

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

F. Pantigny
fpantigny@wanadoo.fr

June 22, 2020

Abstract

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

In this document, we describe the LaTeX extension `witharrows` (however, `witharrows` can also be used with plain-TeX: see p. 23). This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). This package loads the packages `l3keys2e`, `xparse`, `varwidth`, `tikz` and the Tikz libraries `arrows.meta` and `bending`. The arrows are drawn with Tikz and that's why several compilations may be necessary.

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

```


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


```

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

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

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

1 Options for the shape of the arrows

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

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

```

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

```

*This document corresponds to the version 2.5 of witharrows, at the date of 2020/06/22.

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

```

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

```

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

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

Without the option `displaystyle`:

```

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

```

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

The same example with the option `displaystyle`:

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

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

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

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

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

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

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

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

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

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

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

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

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

2 Numbers of columns

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

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

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

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

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

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

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

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

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

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

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

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

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

Therefore $I = \int_{\frac{\pi}{4}}^0 \ln\left(1 + \tan\left(\frac{\pi}{4} - u\right)\right)(-du)$ This arrow uses the **lr** option.

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

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

```

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

```

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

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

```

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

```

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

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

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

In an environment which uses the option `group` or the option `groups`, it’s still possible to give an option of position (`ll`, `lr`, `rl`, `rr` or `i`) to an individual arrow⁸. Such arrow will be drawn irrespective of the groups. It’s also possible to start a new group by applying the option `new-group` to an given arrow.

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

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

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

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

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

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

4 The option “o” for individual arrows

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

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

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

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

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

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

```

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

```

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

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}$
```

$$\begin{array}{lcl}
 A & = & B \\
 & = & C \\
 & = & D \\
 & = & E + E + E + E + E + E + E
 \end{array}
 \begin{array}{c}
 \downarrow \textit{one} \\
 \downarrow \textit{two} \\
 \downarrow \textit{three}
 \end{array}$$

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

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

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

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

$$\begin{array}{lcl}
 A & = & B \\
 & = & C + C + C + C + C + C + C + C \\
 & = & C + C + C + C + C + C + C + C \\
 & = & E + E
 \end{array}
 \begin{array}{c}
 \xrightarrow{\textit{an arrow of type up}} \\
 \downarrow \\
 \xleftarrow{\textit{an arrow of type down}}
 \end{array}$$

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

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

- The key `radius` is the radius of the rounded corner of the arrow.¹⁰
- The key `width` is the width of the (horizontal part of) the arrow:
 - with the value `max`, the width of the arrow is adjusted with respect of the position of the nodes (that’s the behaviour by default of the arrows `up` and `down` as shown in the previous example);

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

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

```

 $\begin{WithArrows}$ 
 $A = B$ 
 $\xrightarrow[\text{we try}]{\text{radius=0pt,width=2cm}} \quad \backslash \backslash$ 
 $= C + C + C + C + C + C + C + C + C$ 
 $\end{WithArrows}$ 

```

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

```

 $\begin{WithArrows}$ 
 $A = B$ 
 $\xrightarrow[\text{we try}]{\text{width=min}} \quad \backslash \backslash$ 
 $= C + C + C + C + C + C + C + C + C$ 
 $\end{WithArrows}$ 

```

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

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

```

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

```

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 `\x` of end of line (it's also possible to use `\x*` but it has exactly the same effect as `\x` since an environment `{WithArrows}` is always unbreakable). This option is designed to be used with positive values only.

```

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

```

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

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

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

```

 $\begin{WithArrows}[displaystyle,jot=2ex]$ 

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


$$= H + \frac{1}{2}K$$


$$= K$$

 $\end{WithArrows}$ 

```

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

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

```

 $\begin{WithArrows}[jot=2ex]$ 

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


$$\Leftrightarrow \left\{ \begin{array}{l} x+y=0 \\ x+2y=0 \end{array} \right.$$

 $\end{WithArrows}$ 

```

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

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

```

 $\begin{WithArrows}[interline=2ex]$ 

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


$$\Leftrightarrow \left\{ \begin{array}{l} x+y=0 \\ x+2y=0 \end{array} \right.$$

 $\end{WithArrows}$ 

```

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

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

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

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

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

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

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

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

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

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

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

`\WithArrowsOptions{displaystyle}`.

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

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

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

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

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

7 Arrows in nested environments

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

```

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

```

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

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

```

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

```

```

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

```

$$\begin{array}{lcl}
A = B & \leftarrow & \\
= C & \leftarrow & \\
= D & \leftarrow & \text{text} \\
= E & \leftarrow & \\
= F & &
\end{array}$$

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

8 Arrows from outside environments `{WithArrows}`

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

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

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

$$\begin{array}{lcl}
A \triangleleft B + B + B + B + B + B + B + B + B + B + B + B + B + B & & B_{\text{wa-45-1-r}} \\
\triangleleft \left\{ \begin{array}{l} C \triangleleft D_{\text{wa-45-1-1-r}} \\ E \triangleleft F_{\text{wa-45-1-2-r}} \end{array} \right. & & \text{wa-45-2-r} \\
\triangleleft \left\{ \begin{array}{l} G \triangleleft H + H + H + H + H + H + H + H \\ I \triangleleft \left\{ \begin{array}{l} J \triangleleft K_{\text{wa-45-2-1-1-r}} \\ L \triangleleft M_{\text{wa-45-2-1-2-r}} \end{array} \right. \end{array} \right. & & \begin{array}{l} \text{wa-45-2-1-r} \\ \text{wa-45-2-2-r} \end{array} \\
\triangleleft \left\{ \begin{array}{l} N \triangleleft O_{\text{wa-45-3-1-r}} \\ P \triangleleft Q_{\text{wa-45-3-2-r}} \end{array} \right. & & \text{wa-45-4-r}
\end{array}$$

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 arrow¹⁵;
- 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.

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

¹⁵More 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}

```

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

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

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

```

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

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

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

```

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

9 The environment `{DispWithArrows}`

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

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

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

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

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

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

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

A link to the equation [\(★\)](#).¹⁶

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

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

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

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

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

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

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

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

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

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

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

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

With the option `fleqn`, the environment is composed flush left (in a way similar to the option `fleqn` of the standard classes of LaTeX). In this case, the left margin can be controlled with the option `mathindent` (with a name inspired by the parameter `\mathindent` of standard LaTeX¹⁹). The initial value of this parameter is 25 pt.

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

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

$$\tag{5}$$

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

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

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

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

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

First environment.

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

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

Second environment.

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

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

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

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

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

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

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

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

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

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

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

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

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

be used directly after a `\item` of a LaTeX list. In this case, no vertical space is added before the environment.²¹

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

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

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

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

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

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

Concerning the references, the package `witharrows` is compatible with the extensions `autonum`, `cleveref`, `fancyref`, `hyperref`, `listbls`, `prettyref`, `refcheck`, `refstyle`, `showlabels`, `smartref`, `typedref` and `varioref`, and with the options `showonlyrefs` and `showmanualtags` of `mathtools`.²²

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

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

²²We recall that `varioref`, `hyperref`, `cleveref` and `autonum` must be loaded in this order. The package `witharrows` can be loaded anywhere.

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

The environment `{DispWithArrows}` provides an option `left-brace`. When present, the value of this option is composed on the left, followed by a curly brace (hence the name) and the body of the environment.²³

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

The following code is an example of multi-case equations.²⁴

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

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

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

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

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

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

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

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

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

10 Advanced features

10.1 Utilisation with plain-TeX

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

```
\input{witharrows}
```

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

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

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

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

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

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

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

By default, the value is the following:

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

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

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

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

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

The environments `{DispWithArrows}` and its starred version `{DispWithArrows*}` provide a command `\WithArrowsRightX` which can be used in a definition of `tikz-code`. This command gives the x -value of the right side of the composition box (taking into account the eventual tags of the equations). For an example of use, see p. 29.

10.3 The command `\WithArrowsNewStyle`

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

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

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

For an example of use, see p. 29.

10.4 Vertical positioning of the arrows

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

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

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

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

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

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

```

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

```


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

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

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

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

Since the version 2.1 of **witharrows**, an arrow of **jump** equal to 1 has a maximal length²⁶ equal to the parameter **max-length-of-arrow**. The initial value of this parameter is 2 cm.

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

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

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

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

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

10.5 Footnotes in the environments of witharrows

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

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

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

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

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

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

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

10.6 Option no-arrows

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

10.7 Note for the users of AUCTeX

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

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

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

10.8 Note for developpers

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

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

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

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

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

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

²⁷A footnote.

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

11 Examples

11.1 `\MoveEqLeft`

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

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

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

11.2 A command `\DoubleArrow`

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

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

Example of utilisation:

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

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

11.3 Modifying the shape of the nodes

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

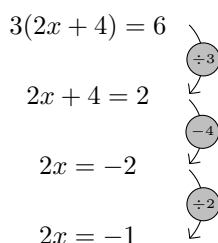
```
\begin{WithArrows}%
  [format = c,
   interline = 4mm,
   tikz = {every node/.style = {circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt,
                    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

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

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

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

  2x = -1
\end{WithArrows}
```



11.4 Examples with the option tikz-code

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

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

11.4.1 Example 1

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

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

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

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

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

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

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

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

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

11.4.2 Example 2

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

```

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

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

```

$$S_n = \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) \quad \overline{\hspace{10em}} \quad (18)$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i \frac{k\pi}{2n}}\right) \quad \overleftarrow{\hspace{10em}} \quad (19)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i \frac{k\pi}{2n}}\right) \quad \overleftarrow{\hspace{10em}} \quad (20)$$

$$= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} \left(e^{i \frac{\pi}{2n}}\right)^k\right) \quad \overleftarrow{\hspace{10em}} \quad (21)$$

$$= \frac{1}{n} \Re\left(\frac{1 - \left(e^{i \frac{\pi}{2n}}\right)^n}{1 - e^{i \frac{\pi}{2n}}}\right) \quad \overleftarrow{\hspace{10em}} \quad (22)$$

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

11.4.3 Example 3

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

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

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

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

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

11.5 Automatic numbered loop

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

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

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

```

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

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

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

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

12 Implementation

12.1 Declaration of the package and extensions loaded

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

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

`<@@=witharrows>`

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

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

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

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

The version of 2020/02/08 of `expl3` has replaced `\l_keys_key_tl` by `\l_keys_key_str`. We have immediatly changed in this file. Now, you test the existence of `\l_keys_key_str` in order to detect if the version of LaTeX used by the final user is up to date.

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

24 \RequirePackage { xparse } [ 2019-01-01 ]
25 \RequirePackage { varwidth }
26 </LaTeX>
27 <*plain-TeX>
28 \ExplSyntaxOn
29 \catcode ` \@ = 11
30 </plain-TeX>
```

12.2 The packages `footnote` and `footnotehyper`

A few options can be given to the package `witharrows` when it is loaded (with `\usepackage`, `\RequirePackage` or `\PassOptionsToPackage`). Currently (version 2.5), there are two such options: `footnote` and `footnotehyper`. With the option `footnote`, `witharrows` loads `footnote` and uses it to extract the footnotes from the environments `{WithArrows}`. Idem for the option `footnotehyper`.

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

```
31 <*LaTeX>
32 \bool_new:N \g_@@_footnotehyper_bool
```

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

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

```
33 \bool_new:N \g_@@_footnote_bool
34 </LaTeX>
```

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

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

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

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

```
58 \ProcessKeysOptions { WithArrows / package }

59 \@@_msg_new:nn { Option~incompatible~with~Beamer }
60   {
61     The~option~'\l_keys_key_str'~is~incompatible~
62     with~Beamer~because~Beamer~has~its~own~system~to~extract~footnotes.
63   }
64 \@@_msg_new:nn { footnote~with~footnotehyper~package }
65   {
66     You~can't~use~the~option~'footnote'~because~the~package~
67     footnotehyper~has~already~been~loaded.~
68     If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
69     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
70     of~the~package~footnotehyper.\\
71     If~you~go~on,~the~package~footnote~won't~be~loaded.
72   }
73 \@@_msg_new:nn { footnotehyper~with~footnote~package }
74   {
75     You~can't~use~the~option~'footnotehyper'~because~the~package~
76     footnote~has~already~been~loaded.~
77     If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
78     within~the~environments~of~witharrows~will~be~extracted~with~the~tools~
79     of~the~package~footnote.\\
80     If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
81   }
```

```

82 \bool_if:NT \g_@@_footnote_bool
83 {
84   \@ifclassloaded { beamer }
85   { \msg_info:nn { witharrows } { Option~incompatible~with~Beamer } }
86   {
87     \@ifpackageloaded { footnotehyper }
88     { \@_error:n { footnote-with-footnotehyper-package } }
89     { \usepackage { footnote } }
90   }
91 }

92 \bool_if:NT \g_@@_footnotehyper_bool
93 {
94   \@ifclassloaded { beamer }
95   { \@_info:n { Option~incompatible~with~Beamer } }
96   {
97     \@ifpackageloaded { footnote }
98     { \@_error:n { footnotehyper-with-footnote-package } }
99     { \usepackage { footnotehyper } }
100  }
101  \bool_gset_true:N \g_@@_footnote_bool
102 }

```

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

12.3 The class option `leqno`

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

```

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

```

12.4 Some technical definitions

```

108 \cs_generate_variant:Nn \tl_put_right:Nn { N v }
109 \cs_generate_variant:Nn \seq_set_split:Nnn { N x x }

```

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

```

110 \AtBeginDocument
111 {
112   \clist_map_inline:nn
113   {
114     amsmath, amsthm, autonum, cleveref, hyperref, mathtools, showlabels,
115     typedref, unicode-math
116   }
117   {

```

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

```

118     \bool_new:c { c_@@_#1_loaded_bool }
119 <*LaTeX>
120     \ifpackageloaded { #1 }
121     { \bool_set_true:c { c_@@_#1_loaded_bool } }
122     { }
123 </LaTeX>
124 <*plain-TeX>
125     \bool_set_false:c { c_@@_#1_loaded_bool }
126 </plain-TeX>
127 }
128 }

```

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

```

129 \sys_if_engine luatex:TF
130 {
131     \cs_new_protected:Npn \@@_strcmp:nn #1 #2
132     { \lua_now:e { l3kernel.strptime('#1','#2') } }
133 }
134 {
135     \cs_new_protected:Npn \@@_strcmp:nn #1 #2
136     { \tex_strcmp:D { #1 } { #2 } }
137 }

```

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

```

138 \cs_new_protected:Npn \@@_sort_seq:N #1
139 {
140     \seq_sort:Nn #1
141     {
142         \int_compare:nNnTF
143         {
144             \@@_strcmp:nn
145             { \str_lower_case:n { ##1 } }
146             { \str_lower_case:n { ##2 } }
147         }
148         > 0
149         \sort_return_swapped:
150         \sort_return_same:
151     }
152 }

```

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

```

153 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
154 {
155     \seq_clear:N \l_tmpa_seq
156     \seq_map_inline:Nn #1
157     {
158         \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
159     }
160     \seq_set_eq:NN #1 \l_tmpa_seq
161 }
162 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
163 {
164     \seq_set_from_clist:Nn #1 { #2 }
165     \@@_convert_to_str_seq:N #1
166 }

```

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

```

167 \cs_new_protected:Npn \@@_save:N #1
168 {
169   \seq_set_split:Nxx \l_tmpa_seq
170     { \char_generate:nn { ` _ } { 12 } }
171     { \cs_to_str:N #1 }
172   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl

```

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

```

173   \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
174   \use:c { \l_tmpa_str _if_exist:cF }
175   { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
176   {
177     \use:c { \l_tmpa_str _new:c }
178     { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
179   }
180   \use:c { \l_tmpa_str _gset_eq:cN }
181   { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ } #1
182 }

```

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

```

183 \cs_new_protected:Npn \@@_restore:N #1
184 {
185   \seq_set_split:Nxx \l_tmpa_seq
186     { \char_generate:nn { ` _ } { 12 } }
187     { \cs_to_str:N #1 }
188   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
189   \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } }
190   \use:c { \l_tmpa_str _set_eq:Nc }
191   #1 { g _\seq_use:Nnnn \l_tmpa_seq _ _ _ }
192 }

```

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.

```

193 \tikzset
194 {
195   @@_node_style / .style =
196   {
197     above = \l_@@_ystart_dim ,
198     inner~sep = \c_zero_dim ,
199     minimum~width = \c_zero_dim ,
200     minimum~height = \l_@@_ygap_dim
201   }
202 }

```

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

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

```

203 \tikzset
204 {
205   @@_standard / .style =
206   {
207     remember~picture ,
208     overlay ,
209     name~prefix = wa - \l_@@_prefix_str -
210   }

```

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

```
211 }
```

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`.

```
212 \tikzset
213 {
214     WithArrows / arrow / tips / .style =
215     { > = { Straight~Barb [ scale = 1.2 , bend ] } }
216 }
```

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`.

```
217 \tikzset
218 {
219     WithArrows / arrow / .style =
220     {
221         align = left ,
```

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

```
222         auto = left ,
223     <*LaTeX>
224         font = \small \itshape ,
225     </LaTeX>
226     WithArrows / arrow / tips ,
227     bend~left = 45 ,
228     ->
229 }
230 }
```

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}`.

```
231 <*LaTeX>
232 \AtBeginDocument
233 {
234     \bool_if:NTF \c_@@_amsmath_loaded_bool
235     {
236         \seq_put_right:Nn \l_@@_options_WithArrowsOptions_seq { subequations }
237         \seq_put_right:Nn \l_@@_options_DispWithArrows_seq { subequations }
238     }
```

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

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

```
239     {
240     </LaTeX>
241     \cs_new_protected:Npn \spread@equation
242     {
243         \openup \jot
244         \cs_set_eq:NN \spread@equation \prg_do_nothing:
245     }
246     <*LaTeX>
247     }
248 }
```

249 \langle /LaTeX \rangle

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

12.5 Variables

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

```
252 \bool_new:N \l_@@_in_WithArrows_bool
253 \bool_new:N \l_@@_in_DispatchWithArrows_bool
254 \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}`.

```
255 \seq_new:N \g_@@_position_in_the_tree_seq
256 \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`.

```
257 \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`).

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

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

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

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

```
259 \int_new:N \l_@@_pos_arrow_int
260 \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.

```
261 \seq_new:N \g_@@_arrow_int_seq
262 \int_new:N \g_@@_arrow_int
263 \seq_new:N \g_@@_line_int_seq
264 \int_new:N \g_@@_line_int
265 \seq_new:N \g_@@_col_int_seq
266 \int_new:N \g_@@_col_int
```

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

```
267 \seq_new:N \g_@@_static_col_int_seq
268 \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`.

```
269 \*LaTeX
270 \clist_new:N \l_@@_tags_clist
271 \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`.

```
272 \cs_new_protected:Npn \@@_test_if_to_tag:
273 {
274   \clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int
275   { \clist_set:Nn \l_@@_tags_clist { all } }
276 }
277 \*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`.

```
278 \str_new:N \l_@@_command_name_str
279 \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`”.

```
280 \str_new:N \l_@@_string_Arrow_for_msg_str
281 \str_set:Nx \l_@@_string_Arrow_for_msg_str { \token_to_str:N \Arrow }
```

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

```
282 \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.

```
283 \*LaTeX
284 \bool_new:N \l_@@_sbwi_bool
285 \*LaTeX
```

```
286 \*LaTeX
287 \bool_new:N \l_@@_tag_star_bool
288 \bool_new:N \l_@@_tag_next_line_bool
289 \bool_new:N \l_@@_qedhere_bool
290 \*LaTeX
291 \bool_new:N \l_@@_in_first_columns_bool
292 \bool_new:N \l_@@_new_group_bool
```

```

293 \bool_new:N \l_@@_initial_r_bool
294 \bool_new:N \l_@@_final_r_bool
295 \tl_new:N \l_@@_initial_tl
296 \tl_new:N \l_@@_final_tl
297 \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}`).

```

298 \str_new:N \l_@@_format_str

```

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

```

299 <*LaTeX>
300 \bool_new:N \l_@@_subequations_bool
301 </LaTeX>

```

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

```

302 \dim_new:N \l_@@_arrow_width_dim
303 \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`.

```

304 \dim_new:N \l_@@_up_and_down_radius_dim
305 \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).

```

306 \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*).

