<return>. }
  }
\str_new:N \l_witharrows_body_str
```

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

```
\cs_new:Npn \@@_potential_body_i:
  {
    \str_if_empty:NF \l_witharrows_body_str
      { \\ If~you~want~to~see~the~body~of~the~environment,~type~H~<return>. }
  }
\cs_new:Npn \@@_potential_body_ii:
  {
    \str_if_empty:NTF \l_witharrows_body_str
      { No~further~help~available }
      {
        The~body~of~your~environment~was:\\
        \l_witharrows_body_str
      }
  }
\str_const:Nn \c_@@_option_ignored_str
  { If~you~go~on,~this~option~will~be~ignored. }
\str_const:Nn \c_@@_command_ignored_str
  { If~you~go~on,~this~command~will~be~ignored. }
⟨*LaTeX⟩
\@@_msg_new:nn { amsmath~not~loaded }
  {
    amsmath~not~loaded.\\
    You~can't~use~the~option~'\l_keys_key_str'~because~the~
    package~'amsmath'~has~not~been~loaded.\\
    If~you~go~on,~this~option~will~be~ignored~in~the~rest~
    of~the~document.
  }
⟨/LaTeX⟩
\@@_msg_new:nn { Bad~value~for~replace~brace~by }
  {
    Incorrect~value.\\
    Bad~value~for~the~option~'\l_keys_key_str'.~The~value~must~begin~
    with~an~extensible~left~delimiter.~The~possible~values~are:~.,
    \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
    \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
    \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
    \token_to_str:N \lfloor\ and~\token_to_str:N \lceil\
    (and~\token_to_str:N \lvert\ and~\token_to_str:N \lVert\
    if~amsmath~or~unicode-math~is~loaded~in~LaTeX).\\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { option~of~cr~negative }
  {
    Bad~value.\\
    The~argument~of~the~command~\token_to_str:N\\~
    should~be~positive~in~the~row~\int_use:N \g_@@_line_int\
    of~your~environment~\{\l_@@_type_env_str\}.\\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { omit~probably~used }
  {
    Strange~problem.\\
    Maybe~you~have~used~a~command~
    \token_to_str:N\omit\ in~the~line~\int_use:N \g_@@_line_int\
    (or~another~line)~of~your~environment~\{\l_@@_type_env_str\}.\\
    You~can~go~on~but~you~may~have~others~errors.
  }
⟨*LaTeX⟩
\@@_msg_new:nnn { newline~at~the~end~of~env }
  {
    Incorrect~end.\\
    The~environments~of~witharrows~(\{WithArrows\}~and~
    \{DispWithArrows\})~should~not~end~by~\token_to_str:N \\.\\
    However,~you~can~go~on~for~this~time.~No~similar~error~will~be~
    raised~in~this~document.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
⟨/LaTeX⟩
\@@_msg_new:nnn { Invalid~option~format }
  {
    Invalide~value.\\
    The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
    must~not~be~empty.\\
    \c_@@_option_ignored_str
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nnn { invalid~key~o }
  {
    Invalid~use~of~a~key.\\
    The~key~'o'~for~individual~arrows~can~be~used~only~in~mode~
    'group'~or~in~mode~'groups'.\\
    \c_@@_option_ignored_str
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nnn { Value~for~a~key }
  {
    Misuse~of~a~key.\\
    The~key~'\l_keys_key_str'~should~be~used~without~value. \\
    However,~you~can~go~on~for~this~time.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nnn { Unknown~option~in~Arrow }
  {
    Unknown~option.\\
    The~key~'\l_keys_key_str'~is~unknown~for~the~command~
    \l_@@_string_Arrow_for_msg_str\ in~the~row~
    \int_use:N \g_@@_line_int\ of~your~environment~
    \{\l_@@_type_env_str\}. \l_tmpa_str \\
    \c_@@_option_ignored_str \\
    \c_@@_available_keys_str
  }
  {
    The~available~keys~are~(in~alphabetic~order):~
    \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
  }
\@@_msg_new:nnn { Unknown~option~WithArrows }
  {
    Unknown~option.\\
    The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
    \c_@@_option_ignored_str \\
    \c_@@_available_keys_str
  }
  {
    The~available~keys~are~(in~alphabetic~order):~
    \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.
  }
\@@_msg_new:nnn { Unknown~option~DispWithArrows }
  {
    Unknown~option.\\
    The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
    \c_@@_option_ignored_str \\
    \c_@@_available_keys_str
  }
  {
    The~available~keys~are~(in~alphabetic~order):~
    \seq_use:Nnnn \l_@@_options_DispWithArrows_seq {~and~} {,~} {~and~}.
  }
\@@_msg_new:nnn { Unknown~option~WithArrowsOptions }
  {
    Unknown~option.\\
    The~key~'\l_keys_key_str'~is~unknown~in~
    \token_to_str:N \WithArrowsOptions. \\
    \c_@@_option_ignored_str \\
    \c_@@_available_keys_str
  }
  {
    The~available~keys~are~(in~alphabetic~order):~
    \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
  }
\@@_msg_new:nnn { Unknown~option~Arrow~in~code-after }
  {
    Unknown~option.\\
    The~key~'\l_keys_key_str'~is~unknown~in~
    \token_to_str:N \Arrow\ in~code-after. \\
    \c_@@_option_ignored_str \\
    \c_@@_available_keys_str
  }
  {
    The~available~keys~are~(in~alphabetic~order):~
    \seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
  }
\@@_msg_new:nnn { Too~much~columns~in~WithArrows }
  {
    Too~much~columns.\\
    Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
    \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
    Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
    If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
    the~option~'more-columns'~at~a~global~level~or~for~an~environment. \\
    However,~you~can~go~one~for~this~time.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
```

