The package witharrows*

F. Pantigny
fpantigny@wanadoo.fr

July 18, 2018

Abstract

The LaTeX package witharrows provides an environment `{WithArrows}` which is similar to environment `{aligned}` of amsmath (and mathtools) but gives 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.

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two compilations may be necessary. This package requires the packages expl3, xparse and tikz. The following Tikz libraries arrows.meta and bending are also required.

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

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

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

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. 13.

1 Options for the shape of the arrows

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

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

*This document corresponds to the version 1.7 of witharrows, at the date of 2018/07/18.

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

$$\begin{aligned} A &= \bigl((a+b)+1\bigr)^2 \\ &= (a+b)^2 + 2(a+b) + 1 \\ &= a^2 + 2ab + b^2 + 2a + 2b + 1 \end{aligned}$$ *we expand*

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

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

$$\begin{aligned} A &= \bigl((a+b)+1\bigr)^2 \\ &= (a+b)^2 + 2(a+b) + 1 \\ &= a^2 + 2ab + b^2 + 2a + 2b + 1 \end{aligned}$$

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

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

$$\begin{aligned} A &= \bigl((a+b)+1\bigr)^2 \\ &= (a+b)^2 + 2(a+b) + 1 \end{aligned}$$ *with* ***xoffset=1cm***

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

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

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

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

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

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

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

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

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

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

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

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

In fact, it's possible to change more drastically the shape or the arrows with the option `TikzCode` presented p. 15.

One of the most useful options is "`text width`" to control the width of the text associated to the arrow.

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

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

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

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

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

If we put commands `\\` in the text to force newlines, 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 a option `font` to modify the font of the whole text. Nevertheless, if we use the option `tikz={font={\bfseries}}`, the default specification of `\small` and `\itshape` will be overwritten.

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

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

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

Almost all the options can be given directly to the environment `{WithArrows}` (between square brackets). In this case, they apply to all the arrows of the environment.[2]

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

$$\begin{aligned} A &= \big((a+b)+1\big)^2 \\ &= (a+b)^2 + 2(a+b) + 1 \quad \text{⤵ } \textit{First expansion.} \\ &= a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \text{⤵ } \textit{Second expansion.} \end{aligned}$$

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

Without the option `displaystyle`:

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

$$\begin{aligned} \textstyle\int_0^1 (x+1)^2 dx &= \textstyle\int_0^1 (x^2+2x+1) dx \\ &= \textstyle\int_0^1 x^2 dx + 2\int_0^1 x dx + \int_0^1 dx \quad \text{⤵ } \textit{linearity of integration} \\ &= \tfrac13 + 2\tfrac12 + 1 \\ &= \tfrac73 \end{aligned}$$

The same example with the option `displaystyle`:

$$\begin{aligned} \int_0^1 (x+1)^2 dx &= \int_0^1 (x^2+2x+1) dx \\ &= \int_0^1 x^2 dx + 2\int_0^1 x dx + \int_0^1 dx \quad \text{⤵ } \textit{linearity of integration} \\ &= \frac13 + 2\frac12 + 1 \\ &= \frac73 \end{aligned}$$

The version 1.3 of witharrows give two options for a fine tuning of the arrows:

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

[2] They also apply to the nested environments `{WithArrows}` (with the logical exceptions of `interline`, `CodeBefore` and `CodeAfter`).

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

ystart

ygap

Remark: It's also possible to use the options "`shorten <`" and "`shorten >`" of Tikz (via the option `tikz` of witharrows).

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}`.[3]

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

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

by linearity

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

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

$$\begin{aligned}f &= \bigl(x\longmapsto(x+1)^2\bigr)\\ &= \bigl(x\longmapsto x^2+2x+1\bigr)\end{aligned}$$

we work directly on fonctions

The environment `{WithArrows}` gives also two options `CodeBefore` and `CodeAfter` for LaTeX code that will be executed at the beginning and at the end of the environment. Theses options are not designed to be hooks (they are avalaible only at the environment level and they are not applied to the nested environments).

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

$$\begin{aligned}A &= (a+b)^2\\ &= a^2+2ab+b^2\end{aligned}$$

we expand

Special commands are available in `CodeAfter` : a command `\NbLines` 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. 10.

[3] Since version 1.4 of witharrows, it's no longer possible to give these options directly when loading the package, *i.e.* with the command `\usepackage` in the preamble.

2 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.[4]

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

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

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

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

Therefore

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

This arrow uses the `lr` option.

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

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

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

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

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

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

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

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

The environment `{WithArrows}` gives also a `groups` option (with a *s* in the name). With this option, the arrows are divided into several "groups". Each group is a set of connected[5] arrows. All the arrows of a given group are grouped on a same vertical line and at a leftmost position.

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

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}`

3 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}`.[6]

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

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

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

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

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

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 an given environment.[7]

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

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

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

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

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

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

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

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

$$\varphi(x,y)=0 \Leftrightarrow (x+y)^2+(x+2y)^2=0 \quad \Big) \; x \text{ and } y \text{ are real}$$
$$\Leftrightarrow \begin{cases} x+y=0 \\ x+2y=0 \end{cases}$$

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

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

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

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

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

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

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

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

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

The following is composed with `{WithArrows}[c,displaystyle]`. The results are stricly identical.[8]

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

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

4 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`, `CodeBefore` and `CodeAfter`). The command `Arrow` can be used as usual in each environment `{WithArrows}`.

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

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

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

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

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

A command `\Arrow` in `CodeAfter` takes three arguments :

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

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

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

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

$$\varphi(x,y)=0 \Leftrightarrow (x+2y)^2+(2x+4y)^2=0$$
$$\Leftrightarrow \left\{\begin{array}{l} x+2y=0 \\ 2x+4y=0 \end{array}\right. \quad \textit{environment number 1}$$
$$\Leftrightarrow \left\{\begin{array}{l} x+2y=0 \\ x+2y=0 \end{array}\right. \quad \textit{environment number 2}$$
$$\Leftrightarrow x+2y=0$$

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

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

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

The options allowed for a command `\Arrow` in `CodeAfter` are : `ll`, `lr`, `rl`, `rr`, `v`, `xoffset`, `tikz` and `TikzCode`. Except `v`, which is specific to `\Arrow` in `CodeAfter`, 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 `CodeAfter`. However, it's always possible to translate an arrow with `xoffset` (or `xshift` of Tikz).

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

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

The package witharrows gives also another command available only in `CodeAfter` : 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 pgfkeys.

```
$\begin{WithArrows}[tikz = rounded corners,
                    CodeAfter = {\MultiArrow{1,...,4}{text}}]
A & = B \\
  & = C \\
```

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

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

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

5 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.[9]

$$\begin{aligned} A \lhd B + B + B + B + B + B + B + B + B + B + B + B \text{wa-37-1} \\ \lhd \begin{cases} C \lhd D \text{wa-37-1-1} \\ E \lhd F \text{wa-37-1-2} \end{cases} \text{wa-37-2} \\ \lhd \begin{cases} G \lhd H + H + H + H + H + H + H \text{wa-37-2-1} \\ I \lhd \begin{cases} J \lhd K \text{wa-37-2-1-1} \\ L \lhd M \text{wa-37-2-1-2} \end{cases} \text{wa-37-2-2} \end{cases} \text{wa-37-3} \\ \lhd \begin{cases} N \lhd O \text{wa-37-3-1} \\ P \lhd Q \text{wa-37-3-2} \end{cases} \text{wa-37-4} \end{aligned}$$

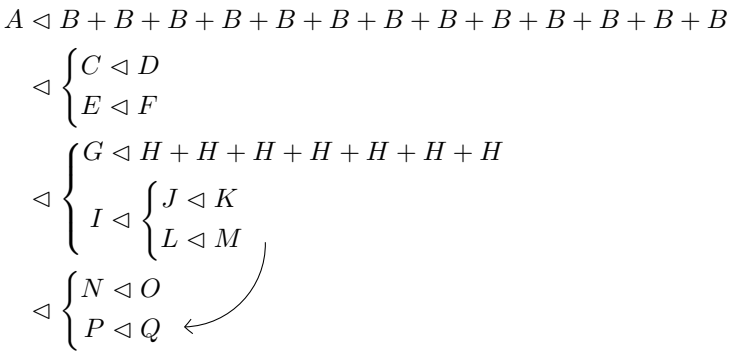
The package witharrows provides some tools facilitating the use of these nodes :

- the command `\WithArrowsLastEnv` gives the number of the last environment of level 0 ;
- a name can be given to a given environment with the option `name` and, in this case, the nodes created in the environment will have aliases constructed with this name ;
- the Tikz style `WithArrows/arrow` is the style used by witharrows when drawing an arrow[10] ;
- the Tikz style `WithArrows/arrow/tips` is the style for the style of the arrow (loaded by `WithArrows/arrow`).