```

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

```

12.6 The definition of the options

There are four levels where options can be set:

- with `\usepackage[...]{witharrows}`: this level will be called *package* level;
- with `\WithArrowsOptions{...}`: this level will be called *global* level³²;
- with `\begin{WithArrows}[...]`: this level will be called *environment* level;
- with `\Arrow[...]` (included in `code-after`): this level will be called *local* level.

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

When we scan a list of options, we want to be able to raise an error if two options of position (11, r1, i, etc.) of the arrows are present. That's why we keep the first option of position in a variable called `\l_@@_previous_key_str`. The following function `\@@_eval_if_allowed:n` will execute its argument only if a first key of position has not been set (and raise an error elsewhere).

```

309 \cs_new_protected:Npn \@@_eval_if_allowed:n #1
310 {
311   \str_if_empty:NTF \l_@@_previous_key_str
312   {
313     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
314     #1
315   }
316   { \@@_error:n { Incompatible~options } }
317 }

318 \cs_new_protected:Npn \@@_fix_pos_option:n #1
319 { \@@_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.

```

320 \keys_define:nn { WithArrows / Global }
321 {
322   max-length-of-arrow .dim_set:N = \l_@@_max_length_of_arrow_dim ,
323   max-length-of-arrow .value_required:n = true ,
324   max-length-of-arrow .initial:n = 2 cm ,
325   ygap .dim_set:N = \l_@@_ygap_dim ,
326   ygap .initial:n = 0.4 ex ,
327   ygap .value_required:n = true ,
328   ystart .dim_set:N = \l_@@_ystart_dim ,
329   ystart .value_required:n = true ,
330   ystart .initial:n = 0.4 ex ,
331   more-columns .code:n =
332     \@@_msg_redirect_name:nn { Too-much~columns~in-WithArrows } { none } ,
333   more-columns .value_forbidden:n = true ,
334   command-name .code:n =
335     \str_set:Nn \l_@@_command_name_str { #1 }
336     \str_set:Nx \l_@@_string_Arrow_for_msg_str
337       { \c_backslash_str Arrow-alias~\c_backslash_str #1 } ,
338   command-name .value_required:n = true ,
339   tikz-code .tl_set:N = \l_@@_tikz_code_tl,
340   tikz-code .initial:n = \draw~(#{1})~to~node{#{3}}~(#{2})~; ,
341   tikz-code .value_required:n = true ,
342   displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
343   displaystyle .default:n = true ,
344   show-nodes .code:n =
345     \tikzset { @@_node_style / .append~style = { draw , red } } ,
346   show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
347   show-node-names .default:n = true ,
348   group .code:n =
349     \str_if_empty:NTF \l_@@_previous_key_str
350     {
351       \str_set:Nn \l_@@_previous_key_str { group }
352       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
353       \int_set:Nn \l_@@_pos_arrow_int 7
354     }
355     { \@@_error:n { Incompatible~options } } ,
356   group .value_forbidden:n = true ,
357   groups .code:n =
358     \str_if_empty:NTF \l_@@_previous_key_str
359     {
360       \str_set:Nn \l_@@_previous_key_str { groups }
361       \seq_if_in:NnF \l_@@_options_Arrow_seq { new-group }
362       { \seq_put_right:Nn \l_@@_options_Arrow_seq { new-group } }
363       \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
364       \int_set:Nn \l_@@_pos_arrow_int 6

```

```

365     }
366     { \@@_error:n { Incompatible~options } } ,
367     groups .value_forbidden:n = true ,
368     tikz .code:n = \tikzset { WithArrows / arrow / .append~style = { #1 } } ,
369     tikz .initial:n = \c_empty_tl ,
370     tikz .value_required:n = true ,
371     rr .code:n = \@@_fix_pos_option:n 3 ,
372     rr .value_forbidden:n = true ,
373     ll .code:n = \@@_fix_pos_option:n 1 ,
374     ll .value_forbidden:n = true ,
375     rl .code:n = \@@_fix_pos_option:n 2 ,
376     rl .value_forbidden:n = true ,
377     lr .code:n = \@@_fix_pos_option:n 0 ,
378     lr .value_forbidden:n = true ,
379     i .code:n = \@@_fix_pos_option:n 5 ,
380     i .value_forbidden:n = true ,
381     xoffset .dim_set:N = \l_@@_xoffset_dim ,
382     xoffset .value_required:n = true ,
383     xoffset .initial:n = 3 mm ,
384     jot .dim_set:N = \jot ,
385     jot .value_required:n = true ,
386     interline .skip_set:N = \l_@@_interline_skip ,
387     start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
388     start-adjust .initial:n = 0.4 ex ,
389     start-adjust .value_required:n = true ,
390     end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
391     end-adjust .initial:n = 0.4 ex ,
392     end-adjust .value_required:n = true ,
393     adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
394     adjust .value_required:n = true ,
395     up-and-down .code:n = \keys_set:nn { WithArrows / up-and-down } { #1 } ,
396     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”.

```

397     no-arrows .code:n =
398         \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn
399         \cs_set_eq:NN \@@_draw_arrow:nnn \use_none:nnn ,
400     no-arrows .value_forbidden:n = true
401 }

```

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`.

```

402 \keys_define:nn { WithArrows / WithArrowsSpecific }
403 {
404     t .code:n = \int_set:Nn \l_@@_pos_env_int 0 ,
405     t .value_forbidden:n = true ,
406     c .code:n = \int_set:Nn \l_@@_pos_env_int 1 ,
407     c .value_forbidden:n = true ,
408     b .code:n = \int_set:Nn \l_@@_pos_env_int 2 ,
409     b .value_forbidden:n = true
410 }

```

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

```

411 \clist_new:N \c_@@_extensible_delimiters_clist
412 \clist_set:Nn \c_@@_extensible_delimiters_clist
413 {
414     ., \{, (, [, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
415 }
416 \<LaTeX>

```

```

417 \AtBeginDocument
418 {
419   \bool_lazy_or:nnT
420     \c_@@_amsmath_loaded_bool
421     { \use:c { c_@@_unicode-math_loaded_bool } }
422     {
423       \clist_put_right:Nn \c_@@_extensible_delimiters_clist { \lvert, \lVert }
424     }
425 }
426 </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`.

```

427 \keys_define:nn { WithArrows / DispWithArrowsSpecific }
428 {
429   fleqn .bool_set:N = \l_@@_fleqn_bool ,
430   fleqn .default:n = true ,
431   mathindent .dim_set:N = \l_@@_mathindent_dim ,
432   mathindent .initial:n = 25 pt ,
433   mathindent .value_required:n = true ,
434 <*LaTeX>
435   notag .code:n =
436     \str_if_eq:nnTF { #1 } { true }
437     { \clist_clear:N \l_@@_tags_clist }
438     { \clist_set:Nn \l_@@_tags_clist { all } } ,
439   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.

```

440   subequations .code:n =
441     \bool_if:NTF \c_@@_amsmath_loaded_bool
442     { \bool_set_true:N \l_@@_subequations_bool }
443     {
444       \@@_error:n { amsmath-not-loaded }
445       \group_begin:
446       \globaldefs = 1
447       \@@_msg_redirect_name:nn { amsmath-not-loaded } { info }
448       \group_end:
449     } ,
450   subequations .default:n = true ,
451   subequations .value_forbidden:n = true ,
452   nonumber .meta:n = notag ,
453   allow-multiple-labels .code:n =
454     \@@_msg_redirect_name:nn { Multiple-labels } { none } ,
455   allow-multiple-labels .value_forbidden:n = true ,
456   tagged-lines .code:n =
457     \clist_set:Nn \l_@@_tags_clist { #1 }
458     \clist_if_in:NnT \l_@@_tags_clist { first }
459     {
460       \clist_remove_all:Nn \l_@@_tags_clist { first }
461       \clist_put_left:Nn \l_@@_tags_clist 1
462     } ,
463   tagged-lines .value_required:n = true ,
464 </LaTeX>
465   wrap-lines .bool_set:N = \l_@@_wrap_lines_bool ,
466   wrap-lines .default:n = true ,
467   replace-left-brace-by .code:n =
468     {
469       \tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
470       \clist_if_in:NVTF
471         \c_@@_extensible_delimiters_clist
472         \l_tmpa_tl
473         { \tl_set:Nn \l_@@_replace_left_brace_by_tl { #1 } }

```

```

474     { \@@_error:n { Bad-value-for-replace-brace-by } }
475   } ,
476   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`).

```

477 \*LaTeX
478   standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool ,
479   standard-behaviour-with-items .default:n = true
480 \*LaTeX
481 }

```

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

```

482 \keys_define:nn { WithArrows / Env }
483 {
484   name .code:n =

```

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

```

485   \str_set:Nn \l_tmpa_str { #1 }
486   \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
487     { \@@_error:n { Duplicate-name } }
488     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
489   \str_set_eq:NN \l_@@_name_str \l_tmpa_str ,
490   name .value_required:n = true ,
491   code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 } ,
492   code-before .value_required:n = true ,
493   CodeBefore .meta:n = { code-before = #1 } ,
494   code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 } ,
495   code-after .value_required:n = true ,
496   CodeAfter .meta:n = { code-after = #1 } ,
497   format .code:n =
498     \tl_if_empty:nTF { #1 }
499     { \@@_error:n { Invalid-option-format } }
500     {
501       \regex_match:nnTF { \A[rcl]*\Z } { #1 }
502       { \tl_set:Nn \l_@@_format_str { #1 } }
503       { \@@_error:n { Invalid-option-format } }
504     } ,
505   format .value_required:n = true
506 }

```

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.

```

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

```

```

523 WithArrowsOptions .inherit:n =
524 {
525     WithArrows / Global ,
526     WithArrows / WithArrowsSpecific ,
527     WithArrows / DispWithArrowsSpecific ,
528 } ,
529 WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
530 }

```

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

```

531 \seq_new:N \l_@@_options_WithArrows_seq
532 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrows_seq
533 {
534     adjust, b, c, code-after, code-before, command-name,
535     displaystyle, end-adjust,
536     format, group, groups, i,
537     interline, jot, ll,
538     lr, max-length-of-arrow, more-columns, name,
539     no-arrows, rl, rr, up-and-down,
540     show-node-names, show-nodes, start-adjust,
541     t, tikz, tikz-code,
542     xoffset, ygap, ystart
543 }
544 \@@_convert_to_str_seq:N \l_@@_options_WithArrows_seq

545 \keys_define:nn { WithArrows / WithArrows }
546 {
547     unknown .code:n =
548         \@@_sort_seq:N \l_@@_options_WithArrows_seq
549         \@@_error:n { Unknown~option~WithArrows }
550 }

```

```

551 \keys_define:nn { WithArrows / DispWithArrows }
552 {
553     left-brace .tl_set:N = \l_@@_left_brace_tl ,
554     unknown .code:n =
555         \@@_sort_seq:N \l_@@_options_DispWithArrows_seq
556         \@@_error:n { Unknown~option~DispWithArrows } ,
557 }

```

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

```

558 \seq_new:N \l_@@_options_DispWithArrows_seq
559 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_DispWithArrows_seq
560 {
561     code-after, code-before, command-name, tikz-code, adjust,
562     displaystyle, end-adjust, fleqn, group, format, groups, i, interline, jot,
563     left-brace, ll, lr, max-length-of-arrow, mathindent, name, no-arrows,
564     up-and-down, replace-left-brace-by, rl, rr, show-node-names,
565     show-nodes, start-adjust, tikz, wrap-lines, xoffset, ygap, ystart,
566     (*LaTeX)
567     allow-multiple-labels, tagged-lines, nonumber, notag
568     (/LaTeX)
569 }

570 \keys_define:nn { WithArrows / WithArrowsOptions }
571 {
572     allow-duplicate-names .code:n =
573         \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
574     allow-duplicate-names .value_forbidden:n = true ,

```

```

575 xoffset-for-o-arrows .dim_set:N = \l_@@_xoffset_for_o_arrows_dim ,
576 xoffset-for-o-arrows .value_required:n = true ,
577 unknown .code:n =
578   \@@_sort_seq:N \l_@@_options-WithArrowsOptions_seq
579   \@@_error:n { Unknown-option-WithArrowsOptions }
580 }

```

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

```

581 \seq_new:N \l_@@_options-WithArrowsOptions_seq
582 \@@_set_seq_of_str_from_clist:Nn \l_@@_options-WithArrowsOptions_seq
583 {
584   allow-duplicate-names, b, c, command-name, more-columns, tikz-code, adjust,
585   displaystyle, end-adjust, fleqn, group, groups, i, interline, jot, ll, lr,
586   mathindent, max-length-of-arrow, no-arrows, up-and-down, rl, rr,
587   show-node-names, show-nodes, start-adjust, t, tikz, wrap-lines, xoffset,
588   xoffset-for-o-arrows, ygap, ystart,
589 }*LaTeX
590   allow-multiple-labels, nonumber, notag, standard-behaviour-with-items,
591   tagged-lines
592 }/LaTeX
593 }

```

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”.

```

594 \cs_new_protected:Npn \@@_set_independent:
595 {
596   \str_if_eq:VnF \l_keys_value_tl { NoValue }
597   { \@@_error:n { Value-for~a~key } }
598   \@@_set_independent_bis:
599 }

```

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

```

600 \cs_new_protected:Npn \@@_set_independent_bis:
601 {
602   \str_if_empty:NTF \l_@@_previous_key_str
603   {
604     \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
605     \str_set:Nn \l_@@_status_arrow_str { independent }
606   }
607   { \@@_error:n { Incompatible~options-in~Arrow } }
608 }

```

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”.

```

609 \keys_define:nn { WithArrows / Arrow / FirstPass }
610 {
611   jump .code:n =
612     \int_compare:nTF { #1 > 0 }
613     { \int_set:Nn \l_@@_jump_int { #1 } }
614     { \@@_error:n { Negative-jump } } ,
615   jump .value_required:n = true,
616   rr .code:n = \@@_set_independent: ,
617   ll .code:n = \@@_set_independent: ,

```

```

618 rl .code:n = \@@_set_independent: ,
619 lr .code:n = \@@_set_independent: ,
620 i .code:n = \@@_set_independent: ,
621 rr .default:n = NoValue ,
622 ll .default:n = NoValue ,
623 rl .default:n = NoValue ,
624 lr .default:n = NoValue ,
625 i .default:n = NoValue ,
626 new-group .value_forbidden:n = true ,
627 new-group .code:n =
628   \int_compare:nTF { \l_@@_pos_arrow_int = 6 }
629   { \str_set:Nn \l_@@_status_arrow_str { new-group } }
630   { \@@_error:n { new-group-without-groups } } ,
631 o .code:n =
632   \str_if_empty:NTF \l_@@_previous_key_str
633   {
634     \int_compare:nNnTF \l_@@_pos_arrow_int < 6
635     { \@@_error:n { invalid-key-o } }
636     {
637       \str_set:Nn \l_@@_status_arrow_str { over }
638       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
639     }
640   }
641   { \@@_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.

```

642 tikz-code .code:n = \prg_do_nothing: ,
643 tikz-code .value_required:n = true ,
644 tikz .code:n = \prg_do_nothing: ,
645 tikz .value_required:n = true ,
646 start-adjust .code:n = \prg_do_nothing: ,
647 start-adjust .value_required:n = true ,
648 end-adjust .code:n = \prg_do_nothing: ,
649 end-adjust .value_required:n = true ,
650 adjust .code:n = \prg_do_nothing: ,
651 adjust .value_required:n = true ,
652 xoffset .code:n = ,
653 unknown .code:n =
654   \@@_sort_seq:N \l_@@_options_Arrow_seq
655   \seq_if_in:NVTF \l_@@_options-WithArrows_seq \l_keys_key_str
656   {
657     \str_set:Nn \l_tmpa_str
658     { ~However,~this~key~can~be~used~in~the~options~of~{WithArrows}. }
659   }
660   { \str_clear:N \l_tmpa_str }
661   \@@_error:n { Unknown-option-in-Arrow }
662 }

```

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

```

663 \seq_new:N \l_@@_options_Arrow_seq
664 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_seq
665 {
666   adjust, end-adjust, i, jump, ll, lr, o , rl, rr, start-adjust, tikz,
667   tikz-code, xoffset
668 }

669 \cs_new_protected:Npn \@@_fix_pos_arrow:n #1
670 {
671   \str_if_empty:NT \l_@@_previous_key_str

```

```

672     {
673       \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
674       \int_set:Nn \l_@@_pos_arrow_int { #1 }
675     }
676   }

```

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.

```

677 \keys_define:nn {WithArrows / Arrow / SecondPass }
678 {
679   tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
680   tikz-code .initial:n = \draw~(##1)~to~node{##3}~(##2)~; ,
681   tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
682   tikz .initial:n = \c_empty_tl ,
683   rr .code:n = \@@_fix_pos_arrow:n 3 ,
684   ll .code:n = \@@_fix_pos_arrow:n 1 ,
685   rl .code:n = \@@_fix_pos_arrow:n 2 ,
686   lr .code:n = \@@_fix_pos_arrow:n 0 ,
687   i .code:n = \@@_fix_pos_arrow:n 5 ,
688   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.

```

689   xoffset .code:n =
690     \bool_if:nTF
691     {
692       \int_compare_p:nNn \g_@@_arrow_int > 1
693       &&
694       \int_compare_p:nNn \l_@@_pos_arrow_int > 5
695       &&
696       ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
697     }
698     { \@@_error:n { Option~xoffset~forbidden } }
699     { \dim_set:Nn \l_@@_xoffset_dim { #1 } } ,
700   xoffset .value_required:n = true ,
701   start-adjust .dim_set:N = \l_@@_start_adjust_dim,
702   end-adjust .dim_set:N = \l_@@_end_adjust_dim,
703   adjust .code:n =
704     \dim_set:Nn \l_@@_start_adjust_dim { #1 }
705     \dim_set:Nn \l_@@_end_adjust_dim { #1 } ,
706 }

```

`\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`.

```

707 <LaTeX>
708 \NewDocumentCommand \WithArrowsOptions { m }
709 </LaTeX>
710 <plain-TeX>
711 \cs_set_protected:Npn \WithArrowsOptions #1
712 </plain-TeX>
713 {
714   \str_clear_new:N \l_@@_previous_key_str
715   \keys_set:nn { WithArrows / WithArrowsOptions } { #1 }
716 }

```


12.7 The command `\Arrow`

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

```

717 <*LaTeX>
718 \NewDocumentCommand \@@_Arrow { 0 { } m ! 0 { } }
719 </LaTeX>
720 <*plain-TeX>
721 \cs_new_protected:Npn \@@_Arrow
722 {
723   \peek_meaning:NTF [
724     { \@@_Arrow_i }
725     { \@@_Arrow_i [ ] }
726   }
727   \cs_new_protected:Npn \@@_Arrow_i [ #1 ] #2
728   {
729     \peek_meaning:NTF [
730       { \@@_Arrow_ii [ #1 ] { #2 } }
731       { \@@_Arrow_ii [ #1 ] { #2 } [ ] }
732     }
733     \cs_new_protected:Npn \@@_Arrow_ii [ #1 ] #2 [ #3 ]
734 </plain-TeX>
735   {

```

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.

```

736   \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`.

```

737   \str_clear_new:N \l_@@_previous_key_str
738   \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:

```

739   \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`):

```

740   \int_set:Nn \l_tmpa_int { \g_@@_line_int + \l_@@_jump_int }
741   \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`.

```

742   \prop_put:NnV \l_tmpa_prop { status } \l_@@_status_arrow_str

```

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

```

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

```

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

```
744 \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).

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

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

```
746 \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.

```
747 \prop_gclear_new:c
748 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
749 \prop_gset_eq:cN
750 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \g_@@_arrow_int _ prop }
751 \l_tmpa_prop
752 }
```

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.

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

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

12.8.1 Code before the `\halign`

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

```
755 \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.

```
756 {
757 <*LaTeX>
758 \str_clear_new:N \l_@@_type_env_str
759 \str_set:NV \l_@@_type_env_str \@currentenv
760 </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`.

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

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

```
763 \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.

```
764 \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.

```
765 \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.

```
766 \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.

```
767 \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
768 \int_gzero:N \g_@@_arrow_int
769 \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
770 \int_gzero:N \g_@@_line_int
771 \seq_gput_right:NV \g_@@_col_int_seq \g_@@_col_int
772 \int_gzero:N \g_@@_col_int
773 \seq_gput_right:NV \g_@@_static_col_int_seq \g_@@_static_col_int
774 \int_gzero:N \g_@@_static_col_int
```

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

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

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

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

```
775 \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).