```
\@@_msg_new:nnn { Too~much~columns~in~DispWithArrows }
  {
    Too~much~columns.\\
    Your~environment~\{\l_@@_type_env_str\}~has~\int_use:N
    \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
    Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\
    at~the~end~of~row~\int_use:N \g_@@_line_int. \\
    This~error~is~fatal.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nn { Negative~jump }
  {
    Incorrect~value.\\
    You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
    \l_@@_string_Arrow_for_msg_str\
    in~the~row~\int_use:N \g_@@_line_int\
    of~your~environment~\{\l_@@_type_env_str\}.~
    You~can~create~an~arrow~going~backwards~with~the~option~'<-'~of~Tikz. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { new-group~without~groups }
  {
    Misuse~of~a~key.\\
    You~can't~use~the~option~'new-group'~for~the~command~
    \l_@@_string_Arrow_for_msg_str\
    because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
    'groups'~in~your~environment~\{\l_@@_type_env_str\}. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nnn
  { Too~few~lines~for~an~arrow }
  {
    Impossible~arrow.\\
    Line~\l_@@_input_line_str\
    :~an~arrow~specified~in~the~row~\int_use:N \l_@@_initial_int\
    of~your~environment~\{\l_@@_type_env_str\}~can't~be~drawn~
    because~it~arrives~after~the~last~row~of~the~environment. \\
    If~you~go~on,~this~arrow~will~be~ignored.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nn { o~arrow~with~no~arrow~under }
  {
    Problem~with~the~key~'o'.\\
    Line~\l_@@_input_line_str\
    :~there~is~no~arrow~'under'~your~arrow~of~type~'o'.\\
    If~you~go~on,~this~arrow~won't~be~drawn.
  }
\@@_msg_new:nnn { WithArrows~outside~math~mode }
  {
    You~are~outside~math~mode.\\
    The~environment~\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
    like~the~environment~\{aligned\}~of~amsmath. \\
    Nevertheless,~you~can~go~on.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nnn { DispWithArrows~in~math~mode }
  {
    You~are~in~math~mode.\\
    The~environment~\{\l_@@_type_env_str\}~should~be~used~only~outside~math~
    mode~like~the~environments~\{align\}~and~\{align*\}~of~amsmath. \\
    This~error~is~fatal.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nn { Incompatible~options~in~Arrow }
  {
    Incompatible~options.\\
    You~try~to~use~the~option~'\l_keys_key_str'~but~
    this~option~is~incompatible~or~redundant~with~the~option~
    '\l_@@_previous_key_str'~set~in~the~same~command~
    \l_@@_string_Arrow_for_msg_str. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { Incompatible~options a}
  {
    Incompatible~options.\\
    You~try~to~use~the~option~'\l_keys_key_str'~but~
    this~option~is~incompatible~or~redundant~with~the~option~
    '\l_@@_previous_key_str'~set~in~the~same~command~
    \bool_if:NT \l_@@_in_code_after_bool
      {
        \l_@@_string_Arrow_for_msg_str\
        in~the~code-after~of~your~environment~\{\l_@@_type_env_str\}
      }. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nnn { Arrow~not~in~last~column }
  {
    Bad~use~of~\l_@@_string_Arrow_for_msg_str.\\
    You~should~use~the~command~\l_@@_string_Arrow_for_msg_str\
    only~in~the~last~column~(column~\int_use:N\l_@@_nb_cols_int)~
    in~the~row~\int_use:N \g_@@_line_int\
    of~your~environment~\{\l_@@_type_env_str\}.\\
    However~you~can~go~on~for~this~time.
    \@@_potential_body_i:
  }
  { \@@_potential_body_ii: }
\@@_msg_new:nn { Wrong~line~in~Arrow }
  {
    Wrong~line.\\
    The~specification~of~line~'#1'~you~use~in~the~command~
    \l_@@_string_Arrow_for_msg_str\
    in~the~'code-after'~of~\{\l_@@_type_env_str\}~doesn't~exist. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { Both~lines~are~equal }
  {
    Both~lines~are~equal.\\
    In~the~'code-after'~of~\{\l_@@_type_env_str\}~you~try~to~
    draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }
  {
    Wrong~line~specification.\\
    The~specification~of~line~'#1'~doesn't~exist. \\
    If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
  }
\@@_msg_new:nn { Too~small~specification~for~MultiArrow }
```