For example, we can draw an arrow from `wa-37-2-1-2-r.south` to `wa-37-3-2-r.north` with the following Tikz command (requires the Tikz library `calc`).

```
\begin{tikzpicture}[remember picture,overlay]
\draw [WithArrows/arrow]
      ($(wa-\WithArrowsLastEnv-2-1-2-r.south)+(3mm,0)$)
   to ($(wa-\WithArrowsLastEnv-3-2-r.north)+(3mm,0)$) ;
\end{tikzpicture}
```

[9] There is an option `shownodenames` to show the names of these nodes.
[10] More precisely, this style is given to the Tikz option "`every path`" before drawing the arrow with the code of the option `TikzCode`. This style is modified (in TeX scopes) by the option `tikz` of witharrows.



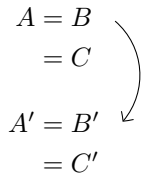
In this case, it would be easier to use a command `\Arrow` in `CodeAfter` but this is an example to explain how the Tikz nodes created by witharrows can be used.

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]
      ($(first-1-r.south)+(3mm,0)$)
   to ($(second-1-r.north)+(3mm,0)$) ;
\end{tikzpicture}
```



6 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.

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

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

It's possible to use the command `\notag` (or `\nonumber`) to suppress a tag.
It's possible to use the command `\tag` to put a special tag (e.g. $\star$).[11]
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}
```

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

A link to the equation ($\star$). This link has been composed with `\eqref{my-equation}` (the command `\eqref` is a command of amsmath).

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

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

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

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

$$\begin{aligned} A &= (a+1)^2 && (3)\\ &= a^2+2a+1 \quad \text{we expand} && (4)\end{aligned}$$

Remark : By design, the option `fleqn` of witharrows is independant 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.

[11] If amsmath (or mathtools) is loaded, it's also possible to use `\tag*` which, as in amsmath, typesets the tag without the parenthesis.

If the package amsmath is loaded, it's possible to use the command `\intertext` in the environments `{DispWithArrows}` and `{DispWithArrows*}` (and even the `\intertext` of nccmath if this package is loaded).
If the option `leqno` is used (for the document or when the package amsmath is loaded), the labels will be composed on the left also for the environments `{DispWithArrows}` et `{DispWithArrows*}`.

The environment `{DispWithArrows}` is similar to the environment `{align}` of amsmath. However, `{DispWithArrows}` is not constructed upon `{align}` (in fact, it's possible witharrows without amsmath).
There are differences between `{DispWithArrows}` and `{align}`. The environment `{DispWithArrows}` allows only 2 columns. With `{DispWithArrows}`, there is no control of a collision between an equation and its tag. An environment `{DispWithArrows}` is always unbreakable (even with `\allowdisplaybreaks` of amsmath). The commands `\label`, `\tag`, `\notag`, `\nonumber` are allowed only in the second column. The labels of `{DispWithArrows}` are not always drawn by showkeys. **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`).**

7 Advanced features

7.1 The option TikzCode : how to change the shape of the arrows

The option `TikzCode` allows the user to change the shape of the arrows.

The value of this option must be a valid Tikz drawing instruction (with the final semi-colon) 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}[ygap=5pt,interline=4mm,
      TikzCode = {\draw[rounded corners]
                        (#1) -- ($(#1) + (5mm,0)$)
                        -- node[circle,
                                draw,
                                auto = false,
                                fill = gray!50,
                                inner sep = 1pt] {\tiny #3}
                        ($(#2) + (5mm,0)$)
                        -- (#2) ; }]
E & \Longleftrightarrow 3 (2x+4) = 6   \Arrow{$\div 3$} \\
  & \Longleftrightarrow 2x+4 = 2       \Arrow{$-4$}     \\
  & \Longleftrightarrow 2x = -2        \Arrow{$\div 2$} \\
  & \Longleftrightarrow  x = -1
\end{WithArrows}
```

$$
\begin{aligned}
E &\Longleftrightarrow 3(2x+4) = 6 \\
&\Longleftrightarrow 2x+4 = 2 \\
&\Longleftrightarrow 2x = -2 \\
&\Longleftrightarrow 2x = -1
\end{aligned}
\qquad \begin{matrix} \div 3 \\ -4 \\ \div 2 \end{matrix}
$$

7.2 Footnotes in the environments of witharrows

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

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

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

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

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

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

$$\begin{aligned} A &= (a+b)^2 \\ &= a^2+b^2+2ab \end{aligned}$$ *We expand*[12]

8 Examples

8.1 With only one column

It's possible to use the environment `{WithArrows}` with making use of the left column only, or the right column only.

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

$$\begin{aligned} f(x) &\geq g(x) \\ f(x)^2 &\geq g(x)^2 \\ f(x)^2 - g(x)^2 &\geq 0 \end{aligned}$$ *by squaring both sides* *by moving to left side*

8.2 MoveEqLeft

It's possible to use `\MoveEqLeft` of mathtools (if we don't want ampersand on the first line):

```
$\begin{WithArrows}[interline=0.5ex]
\MoveEqLeft \arccos(x) = \arcsin \frac45 + \arcsin \frac5{13}
\Arrow{because both are in $[-\frac{\pi}2,\frac{\pi}2]$} \\
& \Leftrightarrow x = \sin\left(\arcsin\frac45 + \arcsin\frac5{13}\right) \\
& \Leftrightarrow x = \frac45\cos\arcsin\frac5{13} + \frac5{13} \cos\arcsin\frac45
\Arrow{$\forall x \in [-1,1], \cos(\arcsin x) = \sqrt{1-x^2}$} \\
& \Leftrightarrow x = \frac45\sqrt{1-\bigl(\frac5{13}\bigr)^2}
+ \frac5{13}\sqrt{1-\bigl(\frac45\bigr)^2}
\end{WithArrows}$
```

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

[12] A footnote.

8.3 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}[%
      interline = 4mm,
      tikz = {every node/.style = {circle,
                                   draw,
                                   auto = false,
                                   fill = gray!50,
                                   inner sep = 1pt,
                                   font = \tiny}}]
E & \Longleftrightarrow 3 (2x+4) = 6
\Arrow{$\div 3$}\\
  & \Longleftrightarrow 2x+4 = 2
\Arrow{$-4$}\\
  & \Longleftrightarrow 2x = -2
\Arrow{$\div 2$} \\
  & \Longleftrightarrow 2x = -1
\end{WithArrows}
```

$$
\begin{aligned}
E &\Longleftrightarrow 3(2x+4) = 6 \\
&\Longleftrightarrow 2x+4 = 2 \\
&\Longleftrightarrow 2x = -2 \\
&\Longleftrightarrow 2x = -1
\end{aligned}
$$

8.4 Examples with the option TikzCode

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

The value by defaut of `TikzCode` is `\draw (#1) to node {#3} (#2) ;` where the three markers `#1`, `#2` and `#3` represent the start raw, the end raw and the label of the arrow.