```
776 \seq_set_eq:NN \l_tmpa_seq \g_@@_position_in_the_tree_seq
777 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
778 \str_clear_new:N \l_@@_prefix_str
779 \str_set:Nx \l_@@_prefix_str { \seq_use:Nnnn \l_tmpa_seq - - - }
```

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

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

These counters will be used later as variables.

```
782 \int_zero_new:N \l_@@_initial_int
783 \int_zero_new:N \l_@@_final_int
784 \int_zero_new:N \l_@@_arrow_int
785 \int_zero_new:N \l_@@_pos_of_arrow_int
786 \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.

```
787 \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.

```
788 \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`.

```
789 \*LaTeX\
790 \seq_clear_new:N \l_@@_labels_seq
791 \bool_set_false:N \l_@@_tag_next_line_bool
792 \*LaTeX\
```

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

```
793 \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`.

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

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

```
796 \str_clear_new:N \l_@@_previous_key_str
797 \bool_if:NT \l_@@_in-WithArrows_bool
798 { \keys_set:nn { WithArrows / WithArrows } { #1 } }
799 \bool_if:NT \l_@@_in-DispWithArrows_bool
800 { \keys_set:nn { WithArrows / DispWithArrows } { #1 } }
```

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

```
801 \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`).

```
802 \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.

```
803 \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.

```
804 \seq_clear_new:N \l_@@_format_seq
805 \seq_set_split:NnV \l_@@_format_seq { } \l_@@_format_str
```

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

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`).

```

806 \*LaTeX
807   \bool_if:NT \g_@@_footnote_bool { \begin { savenotes } }
808 \*LaTeX

```

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

```

809   \l_@@_code_before_tl
810 \*LaTeX
811   \cs_set_eq:NN \notag \@@_notag:
812   \cs_set_eq:NN \nonumber \@@_nonumber:
813   \cs_set_eq:NN \tag \@@_tag
814   \cs_set_eq:NN \@@_old_label \label
815   \cs_set_eq:NN \label \@@_label:n
816   \cs_set_eq:NN \tagnextline \@@_tagnextline:
817 \*LaTeX
818 }

```

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

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

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

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

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

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

```

819 \cs_new_protected:Npn \@@_construct_halign:
820 {
821   \seq_pop_right:NNTF \l_@@_format_seq \l_@@_type_col_str
822   {

```

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

```

823   \use:x
824   {

```

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

We begin the construction of a generic column.

```

825         \int_gdecr:N \g_@@_col_int
826         \@@_construct_halign:
827         \int_compare:nNnT \g_@@_col_int = \l_@@_nb_cols_int
828         {

```

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

```

829         \cs_set_eq:cN { \l_@@_command_name_str } \@@_Arrow
830 (*LaTeX)
831         \bool_if:NT \l_@@_in_DisplayWithArrows_bool
832         {

```

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).

```

833         \@@_test_if_to_tag:

```

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

```

834         \bool_if:NT \c_@@_amsthm_loaded_bool \@@_set_qedhere:
835     }
836 (/LaTeX)
837 }
838 \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
839 \str_if_eq:VnT \l_@@_type_col_str { r } \hfill
840 \int_gincr:N \g_@@_col_int
841 \int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
842 \c_math_toggle_token
843 {
844     { }
845     \bool_if:NT \l_@@_displaystyle_bool \displaystyle
846     ####
847 }
848 \c_math_toggle_token
849 \int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
850 { \@@_construct_nodes: }
851 {

```

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:`).

```

852     \str_if_eq:VnT \l_@@_type_col_str { l } \hfil
853     \str_if_eq:VnT \l_@@_type_col_str { c } \hfil
854     \bool_if:NT \l_@@_in_DisplayWithArrows_bool { \tabskip = \c_zero_skip }
855     &
856 }
857 }
858 }

```

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

```

859 {
860     \bool_if:NTF \l_@@_in-WithArrows_bool
861     {
862         \ialign
863         \bgroup
864     }
865     {
866         \halign to \l_@@_linewidth_dim

```

```

867         \bgroup
868         \bool_if:NT \l_@@_fleqn_bool
869         { \skip_horizontal:N \l_@@_mathindent_dim }
870     }
871     \int_gincr:N \g_@@_line_int
872     \int_gzero:N \g_@@_col_int
873     \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
874     {
875         \skip_horizontal:n
876         { \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
877     }
878     \strut
879 }
880 }

```

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.

```

881 \cs_new_protected:Npn \@@_construct_nodes:
882 {

```

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).

```

883     \tikz [ remember~picture , overlay ]
884     \node
885     [
886         node~contents = { } ,
887         @@_node_style ,
888         name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - l ,
889         alias =
890         {
891             \str_if_empty:NF \l_@@_name_str
892             { \l_@@_name_str - \int_use:N \g_@@_line_int - l }
893         }
894     ]
895     ;
896     \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).

```

897     \tikz [ remember~picture , overlay ]
898     \node
899     [
900         node~contents = { } ,
901         @@_node_style ,
902         name = wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r ,
903         alias =
904         {
905             \str_if_empty:NF \l_@@_name_str
906             { \l_@@_name_str - \int_use:N \g_@@_line_int - r }
907         }
908     ]
909     ;
910     \bool_if:NT \l_@@_show_node_names_bool
911     {
912         \hbox_overlap_right:n
913         { \small wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - r }
914     }
915 }

```

12.8.3 The environment `{WithArrows}`

```

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

```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`³⁴ depending of the value of `\l_@@_pos_env_int` (fixed by the options `t`, `c` or `b`). The environment `{WithArrows}` must be used in math mode³⁵ and therefore, we can use `\vcenter`.

```

940   \int_case:nn \l_@@_pos_env_int { 0 \vtop 1 \vcenter 2 \vbox }
941   \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).

```

942   \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.

```

943   \@@_construct_halign:

```

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

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

```

944   &&
945   \@@_error:n { Too~much~columns~in~WithArrows }
946   \c_math_toggle_token
947   \bool_if:NT \l_@@_displaystyle_bool \displaystyle
948   { ## }
949   \c_math_toggle_token
950   \cr
951 }

```

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

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

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


```

952 <*plain-TeX>
953 \cs_new_protected:Npn \endWithArrows
954 </plain-TeX>
955 {
956   \\\
957   \egroup
958   \egroup
959   \@@_post_halign:

```

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

```

960 <*LaTeX>
961   \bool_if:NT \g_@@_footnote_bool { \end { savenotes } }
962 </LaTeX>
963 <*plain-TeX>
964   \group_end:
965 </plain-TeX>
966 }

```

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

12.8.4 After the construction of the `\halign`

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

```

967 \cs_new_protected:Npn \@@_post_halign:

```

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

```

968 {
969   \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.

```

970   \normalbaselines

```

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

```

971   \int_compare:nNnT \g_@@_arrow_int > 0
972   {

```

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).

```

973     \int_compare:nNnT \g_@@_arrow_int = 1
974     {
975       \int_compare:nNnT \l_@@_pos_arrow_int > 5
976       { \int_set:Nn \l_@@_pos_arrow_int 5 }
977     }
978     \@@_scan_arrows:
979   }

```

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

```

980   \group_begin:
981   \tikzset { every-picture / .style = @@_standard }

```

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

```

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

```

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

```

983 \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
984 \cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
985 \bool_set_true:N \l_@@_in_code_after_bool
986 \l_@@_code_after_tl
987 \group_end:

```

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

```

988 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
989 \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
990 \seq_gput_right:Nx \g_@@_position_in_the_tree_seq
991 { \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`.

```

992 \int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
993 { \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.

```

994 \seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
995 \int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
996 \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
997 \int_gset:Nn \g_@@_line_int \l_tmpa_tl
998 \seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
999 \int_gset:Nn \g_@@_col_int \l_tmpa_tl
1000 \seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
1001 \int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
1002 }

```

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

12.8.5 The command of end of row

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

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

```

1003 \cs_new_protected:Npn \@@_cr:
1004 {
1005 \scan_stop:

```

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

```

1006 \int_compare:nNnF \g_@@_col_int = \g_@@_static_col_int
1007 { \@@_error:n { omit~probably~used } }
1008 \prg_replicate:nn { \l_@@_nb_cols_int - \g_@@_static_col_int } { & { } }
1009 \group_align_safe_begin:
1010 \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:
1011 }

```

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

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

```
1012 \cs_new_protected:Npn \@@_cr_i:
1013   { \peek_meaning:NTF [ \@@_cr_ii: { \@@_cr_ii: [ \c_zero_dim ] } }
```

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

```
1014 \<LaTeX>
1015 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1016   {
1017     \peek_meaning_ignore_spaces:NTF \end
1018     {
1019       \@@_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`.

```
1020       \@@_analyze_end:Nn
1021     }
1022     { \@@_cr_iii:n { #1 } }
1023   }
1024 \cs_new_protected:Npn \@@_cr_iii:n #1
1025 \</LaTeX>
1026 \<*plain-TeX>
1027 \cs_new_protected:Npn \@@_cr_ii: [ #1 ]
1028 \</plain-TeX>
1029   {
1030     \group_align_safe_end:
```

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

```
1031   \bool_if:NT \l_@@_in_DispWithArrows_bool
```

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

```
1032   {
1033 \<*LaTeX>
1034     \clist_if_in:NnTF \l_@@_tags_clist { all }
1035     {
```

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).

```
1036       \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 `&`).

```
1037       \cs_gset:Npx \g_tmpa_tl
1038       { \tl_if_empty:NTF \l_@@_tag_tl \theequation \l_@@_tag_tl }
```

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

```
1039       \seq_if_empty:NF \l_@@_labels_seq
1040       {
```

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

Now, we do the job done by `\refstepcounter` and by the redefinitions of `\refstepcounter` done by some packages (the incrementation of the counter has been done yet).

First an action which is in the definition of `\refstepcounter`.

```
1041 \cs_set:Npx \c_currentlabel { \p@equation \g_tmpa_tl }
```

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

```
1042 \bool_if:NT \c_@@_hyperref_loaded_bool
1043 {
1044   \str_set:Nn \This@name { equation }
1045   \hyper@refstepcounter { equation }
1046 }
```

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

```
1047 \bool_if:NT \c_@@_cleveref_loaded_bool
1048 {
1049   \cref@constructprefix { equation } \cref@result
1050   \protected@edef \cref@currentlabel
1051   {
1052     [
1053       \cs_if_exist:NTF \cref@equation@alias
1054       \cref@equation@alias
1055       { equation }
1056     ]
1057     [ \arabic { equation } ] [ \cref@result ]
1058     \p@equation \g_tmpa_tl
1059   }
1060 }
```

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`).

```
1061 \seq_map_function:NN \l_@@_labels_seq \@@_old_label
1062 }
```

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.

```
1063 \@@_save:N \l_@@_tag_star_bool
1064 \@@_save:N \l_@@_qedhere_bool
1065 \bool_if:NT \l_@@_tag_next_line_bool
1066 {
1067   \openup -\jot
1068   \bool_set_false:N \l_@@_tag_next_line_bool
1069   \notag \&
1070 }
1071 &
1072 \@@_restore:N \l_@@_tag_star_bool
1073 \@@_restore:N \l_@@_qedhere_bool
1074 \bool_if:NT \l_@@_qedhere_bool
1075 { \hbox_overlap_left:n \@@_qedhere_i: }
1076 \cs_set_eq:NN \theequation \g_tmpa_tl
1077 \bool_if:NT \l_@@_tag_star_bool
1078 { \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`.

```
1079 \hbox_overlap_left:n
1080 {
1081   \bool_if:NF \c_@@_leqno_bool
1082   {
1083     \pgfpicture
1084     \pgfrememberpicturepositiononpagetrue
```

```

1085         \pgfcoordinate
1086         { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1087         \pgfpointorigin
1088         \endpgfpicture
1089     }
1090     \quad
1091     \@eqnnum
1092 }
1093 \bool_if:NT \c_@@_leqno_bool
1094 {
1095     \pgfpicture
1096     \pgfrememberpicturepositiononpagetrue
1097     \pgfcoordinate
1098     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1099     \pgfpointorigin
1100     \endpgfpicture
1101 }
1102 }
1103 {
1104     \@@_save:N \l_@@_qedhere_bool
1105 </LaTeX>
1106 &
1107 <*LaTeX>
1108     \@@_restore:N \l_@@_qedhere_bool
1109     \bool_if:NT \l_@@_qedhere_bool
1110     { \hbox_overlap_left:n \@@_qedhere_i: }
1111 </LaTeX>
1112     \pgfpicture
1113     \pgfrememberpicturepositiononpagetrue
1114     \pgfcoordinate
1115     { wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
1116     \pgfpointorigin
1117     \endpgfpicture
1118 <*LaTeX>
1119 }
1120 </LaTeX>
1121 }
1122 \dim_compare:nNnT { #1 } < \c_zero_dim
1123 { \@@_error:n { option-of-cr-negative } }
1124
1125 \cr
1126 \noalign
1127 {
1128     \dim_set:Nn \l_tmpa_dim { \dim_max:nn { #1 } \c_zero_dim }
1129     \skip_vertical:N \l_tmpa_dim
1130     \skip_vertical:N \l_@@_interline_skip
1131     \scan_stop:
1132 }
1133 }

```

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

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

```

1134 <*LaTeX>
1135 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1136 {
1137     \str_if_eq:VnT \l_@@_type_env_str { #2 }
1138     {
1139         \@@_error:n { newline-at-the-end-of-env }

```

```

1140     \group_begin:
1141     \globaldefs = 1
1142     \@@_msg_redirect_name:nn { newline-at-the-end-of-env } { none }
1143     \group_end:
1144 }

```

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

```

1145     \end { #2 }
1146 }
1147 </LaTeX>

```

12.8.6 The environment `{DispWithArrows}`

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

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

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

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

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

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

```

1148 <*LaTeX>
1149 \bool_new:N \l_@@_in_label_or_minipage_bool
1150 </LaTeX>
1151 <*LaTeX>
1152 \NewDocumentEnvironment { DispWithArrows } { ! d < > ! 0 { } }
1153 </LaTeX>
1154 <*plain-TeX>
1155 \cs_new_protected:Npn \DispWithArrows
1156 {
1157     \group_begin:
1158     \peek_meaning:NTF <
1159     { \DispWithArrows_i }
1160     { \DispWithArrows_i < \c_novalue_tl > }
1161 }
1162 \cs_new_protected:Npn \DispWithArrows_i < #1 >
1163 {
1164     \peek_meaning:NTF [
1165     { \DispWithArrows_ii < #1 > }
1166     { \DispWithArrows_ii < #1 > [ ] }
1167 }
1168 \cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
1169 </plain-TeX>
1170 {
1171     \bool_set_true:N \l_@@_in_DispWithArrows_bool
1172 <*plain-TeX>
1173     \str_clear_new:N \l_@@_type_env_str
1174     \str_set:Nn \l_@@_type_env_str { DispWithArrows }
1175 </plain-TeX>

```

Since the version 1.16 of `witharrows`, no space is added between an `\item` of a LaTeX list and an environment `{DispWithArrows}` except with the option `standard-behaviour-with-items` stored in the boolean `\l_@@_sbwi_bool`. We have to know if we are just after an `\item` and this information will be stored in `\l_@@_in_label_or_minipage_bool`. We have to do this test quickly after the beginning of the environment (in particular, because it must be done before the execution of the `code-before`³⁸).

³⁸The `code-before` is not meant to contain 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}`.

```

1176 <*LaTeX>
1177   \bool_if:NF \l_@@_sbwi_bool
1178   {
1179     \legacy_if:nT { @inlabel }
1180     { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1181     \legacy_if:nT { @minipage }
1182     { \bool_set_true:N \l_@@_in_label_or_minipage_bool }
1183   }
1184 </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).

```

1185 <*LaTeX>
1186   \bool_if:nT \c_@@_mathtools_loaded_bool
1187   {
1188     \MH_if_boolean:nT { show_only_refs }
1189     {
1190       \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`.

```

1191       \MH_set_boolean:T:n { show_only_refs }
1192     }
1193   }

```

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`.

```

1194   \bool_if:NT \c_@@_typedref_loaded_bool { \str_set:Nn \sr@name { equation } }

```

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

```

1195   \bool_if:NT \c_@@_amsmath_loaded_bool \intertext@
1196 </LaTeX>
1197   \exp_args:No \tl_if_novalue:nF { #1 } { \tl_set:Nn \l_@@_left_brace_tl { #1 } }
1198   \@@_pre_halign:n { #2 }

```

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

```

1199 <*LaTeX>
1200   \bool_if:NT \l_@@_subequations_bool { \begin { subequations } }
1201 </LaTeX>
1202   \tl_if_eq:NNF \l_@@_left_brace_tl \c_novalue_tl
1203   {

```

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

```

1204     \hbox_set:Nn \l_tmpa_box
1205     {

```

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

```

1206       \group_begin:
1207       \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
1208       \c_math_toggle_token
1209       \left \l_@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
1210       \c_math_toggle_token
1211       \group_end:
1212     }
1213     \dim_zero_new:N \l_@@_delim_wd_dim
1214     \dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
1215     \box_clear_new:N \l_@@_left_brace_box
1216     \hbox_set:Nn \l_@@_left_brace_box
1217     {
1218       \group_begin:
1219       \cs_set_eq:NN \label \@@_old_label
1220       \c_math_toggle_token
1221       \bool_if:NT \l_@@_displaystyle_bool \displaystyle

```

```

1222         \l_@@_left_brace_tl
1223         { }
1224         \c_math_toggle_token
1225     \group_end:
1226 }
1227 }

```

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

```

1228 (*LaTeX)
1229     \tl_clear_new:N \l_@@_tag_tl

1230     \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.

```

1231     \bool_set_false:N \l_@@_tag_star_bool
1232 </LaTeX>

1233     \if_mode_math:
1234         \@@_fatal:n { DisWithArrows~in~math~mode }
1235     \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`.

```

1236 (*plain-TeX)
1237     \dim_zero_new:N \linewidth
1238     \dim_set_eq:NN \linewidth \displaywidth
1239 </plain-TeX>
1240 (*LaTeX)
1241     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1242         { \c_math_toggle_token }
1243         {
1244 </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.

```

1245     \if_mode_vertical:
1246         \nointerlineskip
1247         \hbox_to_wd:nn { .6 \linewidth } { }
1248     \fi:
1249     \c_math_toggle_token \c_math_toggle_token
1250 (*LaTeX)
1251 }
1252 </LaTeX>

1253     \dim_zero_new:N \l_@@_linewidth_dim
1254 (*LaTeX)
1255     \bool_if:NTF \l_@@_in_label_or_minipage_bool
1256         { \dim_set_eq:NN \l_@@_linewidth_dim \linewidth }
1257         { \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth }
1258 </LaTeX>
1259 (*plain-TeX)
1260     \dim_set_eq:NN \l_@@_linewidth_dim \displaywidth
1261 </plain-TeX>

1262     \box_clear_new:N \l_@@_halign_box
1263     \setbox \l_@@_halign_box \vtop \bgroup
1264     \tabskip =
1265         \bool_if:NTF \l_@@_fleqn_bool
1266             \c_zero_skip
1267             { 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).

```

1268 \spread@equation
1269 \@@_construct_halign:
1270 \tabskip = 0 pt plus 1000 pt minus 1000 pt
1271 &

```

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.