```
  {
    Too~small~specification.\\
    The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
    is~too~small:~you~need~at~least~two~lines. \\
    \c_@@_command_ignored_str
  }
\@@_msg_new:nn { Not~allowed~in~DispWithArrows }
  {
    Forbidden~command.\\
    The~command~\token_to_str:N #1
    is~allowed~only~in~the~last~column~
    (column~\int_use:N\l_@@_nb_cols_int)~of~\{\l_@@_type_env_str\}. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nn { Not~allowed~in~WithArrows }
  {
    Forbidden~command.\\
    The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
    (it's~allowed~in~the~last~column~of~\{DispWithArrows\}). \\
    \c_@@_option_ignored_str
  }
⟨*LaTeX⟩
\@@_msg_new:nn { tag*~without~amsmath }
  {
    amsmath~not~loaded.\\
    We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
    (or~mathtools). \\
    If~you~go~on,~the~command~\token_to_str:N\tag\
    will~be~used~instead.
  }
\@@_msg_new:nn { Multiple~tags }
  {
    Multiple~tags.\\
    You~can't~use~twice~the~command~\token_to_str:N\tag\
    in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
    If~you~go~on,~the~tag~'#1'~will~be~used.
  }
\@@_msg_new:nn { Multiple~labels }
  {
    Multiple~labels.\\
    Normally,~we~can't~use~the~command~\token_to_str:N\label\
    twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
    However,~you~can~go~on.~
    \IfPackageLoadedT { showlabels }
      { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
    If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
    'allow-multiple-labels'~at~the~global~or~environment~level.
  }
\@@_msg_new:nn { Multiple~labels~with~cleveref }
  {
    Multiple~labels.\\
    Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
    twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
    If~you~go~on,~you~may~have~undefined~references.
  }
⟨/LaTeX⟩
\@@_msg_new:nn { Inexistent~v-node }
  {
    There~is~a~problem.\\
    Maybe~you~have~put~a~command~\token_to_str:N\cr\
    instead~of~a~command~\token_to_str:N\\~at~the~end~of~
    the~row~\l_tmpa_int\
    of~your~environment~\{\l_@@_type_env_str\}. \\
    This~error~is~fatal.
  }
```

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`).