8.4.1 Example 1

In the following example, we define the value of `TikzCode` 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.

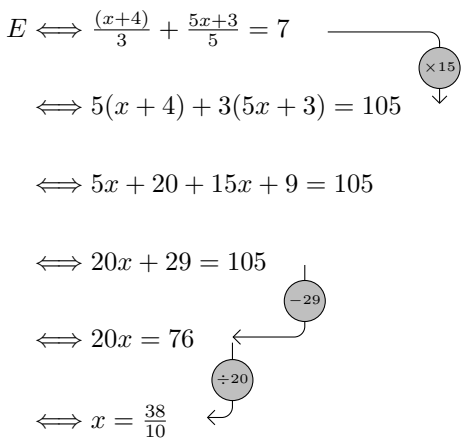
```
$\begin{WithArrows}[
       displaystyle,
       ygap = 2mm,
       ystart = 0mm,
       TikzCode = {\path[draw] (#1) -- ++(4.5cm,0) |- (#2) ;
                   \path (#1) -- (#2)
                         node[text width = 4.2cm, right, midway] {#3} ;}]
S_n
& = \frac1n \sum_{k=0}^{n-1}\cos\bigl(\tfrac{\pi}2\cdot\tfrac kn\bigr)
.....
```

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

8.4.2 Example 2

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

```
\begin{WithArrows}[ll,interline=5mm,xoffset=5mm,
      TikzCode  = {\draw[rounded corners,
                        every node/.style = {circle,
                                             draw,
                                             auto = false,
                                             inner sep = 1pt,
                                             fill = gray!50,
                                             font = \tiny }]
                        let \p1 = (#1),
                            \p2 = (#2)
                        in \ifdim \x1 > \x2
                             (\p1) -- node {#3} (\x1,\y2) -- (\p2)
                           \else
                             (\p1) -- (\x2,\y1) -- node {#3} (\p2)
                           \fi ;}]
E & \Longleftrightarrow \frac{(x+4)}3 + \frac{5x+3}5 = 7
\Arrow{$\times 15$}\\
  & \Longleftrightarrow 5(x+4) + 3(5x+3) = 105 \\
  & \Longleftrightarrow 5x+20 + 15x+9 = 105 \\
  & \Longleftrightarrow 20x+29 = 105
\Arrow{$-29$}\\
  & \Longleftrightarrow 20x = 76
\Arrow{$\div 20$}\\
  & \Longleftrightarrow x = \frac{38}{10}
\end{WithArrows}
```



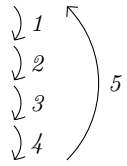
8.5 Automatic numbered loop

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 `CodeAfter`. 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,...,\NbLines}
              { \pgfmathtruncatemacro{\i}{\j-1}
                \Arrow[rr]{\i}{\j}{\i} }
         \Arrow[rr,xoffset=1cm,tikz=<-]{1}{\NbLines}{\NbLines}}
```

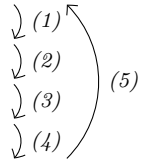
The command `\NbLines` is a command available in `CodeAfter` which gives the total number of lines (=rows) of the current environment (it's a command and not a counter).

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



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 parenthesis, the best way is to change the value of `TikzCode`:

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



9 Implementation

9.1 Declaration of the package and extensions loaded

First, tikz and some Tikz libraries are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.[13]

```
\RequirePackage{tikz}
\usetikzlibrary{arrows.meta,bending}
```

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

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

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

```
\RequirePackage{xparse}
```

9.2 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 1.7), 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.

```
\bool_new:N \g_@@_footnotehyper_bool
```

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

```
\bool_new:N \g_@@_footnote_bool
```

We define a set of keys `WithArrows/package` for these options. However, first, we define a "level of options" `\l_@@_level_int` even if, in the version 1.7 of witharrows, this integer is not used by the options of the set `WithArrows/package`.

```
\int_new:N \l_@@_level_int
\keys_define:nn {WithArrows/package}
      {footnote      .bool_gset:N = \g_@@_footnote_bool,
       footnotehyper .bool_gset:N = \g_@@_footnotehyper_bool}
```

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

```
\ProcessKeysOptions {WithArrows/package}

\msg_new:nnn {witharrows}
             {Option~incompatible~with~Beamer}
             {The~option~"\tl_use:N \l_keys_key_tl"\ is~incompatible~
              with~Beamer~because~Beamer~has~its~own~system~to~extract~footnotes.}
\msg_new:nnn {witharrows}
             {footnote~with~footnotehyper~package}
             {You~can't~use~the~option~footnote~because~the~package~
              footnotehyper~has~already~been~loaded.~
              If~you~want,~you~can~use~the~option~"footnotehyper"~and~the~footnotes~
              within~the~environments~{WithArrows}~will~be~extracted~with~the~tools~
              of~the~package~footnotehyper.}
```

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

```
\msg_new:nnn {witharrows}
             {footnotehyper~with~footnote~package}
             {You~can't~use~the~option~"footnotehyper"~because~the~package~
              footnote~has~already~been~loaded.~
              If~you~want,~you~can~use~the~option~"footnote"~and~the~footnotes~
              within~the~environments~{WithArrows}~will~be~extracted~with~the~tools~
              of~the~package~footnote.}


\bool_if:NT \g_@@_footnote_bool
      {\@ifclassloaded {beamer}
                {\msg_fatal:nn {witharrows}
                               {Option~incompatible~with~Beamer}}
                {}
       \@ifpackageloaded{footnotehyper}
           {\msg_fatal:nn {witharrows}
                          {footnote~with~footnotehyper~package}}
           {}
       \usepackage{footnote}}
\bool_if:NT \g_@@_footnotehyper_bool
      {\@ifclassloaded {beamer}
                {\msg_fatal:nn {witharrows}
                               {Option~incompatible~with~Beamer}}
                {}
       \@ifpackageloaded{footnote}
           {\msg_fatal:nn {witharrows}
                          {footnotehyper~with~footnote~package}}
           {}
       \usepackage{footnotehyper}
       \bool_gset_true:N \g_@@_footnote_bool}
```

The flag `\g_@@_footnote_bool` is raised and so, we will only have to test `\g_@@_footnote_bool` in order to known if we have to insert an environnement `{savenotes}` (see the definition of environnement `{WithArrows}`).

9.3 Some technical definitions

We define a Tikz style `@@_node_style` for the nodes that will be created in the `\halign`. The nodes are Tikz nodes of shape "rectangle" but with zero width. An arrow between two nodes starts from the *south* anchor of the first node and arrives at the *north* anchor of the second node.

```
\tikzset{@@_node_style/.style= {
               above = \l_@@_ystart_dim,
               inner~sep = 0 pt,
               minimum~width = 0pt,
               minimum~height = \l_@@_ygap_dim,
               red,
               \bool_if:NT \l_@@_shownodes_bool {draw} }}
```

The color of the nodes is red, but in fact, the nodes will be drawn only when the option `shownodes` is used (this option is useful for debugging).

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

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

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`).

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

```
\tikzset{WithArrows/arrow/.style  = { align = left,
                                      auto = left,
                                      font = {\small\itshape},
                                      WithArrows/arrow/tips,
                                      bend~left = 45,
                                      -> }}
```

In order to increase the interline in the environments `{WithArrows}`, 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` (this definition will be loaded only if amsmath — or mathtools — has not been loaded yet).