```

1272 $ ## $
1273 \tabskip = \c_zero_skip
1274 &&
1275 \@@_fatal:n { Too~much~columns~in~DispWithArrows }
1276 \bool_if:nT \c_false_bool { ## }
1277 \cr
1278 }

```

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

```

1279 <*plain-TeX>
1280 \cs_new_protected:Npn \endDispWithArrows
1281 </plain-TeX>
1282 {
1283 <*LaTeX>
1284 \clist_if_in:NnT \l_@@_tags_clist { last }
1285 { \clist_set:Nn \l_@@_tags_clist { all } }
1286 </LaTeX>
1287 \}

```

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

```

1288 \egroup
1289 \unskip \unpenalty \unskip \unpenalty
1290 \box_set_to_last:N \l_tmpa_box
1291 \nointerlineskip
1292 \box_use:N \l_tmpa_box
1293 \dim_gzero_new:N \g_@@_alignment_dim
1294 \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
1295 \box_clear_new:N \l_@@_new_box
1296 \hbox_set:Nn \l_@@_new_box { \hbox_unpack_clear:N \l_tmpa_box }
1297 \dim_compare:nNnT
1298 { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
1299 { \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }

```

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

```

1300 \egroup
1301 \tl_if_eq:NNTF \l_@@_left_brace_tl \c_novaluel_tl
1302 { \box_use_drop:N \l_@@_halign_box }
1303 {
1304 \hbox_to_wd:nn \l_@@_linewidth_dim
1305 {
1306 \bool_if:NNTF \l_@@_fleqn_bool
1307 { \skip_horizontal:N \l_@@_mathindent_dim }
1308 \hfil
1309 \hbox_to_wd:nn \g_@@_alignment_dim
1310 {
1311 \box_use_drop:N \l_@@_left_brace_box
1312 \dim_set:Nn \l_tmpa_dim
1313 {
1314 \box_ht:N \l_@@_halign_box
1315 + \box_dp:N \l_@@_halign_box

```

```

1316     }
1317     \group_begin:
1318     \dim_set_eq:NN \nullldelimiterspace \c_zero_dim
1319     \c_math_toggle_token
1320     \left \l_@@_replace_left_brace_by_tl
1321     \vcenter to \l_tmpa_dim { \vfil }
1322     \right.
1323     \c_math_toggle_token
1324     \group_end:
1325     \hfil
1326   }
1327   \hfil
1328 }
1329 \skip_horizontal:N -\l_@@_linewidth_dim
1330 \vcenter { \box_use_drop:N \l_@@_halign_box }
1331 }

```

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.

```

1332 \dim_gzero_new:N \g_@@_right_x_dim
1333 \dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
1334 \pgfpicture
1335 \pgfrememberpicturepositiononpagetrue
1336 \int_step_variable:nNn \g_@@_line_int \l_tmpa_int
1337 {
1338   \cs_if_free:cTF
1339   { pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
1340   { \@@_fatal:n { Inexistent~v-node } }
1341   {
1342     \pgfpointanchor
1343     { wa - \l_@@_prefix_str - \l_tmpa_int - v }
1344     { center }
1345     \dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
1346     { \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x }
1347   }
1348 }
1349 \endpgfpicture

```

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

```

1350 \@@_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).

```

1351 (*LaTeX)
1352 \bool_if:nT \c_@@_mathtools_loaded_bool
1353 { \MH_if_boolean:nT { show_only_refs } \MT_showonlyrefs_true: }
1354 \bool_if:NTF \l_@@_in_label_or_minipage_bool
1355 {
1356   \c_math_toggle_token
1357   \skip_vertical:N \belowdisplayskip
1358 }
1359 { \c_math_toggle_token \c_math_toggle_token }
1360 
1361 (*plain-TeX)
1362 \c_math_toggle_token \c_math_toggle_token
1363 
1364 (*LaTeX)
1365 \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`).

```

1366   \bool_if:NT \g_@@_footnote_bool { \end { savenotes } }
1367 </LaTeX>
1368 <*plain-TeX>
1369   \group_end:
1370 </plain-TeX>
1371 <*LaTeX>
1372   \ignorespacesafterend
1373 </LaTeX>
1374 }

```

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.

```

1375 <*LaTeX>
1376 \NewDocumentEnvironment { DispWithArrows* } { }
1377 {
1378   \WithArrowsOptions { notag }
1379   \DispWithArrows
1380 }
1381 \endDispWithArrows
1382 </LaTeX>

```

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

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

```

1383 \cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
1384 {
1385   \bool_if:NTF \l_@@_in-WithArrows_bool
1386   { \@@_error:nn { Not~allowed~in-WithArrows } { #1 } }
1387   {
1388     \int_compare:nNnTF \g_@@_col_int < \l_@@_nb_cols_int
1389     { \@@_error:nn { Not~allowed~in-DispWithArrows } { #1 } }
1390     { #2 }
1391   }
1392 }

```

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

```

1393 <*LaTeX>
1394 \cs_new_protected:Npn \@@_notag:
1395 { \@@_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}`.

```

1396 \cs_new_protected:Npn \@@_nonumber:
1397 { \@@_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.

```

1398 \NewDocumentCommand \@@_tag { s m }
1399 {

```

```

1400 \@@_if_in_last_col_of_disp:Nn \tag
1401 {
1402   \tl_if_empty:NF \l_@@_tag_tl
1403   { \@@_error:nn { Multiple~tags } { #2 } }
1404   \clist_set:Nn \l_@@_tags_clist { all }
1405   \bool_if:nT \c_@@_mathtools_loaded_bool
1406   {
1407     \MH_if_boolean:nT { show_only_refs }
1408     {
1409       \MH_if_boolean:nF { show_manual_tags }
1410       { \clist_clear:N \l_@@_tags_clist }
1411     }
1412   }
1413   \tl_set:Nn \l_@@_tag_tl { #2 }
1414   \bool_set:Nn \l_@@_tag_star_bool { #1 }

```

The starred version `\tag*` can't be used if `amsmath` has not been loaded because this version does the job by deactivating the command `\tagform@` inserted by `amsmath` in the (two versions of the) command `\@eqnnum`.³⁹

```

1415   \bool_if:nT { #1 && ! \bool_if_p:N \c_@@_amsmath_loaded_bool }
1416   { \@@_error:n { tag*~without~amsmath } }
1417 }
1418 }

```

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

```

1419 \cs_new_protected:Npn \@@_label:n #1
1420 {
1421   \@@_if_in_last_col_of_disp:Nn \label
1422   {
1423     \seq_if_empty:NF \l_@@_labels_seq
1424     {
1425       \bool_if:NTF \c_@@_cleveref_loaded_bool
1426       { \@@_error:n { Multiple~labels~with~cleveref } }
1427       { \@@_error:n { Multiple~labels } }
1428     }
1429     \seq_put_right:Nn \l_@@_labels_seq { #1 }
1430     \bool_if:nT \c_@@_mathtools_loaded_bool
1431     {
1432       \MH_if_boolean:nT { show_only_refs }
1433       {
1434         \cs_if_exist:cTF { MT_r_#1 }
1435         { \clist_set:Nn \l_@@_tags_clist { all } }
1436         { \clist_clear:N \l_@@_tags_clist }
1437       }
1438     }
1439     \bool_if:nT \c_@@_autonum_loaded_bool
1440     {
1441       \cs_if_exist:cTF { autonum@#1Referenced }
1442       { \clist_set:Nn \l_@@_tags_clist { all } }
1443       { \clist_clear:N \l_@@_tags_clist }
1444     }
1445   }
1446 }

```

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

```

1447 \cs_new_protected:Npn \@@_tagnextline:

```

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

```

1448 {
1449   \@@_if_in_last_col_of_disp:Nn \tagnextline
1450   { \bool_set_true:N \l_@@_tag_next_line_bool }
1451 }

```

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`.

```

1452 \cs_new_protected:Npn \@@_qedhere: { \bool_set_true:N \l_@@_qedhere_bool }
1453 \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`.

```

1454 \cs_new_protected:Npn \@@_qedhere_i:
1455 {
1456   \group_begin:
1457   \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.

```

1458   \cs_set_eq:NN \qed@elt \setQED@elt
1459   \QED@stack \relax \relax
1460   \group_end:
1461 }
1462 </LaTeX>

```

12.10 We draw the arrows

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

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

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

```

1463 \cs_new_protected:Npn \@@_scan_arrows:
1464 {
1465   \group_begin:
1466   \int_compare:nNnT \l_@@_pos_arrow_int = 7
1467   {
1468     \@@_scan_arrows_i:
1469     \int_set:Nn \l_@@_pos_arrow_int 8
1470   }
1471   \@@_scan_arrows_i:
1472   \group_end:
1473 }

1474 \cs_new_protected:Npn \@@_scan_arrows_i:
1475 {

```

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

```

1476   \int_zero_new:N \l_@@_first_arrow_of_group_int
1477   \int_zero_new:N \l_@@_first_line_of_group_int
1478   \int_zero_new:N \l_@@_last_line_of_group_int
1479   \seq_clear_new:N \l_@@_first_arrows_seq
1480   \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.

```

1481   \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.

```

1482   \int_set:Nn \l_@@_arrow_int 1
1483   \int_until_do:nNnn \l_@@_arrow_int > \g_@@_arrow_int
1484   {

```

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.

```

1485       \prop_get:cnN
1486       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1487       { initial } \l_tmpa_tl
1488       \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1489       \prop_get:cnN
1490       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1491       { final } \l_tmpa_tl
1492       \int_set:Nn \l_@@_final_int \l_tmpa_tl
1493       \prop_get:cnN
1494       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1495       { status } \l_@@_status_arrow_str
1496       \prop_get:cnN
1497       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1498       { 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`.

```

1499       \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
1500       {
1501         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1502         { \@@_error:n { Too~few~lines~for~an~arrow } }
1503       }
1504       \@@_treat_an_arrow_in_scan:

```

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

```

1505       \int_incr:N \l_@@_arrow_int
1506   }

```

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).

```

1507 \bool_if:nT
1508 {
1509     ! \int_compare_p:nNn \l_@@_pos_arrow_int = 7
1510     &&
1511     \int_compare_p:nNn \l_@@_first_arrow_of_group_int > 0
1512 }
1513 { \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int }
1514 }

```

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`)

```

1515 \cs_new_protected:Npn \@@_treat_an_arrow_in_scan:
1516 {

```

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`).

```

1517 \bool_lazy_and:nnT
1518 { \int_compare_p:nNn \l_@@_arrow_int > 1 }
1519 {
1520     \bool_lazy_or_p:nn
1521     {
1522         \bool_lazy_and_p:nn
1523         {
1524             \int_compare_p:nNn
1525             \l_@@_initial_int > \l_@@_last_line_of_group_int
1526         }
1527         { \bool_not_p:n { \int_compare_p:nNn \l_@@_pos_arrow_int = 7 } }
1528     }
1529     { \str_if_eq_p:Vn \l_@@_status_arrow_str { new-group } }
1530 }
1531 {
1532     \int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
1533     {
1534         \@@_draw_arrows:nn
1535         \l_@@_first_arrow_of_group_int
1536         { \l_@@_arrow_int - 1 }
1537     }
1538     \bool_set_true:N \l_@@_new_group_bool
1539 }

```

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

```

1540 \bool_if:nTF \l_@@_new_group_bool
1541 {
1542     \bool_set_false:N \l_@@_new_group_bool
1543     \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
1544     \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
1545     \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1546     \seq_clear:N \l_@@_first_arrows_seq
1547     \seq_put_left:Nv \l_@@_first_arrows_seq \l_@@_arrow_int

```

```

1548     \seq_clear:N \l_@@_last_arrows_seq
1549     \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.

```

1550     \int_compare:nNnF \l_@@_pos_arrow_int = 8
1551       { \dim_set:Nn \l_@@_x_dim { - \c_max_dim } }
1552   }

```

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

```

1553   {

```

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

```

1554     \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1555     {

```

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

```

1556         \int_compare:nT
1557           { \l_@@_initial_int = \l_@@_first_line_of_group_int }
1558           { \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int }
1559         \int_compare:nNnTF \l_@@_final_int > \l_@@_last_line_of_group_int
1560         {
1561             \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
1562             \seq_clear:N \l_@@_last_arrows_seq
1563             \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
1564         }
1565         {
1566             \int_compare:nNnT \l_@@_final_int = \l_@@_last_line_of_group_int
1567               { \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int }
1568         }
1569     }
1570 }

```

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.

```

1571     \str_if_eq:VnF \l_@@_status_arrow_str { independent }
1572     {
1573         \int_compare:nNnF \l_@@_pos_arrow_int = 8
1574           { \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
1575     }
1576 }

```

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

```

1577 \cs_generate_variant:Nn \keys_set:nn { n o }
1578 \cs_new_protected:Npn \@@_keys_set:
1579   { \keys_set_known:no { WithArrows / Arrow / SecondPass } }

```

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

```

1580 \cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
1581 {
1582   \group_begin:
1583   \int_zero_new:N \l_@@_first_arrow_int
1584   \int_set:Nn \l_@@_first_arrow_int { #1 }
1585   \int_zero_new:N \l_@@_last_arrow_int
1586   \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.

```

1587   \int_set:Nn \l_@@_arrow_int \l_@@_first_arrow_int
1588   \int_until_do:nNnn \l_@@_arrow_int > \l_@@_last_arrow_int
1589   {

```

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.

```

1590       \prop_get:cnN
1591       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1592       { initial } \l_tmpa_tl
1593   \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1594   \prop_get:cnN
1595   { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1596   { final } \l_tmpa_tl
1597   \int_set:Nn \l_@@_final_int \l_tmpa_tl
1598   \prop_get:cnN
1599   { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1600   { 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).

```

1601   \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).

```

1602   {
1603       \str_if_eq:VnTF \l_@@_status_arrow_str { over }
1604       { \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
1605       \@@_draw_arrow:
1606   }
1607   \int_incr:N \l_@@_arrow_int
1608   }
1609   \@@_draw_o_arrows_of_the_group:
1610   \group_end:
1611   }

```

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).

```

1612   \cs_new_protected:Npn \@@_draw_arrow:
1613   {
1614       \group_begin:

```

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

```

1615       \prop_get:cnN
1616       { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1617       { options } \l_tmpa_tl
1618   \str_clear_new:N \l_@@_previous_key_str
1619   \exp_args:NNo \exp_args:No
1620   \@@_keys_set: { \l_tmpa_tl , tikz = { xshift = \l_@@_xoffset_dim } }

```

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

```

1621 \bool_set_false:N \l_@@_initial_r_bool
1622 \bool_set_false:N \l_@@_final_r_bool
1623 \int_case:nn \l_@@_pos_arrow_int
1624 {
1625   0 { \bool_set_true:N \l_@@_final_r_bool }
1626   2 { \bool_set_true:N \l_@@_initial_r_bool }
1627   3
1628   {
1629     \bool_set_true:N \l_@@_initial_r_bool
1630     \bool_set_true:N \l_@@_final_r_bool
1631   }
1632 }

```

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).

```

1633 \int_compare:nNnT \l_@@_pos_arrow_int = 5
1634 {
1635   \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
1636   \@@_update_x:nn \l_@@_initial_int \l_@@_final_int
1637 }

```

\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.

```

1638 \tl_set:Nx \l_@@_initial_tl
1639 { \int_use:N \l_@@_initial_int - \bool_if:NTF \l_@@_initial_r_bool rl }
1640 \tl_set:Nx \l_@@_final_tl
1641 { \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.

```

1642 \prop_get:cnN
1643 { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1644 { label }
1645 \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.

```

1646 \seq_if_in:NxTF \l_@@_first_arrows_seq
1647 { \int_use:N \l_@@_arrow_int }
1648 { \bool_set_true:N \l_tmpa_bool }
1649 { \bool_set_false:N \l_tmpa_bool }
1650 \seq_if_in:NxTF \l_@@_last_arrows_seq
1651 { \int_use:N \l_@@_arrow_int }
1652 { \bool_set_true:N \l_tmpb_bool }
1653 { \bool_set_false:N \l_tmpb_bool }
1654 \int_compare:nNnT \l_@@_pos_arrow_int = 5

```

```

1655 {
1656   \bool_set_true:N \l_tmpa_bool
1657   \bool_set_true:N \l_tmpb_bool
1658 }

```

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

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

```

1659 \dim_gzero_new:N \g_@@_x_initial_dim
1660 \dim_gzero_new:N \g_@@_x_final_dim
1661 \dim_gzero_new:N \g_@@_y_initial_dim
1662 \dim_gzero_new:N \g_@@_y_final_dim
1663 \pgfpicture
1664   \pgfrememberpicturepositiononpagetrue
1665   \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south }
1666   \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1667   \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1668   \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north }
1669   \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1670   \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1671 \endpgfpicture
1672 \bool_lazy_and:nnTF
1673 {
1674   \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim }
1675     > \l_@@_max_length_of_arrow_dim
1676 }
1677 { \int_compare_p:nNn { \l_@@_final_int - \l_@@_initial_int } = 1 }
1678 {
1679   \tl_gset:Nx \g_tmpa_tl
1680   {
1681     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1682     { \dim_use:N \g_@@_x_initial_dim }
1683     { \dim_use:N \l_@@_x_dim } ,
1684     \dim_eval:n
1685     {
1686       ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1687       + 0.5 \l_@@_max_length_of_arrow_dim
1688     }
1689   }
1690   \tl_gset:Nx \g_tmpb_tl
1691   {
1692     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1693     { \dim_use:N \g_@@_x_final_dim }
1694     { \dim_use:N \l_@@_x_dim } ,
1695     \dim_eval:n
1696     {
1697       ( \g_@@_y_initial_dim + \g_@@_y_final_dim ) / 2
1698       - 0.5 \l_@@_max_length_of_arrow_dim
1699     }
1700   }
1701 }
1702 {
1703   \tl_gset:Nx \g_tmpa_tl
1704   {
1705     \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1706     { \dim_use:N \g_@@_x_initial_dim }
1707     { \dim_use:N \l_@@_x_dim } ,

```

```

1708         \bool_if:NTF \l_tmpa_bool
1709         { \dim_eval:n { \g_@@_y_initial_dim + \l_@@_start_adjust_dim } }
1710         { \dim_use:N \g_@@_y_initial_dim }
1711     }
1712     \tl_gset:Nx \g_tmpb_tl
1713     {
1714         \int_compare:nNnTF \l_@@_pos_arrow_int < 5
1715         { \dim_use:N \g_@@_x_final_dim }
1716         { \dim_use:N \l_@@_x_dim } ,
1717         \bool_if:NTF \l_tmpb_bool
1718         { \dim_eval:n { \g_@@_y_final_dim - \l_@@_end_adjust_dim } }
1719         { \dim_use:N \g_@@_y_final_dim }
1720     }
1721 }

```

Eventually, we can draw the arrow with the code in `\l_@@_tikz_code_tl`. We recall that the value by default for this token list is: “`\draw (#1) to node {#3} (#2) ;`”. This value can be modified with the option `tikz-code`. We use the variant `@@_draw_arrow:nno` of the macro `@@_draw_arrow:nnn` because of the characters *underscore* in the name `\l_tmpa_tl`: if the user uses the Tikz library `babel`, the third argument of the command `@@_draw_arrow:nno` will be rescanned because this third argument will be in the argument of a command `node` of an instruction `\draw` of Tikz... and we will have an error because of the characters *underscore*.⁴⁰

```

1722     @@_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).

```

1723     \group_end:
1724 }

```

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

```

1725 \cs_new_protected:Npn @@_def_function_tmpa:n #1
1726 {
1727     \cs_set:Npn @@_tmpa:nnn ##1 ##2 ##3
1728     {
1729         <*LaTeX>
1730         \begin{tikzpicture}
1731         </LaTeX>
1732         <*plain-TeX>
1733         \tikzpicture
1734         </plain-TeX>
1735         [
1736             @@_standard ,
1737             every~path / .style = WithArrows / arrow
1738         ]

```

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`.

```

1739         \pgf@relevantforpicturesizetrue
1740         #1
1741         \dim_compare:nNnTF \pgf@picminx = { 16000 pt }
1742         { \dim_set_eq:NN \l_tmpa_dim \c_zero_dim }
1743         { \dim_set:Nn \l_tmpa_dim { \pgf@picmaxx - \pgf@picminx } }
1744         \dim_add:Nn \l_tmpa_dim \l_@@_xoffset_dim
1745         \prop_gput:cnV
1746         { g_@@_arrow _ \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop }
1747         { width }

```

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

```

1748         \l_tmpa_dim
1749     \pgfresetboundingbox
1750 < *LaTeX>
1751     \end{tikzpicture}
1752 < /LaTeX>
1753 < *plain-TeX>
1754     \endtikzpicture
1755 < /plain-TeX>
1756 }
1757 }

```

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

```

1758 \cs_new_protected:Npn \@@_draw_arrow:nnn #1 #2 #3
1759 {

```

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`).

```

1760     \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
1761     { \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_wrap_lines_tl }

```

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

```

1762     \exp_args:NV \@@_def_function_tmpa:n \l_@@_tikz_code_tl
1763     \@@_tmpa:nnn { #1 } { #2 } { #3 }
1764 }
1765 \cs_generate_variant:Nn \@@_draw_arrow:nnn { n n o }

```

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

```

1766 \tl_const:Nn \c_@@_tikz_code_wrap_lines_tl
1767 {

```

First, we draw the arrow without the label.