```
\@@_msg_new:nn { Option~xoffset~forbidden }
  {
    Incorrect~key.\\
    You~can't~use~the~option~'xoffset'~in~the~command~
    \l_@@_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g_@@_line_int\
    of~your~environment~\{\l_@@_type_env_str\}~
    because~you~are~using~the~option~
    ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7
        { group }
        { groups } '.~It's~possible~for~an~independent~arrow~or~if~there~is~
    only~one~arrow. \\
    \c_@@_option_ignored_str
  }
\@@_msg_new:nnn { Duplicate~name }
  {
    Duplicate~name.\\
    The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
    the~same~environment~name~twice.~You~can~go~on,~but,~
    maybe,~you~will~have~incorrect~results. \\
    For~a~list~of~the~names~already~used,~type~H~<return>. \\
    If~you~don't~want~to~see~this~message~again,~use~the~option~
    'allow-duplicate-names'.
  }
  {
    The~names~already~defined~in~this~document~are:~
    \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
  }
\@@_msg_new:nn { Invalid~specification~for~MultiArrow }
  {
    Invalid~specification.\\
    The~specification~of~rows~for~\token_to_str:N\MultiArrow\
    (i.e.~#1)~is~invalid. \\
    \c_@@_command_ignored_str
  }
```

12.15 The command \WithArrowsNewStyle

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

```
⟨*LaTeX⟩
\NewDocumentCommand \WithArrowsNewStyle { m m }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\cs_new_protected:Npn \WithArrowsNewStyle #1 #2
⟨/plain-TeX⟩
  {
    \keys_if_exist:nnTF { WithArrows / Global } { #1 }
      { \@@_error:nn { Key~already~defined } { #1 } }
      {
```

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

```
        \seq_clear:N \l_tmpa_seq
        \keyval_parse:NNn \@@_valid_key:n \@@_valid_key:nn { #2 }
        \seq_if_empty:NTF \l_tmpa_seq
          {
            \seq_put_right:Ne \l_@@_options_WithArrows_seq
              { \tl_to_str:n { #1 } }
            \seq_put_right:Ne \l_@@_options_DispWithArrows_seq
              { \tl_to_str:n { #1 } }
            \seq_put_right:Ne \l_@@_options_WithArrowsOptions_seq
              { \tl_to_str:N { #1 } }
            \keys_precompile:nnc
              { WithArrows / WithArrowsOptions }
              { #2 }
              { @@ _ style _ #1 _ tl }
            \keys_define:nn { WithArrows / Global }
              { #1 .code:n = \use:c { @@ _ style _ #1 _ tl } }
          }
          { \@@_error:nn { Impossible~style } { #1 } }
      }
  }
\@@_msg_new:nn { Impossible~style }
  {
    Impossible~style.\\
    It's~impossible~to~define~the~style~'#1'~
    because~it~contains~unknown~keys:~'
    \seq_use:Nnnn \l_tmpa_seq { '~and~' } { ',~'} { ',~and~}'.
  }
\cs_new_protected:Npn \@@_valid_key:n #1
  {
    \keys_if_exist:nnF { WithArrows / Global } { #1 }
      { \seq_put_right:Nn \l_tmpa_seq { #1 } }
  }
\cs_new_protected:Npn \@@_valid_key:nn #1 #2
  {
    \keys_if_exist:nnF { WithArrows / Global } { #1 }
      { \seq_put_right:Nn \l_tmpa_seq { #1 } }
  }
\@@_msg_new:nn { Key~already~defined }
  {
    Key~already~defined.\\
    The~key~'#1'~is~already~defined. \\
    If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
    will~be~ignored.
  }
```

12.16 The options up and down

The options up and down are available for individual arrows. The corresponding code is given here. It is independent of the main code of the extension witharrows.
This code is the only part of the code of witharrows which uses the the Tikz library calc. That's why we have decided not to load by default this library. If it is not loaded, the user will have an error only when using the option up or the option down.

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

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

```
\keys_define:nn { WithArrows / up-and-down }
  {
    radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
    radius .value_required:n = true ,
    width .code:n =
      \str_case:nnF { #1 }
        {
          { min } { \dim_zero:N \l_@@_arrow_width_dim }
          { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
        }
        { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
    width .value_required:n = true ,
    unknown .code:n = \@@_error:n { Option~unknown~for~up-and-down }
  }
\@@_msg_new:nn { Option~unknown~for~up-and-down }
  {
    Unknown~option.\\
    The~option~'\l_keys_key_str'~is~unknown.~\c_@@_option_ignored_str
  }
```

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

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

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

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

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

```
                  \narrowragged
                  #3
                \end { varwidth }
              }
              (\x2,\y1) -- (\p2) ;
      }
```

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`.