```
\cs_if_free:NT \spread@equation
    {\cs_set_protected:Npn \spread@equation
                {\openup\jot
                 \cs_set_protected:Npn \spread@equation {}}}
```

Don't put `\cs_set_eq:NN \spread@equation \prog_do_nothing:` in the last line because this would raise errors with nested environments.

9.4 Variables

The boolean `\l_@@_in_witharrows_bool` will be raised if (and only if) we are in an environment `{WithArrows}` (and not in an environment `{DispWithArrows}` or `{DispWithArrows*}`).

```
\bool_new:N \l_@@_in_witharrows_bool
```

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

```
\seq_new:N \g_@@_position_in_the_tree_seq
\seq_gput_right:Nn \g_@@_position_in_the_tree_seq 1
```

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

```
\int_new:N \g_@@_last_env_int
```

The following skip (=glue) is the vertical space inserted between two lines (=rows) of the `\halign`.

```
\skip_new:N \l_@@_interline_skip
```

The following integer indicates the position of the box that will be created: 0 (=`t`=`\vtop`), 1 (=`c`=`\vcenter`) or 2 (=`b`=`\vbox`).

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

```
\dim_new:N \l_@@_xoffset_dim
\dim_set:Nn \l_@@_xoffset_dim {3mm}
```

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

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

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

```
\int_new:N \l_@@_pos_arrows_int
```

When we scan a list of options, we want to be able to raise an error if two options of position of the arrows are present. That's why we keep the code of the first option of position in a variable called `\l_@@_previous_pos_arrows_int`. This variable will be set to -1 each time we start the scanning of a list of options.

```
\int_new:N \l_@@_previous_pos_arrows_int
```

At each possible level for the options (*global*, *environment* or *local*: see below), the new values will be appended on the right of this token list.

The dimension `\g_@@_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_arrows_int`) is used.

```
\dim_new:N \g_@@_x_dim
```

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

- `\g_@@_arrow_int` to count the arrows created in the environment ;
- `\g_@@_line_int` to count the lines of the `\halign` ;
- `\g_@@_line_bis_int` to count the lines of the `\halign` which have a second column.[14]

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

```
\seq_new:N \g_@@_arrow_int_seq
\int_new:N \g_@@_arrow_int
\seq_new:N \g_@@_line_int_seq
\int_new:N \g_@@_line_int
\seq_new:N \g_@@_line_bis_int_seq
\int_new:N \g_@@_line_bis_int
```

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

```
\tl_new:N \l_@@_name_tl
```

The boolean `\l_@@_notag_bool` will be used in `{DispWithArrows}`. In particular, it will be raised when the command `\notag` is used.

```
\bool_new:N \l_@@_notag_bool
```

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

```
\tl_new:N \l_@@_tag_tl
```

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

```
\bool_new:N \l_@@_tag_star_bool
```

The command `\@@_label:n` will be linked to `\label` in the second column of the `\halign` of the environment `{DispWithArrows}`. This command will store its argument in the token list `\l_@@_label_tl`.

```
\tl_new:N \l_@@_label_tl
\cs_set:Nn \@@_label:n {\tl_set:Nn \l_@@_label_tl {#1}}
```

The boolean `\l_@@_fleqn_bool` indicates wether the environments `{DispWithArrows}` must be composed flush left or centered. It corresponds to the option `fleqn`.

```
\bool_new:N \l_@@_fleqn_bool
```

[14] This counter is used in order to raise an error if there is a line without the second column (such an situation could raise a PGF error for an undefined node).

The dimension `\l_@@_mathindent_dim` is used only by the environments `{DispWithArrows}` : it's the left margin of the environments `{DispWithArrows}` if the environment `{DispWithArrows}` is composed flush left (option `fleqn`).

```
\dim_new:N \l_@@_mathindent_dim
\dim_set:Nn \l_@@_mathindent_dim {25pt}
```

9.5 The definition of the options

There are four levels where options can be set:

- with `\usepackage[...]{witharrows}`: this level will be called *package* level (number 0);
- with `\WithArrowsOptions{...}`: this level will be called *global* level (number 1);
- with `\begin{WithArrows}[...]`: this level will be called *environment* level (number 2);
- with `\Arrow[...]` (included in `CodeAfter`): this level will be called *local* level (number 3).

The level is specified in the variable `\l_@@_level_int` and the code attached to the options can use this information to alter its actions.

```
\int_set:Nn \l_@@_level_int 1
```

We start with a submodule which will be loaded only at the global or the environment level.

The options `t`, `c` and `b` indicate if we will create a `\vtop`, a `\vcenter` or a `\vbox`. This information is stored in the variable `\l_@@_pos_env_int`. Of course, they are available only in `{WithArrows}` and not in `{DispWithArrows}` or `{DispWithArrows*}`

```
\keys_define:nn {WithArrows/GlobalOrEnv}
   {  t   .code:n            = {\bool_if:NTF \l_@@_in_witharrows_bool
                                    {\int_set:Nn \l_@@_pos_env_int 0}
                                    {\msg_error:nn {witharrows} {Option~will~be~ignored}}},
      t   .value_forbidden:n = true,
      c   .code:n            = {\bool_if:NTF \l_@@_in_witharrows_bool
                                    {\int_set:Nn \l_@@_pos_env_int 1}
                                    {\msg_error:nn {witharrows} {Option~will~be~ignored}}},
      c   .value_forbidden:n = true,
      b   .code:n            = {\bool_if:NTF \l_@@_in_witharrows_bool
                                    {\int_set:Nn \l_@@_pos_env_int 2}
                                    {\msg_error:nn {witharrows} {Option~will~be~ignored}}},
      b   .value_forbidden:n = true,
```

The gap between two consecutive arrows.

```
      ygap .dim_set:N        = \l_@@_ygap_dim,
      ygap .value_required:n = true,
      ygap .initial:n        = 0.4 ex,
```

The vertical position of the start point of an arrow.

```
      ystart .dim_set:N        = \l_@@_ystart_dim,
      ystart .value_required:n = true,
      ystart .initial:n        = 0.4 ex,
```

Usually, the number of columns in a `{WithArrows}` environment is limited to 2. Nevertheless, it's possible to have more columns with the option `MoreColumns`.

```
      MoreColumns .code:n            = { \msg_redirect_name:nnn
                                         {witharrows}
                                         {Third~column~in~an~environment~{WithArrows}}
                                         {none} },
      MoreColumns .value_forbidden:n = true,
```

By default, an error message is raised if there is a line without ampersand (&). However, it's possible to suppress this error with the option AllowLineWithoutAmpersand.

```
AllowLineWithoutAmpersand .code:n = { \msg_redirect_name:nnn
                                        {witharrows}
                                        {All~lines~must~have~an~ampersand}
                                        {none} },
AllowLineWithoutAmpersand .value_forbidden:n = true,
```

If the user wants to give a new name to the \Arrow command (and the name \Arrow remains free).

```
CommandName .tl_set:N         = \l_@@_CommandName_tl,
CommandName .initial:n        = Arrow ,
CommandName .value_required:n = true,
```

```
TikzCode .tl_set:N         = \l_@@_tikz_code_tl,
TikzCode .initial:n        = \draw~(#1)~to~node{#3}~(#2)~; ,
TikzCode .value_required:n = true,
```

With the option displaystyle, the environments will be composed in \displaystyle.

```
displaystyle .bool_set:N     = \l_@@_displaystyle_bool,
displaystyle .initial:n      = false,
```

With the option shownodes, the nodes will be drawn in red (useful only for debugging).

```
shownodes .bool_set:N        = \l_@@_shownodes_bool,
shownodes .initial:n         = false,
```

With the option shownodenames, the name of the "right nodes" will be written in the document (useful only for debugging).

```
shownodenames .bool_set:N    = \l_@@_shownodenames_bool,
shownodenames .initial:n     = false,
```

With the option group, *all* the arrows of the environment are vertical with the same abscissa and at a leftmost position.