```

1768     \draw ( #1 ) to node ( @@_label ) { } ( #2 ) ;

```

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

```

1769     \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.

```

1770     \dim_set:Nn \l_tmpa_dim
1771     { \g_@@_right_x_dim - \pgf@x - \pgfkeysvalueof { / pgf / inner~xsep } }

```

We retrieve in `\g_tmpa_tl` the current value of the Tikz parameter “text width”.⁴¹

```

1772     \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;

```

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

```

1773     \tl_if_empty:NF \g_tmpa_tl
1774     {
1775         \dim_set:Nn \l_tmpb_dim \g_tmpa_tl
1776         \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
1777         { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
1778     }

```

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

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

```

1779 \dim_compare:nNnT \l_tmpa_dim > \c_zero_dim
1780 {
1781   \path ( @@_label.west )
1782   (*LaTeX)
1783     node [ anchor = west ]
1784     {
1785       \begin { varwidth } { \l_tmpa_dim }
1786       #3
1787       \end { varwidth }
1788     } ;
1789   (/LaTeX)
1790   (*plain-TeX)
1791     node [ anchor = west , text-width = \dim_use:N \l_tmpa_dim ]
1792     { #3 } ;
1793   (/plain-TeX)
1794 }
1795 }
```

12.10.1 The command `update_x`

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

```

1796 \cs_new_protected:Npn \@@update_x:nn #1 #2
1797 {
1798   \dim_gset_eq:NN \g_tmpa_dim \l_@@_x_dim
1799   \pgfpicture
1800   \pgfrememberpicturepositiononpagetrue
1801   \int_step_inline:nnn { #1 } { #2 }
1802   {
1803     \pgfpointanchor { wa - \l_@@_prefix_str - ##1 - 1 } { center }
1804     \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \pgf@x }
1805   }
1806   \endpgfpicture
1807   \dim_set_eq:NN \l_@@_x_dim \g_tmpa_dim
1808 }
```

12.10.2 We draw the arrows of type `o`

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

```

1809 \cs_new_protected:Npn \@@draw_o_arrows_of_the_group:
1810 {
```

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

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

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

`$\begin{WithArrows}[groups]`

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

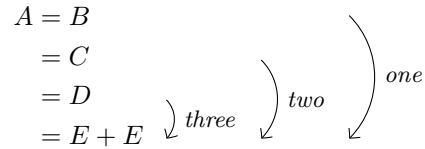
`\end{WithArrows}$`

$$A = B$$

$$= C$$

$$= D$$

$$= E + E$$



```
1811 \seq_sort:Nn \l_@@_o_arrows_seq
1812 {
1813   \prop_get:cnN
1814   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1815   { final } \l_tmpa_tl
```

We recall that `\prop_get:cnN` retrieves token lists (here `\l_tmpa_tl` and `\l_tmpb_tl`). We don't need to do an explicit conversion in `expl3` integers because such token lists can be used directly in `\int_compare:nNnTF`.

```
1816   \prop_get:cnN
1817   { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1818   { final } \l_tmpb_tl
1819   \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1820   \sort_return_same:
1821   {
1822     \int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
1823     \sort_return_swapped:
1824     {
1825       \prop_get:cnN
1826       { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1827       { initial } \l_tmpa_tl
1828       \prop_get:cnN
1829       { g_@@_arrow _ \l_@@_prefix_str _ ##2 _ prop }
1830       { initial } \l_tmpb_tl
1831       \int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
1832       \sort_return_swapped:
1833       \sort_return_same:
1834     }
1835   }
1836 }
```

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

```
1837 \seq_map_inline:Nn \l_@@_o_arrows_seq
1838 {
```

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

```
1839   \prop_get:cnN
1840   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1841   { initial } \l_tmpa_tl
1842   \int_set:Nn \l_@@_initial_int \l_tmpa_tl
1843   \prop_get:cnN
1844   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1845   { final } \l_tmpa_tl
1846   \int_set:Nn \l_@@_final_int \l_tmpa_tl
```

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

```
1847   \prop_get:cnN
1848   { g_@@_arrow _ \l_@@_prefix_str _ ##1 _ prop }
1849   { input-line } \l_@@_input_line_str
```

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

```
1850   \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 error⁴²).

```

1851 \bool_set_false:N \g_tmpa_bool
1852 \pgfpicture
1853 \pgfrememberpicturepositiononpagetrue
1854 \int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
1855 {
1856   \prop_get:cnN
1857   { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1858   { initial } \l_tmpa_tl
1859   \prop_get:cnN
1860   { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1861   { final } \l_tmpb_tl
1862   \prop_get:cnN
1863   { g_@@_arrow _ \l_@@_prefix_str _ #####1 _ prop }
1864   { status } \l_@@_status_arrow_str
1865   \bool_if:nT
1866   {
1867     ! \int_compare_p:n { ##1 = #####1 }
1868     && \int_compare_p:n { \l_@@_initial_int <= \l_tmpa_tl }
1869     && \int_compare_p:n { \l_tmpb_tl <= \l_@@_final_int }

```

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

```

1870     && ! \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
1871   }
1872 {

```

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

```

1873 \bool_gset_true:N \g_tmpa_bool
1874 \prop_get:cnN
1875 { g_@@ _ arrow _ \l_@@_prefix_str _ #####1 _ prop }
1876 { width }
1877 \l_tmpa_tl

```

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

```

1878 \dim_gset:Nn \g_tmpa_dim { \dim_max:nn \g_tmpa_dim \l_tmpa_tl }
1879 }
1880 }
1881 \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).

```

1882 \bool_if:NTF \g_tmpa_bool
1883 {
1884   \int_set:Nn \l_@@_arrow_int { ##1 }
1885   \dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
1886   \dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
1887   \@@_draw_arrow:
1888 }
1889 { \@@_error:n { o~arrow~with~no~arrow~under } }
1890 }
1891 }

```

The command `\WithArrowsLastEnv` is not used by the package `witharrows`. It’s only a facility given to the final user. It gives the number of the last environment `{WithArrows}` at level 0 (to the sense of the nested environments). This macro is fully expandable and, thus, can be used directly in the name of a Tikz node.

```

1892 \*LaTeX
1893 \NewExpandableDocumentCommand \WithArrowsLastEnv { }

```

⁴²Maybe we will change that in future versions.


```

1894 { \int_use:N \g_@@_last_env_int }
1895 </LaTeX>
1896 <*plain-TeX>
1897 \cs_new:Npn \WithArrowsLastEnv { \int_use:N \g_@@_last_env_int }
1898 </plain-TeX>

```

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

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

```

1899 \keys_define:nn { WithArrows / Arrow / code-after }
1900 {
1901   tikz      .code:n =
1902     \tikzset { WithArrows / arrow / .append-style = { #1 } } ,
1903   tikz      .value_required:n = true ,
1904   rr        .value_forbidden:n = true ,
1905   rr        .code:n      = \@@_fix_pos_option:n 0 ,
1906   ll        .value_forbidden:n = true ,
1907   ll        .code:n      = \@@_fix_pos_option:n 1 ,
1908   rl        .value_forbidden:n = true ,
1909   rl        .code:n      = \@@_fix_pos_option:n 2 ,
1910   lr        .value_forbidden:n = true ,
1911   lr        .code:n      = \@@_fix_pos_option:n 3 ,
1912   v         .value_forbidden:n = true ,
1913   v         .code:n      = \@@_fix_pos_option:n 4 ,
1914   tikz-code .tl_set:N      = \l_@@_tikz_code_tl ,
1915   tikz-code .value_required:n = true ,
1916   xoffset   .dim_set:N      = \l_@@_xoffset_dim ,
1917   xoffset   .value_required:n = true ,
1918   unknown   .code:n =
1919     \@@_sort_seq:N \l_@@_options_Arrow_code_after_seq
1920     \@@_error:n { Unknown-option-Arrow-in-code-after }
1921 }

```

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.

```

1922 \seq_new:N \l_@@_options_Arrow_code_after_seq
1923 \@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq
1924 { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }

```

```

1925 <*LaTeX>
1926 \NewDocumentCommand \@@_Arrow_code_after { 0 { } m m m ! 0 { } }
1927 </LaTeX>
1928 <*plain-TeX>
1929 \cs_new_protected:Npn \@@_Arrow_code_after
1930 {
1931   \peek_meaning:NTF [
1932     { \@@_Arrow_code_after_i }
1933     { \@@_Arrow_code_after_i [ ] }
1934   }
1935   \cs_new_protected:Npn \@@_Arrow_code_after_i [ #1 ] #2 #3 #4
1936   {
1937     \peek_meaning:NTF [
1938       { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } }
1939       { \@@_Arrow_code_after_ii [ #1 ] { #2 } { #3 } { #4 } [ ] }
1940     }
1941     \cs_new_protected:Npn \@@_Arrow_code_after_ii [ #1 ] #2 #3 #4 [ #5 ]
1942   </plain-TeX>

```

```

1943 {
1944   \int_set:Nn \l_@@_pos_arrow_int 1
1945   \str_clear_new:N \l_@@_previous_key_str
1946   \group_begin:
1947     \keys_set:nn { WithArrows / Arrow / code-after }
1948     { #1, #5, tikz = { xshift = \l_@@_xoffset_dim } }
1949     \bool_set_false:N \l_@@_initial_r_bool
1950     \bool_set_false:N \l_@@_final_r_bool
1951     \int_case:nn \l_@@_pos_arrow_int
1952     {
1953       0
1954       {
1955         \bool_set_true:N \l_@@_initial_r_bool
1956         \bool_set_true:N \l_@@_final_r_bool
1957       }
1958       2 { \bool_set_true:N \l_@@_initial_r_bool }
1959       3 { \bool_set_true:N \l_@@_final_r_bool }
1960     }

```

We prevent drawing an arrow from a line to itself.

```

1961   \tl_if_eq:nnTF { #2 } { #3 }
1962   { \@@_error:nn { Both-lines-are-equal } { #2 } }

```

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

```

1963 {
1964   \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #2 - 1 }
1965   { \@@_error:nx { Wrong-line-in-Arrow } { #2 } }
1966   {
1967     \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - #3 - 1 }
1968     { \@@_error:nx { Wrong-line-in-Arrow } { #3 } }
1969     {
1970       \int_compare:nNnTF \l_@@_pos_arrow_int = 4
1971       {
1972         \pgfpicture
1973         \pgfrememberpicturepositiononpagetrue
1974         \pgfpointanchor { wa - \l_@@_prefix_str - #2 - 1 }
1975         { south }
1976         \dim_set_eq:NN \l_tmpa_dim \pgf@x
1977         \dim_set_eq:NN \l_tmpb_dim \pgf@y
1978         \pgfpointanchor { wa - \l_@@_prefix_str - #3 - 1 }
1979         { north }
1980         \dim_set:Nn \l_tmpa_dim
1981         { \dim_max:nn \l_tmpa_dim \pgf@x }
1982         \tl_gset:Nx \g_tmpa_tl
1983         { \dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim }
1984         \tl_gset:Nx \g_tmpb_tl
1985         { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
1986         \endpgfpicture
1987       }
1988       {
1989         \pgfpicture
1990         \pgfrememberpicturepositiononpagetrue
1991         \pgfpointanchor
1992         {
1993           wa - \l_@@_prefix_str -
1994           #2 - \bool_if:NTF \l_@@_initial_r_bool r l
1995         }
1996         { south }
1997         \tl_gset:Nx \g_tmpa_tl
1998         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
1999         \pgfpointanchor
2000         {
2001           wa - \l_@@_prefix_str -

```

```

2002             #3 - \bool_if:NTF \l_@@_final_r_bool r l
2003         }
2004         { north }
2005         \tl_gset:Nx \g_tmpb_tl
2006         { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
2007         \endpgfpicture
2008     }
2009     \@@_draw_arrow:nnn \g_tmpa_tl \g_tmpb_tl { #4 }
2010 }
2011 }
2012 }
2013 \group_end:
2014 }

```

12.12 The command `\MultiArrow` in code-after

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

```

2015 \cs_new_protected:Npn \@@_MultiArrow:nn #1 #2
2016 {

```

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.

```

2017     \exp_args:Nnx
2018     \regex_match:nnTF
2019     { \A \d+ (\,\d+)* ( \, \.\.\. (\,\d+)+ )* \Z }
2020     { #1 }
2021     { \@@_MultiArrow_i:nn { #1 } { #2 } }
2022     { \@@_error:nx { Invalid-specification-for-MultiArrow } { #1 } }
2023 }
2024 \cs_new_protected:Npn \@@_MultiArrow_i:nn #1 #2
2025 {

```

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

```

2026     \foreach \x in { #1 }
2027     {
2028         \cs_if_free:cTF { pgf@sh@ns@wa - \l_@@_prefix_str - \x - 1 }
2029         { \@@_error:nx { Wrong-line-specification-in-MultiArrow } \x }
2030         { \clist_gput_right:Nx \g_tmpa_clist \x }
2031     }

```

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

```

2032     \int_compare:nTF { \clist_count:N \g_tmpa_clist < 2 }
2033     { \@@_error:n { Too-small-specification-for-MultiArrow } }
2034     {
2035         \clist_sort:Nn \g_tmpa_clist
2036         {
2037             \int_compare:nTF { ##1 > ##2 }
2038             \sort_return_swapped:
2039             \sort_return_same:
2040         }

```

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

```

2041     \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...

```

2042     \clist_reverse:N \g_tmpa_clist
2043     \clist_pop:NN \g_tmpa_clist \l_tmpb_tl

```

We draw the teeth of the rak (except the first one and the last one) with the auxiliary function `\@@_MultiArrow_i:n`. This auxiliary function is necessary to expand the specification of the list in the `\foreach` loop. The first and the last teeth of the rak can't be drawn the same way as the others (think, for example, to the case of the option “rounded corners” is used).

```
2044 \exp_args:NV \@@_MultiArrow_i:n \g_tmpa_clist
```

Now, we draw the rest of the structure.

```
2045 <*LaTeX>
2046 \begin { tikzpicture }
2047 </LaTeX>
2048 <*plain-TeX>
2049 \tikzpicture
2050 </plain-TeX>
2051 [
2052 \@@_standard ,
2053 every~path / .style = { WithArrows / arrow }
2054 ]
2055 \draw [<->] ([xshift = \l_@@_xoffset_dim]\l_tmpa_tl-r.south)
2056 -- ++(5mm,0)
2057 -- node (@@_label) {}
2058 ([xshift = \l_@@_xoffset_dim+5mm]\l_tmpb_tl-r.south)
2059 -- ([xshift = \l_@@_xoffset_dim]\l_tmpb_tl-r.south) ;
2060
2061 \pgfpointanchor { wa - \l_@@_prefix_str - @@_label } { west }
2062 \dim_set:Nn \l_tmpa_dim { 20 cm }
2063 \path \pgfextra { \tl_gset:Nx \g_tmpa_tl \tikz@text@width } ;
2064 \tl_if_empty:NF \g_tmpa_tl { \dim_set:Nn \l_tmpa_dim \g_tmpa_tl }
2065 \bool_lazy_and:nnT \l_@@_wrap_lines_bool \l_@@_in_DispWithArrows_bool
2066 {
2067 \dim_set:Nn \l_tmpb_dim
2068 { \g_@@_right_x_dim - \pgf@x - 0.3333 em }
2069 \dim_compare:nNnT \l_tmpb_dim < \l_tmpa_dim
2070 { \dim_set_eq:NN \l_tmpa_dim \l_tmpb_dim }
2071 }
2072 \path (@@_label.west)
2073 node [ anchor = west, text-width = \dim_use:N \l_tmpa_dim ] { #2 } ;
2074 <*LaTeX>
2075 \end { tikzpicture }
2076 </LaTeX>
2077 <*plain-TeX>
2078 \endtikzpicture
2079 </plain-TeX>
2080 }
2081 }
2082 \cs_new_protected:Npn \@@_MultiArrow_i:n #1
2083 {
2084 <*LaTeX>
2085 \begin { tikzpicture }
2086 </LaTeX>
2087 <*plain-TeX>
2088 \tikzpicture
2089 </plain-TeX>
2090 [
2091 \@@_standard ,
2092 every~path / .style = { WithArrows / arrow }
2093 ]
2094 \foreach \k in { #1 }
2095 {
2096 \draw [ <- ]
2097 ( [xshift = \l_@@_xoffset_dim]\k-r.south ) -- ++(5mm,0) ;
2098 } ;
2099 <*LaTeX>
2100 \end{tikzpicture}
```

```

2101 </LaTeX>
2102 <*plain-TeX>
2103     \endtikzpicture
2104 </plain-TeX>
2105 }

```

12.13 The error messages of the package

```

2106 \str_const:Nn \c_@@_option_ignored_str
2107 { If-you-go-on,~this-option-will-be-ignored. }
2108 \str_const:Nn \c_@@_command_ignored_str
2109 { If-you-go-on,~this-command-will-be-ignored. }
2110 <*LaTeX>
2111 \@@_msg_new:nn { amsmath-not-loaded }
2112 {
2113     You~can't~use~the~option~'\l_keys_key_str'~because~the~
2114     package~'amsmath'~has~not~been~loaded.\\
2115     If~you~go~on,~this~option~will~be~ignored~in~the~rest~
2116     of~the~document.
2117 }
2118 </LaTeX>
2119 \@@_msg_new:nn { Bad-value-for-replace-brace-by }
2120 {
2121     Bad-value-for-the-option~'\l_keys_key_str'.~The-value-must-begin~
2122     with~an~extensible~left~delimiter.~The-possible-values-are:~.,
2123     \token_to_str:N \{,~(,~[,~\token_to_str:N \lbrace,~
2124     \token_to_str:N \lbrack,~\token_to_str:N \lgroup,~
2125     \token_to_str:N \langle,~\token_to_str:N \lmoustache,~
2126     \token_to_str:N \lfloor~and~\token_to_str:N \lceil~
2127     (and~\token_to_str:N \lvert~and~\token_to_str:N \lVert~
2128     if~amsmath~or~unicode-math~is~loaded~in~LaTeX).\\
2129     \c_@@_option_ignored_str
2130 }
2131 \@@_msg_new:nn { option-of-cr-negative }
2132 {
2133     The~argument~of~the~command~\token_to_str:N~~
2134     should~be~positive~in~the~row~\int_use:N \g_@@_line_int~
2135     of~your~environment~\{\l_@@_type_env_str\}.\\
2136     \c_@@_option_ignored_str
2137 }
2138 \@@_msg_new:nn { omit-probably-used }
2139 {
2140     There~is~a~problem.~Maybe~you~have~used~a~command~
2141     \token_to_str:N\omit~ in~the~line~\int_use:N \g_@@_line_int~
2142     (or~another~line)~of~your~environment~\{\l_@@_type_env_str\}.\\
2143     You~can~go~on~but~you~may~have~others~errors.
2144 }
2145 <*LaTeX>
2146 \@@_msg_new:nn { newline-at-the-end-of-env }
2147 {
2148     The~environments~of~witharrows~(\{WithArrows\})~and~
2149     \{DispWithArrows\})~should~not~end~by~\token_to_str:N \\.\\
2150     However,~you~can~go~on~for~this~time.~No~similar~error~will~be~
2151     raised~in~this~document.
2152 }
2153 </LaTeX>
2154 \@@_msg_new:nn { Invalid-option-format }
2155 {
2156     The~key~'format'~should~contain~only~letters~r,~c~and~l~and~
2157     must~not~be~empty.\\

```