```
      {
        \path
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in node
              {
```

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

```
                \dim_set:Nn \l_tmpa_dim
                  { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
                \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
                  {
                    \dim_set:Nn \l_tmpa_dim
                      { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
                  }
```

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.

```
                \hbox_gset:Nn \g_tmpa_box
                  {
                    \begin { varwidth } \l_tmpa_dim
                      \narrowragged
                      #3
                    \end { varwidth }
                  }
```

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

```
                \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
                  { \dim_gset_eq:NN  \g_tmpa_dim \l_@@_arrow_width_dim }
                  { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
                \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
              } ;
       \draw
         let \p1 = ( #1 ) , \p2 = ( #2 )
         in (\x2-\g_tmpa_dim,\y1)
            -- node { \box_use:N \g_tmpa_box }
            (\x2-\l_@@_up_and_down_radius_dim,\y1)
            [ rounded~corners = \l_@@_up_and_down_radius_dim ]
            -| (\p2) ;
      }
  }
⟨/LaTeX⟩
⟨*plain-TeX⟩
\tl_const:Nn \c_@@_tikz_code_up_tl
  {
    \dim_case:nnF \l_@@_arrow_width_dim
      {
        \c_max_dim
          {
            \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
              let \p1 = ( #1 ) , \p2 = ( #2 )
              in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
          }
        \c_zero_dim
          {
            \path node
              {
                \hbox_gset:Nn \g_tmpa_box { #3 }
                \dim_gset:Nn \g_tmpa_dim
                  { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
              } ;
            \draw
              let \p1 = ( #1 ) , \p2 = ( #2 )
              in (\x2-\g_tmpa_dim,\y1)
                 -- node { \box_use:N \g_tmpa_box }
                 (\x2-\l_@@_up_and_down_radius_dim,\y1)
                 [ rounded~corners = \l_@@_up_and_down_radius_dim ]
                 -| (\p2) ;
          }
      }
      {
        \draw
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
             -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
             [ rounded~corners = \l_@@_up_and_down_radius_dim ]
             -| (\p2) ;
      }
  }
⟨/plain-TeX⟩
```

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

```
⟨*LaTeX⟩
\tl_const:Nn \c_@@_tikz_code_down_tl
  {
    \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
      {
        \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in (\p1) -- (\x1,\y2) -- node
             {
               \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
               \begin { varwidth } \l_tmpa_dim
                 \narrowragged
                 #3
               \end { varwidth }
             }
             (\p2) ;
      }
      {
        \path
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in node
              {
                \hbox_gset:Nn \g_tmpa_box
                  {
                    \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.).

```
                      { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
                    \begin { varwidth } \l_tmpa_dim
                      \narrowragged
                      #3
                    \end { varwidth }
                  }
                \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
                  { \dim_gset_eq:NN  \g_tmpa_dim \l_@@_arrow_width_dim }
                  { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
                \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
              } ;

        \draw
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in (\p1)
             { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
             -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
             -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
             -- ++ (-2mm,0) ;
      }
  }
⟨/LaTeX⟩
%
⟨*plain-TeX⟩
\tl_const:Nn \c_@@_tikz_code_down_tl
  {
    \dim_case:nnF \l_@@_arrow_width_dim
      {
        \c_max_dim
          {
            \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
              let \p1 = ( #1 ) , \p2 = ( #2 )
              in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
          }
        \c_zero_dim
          {
            \path node
              {
                \hbox_gset:Nn \g_tmpa_box { #3 }
                \dim_gset:Nn \g_tmpa_dim
                  { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
              } ;
            \draw
              let \p1 = ( #1 ) , \p2 = ( #2 )
              in (\p1)
                 { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
                 -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
                 -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
                 -- ++ (-2mm,0) ;
          }
      }
      {
        \draw
          let \p1 = ( #1 ) , \p2 = ( #2 )
          in (\p1)
             { [ rounded~corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
             -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
             -- node { #3 }
                (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
             -- ++ (-2mm,0) ;
      }
  }
⟨/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.