```
group    .code:n    = {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
                                {\msg_error:nn {witharrows}
                                               {Two~options~are~incompatible}}
                           \int_set:Nn \l_@@_previous_pos_arrows_int 6
                           \int_set:Nn \l_@@_pos_arrows_int 6} ,
group    .value_forbidden:n = true,
```

With the option groups (with a *s*), the arrows of the environment are divided in groups by an argument of connexity, and, in each group, the arrows are vertical with the same abscissa and at a leftmost position. When the option group or groups is used, it's not possible to another option of position like ll, lr, etc. for a individual key.

```
groups   .code:n   = {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
                                {\msg_error:nn {witharrows}
                                               {Two~options~are~incompatible}}
                           \int_set:Nn \l_@@_previous_pos_arrows_int 7
                           \int_set:Nn \l_@@_pos_arrows_int 7} ,
groups   .value_forbidden:n = true,
```

The option CodeBefore gives a code that is executed at the beginning of the environment {WithArrows} (after the eventual \begin{savenotes}).

```
CodeBefore  .code:n = {\int_compare:nNnTF \l_@@_level_int = 1
                           {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                           {\tl_put_right:Nn \l_@@_code_before_tl {#1}}} ,
CodeBefore  .value_required:n = true,
```

The option `CodeAfter` gives a code that is executed at the end of the environment `{WithArrows}` (after the eventual `\end{savenotes}`).

```
      CodeAfter .code:n = {\int_compare:nNnTF \l_@@_level_int = 1
                            {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                            {\tl_put_right:Nn \l_@@_code_after_tl {#1}}} ,
      CodeAfter .value_required:n = true,
```

The option `name` is a name given to the environment. If this option is used, the nodes created in the environment will have aliases constructed with this name (and it will be easier to use these nodes from outside the environment).

```
      name .code:n = {\int_compare:nNnTF \l_@@_level_int = 1
                            {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                            {\tl_set:Nn \l_@@_name_tl {#1}}} ,
      name .value_required:n = true,
```

The option `fleqn` indicates wether the environments `{DispWithArrows}` are composed centered or flush left.

```
      fleqn   .code:n = {\bool_if:NTF \l_@@_in_witharrows_bool
                           {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                           {\tl_if_eq:nnTF {#1} {true}
                               {\bool_set_true:N \l_@@_fleqn_bool}
                               {\bool_set_false:N \l_@@_fleqn_bool}}},
      fleqn   .default:n = true,
```

The option `mathindent` is the left margin of the environments `{DispWithArrows}` when the option `fleqn` is used.

```
      mathindent   .code:n = {\bool_if:NTF \l_@@_in_witharrows_bool
                              {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                              {\dim_set:Nn \l_@@_mathindent_dim {#1}}},
      mathindent   .value_required:n = true,
```

The option `notag` indicates wether the environments `{DispWithArrows}` will be without tags (like `{DispWithArrows*}`).

```
      notag   .code:n = {\bool_if:NTF \l_@@_in_witharrows_bool
                           {\msg_error:nn {witharrows} {Option~will~be~ignored}}
                           {\tl_if_eq:nnTF {#1} {true}
                               {\bool_set_true:N \l_@@_notag_bool}
                               {\bool_set_false:N \l_@@_notag_bool}}},
      notag   .default:n = true,
      nonumber .meta:n   = notag,
      unknown .code:n  = \msg_error:nn {witharrows} {Option~unknown}
     }
```

Then we define the main module called `WithArrows/General` which will be loaded at all the levels.

The option `tikz` gives Tikz parameters that will be given to the arrow when it is drawn (more precisely, the parameters will be given to the command `\path` of Tikz).

```
\keys_define:nn {WithArrows/General}
     {tikz     .code:n            = \tikzset {WithArrows/arrow/.append~style = {#1}},
      tikz     .initial:n         = {},
      tikz     .value_required:n  = true,
```

The other options are for the position of the arrows. The treatment is the same for the options `ll`, `rr`, `lr`, `lr` and `i` and that's why a dedicated fonction `\@@_analyze_option_position:n` has been written (see below).

```
      rr       .value_forbidden:n = true,
      rr       .code:n            = \@@_analyze_option_position:n 0 ,
      ll       .value_forbidden:n = true,
      ll       .code:n            = \@@_analyze_option_position:n 1 ,
```

```
      rl       .value_forbidden:n = true,
      rl       .code:n            = \@@_analyze_option_position:n 2 ,
      lr       .value_forbidden:n = true,
      lr       .code:n            = \@@_analyze_option_position:n 3 ,
      i        .value_forbidden:n = true,
      i        .code:n            = \@@_analyze_option_position:n 5 ,
```

The option `xoffset` change the x-offset of the arrows (towards the right). It's a dimension and not a skip. It's not possible to change the value of this parameter for a individual arrow if the option `group` or the option `groups` is used. When we will treat an individual arrow, we will give it the option `tikz={xshift=\l_@@_xoffset_dim}` (we can't to it at the global or the environment level because the Tikz options `xshift` are cumulative.

```
      xoffset  .code:n  = {\bool_if:nTF {\int_compare_p:nNn \l_@@_level_int = 3 &&
                                         \int_compare_p:nNn \l_@@_pos_arrows_int > 5}
                              {\msg_error:nn {witharrows}
                                             {Option~incompatible~with~"group(s)"}}
                              {\dim_set:Nn \l_@@_xoffset_dim {#1}}} ,
      xoffset  .value_required:n  = true,
```

The option `jot` exists for compatibility. It changes directly the value of the parameter `\jot`, which is a LaTeX parameter and not a parameter specific to witharrows. It's allowed only at the level of the environment (maybe we should suppress completely this option in the future).

```
      jot       .code:n     = {\int_compare:nNnTF \l_@@_level_int = 2
                                  {\dim_set:Nn \jot {#1}}
                                  {\msg_error:nn {witharrows}
                                                 {Option~will~be~ignored}}} ,
      jot       .value_required:n  = true,
```

The option `interline` gives the vertical skip (=glue) inserted between two lines (independently of `\jot`). It's accepted only at the level of the environment (this last point is a kind of security). Futhermore, this option has a particular behaviour: it applies only to the current environment and doesn't apply to the nested environments.

```
      interline   .code:n       = {\int_compare:nNnTF \l_@@_level_int = 2
                                      {\skip_set:Nn \l_@@_interline_skip {#1}}
                                      {\msg_error:nn {witharrows}
                                                     {Option~will~be~ignored}}} ,
      interline   .value_required:n = true,
```

Eventually, a key `jump` (see below) and a key for unknown keys.

```
      jump    .code:n  = \msg_error:nn {witharrows} {Option~will~be~ignored} ,
      unknown .code:n  = \msg_error:nn {witharrows} {Option~unknown}
}
```

The key `jump` indicates the number of lines jumped by the arrow (1 by default). This key will be extracted when the command `\Arrow` will be executed. That's why there is a special module for this key. The key `jump` is extracted in the command `\Arrow` because we want to compute right away the final line of the arrow (this will be useful for the options `group` and `groups`).

```
\keys_define:nn {WithArrows/jump}
     {jump  .code:n = {\int_set:Nn \l_@@_jump_int {#1}
                       \int_compare:nNnF \l_@@_jump_int > 0
                           {\msg_error:nn {witharrows}
                                          {The~option~"jump"~must~be~non~negative}}} ,
      jump  .value_required:n  = true}
```

The following command is for technical reasons. It's used for the following options of position: `ll`, `lr`, `rl`, `rr` and `i`. The argument is the corresponding code for the position of the arrows.