```

2158     \c_@@_option_ignored_str
2159 }
2160 \@@_msg_new:nn { invalid~key~o }
2161 {
2162     The~key~'o'~for~individual~arrows~can~be~used~only~in~mode~
2163     'group'~or~in~mode~'groups'.\\
2164     \c_@@_option_ignored_str
2165 }
2166 \@@_msg_new:nn { Value~for~a~key }
2167 {
2168     The~key~'\l_keys_key_str'~should~be~used~without~value. \\
2169     However,~you~can~go~on~for~this~time.
2170 }
2171 \@@_msg_new:nnn { Unknown~option~in~Arrow }
2172 {
2173     The~key~'\l_keys_key_str'~is~unknown~for~the~command~
2174     \l_@@_string_Arrow_for_msg_str\ in~the~row~
2175     \int_use:N \g_@@_line_int\ of~your~environment~
2176     \{\l_@@_type_env_str\}. \l_tmpa_str \\
2177     \c_@@_option_ignored_str \\
2178     For~a~list~of~the~available~keys,~type~H~<return>.
2179 }
2180 {
2181     The~available~keys~are~(in~alphabetic~order):~
2182     \seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.
2183 }
2184 \@@_msg_new:nnn { Unknown~option~WithArrows }
2185 {
2186     The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
2187     \c_@@_option_ignored_str \\
2188     For~a~list~of~the~available~keys,~type~H~<return>.
2189 }
2190 {
2191     The~available~keys~are~(in~alphabetic~order):~
2192     \seq_use:Nnnn \l_@@_options_WithArrows_seq {~and~} {,~} {~and~}.
2193 }
2194 \@@_msg_new:nnn { Unknown~option~DispWithArrows }
2195 {
2196     The~key~'\l_keys_key_str'~is~unknown~in~\{\l_@@_type_env_str\}. \\
2197     \c_@@_option_ignored_str \\
2198     For~a~list~of~the~available~keys,~type~H~<return>.
2199 }
2200 {
2201     The~available~keys~are~(in~alphabetic~order):~
2202     \seq_use:Nnnn \l_@@_options_DispatchWithArrows_seq {~and~} {,~} {~and~}.
2203 }
2204 \@@_msg_new:nnn { Unknown~option~WithArrowsOptions }
2205 {
2206     The~key~'\l_keys_key_str'~is~unknown~in~
2207     \token_to_str:N \WithArrowsOptions. \\
2208     \c_@@_option_ignored_str \\
2209     For~a~list~of~the~available~keys,~type~H~<return>.
2210 }
2211 {
2212     The~available~keys~are~(in~alphabetic~order):~
2213     \seq_use:Nnnn \l_@@_options_WithArrowsOptions_seq {~and~} {,~} {~and~}.
2214 }
2215 \@@_msg_new:nnn { Unknown~option~Arrow~in~code~after }
2216 {
2217     The~key~'\l_keys_key_str'~is~unknown~in~
2218     \token_to_str:N \Arrow\ in~code~after. \\

```

```

2219 \c_@@_option_ignored_str \\
2220 For~a~list~of~the~available~keys,~type~H~<return>.
2221 }
2222 {
2223 The~available~keys~are~(in~alphabetic~order):~
2224 \seq~use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
2225 }
2226 \@@_msg_new:nn { Too~much~columns~in~WithArrows }
2227 {
2228 Your~environment~\{\l_@@_type_env_str\}~has~\int~use:N
2229 \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2230 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str.~
2231 If~you~really~want~to~use~more~columns~(after~the~arrows)~you~should~use~
2232 the~option~'more~columns'~at~a~global~level~or~for~an~environment. \\
2233 However,~you~can~go~one~for~this~time.
2234 }
2235 \@@_msg_new:nn { Too~much~columns~in~DispWithArrows }
2236 {
2237 Your~environment~\{\l_@@_type_env_str\}~has~\int~use:N
2238 \l_@@_nb_cols_int\ columns~and~you~try~to~use~one~more.~
2239 Maybe~you~have~forgotten~a~\c_backslash_str\c_backslash_str\
2240 at~the~end~of~row~\int~use:N \g_@@_line_int. \\
2241 This~error~is~fatal.
2242 }
2243 \@@_msg_new:nn { Negative~jump }
2244 {
2245 You~can't~use~a~negative~value~for~the~option~'jump'~of~command~
2246 \l_@@_string_Arrow_for_msg_str\
2247 in~the~row~\int~use:N \g_@@_line_int\
2248 of~your~environment~\{\l_@@_type_env_str\}.~
2249 You~can~create~an~arrow~going~backwards~with~the~option~'<-'~of~Tikz. \\
2250 \c_@@_option_ignored_str
2251 }
2252 \@@_msg_new:nn { new~group~without~groups }
2253 {
2254 You~can't~use~the~option~'new~group'~for~the~command~
2255 \l_@@_string_Arrow_for_msg_str\
2256 because~you~are~not~in~'groups'~mode.~Try~to~use~the~option~
2257 'groups'~in~your~environment~\{\l_@@_type_env_str\}. \\
2258 \c_@@_option_ignored_str
2259 }
2260 \@@_msg_new:nn
2261 { Too~few~lines~for~an~arrow }
2262 {
2263 Line~\l_@@_input_line_str\
2264 :~an~arrow~specifiead~in~the~row~\int~use:N \l_@@_initial_int\
2265 of~your~environment~\{\l_@@_type_env_str\}~can't~be~drawn~
2266 because~it~arrives~after~the~last~row~of~the~environment. \\
2267 If~you~go~on,~this~arrow~will~be~ignored.
2268 }
2269 \@@_msg_new:nn { o~arrow~with~no~arrow~under }
2270 {
2271 Line~\l_@@_input_line_str\
2272 :~there~is~no~arrow~'under'~your~arrow~of~type~'o'.\\
2273 If~you~go~on,~this~arrow~won't~be~drawn.
2274 }
2275 \@@_msg_new:nn { WithArrows~outside~math~mode }
2276 {
2277 The~environment~\{\l_@@_type_env_str\}~should~be~used~only~in~math~mode~
2278 like~the~environment~\{aligned\}~of~amsmath. \\
2279 Nevertheless,~you~can~go~on.

```

```

2280 }
2281 \@@_msg_new:nn { DispWithArrows~in~math~mode }
2282 {
2283   The~environment~\{\l_@@_type_env_str\}~should~be~used~only~outside~math~
2284   mode~like~the~environments~\{align\}~and~\{align*\}~of~amsmath. \\\
2285   This~error~is~fatal.
2286 }
2287 \@@_msg_new:nn { Incompatible~options~in~Arrow }
2288 {
2289   You~try~to~use~the~option~'\l_keys_key_str'~but~
2290   this~option~is~incompatible~or~redundant~with~the~option~
2291   '\l_@@_previous_key_str'~set~in~the~same~command~
2292   \l_@@_string_Arrow_for_msg_str. \\\
2293   \c_@@_option_ignored_str
2294 }
2295 \@@_msg_new:nn { Incompatible~options }
2296 { You~try~to~use~the~option~'\l_keys_key_str'~but~
2297   this~option~is~incompatible~or~redundant~with~the~option~
2298   '\l_@@_previous_key_str'~set~in~the~same~command~
2299   \bool_if:NT \l_@@_in_code_after_bool
2300   {
2301     \l_@@_string_Arrow_for_msg_str\
2302     in~the~code~after~of~your~environment~\{\l_@@_type_env_str\}
2303   }. \\\
2304   \c_@@_option_ignored_str
2305 }
2306 \@@_msg_new:nn { Arrow~not~in~last~column }
2307 {
2308   You~should~use~the~command~\l_@@_string_Arrow_for_msg_str\
2309   only~in~the~last~column~(column~\int_use:N\l_@@_nb_cols_int)~
2310   of~your~environment~\{\l_@@_type_env_str\}. \\\
2311   However~you~can~go~on~for~this~time.
2312 }
2313 \@@_msg_new:nn { Wrong~line~in~Arrow }
2314 {
2315   The~specification~of~line~'#1'~you~use~in~the~command~
2316   \l_@@_string_Arrow_for_msg_str\
2317   in~the~'code~after'~of~\{\l_@@_type_env_str\}~doesn't~exist. \\\
2318   \c_@@_option_ignored_str
2319 }
2320 \@@_msg_new:nn { Both~lines~are~equal }
2321 {
2322   In~the~'code~after'~of~\{\l_@@_type_env_str\}~you~try~to~
2323   draw~an~arrow~going~to~itself~from~the~line~'#1'.~This~is~not~possible. \\\
2324   \c_@@_option_ignored_str
2325 }
2326 \@@_msg_new:nn { Wrong~line~specification~in~MultiArrow }
2327 {
2328   The~specification~of~line~'#1'~doesn't~exist. \\\
2329   If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.
2330 }
2331 \@@_msg_new:nn { Too~small~specification~for~MultiArrow }
2332 {
2333   The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
2334   is~too~small:~you~need~at~least~two~lines. \\\
2335   \c_@@_command_ignored_str
2336 }
2337 \@@_msg_new:nn { Not~allowed~in~DispWithArrows }
2338 {
2339   The~command~\token_to_str:N #1

```



```

2340     is~allowed~only~in~the~last~column~
2341     (column~\int_use:N\l_@@_nb_cols_int)~of~\{\l_@@_type_env_str\}. \\
2342     \c_@@_option_ignored_str
2343   }
2344   \@@_msg_new:nn { Not~allowed~in~WithArrows }
2345   {
2346     The~command~\token_to_str:N #1 is~not~allowed~in~\{\l_@@_type_env_str\}~
2347     (it's~allowed~in~the~last~column~of~\{DispWithArrows\}). \\
2348     \c_@@_option_ignored_str
2349   }
2350   (*LaTeX)
2351   \@@_msg_new:nn { tag*~without~amsmath }
2352   {
2353     We~can't~use~\token_to_str:N\tag*~because~you~haven't~loaded~amsmath~
2354     (or~mathtools). \\
2355     If~you~go~on,~the~command~\token_to_str:N\tag\
2356     will~be~used~instead.
2357   }
2358   \@@_msg_new:nn { Multiple~tags }
2359   {
2360     You~can't~use~twice~the~command~\token_to_str:N\tag\
2361     in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2362     If~you~go~on,~the~tag~'#1'~will~be~used.
2363   }
2364   \@@_msg_new:nn { Multiple~labels }
2365   {
2366     Normally,~we~can't~use~the~command~\token_to_str:N\label\
2367     twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2368     However,~you~can~go~on.~
2369     \bool_if:NT \c_@@_showlabels_loaded_bool
2370       { However,~only~the~last~label~will~be~shown~by~showlabels.~ }
2371     If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
2372     'allow~multiple~labels'~at~the~global~or~environment~level.
2373   }
2374   \@@_msg_new:nn { Multiple~labels~with~cleveref }
2375   {
2376     Since~you~use~cleveref,~you~can't~use~the~command~\token_to_str:N\label\
2377     twice~in~a~line~of~the~environment~\{\l_@@_type_env_str\}. \\
2378     If~you~go~on,~you~may~have~undefined~references.
2379   }
2380   (/LaTeX)
2381   \@@_msg_new:nn { Inexistent~v-node }
2382   {
2383     There~is~a~problem.~Maybe~you~have~put~a~command~\token_to_str:N\cr\
2384     instead~of~a~command~\token_to_str:N\\~at~the~end~of~
2385     the~row~\l_tmpa_int\
2386     of~your~environment~\{\l_@@_type_env_str\}. \\
2387     This~error~is~fatal.
2388   }

```

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`).

```

2389   \@@_msg_new:nn { Option~xoffset~forbidden }
2390   {
2391     You~can't~use~the~option~'xoffset'~in~the~command~
2392     \l_@@_string_Arrow_for_msg_str\ in~the~row~\int_use:N \g_@@_line_int\
2393     of~your~environment~\{\l_@@_type_env_str\}~
2394     because~you~are~using~the~option~
2395     ' \int_compare:nNnTF \l_@@_pos_arrow_int = 7

```

```

2396     { group }
2397     { groups } '~It's possible for an independent arrow or if there is~
2398     only one arrow. \\
2399     \c_@@_option_ignored_str
2400   }
2401 \@@_msg_new:nnn { Duplicate-name }
2402 {
2403   The name '~\l_keys_value_tl' is already used and you shouldn't use~
2404   the same environment name twice.~You can go on,~but,~
2405   maybe,~you will have incorrect results. \\
2406   For a list of the names already used,~type H~<return>. \\
2407   If you don't want to see this message again,~use the option~
2408   'allow-duplicate-names'.
2409 }
2410 {
2411   The names already defined in this document are:~
2412   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2413 }
2414 \@@_msg_new:nn { Invalid-specification-for-MultiArrow }
2415 {
2416   The specification of rows for~\token_to_str:N\MultiArrow\
2417   (i.e.~#1) is invalid. \\
2418   \c_@@_command_ignored_str
2419 }

```

12.14 The command \WithArrowsNewStyle

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

```

2420 <*LaTeX>
2421 \NewDocumentCommand \WithArrowsNewStyle { m m }
2422 </LaTeX>
2423 <*plain-TeX>
2424 \cs_new_protected:Npn \WithArrowsNewStyle #1 #2
2425 </plain-TeX>
2426 {
2427   \keys_if_exist:nnTF { WithArrows / Global } { #1 }
2428   { \@@_error:nn { Key-already-defined } { #1 } }
2429   {
2430     \keys_define:nn { WithArrows / Global }
2431     {
2432       #1 .code:n =
2433       { \keys_set_known:nn { WithArrows / WithArrowsOptions } { #2 } }
2434     }
2435     \seq_put_right:Nx \l_@@_options-WithArrows_seq { \tl_to_str:n { #1 } }
2436     \seq_put_right:Nx \l_@@_options-DispWithArrows_seq
2437     { \tl_to_str:n { #1 } }
2438     \seq_put_right:Nx \l_@@_options-WithArrowsOptions_seq
2439     { \tl_to_str:N { #1 } }

```

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

```

2440     \group_begin:
2441     \msg_set:nnn { witharrows } { Unknown-option-WithArrowsOptions }
2442     {
2443       The key '~\l_keys_key_str' can't be set in the~
2444       definition of a style.~You can go on for this time~
2445       but you should suppress this key.
2446     }
2447     \WithArrowsOptions { #2 }
2448   \group_end:
2449 }
2450 }
2451 \@@_msg_new:nn { Key-already-defined }

```

```

2452 {
2453   The~key~'#1'~is~already~defined. \\
2454   If~you~go~on,~your~instruction~\token_to_str:N\WithArrowsNewStyle\
2455   will~be~ignored.
2456 }

```

12.15 The options up and down

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

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

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

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

```

2457 \keys_define:nn { WithArrows / up-and-down }
2458 {
2459   radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
2460   radius .value_required:n = true ,
2461   width .code:n =
2462     \str_case:nnF { #1 }
2463     {
2464       { min } { \dim_zero:N \l_@@_arrow_width_dim }
2465       { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c_max_dim }
2466     }
2467     { \dim_set:Nn \l_@@_arrow_width_dim { #1 } } ,
2468   width .value_required:n = true ,
2469   unknown .code:n = \@@_error:n { Option-unknown~for~up-and-down }
2470 }
2471 \@@_msg_new:nn { Option-unknown~for~up-and-down }
2472 {
2473   The~option~'\l_keys_key_str'~is~unknown.~\c_@@_option_ignored_str
2474 }

```

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

```

2475 ⟨*LaTeX⟩
2476 \tl_const:Nn \c_@@_tikz_code_up_tl
2477 {

```

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

```

2478   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2479   {
2480     \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2481       let \p1 = ( #1 ) , \p2 = ( #2 )
2482       in (\p1) -- node
2483       {
2484         \dim_set:Nn \l_tmpa_dim { \x2 - \x1 }
2485         \begin { varwidth } \l_tmpa_dim

```

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

```

2486         \narrowragged
2487         #3
2488     \end { varwidth }
2489 }
2490 (\x2,\y1) -- (\p2) ;
2491 }

```

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`.

```

2492 {
2493     \path
2494     let \p1 = ( #1 ) , \p2 = ( #2 )
2495     in node
2496     {

```

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

```

2497         \dim_set:Nn \l_tmpa_dim
2498         { \x2 - \x1 - \l_@@_up_and_down_radius_dim }
2499     \dim_compare:nNnF \l_@@_arrow_width_dim = \c_zero_dim
2500     {
2501         \dim_set:Nn \l_tmpa_dim
2502         { \dim_min:nn \l_tmpa_dim \l_@@_arrow_width_dim }
2503     }

```

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.

```

2504         \hbox_gset:Nn \g_tmpa_box
2505         {
2506             \begin { varwidth } \l_tmpa_dim
2507             \narrowragged
2508             #3
2509             \end { varwidth }
2510         }

```

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

```

2511         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2512         { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
2513         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2514     \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2515 } ;
2516 \draw
2517 let \p1 = ( #1 ) , \p2 = ( #2 )
2518 in (\x2-\g_tmpa_dim,\y1)
2519 -- node { \box_use:N \g_tmpa_box }
2520 (\x2-\l_@@_up_and_down_radius_dim,\y1)
2521 [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2522 -| (\p2) ;
2523 }
2524 }
2525 </LaTeX>
2526 <*plain-TeX>
2527 \tl_const:Nn \c_@@_tikz_code_up_tl
2528 {
2529     \dim_case:nF \l_@@_arrow_width_dim
2530     {
2531         \c_max_dim
2532         {
2533             \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2534             let \p1 = ( #1 ) , \p2 = ( #2 )
2535             in (\p1) -- node { #3 } (\x2,\y1) -- (\p2) ;
2536         }
2537     } \c_zero_dim

```

```

2538 {
2539   \path node
2540   {
2541     \hbox_gset:Nn \g_tmpa_box { #3 }
2542     \dim_gset:Nn \g_tmpa_dim
2543     { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2544   } ;
2545   \draw
2546     let \p1 = ( #1 ) , \p2 = ( #2 )
2547     in (\x2-\g_tmpa_dim,\y1)
2548     -- node { \box_use:N \g_tmpa_box }
2549     (\x2-\l_@@_up_and_down_radius_dim,\y1)
2550     [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2551     -| (\p2) ;
2552   }
2553 }
2554 {
2555   \draw
2556     let \p1 = ( #1 ) , \p2 = ( #2 )
2557     in (\x2 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y1)
2558     -- node { #3 } (\x2-\l_@@_up_and_down_radius_dim,\y1)
2559     [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2560     -| (\p2) ;
2561 }
2562 }
2563 \</plain-TeX>

```

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

```

2564 \*LaTeX>
2565 \tl_const:Nn \c_@@_tikz_code_down_tl
2566 {
2567   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
2568   {
2569     \draw [ rounded~corners = \l_@@_up_and_down_radius_dim ]
2570     let \p1 = ( #1 ) , \p2 = ( #2 )
2571     in (\p1) -- (\x1,\y2) -- node
2572     {
2573       \dim_set:Nn \l_tmpa_dim { \x1 - \x2 }
2574       \begin { varwidth } \l_tmpa_dim
2575         \narrowragged
2576         #3
2577       \end { varwidth }
2578     }
2579     (\p2) ;
2580   }
2581   {
2582     \path
2583     let \p1 = ( #1 ) , \p2 = ( #2 )
2584     in node
2585     {
2586       \hbox_gset:Nn \g_tmpa_box
2587       {
2588         \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.).

```

2589       { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
2590       \begin { varwidth } \l_tmpa_dim
2591         \narrowragged
2592         #3
2593       \end { varwidth }
2594     }

```

```

2595         \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
2596         { \dim_gset_eq:Nn \g_tmpa_dim \l_@@_arrow_width_dim }
2597         { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
2598         \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
2599     } ;
2600
2601     \draw
2602     let \p1 = ( #1 ) , \p2 = ( #2 )
2603     in (\p1)
2604     { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2605     -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2606     -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2607     -- ++ (-2mm,0) ;
2608 }
2609 }
2610 \</LaTeX>
2611 %
2612 \<*plain-TeX>
2613 \tl_const:Nn \c_@@_tikz_code_down_tl
2614 {
2615     \dim_case:nnF \l_@@_arrow_width_dim
2616     {
2617         \c_max_dim
2618         {
2619             \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
2620             let \p1 = ( #1 ) , \p2 = ( #2 )
2621             in (\p1) -- (\x1,\y2) -- node { #3 } (\p2) ;
2622         }
2623     } \c_zero_dim
2624     {
2625         \path node
2626         {
2627             \hbox_gset:Nn \g_tmpa_box { #3 }
2628             \dim_gset:Nn \g_tmpa_dim
2629             { \box_wd:N \g_tmpa_box + \l_@@_up_and_down_radius_dim }
2630         } ;
2631         \draw
2632         let \p1 = ( #1 ) , \p2 = ( #2 )
2633         in (\p1)
2634         { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2635         -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2636         -- node { \box_use:N \g_tmpa_box } (\x1-\g_tmpa_dim,\y2)
2637         -- ++ (-2mm,0) ;
2638     }
2639 }
2640 {
2641     \draw
2642     let \p1 = ( #1 ) , \p2 = ( #2 )
2643     in (\p1)
2644     { [ rounded-corners = \l_@@_up_and_down_radius_dim ] -- (\x1,\y2) }
2645     -- (\x1-\l_@@_up_and_down_radius_dim,\y2)
2646     -- node { #3 }
2647     (\x1 - \l_@@_arrow_width_dim - \l_@@_up_and_down_radius_dim,\y2)
2648     -- ++ (-2mm,0) ;
2649 }
2650 }
2651 \</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.