```
\keys_define:nn { WithArrows / Arrow / FirstPass }
  {
    up   .code:n = \@@_set_independent_bis: ,
    down .code:n = \@@_set_independent_bis: ,
    up   .default:n = NoValue ,
    down .default:n = NoValue
  }
```

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

```
\keys_define:nn { WithArrows / Arrow / SecondPass }
  {
    up .code:n =
      \str_if_empty:NT \l_@@_previous_key_str
        {
          \str_set:Nn \l_@@_previous_key_str { up }
          \cs_if_exist:NTF \tikz@library@calc@loaded
            {
              \keys_set:no { WithArrows / up-and-down } \l_keys_value_tl
              \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.

```
              \bool_set_false:N \l_@@_wrap_lines_bool
```

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

```
              \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
            }
            { \@@_error:n { calc~not~loaded } }
        } ,
    down .code:n =
      \str_if_empty:NT \l_@@_previous_key_str
        {
          \str_set:Nn \l_@@_previous_key_str { down }
          \cs_if_exist:NTF \tikz@library@calc@loaded
            {
              \keys_set:no { WithArrows / up-and-down } \l_keys_value_tl
              \int_set:Nn \l_@@_pos_arrow_int 1
              \bool_set_false:N \l_@@_wrap_lines_bool
              \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
            }
            { \@@_error:n { calc~not~loaded } }
        }
  }
\seq_put_right:Nn \l_@@_options_Arrow_seq { down }
\seq_put_right:Nn \l_@@_options_Arrow_seq { up }
\@@_msg_new:nn { calc~not~loaded }
  {
    calc~not~loaded.\\
    You~can't~use~the~option~'\l_keys_key_str'~because~you~don't~have~loaded~the~
    Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
    ~in~the~preamble~of~your~document. \\
    \c_@@_option_ignored_str
  }
⟨*plain-TeX⟩
\catcode `\@ = 12
\ExplSyntaxOff
⟨/plain-TeX⟩
```

13 History

Changes between 2.8 and 2.9

Argument `<...>` for the command `\Arrow` in the class Beamer.

Changes between 2.7 and 2.8

New key `right-overlap`

Changes between 2.6b and 2.7

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

Changes between 2.6 and 2.6a (and 2.6b)

Replacement of `\hbox_unpack_clear:N` by `\hbox_unpack_drop:N` since `\hbox_unpack_clear:N` is now deprecated in L3.
Version 2.6d: correction of a bug (cf. question 628461 on TeX StackExchange).

Changes between 2.5 and 2.5.1

Correction of the erroneous programmation of the nodes aliases.

Changes between 2.4 and 2.5

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

Changes between 2.3 and 2.4

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

Changes between 2.2 and 2.3

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

Changes between 2.1 and 2.2

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

Changes between 2.0 and 2.1

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

Changes between 1.18 and 2.0

A version of witharrows is available for plain-TeX.

Changes between 1.17 and 1.18

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

Changes between 1.16 and 1.17

Option `format`.

Changes between 1.15 and 1.16

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

Changes between 1.14 and 1.15

Option `new-group` to start a new group of arrows (only available when the environment is composed with the option `groups`).
Tikz externalization is now deactivated in the environments of the extension witharrows.[41]

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

Changes between 1.13 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 1.12 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 1.11 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 1.10 and 1.11

New commands \WithArrowsNewStyle and \WithArrowsRightX.

Changes between 1.9 and 1.10

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

Changes between 1.8 and 1.9

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

Changes between 1.7 and 1.8

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

Changes between 1.6 and 1.7

New environments {DispWithArrows} and {DispWithArrows*}.

Changes between versions 1.5 and 1.6

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

Changes between versions 1.4 and 1.5

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

Changes between versions 1.3 and 1.4

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

Changes between versions 1.2 and 1.3

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

Changes between versions 1.1 and 1.2

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

Changes between versions 1.0 and 1.1

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

Contents