```
\cs_new_protected:Nn \@@_analyze_option_position:n
     {\int_compare:nNnT \l_@@_previous_pos_arrows_int > {-1}
```

```
        {\msg_error:nn {witharrows}
                       {Two~options~are~incompatible}}
      \int_set:Nn \l_@@_previous_pos_arrows_int {#1}
```

It's not possible to use one of the considered options at the level of an arrow (level 2) when the option `group` or the option `groups` is used. However, if we are at the level of an environment, it's possible to override a previous option `group` or `groups` (this previous option `group` or `groups` would necessarily have been set at a global level by `\WithArrowsOptions`).

```
      \bool_if:nTF { \int_compare_p:nNn \l_@@_level_int = 3 &&
                     \int_compare_p:nNn \l_@@_pos_arrows_int > 5}
        {\msg_error:nn {witharrows}
                       {Option~incompatible~with~"group(s)"}}
        {\int_set:Nn \l_@@_pos_arrows_int {#1}}}
```

`\WithArrowsOptions` is the command of the witharrows package to fix options at the document level.

```
\NewDocumentCommand \WithArrowsOptions {m}
    {\int_set:Nn \l_@@_previous_pos_arrows_int {-1}
     \keys_set_known:nnN {WithArrows/General} {#1} \l_tmpa_tl
     \keys_set:nV {WithArrows/GlobalOrEnv} \l_tmpa_tl}
```

9.6 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 `CommandName` and the user command for `\@@_Arrow` will be different. This mechanism can be useful when the user has already a command named `\Arrow` he wants to still be able to use in the environment `{WithArrows}`.

```
\NewDocumentCommand \@@_Arrow {O{} m O{}}
        {
```

The counter `\g_@@_arrow_int` counts the arrows in the environment. The incrementation must be global (`gincr`) because the command `\Arrow` will be used in the cell of a `\halign`. It's recalled that we manage a stack for this counter.

```
         \int_gincr:N \g_@@_arrow_int
```

We decide to extract immediatly the key `jump` in order to compute the end line. That's the reason why there is a module `WithArrows/jump` with this sole key. The remainded key-value pairs are stored in `\l_tmpa_tl` and will be stored further in the properly list of the arrow.

```
         \int_zero_new:N \l_@@_jump_int
         \int_set:Nn \l_@@_jump_int 1
         \keys_set_known:nnN {WithArrows/jump} {#1,#3} \l_tmpa_tl
```

We will construct a global property list to store the informations of the considered arrow. The four fields of this property list are "initial", "final", "options" and "label".

1. First, the line from which the arrow starts:

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

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

```
             \int_set:Nn \l_tmpa_int {\g_@@_line_int + \l_@@_jump_int}
             \prop_put:NnV \l_tmpa_prop {final} \l_tmpa_int
```

3. All the options of the arrow (it's a token list):

```
             \prop_put:NnV \l_tmpa_prop {options} \l_tmpa_tl
```

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

```
             \prop_put:Nnn \l_tmpa_prop {label} {#2}
```

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

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

\cs_new_protected:Nn \@@_Arrow_first_column:
```

All messages of LaTeX3 must be *fully expandable* and that's why we do the affectation (necessary for a comparison) before the `\msg_error:nn`.

```
            {\tl_set:Nn \l_tmpa_tl {Arrow}
             \msg_error:nn {witharrows} {Arrow~in~first~column}
             \@@_Arrow}
```

9.7 The environment {WithArrows}

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

```
\cs_new_protected:Nn \@@_pre_environment:n
```

First the initialisation of the three counters `\g_@@_arrow_int`, `\g_@@_line_int` and `\g_@@_line_bis_int`. However, we have to save their previous values with the three stacks created for this end.

```
      { \seq_gput_right:NV \g_@@_arrow_int_seq \g_@@_arrow_int
        \int_gzero:N \g_@@_arrow_int
        \seq_gput_right:NV \g_@@_line_int_seq \g_@@_line_int
        \int_gzero:N \g_@@_line_int
        \seq_gput_right:NV \g_@@_line_bis_int_seq \g_@@_line_bis_int
        \int_gzero:N \g_@@_line_bis_int
```

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

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

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

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

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

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

These counters will be used later as variables.

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

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.[15]

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

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

The value corresponding to the key `CodeBefore` 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 `CodeAfter`.

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

We process the options given to the `{WithArrows}` environment. The level of options is set to 1.

```
\int_set:Nn \l_@@_previous_pos_arrows_int {-1}
\int_set:Nn \l_@@_level_int 2
\keys_set_known:nnN {WithArrows/General} {#1} \l_tmpa_tl
\keys_set:nV {WithArrows/GlobalOrEnv} \l_tmpa_tl
```

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

```
\bool_if:NT \g_@@_footnote_bool {\begin{savenotes}}
```

We execute the code `\l_@@_code_before_tl` of the option `CodeBefore` of the environment after the eventual `\begin{savenotes}` and, symetricaly, 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 `CodeAfter`).

```
\l_@@_code_before_tl
```

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

```
\cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow}
```

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

Now, we begin the environment `{WithArrows}`.

```
\NewDocumentEnvironment {WithArrows} {O{}}
        { \bool_set_true:N \l_@@_in_witharrows_bool
          \reverse_if:N \if_mode_math:
                            \msg_error:nn {witharrows}
                                          {{WithArrows}~used~outside~math~mode}
                        \fi:
          \cs_set:Npn \notag {\msg_error:nnn {witharrows}
                                             {Command~not~allowed~in~{WithArrows}}
                                             {\notag}}
          \cs_set:Npn \nonumber {\msg_error:nnn {witharrows}
                                                {Command~not~allowed~in~{WithArrows}}
                                                {\nonumber}}
          \cs_set:Npn \tag ##1 {\msg_error:nnn {witharrows}
                                               {Command~not~allowed~in~{WithArrows}}
                                               {\tag}}
          \cs_set:Npn \label ##1 {\msg_error:nnn {witharrows}
                                                 {Command~not~allowed~in~{WithArrows}}
                                                 {\label}}
          \@@_pre_environment:n {#1}
```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`[16] depending of the value of `\l_@@_pos_env_int` (fixed by the options `t`, `c` or `b`). The environment `{WithArrows}` must be used in math mode[17] and therefore, we can use `\vcenter`.

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

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

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

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

```
\spread@equation
```

We begin the \halign and the preamble.

```
\ialign\bgroup
```

We increment the counter \g_@@_line_int which will be used in the names of the Tikz nodes created in the array. This incrementation must be global (gincr) because we are in the cell of a \halign. It's recalled that we manage a stack for this counter.

```
\int_gincr:N \g_@@_line_int
\cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow_first_column:
\strut\hfil
$\bool_if:NT \l_@@_displaystyle_bool \displaystyle {##}$
&
```

In the second column, we increment the counter \g_@@_line_bis_int because we want to count the lines with a second column and raise an error if there is lines without a second column. Once again, the incrementation must be global and it's recalled that we manage a stack for this counter too.

```
\int_gincr:N \g_@@_line_bis_int
$\bool_if:NT \l_@@_displaystyle_bool \displaystyle {{}##}$
```

We create the "left node" of the line (when using macros in Tikz node names, the macros have to be fully expandable: here, \tl_use:N and \int_use:N are fully expandable).

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

If the environment {WithArrows} has a name (given with the option name), the node has also an alias constructed with this name.