```

2652 \keys_define:nn { WithArrows / Arrow / FirstPass }
2653 {

```

```

2654   up   .code:n = \@@_set_independent_bis: ,
2655   down .code:n = \@@_set_independent_bis: ,
2656   up   .default:n = NoValue ,
2657   down .default:n = NoValue
2658 }

```

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

```

2659 \keys_define:nn { WithArrows / Arrow / SecondPass }
2660 {
2661   up .code:n =
2662     \str_if_empty:NT \l_@@_previous_key_str
2663     {
2664       \str_set:Nn \l_@@_previous_key_str { up }
2665       \cs_if_exist:cTF { tikz@library@calc@loaded }
2666       {
2667         \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2668         \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.

```

2669       \bool_set_false:N \l_@@_wrap_lines_bool

```

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

```

2670       \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
2671     }
2672     { \@@_error:n { calc-not-loaded } }
2673   } ,
2674   down .code:n =
2675     \str_if_empty:NT \l_@@_previous_key_str
2676     {
2677       \str_set:Nn \l_@@_previous_key_str { down }
2678       \cs_if_exist:cTF { tikz@library@calc@loaded }
2679       {
2680         \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
2681         \int_set:Nn \l_@@_pos_arrow_int 1
2682         \bool_set_false:N \l_@@_wrap_lines_bool
2683         \tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
2684       }
2685       { \@@_error:n { calc-not-loaded } }
2686     }
2687   }
2688   \seq_put_right:Nn \l_@@_options_Arrow_seq { down }
2689   \seq_put_right:Nn \l_@@_options_Arrow_seq { up }
2690   \@@_msg_new:nn { calc-not-loaded }
2691   {
2692     You~can't~use~the~option~'\l_keys_key_str'~because~you~don't~have~loaded~the~
2693     Tikz~library~'calc'.You~should~add~'\token_to_str:N\usetikzlibrary{calc}'~
2694     ~in~the~preamble~of~your~document. \\\
2695     \c_@@_option_ignored_str
2696   }
2697   \<plain-TeX>
2698   \catcode ` \@ = 12
2699   \ExplSyntaxOff
2700   \</plain-TeX>

```

13 History

Changes between versions 1.0 and 1.1

Option for the command `\\` and option `interline`

Compatibility with `\usetikzlibrary{babel}`
Possibility of nested environments `{WithArrows}`

Changes between versions 1.1 and 1.2

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

Changes between versions 1.2 and 1.3

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

Changes between versions 1.3 and 1.4

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

Changes between versions 1.4 and 1.5

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

Changes between versions 1.5 and 1.6

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

Changes between 1.6 and 1.7

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

Changes between 1.7 and 1.8

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

Changes between 1.8 and 1.9

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

Changes between 1.9 and 1.10

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

The option `wrap-lines` is now fully compatible with the class option `leqno`.
Correction of a bug: `\nointerlineskip` and `\makebox[.6\linewidth]{}` should be inserted in `{DispWithArrows}` only in vertical mode.

Changes between 1.10 and 1.11

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

Changes between 1.11 and 1.12

New command `\tagnextline`.

New option `tagged-lines`.

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

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

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

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

Changes between 1.12 and 1.13

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

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

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

Changes between 1.13 and 1.14

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

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

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

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

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

Changes between 1.14 and 1.15

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

Tikz externalization is now deactivated in the environments of the extension `witharrows`.⁴³

Changes between 1.15 and 1.16

Option `no-arrows`

The behaviour of `{DispWithArrows}` after an `\item` of a LaTeX list has been changed : no vertical is added. The previous behaviour can be restored with the option `standard-behaviour-with-items`.

A given name can no longer be used for two distinct environments. However, it's possible to deactivate this control with the option `allow-duplicate-names`.

Changes between 1.16 and 1.17

Option `format`.

Changes between 1.17 and 1.18

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

Option `subequations`.

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

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

⁴³Before 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 1.18 and 2.0

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

Changes between 2.0 and 2.1

Option `max-length-of-arrow`.

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

Changes between 2.1 and 2.2

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

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

Changes between 2.2 and 2.3

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

Changes between 2.3 and 2.4

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

Changes between 2.4 and 2.5

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

`witharrows` now requires and loads `varwidth`

Index

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

Symbols	
<code>\</code> ,	2019
<code>\.</code>	2019
@@ commands:	
<code>\@@_Arrow</code>	718, 721, 754, 829
<code>\@@_Arrow_code_after</code>	984, 1926, 1929
<code>\@@_Arrow_code_after_i</code> ...	1932, 1933, 1935
<code>\@@_Arrow_code_after_ii</code> ..	1938, 1939, 1941
<code>\@@_Arrow_first_columns:</code>	753, 801
<code>\@@_Arrow_i</code>	724, 725, 727
<code>\@@_Arrow_ii</code>	730, 731, 733
<code>\@@_MultiArrow:nn</code>	983, 2015
<code>\@@_MultiArrow_i:n</code>	2044, 2082
<code>\@@_MultiArrow_i:nn</code>	2021, 2024
<code>\g_@@_alignment_dim</code>	
.....	1293, 1294, 1298, 1299, 1309
<code>\c_@@_amsmath_loaded_bool</code>	
.....	234, 420, 441, 1195, 1415
<code>\c_@@_amsthm_loaded_bool</code>	834
<code>\@@_analyze_end:Nn</code>	1020, 1135
<code>\g_@@_arrow_int</code>	262, 692, 736,
	748, 750, 767, 768, 971, 973, 995, 1483, 1513
<code>\l_@@_arrow_int</code>	
	784, 1482, 1483, 1486, 1490, 1494, 1497,
	1505, 1518, 1536, 1543, 1547, 1549, 1558,
	1563, 1567, 1587, 1588, 1591, 1595, 1599,
	1604, 1607, 1616, 1643, 1647, 1651, 1746, 1884
<code>\g_@@_arrow_int_seq</code>	261, 767, 994
<code>\l_@@_arrow_width_dim</code>	302, 303,
	2464, 2465, 2467, 2478, 2499, 2502, 2511,
	2512, 2529, 2557, 2567, 2595, 2596, 2615, 2647
<code>\c_@@_autonum_loaded_bool</code>	1439
<code>\c_@@_cleveref_loaded_bool</code>	1047, 1425
<code>\l_@@_code_after_tl</code>	494, 795, 986
<code>\l_@@_code_before_tl</code>	491, 794, 809
<code>\g_@@_col_int</code>	266, 771, 772, 803,
	825, 827, 840, 841, 849, 872, 999, 1006, 1388
<code>\g_@@_col_int_seq</code>	265, 771, 998
<code>\c_@@_command_ignored_str</code>	2108, 2335, 2418
<code>\l_@@_command_name_str</code>	
.....	278, 279, 335, 801, 829, 984
<code>\@@_construct_halign:</code> ..	819, 826, 943, 1269
<code>\@@_construct_nodes:</code>	850, 881
<code>\@@_convert_to_str_seq:N</code>	153, 165, 544
<code>\@@_cr:</code>	780, 1003
<code>\@@_cr_i:</code>	1010, 1012
<code>\@@_cr_ii:</code>	1013, 1015, 1027
<code>\@@_cr_iii:n</code>	1019, 1022, 1024
<code>\@@_def_function_tmpa:n</code>	1725, 1762
<code>\l_@@_delim_wd_dim</code>	876, 1213, 1214
<code>\l_@@_displaystyle_bool</code>	342, 845, 947, 1221
<code>\@@_draw_arrow:</code>	1605, 1612, 1887
<code>\@@_draw_arrow:nnn</code>	399, 1722, 1758, 1765, 2009
<code>\@@_draw_arrows:nn</code> ...	398, 1513, 1534, 1580
<code>\@@_draw_o_arrows_of_the_group:</code>	1609, 1809
<code>\l_@@_end_adjust_dim</code> ...	390, 702, 705, 1718

`\@@_error:n` 39, 88, 98, 316, 355,
366, 444, 474, 487, 499, 503, 549, 556, 579,
597, 607, 614, 630, 635, 641, 661, 698, 754,
938, 945, 1007, 1123, 1139, 1416, 1426,
1427, 1502, 1889, 1920, 2033, 2469, 2672, 2685
`\@@_error:nn` 42, 43, 1386,
1389, 1403, 1962, 1965, 1968, 2022, 2029, 2428
`\@@_eval_if_allowed:n` 309, 319
`\c_@@_extensible_delimiters_clist` ...
..... 411, 412, 423, 471
`\@@_fatal:n` 41, 50, 1234, 1275, 1340
`\l_@@_final_int`
783, 1492, 1499, 1545, 1559, 1561, 1566,
1574, 1597, 1601, 1636, 1641, 1677, 1846, 1869
`\l_@@_final_r_bool` 294,
1622, 1625, 1630, 1641, 1950, 1956, 1959, 2002
`\l_@@_final_tl` 296, 1640, 1668
`\l_@@_first_arrow_int` 1583, 1584, 1587, 1854
`\l_@@_first_arrow_of_group_int`
..... 1476, 1511, 1513, 1532, 1535, 1543
`\l_@@_first_arrows_seq`
..... 1479, 1546, 1547, 1558, 1646
`\l_@@_first_line_of_group_int`
..... 1477, 1544, 1557
`\@@_fix_pos_arrow:n`
..... 669, 683, 684, 685, 686, 687
`\@@_fix_pos_option:n` 318, 371, 373,
375, 377, 379, 1905, 1907, 1909, 1911, 1913
`\l_@@_fleqn_bool` 429, 868, 1265, 1306
`\g_@@_footnote_bool`
..... 33, 47, 82, 101, 807, 961, 1366
`\g_@@_footnotehyper_bool` 32, 48, 92
`\l_@@_format_seq` 804, 805, 821
`\l_@@_format_str` 298, 502, 788, 802, 805
`\l_@@_halign_box`
..... 1262, 1263, 1302, 1314, 1315, 1330
`\c_@@_hyperref_loaded_bool` 1042
`\@@_if_in_last_col_of_disp:Nn`
..... 1383, 1395, 1397, 1400, 1421, 1449
`\l_@@_in_DispWithArrows_bool` 253,
799, 831, 854, 931, 1031, 1171, 1760, 2065
`\l_@@_in-WithArrows_bool`
..... 252, 797, 860, 930, 1385
`\l_@@_in_code_after_bool` ... 254, 985, 2299
`\l_@@_in_first_columns_bool` 291
`\l_@@_in_label_or_minipage_bool`
..... 1149, 1180, 1182, 1241, 1255, 1354
`\@@_info:n` 95
`\l_@@_initial_int`
..... 782, 1488, 1525, 1544, 1557,
1574, 1593, 1636, 1639, 1677, 1842, 1868, 2264
`\l_@@_initial_r_bool` 293,
1621, 1626, 1629, 1639, 1949, 1955, 1958, 1994
`\l_@@_initial_tl` 295, 1638, 1665
`\l_@@_input_line_str`
..... 766, 1498, 1849, 2263, 2271
`\l_@@_interline_skip` 386, 793, 1130
`\l_@@_jump_int` 613, 740, 786, 787
`\@@_keys_set:` 1578, 1620
`\@@_label:n` 815, 1419
`\l_@@_labels_seq` 790, 1039, 1061, 1423, 1429
`\l_@@_last_arrow_int` . 1585, 1586, 1588, 1854
`\l_@@_last_arrows_seq`
... 1480, 1548, 1549, 1562, 1563, 1567, 1650
`\g_@@_last_env_int` ... 257, 993, 1894, 1897
`\l_@@_last_line_of_group_int`
..... 1478, 1525, 1545, 1559, 1561, 1566
`\l_@@_left_brace_box` .. 876, 1215, 1216, 1311
`\l_@@_left_brace_tl`
... 250, 251, 553, 873, 1197, 1202, 1222, 1301
`\c_@@_leqno_bool` 103, 104, 1081, 1093
`\g_@@_line_int` 264,
274, 739, 740, 769, 770, 871, 888, 892, 902,
906, 913, 982, 997, 1086, 1098, 1115, 1336,
1499, 1601, 2134, 2141, 2175, 2240, 2247, 2392
`\g_@@_line_int_seq` 263, 769, 996
`\l_@@_linewidth_dim`
... 866, 1253, 1256, 1257, 1260, 1304, 1329
`\l_@@_mathindent_dim` 431, 869, 1307
`\c_@@_mathtools_loaded_bool`
..... 1186, 1352, 1405, 1430
`\l_@@_max_length_of_arrow_dim`
..... 322, 1675, 1687, 1698
`\@@_msg_new:nn` 35, 52, 59,
64, 73, 2111, 2119, 2131, 2138, 2146, 2154,
2160, 2166, 2226, 2235, 2243, 2252, 2260,
2269, 2275, 2281, 2287, 2295, 2306, 2313,
2320, 2326, 2331, 2337, 2344, 2351, 2358,
2364, 2374, 2381, 2389, 2414, 2451, 2471, 2690
`\@@_msg_new:nnn`
... 36, 2171, 2184, 2194, 2204, 2215, 2401
`\@@_msg_redirect_name:nn`
..... 37, 332, 447, 454, 573, 1142
`\l_@@_name_str` .. 489, 763, 891, 892, 905, 906
`\g_@@_names_seq` 282, 486, 488, 2412
`\l_@@_nb_cols_int` 297, 802, 803,
827, 849, 1008, 1388, 2229, 2238, 2309, 2341
`\l_@@_new_box` 1295, 1296, 1298, 1299
`\l_@@_new_group_bool`
..... 292, 1481, 1538, 1540, 1542
`\@@_nonumber:` 812, 1396
`\@@_notag:` 811, 1394
`\l_@@_o_arrows_seq` ... 306, 1604, 1811, 1837
`\@@_old_label` 814, 1061, 1219
`\c_@@_option_ignored_str`
... 2106, 2129, 2136, 2158, 2164, 2177,
2187, 2197, 2208, 2219, 2250, 2258, 2293,
2304, 2318, 2324, 2342, 2348, 2399, 2473, 2695
`\l_@@_options_Arrow_code_after_seq` ..
..... 1919, 1922, 1923, 2224
`\l_@@_options_Arrow_seq` 352,
361, 362, 363, 654, 663, 664, 2182, 2688, 2689
`\l_@@_options_DispWithArrows_seq`
..... 237, 555, 558, 559, 2202, 2436
`\l_@@_options-WithArrowsOptions_seq` ..
..... 236, 578, 581, 582, 2213, 2438
`\l_@@_options-WithArrows_seq`
..... 531, 532, 544, 548, 655, 2192, 2435
`\l_@@_pos_arrow_int` 259,
260, 319, 353, 364, 628, 634, 674, 694,
975, 976, 1466, 1469, 1501, 1509, 1527,
1550, 1573, 1623, 1633, 1654, 1681, 1692,
1705, 1714, 1944, 1951, 1970, 2395, 2668, 2681
`\l_@@_pos_env_int` ... 258, 404, 406, 408, 940
`\l_@@_pos_of_arrow_int` 785

<code>\g_@@_position_in_the_tree_seq</code>	255, 256, 775, 776, 988, 989, 990, 992
<code>\@@_post_halign:</code>	959, 967, 1350
<code>\@@_pre_halign:n</code>	755, 936, 1198
<code>\l_@@_prefix_str</code>	209, 748, 750, 778, 779, 888, 902, 913, 1086, 1098, 1115, 1339, 1343, 1486, 1490, 1494, 1497, 1591, 1595, 1599, 1616, 1643, 1665, 1668, 1746, 1769, 1803, 1814, 1817, 1826, 1829, 1840, 1844, 1848, 1857, 1860, 1863, 1875, 1964, 1967, 1974, 1978, 1993, 2001, 2028, 2061
<code>\l_@@_previous_key_str</code>	311, 313, 349, 351, 358, 360, 602, 604, 632, 638, 671, 673, 688, 714, 737, 796, 1618, 1945, 2291, 2298, 2662, 2664, 2675, 2677
<code>\@@_qedhere:</code>	1452, 1453
<code>\l_@@_qedhere_bool</code>	289, 1064, 1073, 1074, 1104, 1108, 1109, 1230, 1452
<code>\@@_qedhere_i:</code>	1075, 1110, 1454
<code>\l_@@_replace_left_brace_by_tl</code>	473, 1209, 1320
<code>\@@_restore:N</code>	183, 1072, 1073, 1108
<code>\g_@@_right_x_dim</code>	969, 1332, 1333, 1345, 1346, 1771, 2068
<code>\@@_save:N</code>	167, 1063, 1064, 1104
<code>\l_@@_sbwi_bool</code>	284, 478, 1177
<code>\@@_scan_arrows:</code>	978, 1463
<code>\@@_scan_arrows_i:</code>	1468, 1471, 1474
<code>\@@_set_independent:</code>	594, 616, 617, 618, 619, 620
<code>\@@_set_independent_bis:</code>	598, 600, 2654, 2655
<code>\@@_set_qedhere:</code>	834, 1453
<code>\@@_set_seq_of_str_from_clist:Nn</code>	162, 532, 559, 582, 664, 1923
<code>\l_@@_show_node_names_bool</code>	346, 910
<code>\c_@@_showlabels_loaded_bool</code>	2369
<code>\@@_sort_seq:N</code>	138, 548, 555, 578, 654, 1919
<code>\l_@@_start_adjust_dim</code>	387, 701, 704, 1709
<code>\g_@@_static_col_int</code>	268, 773, 774, 841, 1001, 1006, 1008
<code>\g_@@_static_col_int_seq</code>	267, 773, 1000
<code>\l_@@_status_arrow_str</code>	605, 629, 637, 696, 742, 764, 1495, 1529, 1554, 1571, 1600, 1603, 1864, 1870
<code>\@@_strcmp:nn</code>	131, 135, 144
<code>\l_@@_string_Arrow_for_msg_str</code>	280, 281, 336, 2174, 2246, 2255, 2292, 2301, 2308, 2316, 2392
<code>\l_@@_subequations_bool</code>	300, 442, 1200, 1365
<code>\@@_tag</code>	813, 1398
<code>\l_@@_tag_next_line_bool</code>	288, 791, 1065, 1068, 1450
<code>\l_@@_tag_star_bool</code>	287, 1063, 1072, 1077, 1231, 1414
<code>\l_@@_tag_tl</code>	1036, 1038, 1229, 1402, 1413
<code>\@@_tagnextline:</code>	816, 1447
<code>\l_@@_tags_clist</code>	270, 271, 274, 275, 437, 438, 457, 458, 460, 461, 1034, 1284, 1285, 1395, 1397, 1404, 1410, 1435, 1436, 1442, 1443
<code>\@@_test_if_to_tag:</code>	272, 833
<code>\c_@@_tikz_code_down_tl</code>	2565, 2613, 2683
<code>\l_@@_tikz_code_tl</code>	339, 679, 1761, 1762, 1914, 2670, 2683
<code>\c_@@_tikz_code_up_tl</code>	2476, 2527, 2670
<code>\c_@@_tikz_code_wrap_lines_tl</code>	1761, 1766
<code>\@@_tmpa:nnn</code>	1727, 1763
<code>\@@_treat_an_arrow_in_scan:</code>	1504, 1515
<code>\l_@@_type_col_str</code>	821, 838, 839, 852, 853
<code>\l_@@_type_env_str</code>	758, 759, 933, 934, 1137, 1173, 1174, 2135, 2142, 2176, 2186, 2196, 2228, 2237, 2248, 2257, 2265, 2277, 2283, 2302, 2310, 2317, 2322, 2341, 2346, 2361, 2367, 2377, 2386, 2393
<code>\c_@@_typedref_loaded_bool</code>	1194
<code>\l_@@_up_and_down_radius_dim</code>	304, 305, 2459, 2480, 2498, 2514, 2520, 2521, 2533, 2543, 2549, 2550, 2557, 2558, 2559, 2569, 2589, 2598, 2604, 2605, 2619, 2629, 2634, 2635, 2644, 2645, 2647
<code>\@@_update_x:nn</code>	1574, 1636, 1796
<code>\@@_warning:n</code>	40
<code>\l_@@_wrap_lines_bool</code>	465, 1760, 2065, 2669, 2682
<code>\l_@@_x_dim</code>	765, 1551, 1635, 1683, 1694, 1707, 1716, 1798, 1807
<code>\g_@@_x_final_dim</code>	1660, 1669, 1693, 1715
<code>\g_@@_x_initial_dim</code>	1659, 1666, 1682, 1706
<code>\l_@@_xoffset_dim</code>	381, 699, 1620, 1744, 1885, 1886, 1916, 1948, 2055, 2058, 2059, 2097
<code>\l_@@_xoffset_for_o_arrows_dim</code>	307, 308, 575, 1886
<code>\g_@@_y_final_dim</code>	1662, 1670, 1674, 1686, 1697, 1718, 1719
<code>\g_@@_y_initial_dim</code>	1661, 1667, 1674, 1686, 1697, 1709, 1710
<code>\l_@@_ygap_dim</code>	200, 325
<code>\l_@@_ystart_dim</code>	197, 328
<code>\</code>	70, 79, 780, 956, 1069, 1287, 2114, 2128, 2133, 2135, 2142, 2149, 2157, 2163, 2168, 2176, 2177, 2186, 2187, 2196, 2197, 2207, 2208, 2218, 2219, 2232, 2240, 2249, 2257, 2266, 2272, 2278, 2284, 2292, 2303, 2310, 2317, 2323, 2328, 2334, 2341, 2347, 2354, 2361, 2367, 2377, 2384, 2386, 2398, 2405, 2406, 2417, 2453, 2694
<code>\{</code>	414, 2123, 2135, 2142, 2148, 2149, 2176, 2186, 2196, 2228, 2237, 2248, 2257, 2265, 2277, 2278, 2283, 2284, 2302, 2310, 2317, 2322, 2341, 2346, 2347, 2361, 2367, 2377, 2386, 2393
<code>\}</code>	2135, 2142, 2148, 2149, 2176, 2186, 2196, 2228, 2237, 2248, 2257, 2265, 2277, 2278, 2283, 2284, 2302, 2310, 2317, 2322, 2341, 2346, 2347, 2361, 2367, 2377, 2386, 2393
<code>_</code>	61, 2126, 2127, 2134, 2141, 2174, 2175, 2218, 2229, 2238, 2239, 2246, 2247, 2255, 2263, 2264, 2271, 2301, 2308, 2316, 2333, 2355, 2360, 2366, 2376, 2383, 2385, 2392, 2416, 2454
A	
<code>\A</code>	501, 2019
<code>\arabic</code>	1057
<code>\Arrow</code>	281, 2218
<code>\AtBeginDocument</code>	110, 232, 417

B	
\begin	807, 1200, 1730, 1785, 2046, 2085, 2485, 2506, 2574, 2590
\belowdisplayskip	1357
\bgroup	863, 867, 941, 1263
bool commands:	
\bool_gset_true:N	101, 1873
\bool_if:NTF	82, 92, 234, 441, 797, 799, 807, 831, 834, 845, 854, 860, 868, 910, 947, 961, 1031, 1042, 1047, 1065, 1074, 1077, 1081, 1093, 1109, 1177, 1194, 1195, 1200, 1221, 1241, 1255, 1265, 1306, 1354, 1365, 1366, 1385, 1425, 1639, 1641, 1708, 1717, 1882, 1994, 2002, 2299, 2369
\bool_if:nTF	690, 1186, 1276, 1352, 1405, 1415, 1430, 1439, 1507, 1540, 1865
\bool_if_p:N	1415
\bool_lazy_and:nnTF ..	1517, 1672, 1760, 2065
\bool_lazy_and_p:nn	1522
\bool_lazy_or:nnTF	419
\bool_lazy_or_p:nn	1520
\bool_new:N	32, 33, 103, 118, 252, 253, 254, 284, 287, 288, 289, 291, 292, 293, 294, 300, 1149
\bool_not_p:n	1527
\bool_set:Nn	1414
\bool_set_false:N	125, 791, 931, 1068, 1230, 1231, 1542, 1621, 1622, 1649, 1653, 1851, 1949, 1950, 2669, 2682
\bool_set_true:N	104, 121, 442, 930, 985, 1171, 1180, 1182, 1450, 1452, 1481, 1538, 1625, 1626, 1629, 1630, 1648, 1652, 1656, 1657, 1955, 1956, 1958, 1959
\c_false_bool	1276
\g_tmpa_bool	1851, 1873, 1882
\l_tmpa_bool	1648, 1649, 1656, 1708
\l_tmpb_bool	1652, 1653, 1657, 1717
box commands:	
\box_clear_new:N	1215, 1262, 1295
\box_dp:N	1315
\box_ht:N	1314
\box_set_to_last:N	1290
\box_use:N	1292, 2519, 2548, 2606, 2636
\box_use_drop:N	1302, 1311, 1330
\box_wd:N	876, 1214, 1294, 1298, 1299, 2513, 2543, 2597, 2629
\g_tmpa_box	2504, 2513, 2519, 2541, 2543, 2548, 2586, 2597, 2606, 2627, 2629, 2636
\l_tmpa_box	1204, 1214, 1290, 1292, 1294, 1296
C	
\catcode	29, 2698
char commands:	
\char_generate:nn	170, 186
clist commands:	
\clist_clear:N	437, 1395, 1397, 1410, 1436, 1443
\clist_count:N	2032
\clist_gput_right:Nn	2030
\clist_if_in:NnTF ..	274, 458, 470, 1034, 1284
\clist_map_inline:nn	112
\clist_new:N	270, 411
\clist_pop:NN	2041, 2043
\clist_put_left:Nn	461
\clist_put_right:Nn	423
\clist_remove_all:Nn	460
\clist_reverse:N	2042
\clist_set:Nn	271, 275, 412, 438, 457, 1285, 1404, 1435, 1442
\clist_sort:Nn	2035
\g_tmpa_clist	2030, 2032, 2035, 2041, 2042, 2043, 2044
\cr	950, 1125, 1277, 2383
cs commands:	
\cs_generate_variant:Nn	43, 108, 109, 1577, 1765
\cs_gset:Npx	1037
\cs_if_exist:NTF	22, 761, 1053, 1434, 1441, 2665, 2678
\cs_if_free:NTF	1338, 1964, 1967, 2028
\cs_new:Npn	1897
\cs_new_protected:Npn	35, 36, 37, 39, 40, 41, 42, 131, 135, 138, 153, 162, 167, 183, 241, 272, 309, 318, 594, 600, 669, 721, 727, 733, 753, 755, 819, 881, 920, 927, 953, 967, 1003, 1012, 1015, 1024, 1027, 1135, 1155, 1162, 1168, 1280, 1383, 1394, 1396, 1419, 1447, 1452, 1453, 1454, 1463, 1474, 1515, 1578, 1580, 1612, 1725, 1758, 1796, 1809, 1929, 1935, 1941, 2015, 2024, 2082, 2424
\cs_set:Npn	969, 982, 1727
\cs_set:Npx	1041
\cs_set_eq:NN	244, 398, 399, 780, 801, 811, 812, 813, 814, 815, 816, 829, 983, 984, 1076, 1078, 1219, 1453, 1457, 1458
\cs_set_protected:Npn	711
\cs_to_str:N	171, 187
D	
\d	2019
\DeclareOption	104, 105
dim commands:	
\dim_add:Nn	1744, 1886
\dim_case:nnTF	2529, 2615
\dim_compare:nNnTF ..	1122, 1297, 1345, 1741, 1776, 1779, 2069, 2478, 2499, 2511, 2567, 2595
\dim_compare_p:nNn	1674
\dim_eval:n	1684, 1695, 1709, 1718
\dim_gadd:Nn	2514, 2598
\dim_gset:Nn	1294, 1299, 1666, 1667, 1669, 1670, 1804, 1878, 2513, 2542, 2597, 2628
\dim_gset_eq:NN	1333, 1346, 1798, 2512, 2596
\dim_gzero:N	1850
\dim_gzero_new:N	1293, 1332, 1659, 1660, 1661, 1662
\dim_max:nn	1128, 1804, 1878, 1981
\dim_min:nn	2502
\dim_new:N	302, 304, 307
\dim_set:Nn	305, 308, 699, 704, 705, 1128, 1214, 1312, 1551, 1635, 1743, 1770, 1775, 1980, 2062, 2064, 2067, 2467, 2484, 2497, 2501, 2573, 2588
\dim_set_eq:NN	303, 1207, 1238, 1256, 1257, 1260, 1318, 1742, 1777, 1807, 1885, 1976, 1977, 2070, 2465
\dim_use:N	1682, 1683, 1693, 1694, 1706, 1707, 1710, 1715, 1716, 1719, 1791, 1983, 1985, 1998, 2006, 2073

keys commands:			
\keys_define:nn	45, 320, 402, 427, 482, 507, 545, 551, 570, 609, 677, 1899, 2430, 2457, 2652, 2659		
\keys_if_exist:nnTF	2427		
\l_keys_key_str	22, 54, 61, 313, 604, 638, 655, 673, 2113, 2121, 2168, 2173, 2186, 2196, 2206, 2217, 2289, 2296, 2443, 2473, 2692		
\keys_set:nn	395, 715, 738, 798, 800, 1577, 1947, 2667, 2680		
\keys_set_known:nn	1579, 2433		
\l_keys_value_tl	596, 2403, 2667, 2680		
L			
\label	814, 815, 1219, 1421, 2366, 2376		
\langle	414, 2125		
\lbrace	414, 476, 2123		
\lbrack	414, 2124		
\lceil	414, 2126		
\left	1209, 1320		
legacy commands:			
\legacy_if:nTF	1179, 1181		
\lfloor	414, 2126		
\lggroup	414, 2124		
\linewidth	1237, 1238, 1247, 1256		
\lmoustache	414, 2125		
lua commands:			
\lua_now:n	132		
\lVert	423, 2127		
\lvert	423, 2127		
M			
math commands:			
\c_math_toggle_token	842, 848, 946, 949, 1208, 1210, 1220, 1224, 1242, 1249, 1319, 1323, 1356, 1359, 1362		
\mathsurround	781		
MH commands:			
\MH_if_boolean:nTF	1188, 1353, 1407, 1409, 1432		
\MH_set_boolean_T:n	1191		
msg commands:			
\msg_error:nn	23, 39		
\msg_error:nnn	42		
\msg_fatal:nn	41		
\msg_info:nn	85		
\msg_line_number:	745		
\msg_new:nnn	16, 35		
\msg_new:nnnn	36		
\msg_redirect_name:nnn	38		
\msg_set:nnn	2441		
\msg_warning:nn	40		
MT commands:			
\MT_showonlyrefs_false:	1190		
\MT_showonlyrefs_true:	1353		
\MultiArrow	983, 2329, 2333, 2416		
\myfiledate	13		
\myfileversion	14		
N			
\narrowragged	2486, 2507, 2575, 2591		
\NewDocumentCommand	708, 718, 1398, 1926, 2421		
\NewDocumentEnvironment	917, 1152, 1376		
\NewExpandableDocumentCommand	1893		
\noalign	1126		
\node	884, 898		
\nointerlineskip	1246, 1291		
\nonumber	812, 1397		
\normalbaselines	970		
\notag	811, 1069, 1395		
\nulldelimiterspace	1207, 1318		
O			
\omit	2141		
\openup	243, 1067		
P			
\p	2481, 2482, 2490, 2494, 2517, 2522, 2534, 2535, 2546, 2551, 2556, 2560, 2570, 2571, 2579, 2583, 2602, 2603, 2620, 2621, 2632, 2633, 2642, 2643		
\path	1772, 1781, 2063, 2072, 2493, 2539, 2582, 2625		
peek commands:			
\peek_meaning:NTF	723, 729, 923, 1013, 1158, 1164, 1931, 1937		
\peek_meaning_ignore_spaces:NTF	1017		
\peek_meaning_remove:NTF	1010		
\pgfcoordinate	1085, 1097, 1114		
\pgfextra	1772, 2063		
\pgfkeysvalueof	1771		
\pgfpicture	1083, 1095, 1112, 1334, 1663, 1799, 1852, 1972, 1989		
\pgfpointanchor	1342, 1665, 1668, 1769, 1803, 1974, 1978, 1991, 1999, 2061		
\pgfpointorigin	1087, 1099, 1116		
\pgfrememberpicturerepositiononpagetrue	1084, 1096, 1113, 1335, 1664, 1800, 1853, 1973, 1990		
\pgfresetboundingbox	1749		
prg commands:			
\prg_do_nothing:	244, 642, 644, 646, 648, 650, 1078		
\prg_replicate:nn	1008		
\ProcessKeysOptions	58		
\ProcessOptions	106		
prop commands:			
\prop_gclear_new:N	747		
\prop_get:NnN	1485, 1489, 1493, 1496, 1590, 1594, 1598, 1615, 1642, 1813, 1816, 1825, 1828, 1839, 1843, 1847, 1856, 1859, 1862, 1874		
\prop_gput:Nnn	1745		
\prop_gset_eq:NN	749		
\prop_put:Nnn	739, 741, 742, 743, 744, 745, 746		
\l_tmpa_prop	739, 741, 742, 743, 744, 745, 746, 751		
\ProvidesExplPackage	11		
Q			
\qed	1457		
\qedhere	1453		
\qedsymbol	1457		
\quad	1090		
R			
regex commands:			
\regex_match:nnTF	501, 2018		
\relax	1459		
\RequirePackage	2, 10, 24, 25		
\right	1209, 1322		

S

scan commands:

- `\scan_stop:` 1005, 1131

seq commands:

- `\seq_clear:N` 155, 1546, 1548, 1562
- `\seq_clear_new:N` 790, 804, 1479, 1480
- `\seq_count:N` 992
- `\seq_gpop_right:NN` 988, 989, 994, 996, 998, 1000
- `\seq_gput_left:Nn` 488
- `\seq_gput_right:Nn` 256, 767, 769, 771, 773, 775, 990
- `\seq_if_empty:NTF` 1039, 1423
- `\seq_if_in:NnTF` ... 361, 486, 655, 1646, 1650
- `\seq_item:Nn` 173, 189
- `\seq_map_function:NN` 1061
- `\seq_map_inline:Nn` 156, 1837
- `\seq_new:N` 255, 261, 263, 265, 267, 282, 306, 531, 558, 581, 663, 1922
- `\seq_pop_left:NN` 172, 188
- `\seq_pop_right:NN` 777
- `\seq_pop_right:NNTF` 821
- `\seq_put_left:Nn` 158, 1547, 1549, 1558, 1563, 1567
- `\seq_put_right:Nn` 236, 237, 362, 1429, 1604, 2435, 2436, 2438, 2688, 2689
- `\seq_remove_all:Nn` 352, 363
- `\seq_set_eq:NN` 160, 776
- `\seq_set_from_clist:Nn` 164
- `\seq_set_split:Nnn` 109, 169, 185, 805
- `\seq_sort:Nn` 140, 1811
- `\seq_use:Nnnn` 175, 178, 181, 191, 779, 2182, 2192, 2202, 2213, 2224, 2412
- `\l_tmpa_seq` .. 155, 158, 160, 169, 172, 173, 175, 178, 181, 185, 188, 189, 191, 776, 777, 779

`\setbox` 1263

skip commands:

- `\skip_horizontal:N` 869, 1307, 1329
- `\skip_horizontal:n` 875
- `\skip_vertical:N` 1129, 1130, 1357
- `\skip_zero:N` 793
- `\c_zero_skip` 854, 1266, 1273

`\small` 224, 913

sort commands:

- `\sort_return_same:` ... 150, 1820, 1833, 2039
- `\sort_return_swapped:` . 149, 1823, 1832, 2038

str commands:

- `\c_backslash_str` 337, 2230, 2239
- `\str_case:nnTF` 2462
- `\str_clear:N` 660
- `\str_clear_new:N` 714, 737, 758, 763, 764, 766, 778, 796, 933, 1173, 1618, 1945
- `\str_const:Nn` 2106, 2108
- `\str_count:N` 802
- `\str_if_empty:NTF` 311, 349, 358, 602, 632, 671, 891, 905, 2662, 2675
- `\str_if_eq:nnTF` 436, 596, 838, 839, 852, 853, 1137, 1554, 1571, 1603
- `\str_if_eq_p:nn` 696, 1529, 1870
- `\str_lower_case:n` 145, 146
- `\str_new:N` 278, 280, 298

- `\str_set:Nn` .. 173, 189, 279, 281, 335, 336, 351, 360, 485, 605, 629, 637, 657, 688, 759, 779, 788, 934, 1044, 1174, 1194, 2664, 2677
- `\str_set_eq:NN` 313, 489, 604, 638, 673
- `\l_tmpa_str` 173, 174, 177, 180, 189, 190, 485, 486, 488, 489, 657, 660, 2176
- `\strut` 878

sys commands:

- `\sys_if_engine luatex:TF` 129

T

`\tabskip` 854, 1264, 1270, 1273

`\tag` 813, 1400, 2353, 2355, 2360

`\tagnextline` 816, 1449

TEX and L^AT_EX 2_ε commands:

- `\@` 29, 2698
- `\@currentlabel` 1041
- `\@currentenv` 759
- `\@eqnnum` 1091
- `\@ifclassloaded` 84, 94
- `\@ifpackageloaded` 87, 97, 120
- `\@cequation` 1036
- `\cref@constructprefix` 1049
- `\cref@currentlabel` 1050
- `\cref@equation@alias` 1053, 1054
- `\cref@result` 1049, 1057
- `\hyper@refstepcounter` 1045
- `\intertext@` 1195
- `\p@equation` 1041, 1058
- `\pgf@picmaxx` 1743
- `\pgf@picminx` 1741, 1743
- `\pgf@relevantforpicturesizetrue` 1739
- `\pgf@x` 1345, 1346, 1666, 1669, 1771, 1804, 1976, 1981, 1998, 2006, 2068
- `\pgf@y` 1667, 1670, 1977, 1985, 1998, 2006
- `\protected@edef` 1050
- `\qed@elt` 1458
- `\QED@stack` 1459
- `\setQED@elt` 1458
- `\spread@equation` 241, 244, 942, 1268
- `\sr@name` 1194
- `\tagform@` 1078
- `\This@name` 1044
- `\tikz@library@external@loaded` 761
- `\tikz@text@width` 1772, 2063

tex commands:

- `\tex_strcmp:D` 136
- `\theequation` 1038, 1076
- `\tikz` 883, 897
- `\tikzpicture` 1733, 2049, 2088
- `\tikzset` 193, 203, 212, 217, 345, 368, 681, 762, 981, 1902

tl commands:

- `\c_empty_tl` 369, 682
- `\c_novalue_tl` 251, 873, 1160, 1202, 1301
- `\tl_clear_new:N` 794, 795, 1229
- `\tl_const:Nn` ... 1766, 2476, 2527, 2565, 2613
- `\tl_gset:Nn` 1679, 1690, 1703, 1712, 1772, 1982, 1984, 1997, 2005, 2063
- `\tl_head:n` 469
- `\tl_if_empty:NTF` 1036, 1038, 1402, 1773, 2064
- `\tl_if_empty:nTF` 498
- `\tl_if_eq:NNTF` 873, 1202, 1301
- `\tl_if_eq:nnTF` 1961

10.6	Option no-arrows	26
10.7	Note for the users of AUCTeX	26
10.8	Note for developers	26
11	Examples	27
11.1	<code>\MoveEqLeft</code>	27
11.2	A command <code>\DoubleArrow</code>	27
11.3	Modifying the shape of the nodes	28
11.4	Examples with the option <code>tikz-code</code>	28
11.4.1	Example 1	28
11.4.2	Example 2	29
11.4.3	Example 3	30
11.5	Automatic numbered loop	31
12	Implementation	32
12.1	Declaration of the package and extensions loaded	32
12.2	The packages <code>footnote</code> and <code>footnotehyper</code>	32
12.3	The class option <code>leqno</code>	34
12.4	Some technical definitions	34
12.5	Variables	38
12.6	The definition of the options	40
12.7	The command <code>\Arrow</code>	49
12.8	The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code>	50
12.8.1	Code before the <code>\halign</code>	50
12.8.2	The construction of the preamble of the <code>\halign</code>	53
12.8.3	The environment <code>{WithArrows}</code>	56
12.8.4	After the construction of the <code>\halign</code>	57
12.8.5	The command of end of row	58
12.8.6	The environment <code>{DispWithArrows}</code>	62
12.9	The commands <code>\tag</code> , <code>\notag</code> , <code>\label</code> , <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code>	67
12.10	We draw the arrows	69
12.10.1	The command <code>update_x</code>	78
12.10.2	We draw the arrows of type <code>o</code>	78
12.11	The command <code>\Arrow</code> in code-after	81
12.12	The command <code>\MultiArrow</code> in code-after	83
12.13	The error messages of the package	85
12.14	The command <code>\WithArrowsNewStyle</code>	90
12.15	The options <code>up</code> and <code>down</code>	91
13	History	95
	Index	98