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

Now, after the \hfil, we create the "right node" and, if the option shownodenames is raised, the name of the node is written in the document (useful for debugging).

```
\tl_if_empty:NTF \l_@@_name_tl
   {\tikz [remember~picture,overlay]
         \node [@@_node_style,
                name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-r] {} ;}
   {\tikz [remember~picture,overlay]
         \node [@@_node_style,
                name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-r,
                alias = \l_@@_name_tl-\int_use:N\g_@@_line_int-r] {} ;}
\bool_if:NT \l_@@_shownodenames_bool
     {\hbox_overlap_right:n {\small wa-\l_@@_prefix_str
                                    -\int_use:N\g_@@_line_int}}
```

Usually, the \halign of an environment {WithArrows} will have exactly two columns. Nevertheless, if the user wants to use more columns (without arrows) it's possible with the option MoreColumns.

```
&& \msg_error:nn {witharrows} {Third~column~in~an~environment~{WithArrows}}
$\bool_if:NT \l_@@_displaystyle_bool \displaystyle {##}$
\cr
}
```

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).

```
        {\crcr
         \egroup
         \egroup
         \@@_post_environment:}
```

This is the end of the environment {WithArrows}.

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

```
\cs_new_protected:Nn \@@_post_environment:
```

If there is a line without the second column, we raise an error (a line without the second column could generate an PGF error for an unknown node since the nodes are created in the second column).

```
        {\int_compare:nNnT \g_@@_line_bis_int < \g_@@_line_int
                {\msg_error:nn {witharrows} {All~lines~must~have~an~ampersand}}
```

It there is really arrows in the environment, we draw the arrows:

- if neither option group or groups is used, we can draw directly ;
- if option group or option groups is used (\l_@@_pos_arrows_int > 5), we have to draw the arrows group by group ; the macro \@@_draw_arrows: does the work.

```
        \int_compare:nNnT \g_@@_arrow_int > 0
               {\int_compare:nNnTF \l_@@_pos_arrows_int > 5
                 \@@_draw_arrows:
                 {\@@_draw_arrows:nn 1 \g_@@_arrow_int}}
```

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

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

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

```
        \cs_set:Npn \NbLines {\int_use:N \g_@@_line_int}
```

The command \MultiArrow is available in CodeAfter, and we have a special version of \Arrow, called "\Arrow in CodeAfter" in the documentation.[18]

```
        \cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
        \cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow_code_after
        \l_@@_code_after_tl
        \group_end:
```

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).

```
        \bool_if:NT \g_@@_footnote_bool {\end{savenotes}}
```

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

```
        \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
        \seq_gpop_right:NN \g_@@_position_in_the_tree_seq \l_tmpa_tl
        \seq_gput_right:Nx \g_@@_position_in_the_tree_seq {\int_eval:n {\l_tmpa_tl+1}}
```

We update the value of the counter \g_@@_last_env_int. This counter is used only by the user function \WithArrowsLastEnv.

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

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

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

```
            \seq_gpop_right:NN \g_@@_arrow_int_seq {\l_tmpa_tl}
            \int_gset:Nn \g_@@_arrow_int {\l_tmpa_tl}
            \seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
            \int_gset:Nn \g_@@_line_int {\l_tmpa_tl}
            \seq_gpop_right:NN \g_@@_line_bis_int_seq \l_tmpa_tl
            \int_gset:Nn \g_@@_line_bis_int {\l_tmpa_tl}
            }
```

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

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 a \halign structure.
First, we remove an eventual token * since the commands \\ and * are equivalent in an environment {WithArrows} (an environment {WithArrows}, like an environment {aligned} of amsmath, is always unbreakable).

```
\cs_new_protected:Nn \@@_cr:
      {\scan_stop:
       \group_align_safe_begin:
       \peek_meaning_remove:NTF * \@@_cr_i: \@@_cr_i:}
```

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

```
\cs_new_protected:Nn \@@_cr_i:
      {\peek_meaning:NTF [ {\@@_cr_ii:} {\@@_cr_ii:[\c_zero_dim]} }
\cs_new_protected:Npn \@@_cr_ii:[#1]
      {\group_align_safe_end:
```

For the environment {DispWithArrows}, the behaviour of \\ is different because we add the third column which is the column for the tag (number of the equation).

```
       \bool_if:NF \l_@@_in_witharrows_bool
           {\bool_if:NTF \l_@@_notag_bool
```

If there is no tag to put, we use as well the third column because you want to raise an error if the user uses more than two columns.

```
                {&}
                {
```

Here, we can't use \refstepcounter{equation} (even the \refstepcounter modified by \hyperref) because if the user has issued a \tag command, we have to use \l_@@_tag_tl and not \theequation. That's why we do the incrementation of the counter "manually" and, the, we insert some code inspired by the code of \refstepcounter.

```
                 \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 in the line. We use a global variable because we will use it in the *next* cell (after the &).

```
                 \cs_gset:Npx \g_tmpa_tl
                      {\tl_if_empty:NTF \l_@@_tag_tl
                          \theequation
                          \l_@@_tag_tl}
                 \tl_if_empty:NF \l_@@_label_tl
                      {
```

The following code is inspired from the definition of \refstepcounter in source2e.

```
                       \cs_set_eq:NN \@currentlabel \g_tmpa_tl
```

The following code is inspired from the redefinition of \refstepcounter done by hyperref. It is executed only if hyperref has been loaded.

```
                       \cs_if_exist:NT \hyper@refstepcounter
                            {\cs_set:Npn \This@name {equation}
                             \hyper@refstepcounter{equation}}
```

Now, we can issue the command \label.

```
                        \@@_old_label {\l_@@_label_tl}}
```

We store the boolean \l_@@_tag_star_bool in the global variable \g_tmpa_bool because we will use it in the *next* cell (after the &).

```
                  \bool_gset_eq:NN \g_tmpa_bool \l_@@_tag_star_bool
                  & \cs_set_eq:NN \theequation \g_tmpa_tl
                    \bool_if:NT \g_tmpa_bool {\cs_set:Npn \tagform@ {}}
```

We use \@eqnnum (we recall that there are two definitions of \@eqnnum, a standard definition and another, loaded if the option \leqno is used).

```
                    \hbox_overlap_left:n \@eqnnum
              }}
        \cr\noalign{\skip_vertical:n {#1 + \l_@@_interline_skip}
        \scan_stop:}}
```

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

9.8 The environment {DispWithArrows}

For the environment {DispWithArrows}, the construction is a construction of the type:
\[\vcenter{\halign to \displaywith {...}}\]
The purpose of the \vcenter is to have an environment unbreakable.

```
\NewDocumentEnvironment {DispWithArrows} {O{}}
          {
```

The command \intertext@ is a command of amsmath which loads the definition of \intertext (if it is loaded, the package nccmath gives a new defintion of \intertext@).

```
            \cs_if_exist_use:N \intertext@
            \if_mode_math:
                \msg_error:nn {witharrows}
                              {{DispWithArrows}~used~in~math~mode}
            \fi:
            \bool_set_false:N \l_@@_in_witharrows_bool
            \@@_pre_environment:n {#1}
```

We use a \vcenter in order to prevent page breaks in the environment.

```
            \begin{displaymath}
            \vcenter \bgroup
            \spread@equation
            \bool_if:NTF \l_@@_fleqn_bool
                  {\tabskip = \c_zero_skip}
                  {\tabskip = 0 pt plus 1000 pt minus 1000 pt}
```

If amsmath is loaded, the LaTeX version of \label is stored in \ltx@label. If not, it's, of course, stored in \label. We store this definition of \label because we will redefine \label in the environment.

```
            \cs_if_exist:NTF \ltx@label
               {\cs_set_eq:NN \_@@_old_label \ltx@label}
               {\cs_set_eq:NN \_@@_old_label \label}
            \cs_set:Npn \notag {\msg_error:nnn {witharrows}
                                               {Command~not~allowed~in~{DispWithArrows}}
                                               {\notag}}
            \cs_set:Npn \nonumber {\msg_error:nnn {witharrows}
                                                  {Command~not~allowed~in~{DispWithArrows}}
                                                  {\nonumber}}
            \cs_set:Npn \tag ##1 {\msg_error:nnn {witharrows}
                                                 {Command~not~allowed~in~{DispWithArrows}}
                                                 {\tag}}
            \cs_set:Npn \label ##1 {\msg_error:nnn {witharrows}
                                                   {Command~not~allowed~in~{DispWithArrows}}
```

```
                                                {\label}}
            \halign to \displaywidth \bgroup
              \int_gincr:N \g_@@_line_int
              \cs_set_eq:cN \l_@@_CommandName_tl \@@_Arrow_first_column:
              \strut
              \bool_if:NT \l_@@_fleqn_bool
                    {\hspace{\l_@@_mathindent_dim}}
              \hfil
              $\bool_if:NT \l_@@_displaystyle_bool \displaystyle {##}$
              \tabskip = \c_zero_skip
             &
              \cs_set:Npn \notag {\bool_set_true:N \l_@@_notag_bool}
              \cs_set_eq:NN \nonumber \notag
              \cs_set_eq:NN \tag \@@_tag
              \cs_set_eq:NN \label \@@_label:n
              $\bool_if:NT \l_@@_displaystyle_bool \displaystyle {{}##}$
              \tabskip = 0 pt plus 1000 pt minus 1000 pt
              \int_gincr:N \g_@@_line_bis_int
              \tl_if_empty:NTF \l_@@_name_tl
                 {\tikz [remember~picture,overlay]
                       \node [_@@_node_style,
                              name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-l] {} ;}
                 {\tikz [remember~picture,overlay]
                       \node [_@@_node_style,
                              name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-l,
                              alias = \l_@@_name_tl-\int_use:N\g_@@_line_int-l] {} ;}
              \hfil
              \tl_if_empty:NTF \l_@@_name_tl
                 {\tikz [remember~picture,overlay]
                       \node [_@@_node_style,
                              name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-r] {} ;}
                 {\tikz [remember~picture,overlay]
                       \node [_@@_node_style,
                              name = wa-\l_@@_prefix_str-\int_use:N\g_@@_line_int-r,
                              alias = \l_@@_name_tl-\int_use:N\g_@@_line_int-r] {} ;}
              \bool_if:NT \l_@@_shownodenames_bool
                   {\hbox_overlap_right:n {\small wa-\l_@@_prefix_str
                                             -\int_use:N\g_@@_line_int}}
             & ##
              \tabskip = \c_zero_skip
             && \msg_error:nn {witharrows} {Third~column~in~an~environment~{DispWithArrows}}
                \iffalse ## \fi
              \cr}
           {\\
```

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

```
            \egroup
```

The following `\egroup` is for the `\vcenter` (aimed to prevent page breaks).

```
            \egroup
            \end{displaymath}
            \@@_post_environment:
            \ignorespacesafterend
            }
```

The command `\@@_tag` will be linked to `\tag` in the environment `{DispWithArrows}`.

```
\NewDocumentCommand \@@_tag {sm}
    {\bool_set_false:N \l_@@_notag_bool
     \tl_set:Nn \l_@@_tag_tl {#2}
     \bool_set:Nn \l_@@_tag_star_bool {#1}
```

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

command \@eqnnum.[19]

```
    \bool_if:nT {#1 && ! \cs_if_exist_p:N \tagform@}
        { \msg_error:nn {witharrows} {tag*~without~amsmath} }
  }
```

With the environment {DispWithArrows*}, the equations are not numbered.

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

9.9 We draw the arrows

\@@_draw_arrows: draws the arrows when the option group or the option groups is used. In both cases, we have to compute the x-value of a group of arrows before actually drawing the arrows of that group. The arrows will actually be drawn by the macro \@@_draw_arrows:nn.

```
\cs_new_protected:Nn \@@_draw_arrows:
  { \group_begin:
```

\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_@@_last_line_of_group_int will be the last line involved in the group (impossible to guess in advance).

```
    \int_zero_new:N \l_@@_first_arrow_of_group_int
    \int_zero_new:N \l_@@_first_line_of_group_int
    \int_zero_new:N \l_@@_last_line_of_group_int
    \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 lines of that group.

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

We extract from the property list of the current arrow the fields "initial" and "final" and we store these values in \l_@@_initial_int and \l_@@_final_int. However, we have to do a conversion because the components of a property list are token lists.

```
      \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
                    {initial} \l_tmpa_tl
      \int_set:Nn \l_@@_initial_int {\l_tmpa_tl}
      \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
                    {final} \l_tmpa_tl
      \int_set:Nn \l_@@_final_int {\l_tmpa_tl}
```

We test if 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 (with the x-value computed in \g_@@_x_dim).

```
      \bool_if:nT {     \int_compare_p:nNn \l_@@_pos_arrows_int = 7
                     && \int_compare_p:nNn \l_@@_arrow_int > 1
                     && \int_compare_p:nNn
                                \l_@@_initial_int > \l_@@_last_line_of_group_int}
            {\@@_draw_arrows:nn \l_@@_first_arrow_of_group_int {\l_@@_arrow_int - 1}
             \bool_set_true:N \l_@@_new_group_bool}
```

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

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 two circonstancies: if we are at the first arrow of the environment (that's why the flag is raised before the beginning of the loop) an if we have just finished a group (that's why the flag is raised in the previous conditionnal). At the beginning of a group, we have to initialize four variables: `\l_@@_first_arrow_int`, `\l_@@_first_line_of_group_int`, `\l_@@_last_line_of_group` and `\g_@@_x_dim` (global for technical reasons). The last two will evolve during the construction of the group.

```
      \bool_if:nTF \l_@@_new_group_bool
             {\bool_set_false:N \l_@@_new_group_bool
              \int_set:Nn \l_@@_first_arrow_of_group_int \l_@@_arrow_int
              \int_set:Nn \l_@@_first_line_of_group_int \l_@@_initial_int
              \int_set:Nn \l_@@_last_line_of_group_int \l_@@_final_int
              \begin{tikzpicture} [@@_standard]
              \tikz@parse@node\pgfutil@firstofone (\int_use:N\l_@@_initial_int-1)
              \dim_gset:Nn \g_@@_x_dim \pgf@x
              \end{tikzpicture}
              }
```

If we are not at the beginning of a new group, we actualize `\l_@@_last_line_of_group_int`.

```
             {\int_set:Nn \l_@@_last_line_of_group_int
                    {\int_max:nn \l_@@_last_line_of_group_int \l_@@_final_int}}
```

We update the current x-value (in `\g_@@_x_dim`) even if we are at the beginning of a group. Indeed, the previous initialisation of `\g_@@_x_dim` only considers the initial line of the arrows and now we consider all the lines between the initial and the final line. This is done with `\@@_update_x_value:nn`. We have written a command for this because it is also used with the option `i` (`\l_@@_pos_arrows_int` = 5).

```
      \@@_update_x_value:nn \l_@@_initial_int \l_@@_final_int
```

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

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

After the last arrow of the environment, we have to draw the last group of arrows.

```
    \@@_draw_arrows:nn \l_@@_first_arrow_of_group_int \g_@@_arrow_int
    \group_end:
}
```

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

```
\cs_generate_variant:Nn \keys_set:nn {no}
\cs_new_protected:Nn \@@_keys_set: {\keys_set:no {WithArrows/General}}
```

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).

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

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

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

We extract from the property list of the current arrow the fields "initial" and "final" and we store these values in `\l_@@_initial_int` and `\l_@@_final_int`. However, we have to do a conversion because the components of a property list are token lists.

```
      \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
                    {initial} \l_tmpa_tl
      \int_set:Nn \l_@@_initial_int {\l_tmpa_tl}
      \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str _\int_use:N\l_@@_arrow_int _prop}
                    {final} \l_tmpa_tl
      \int_set:Nn \l_@@_final_int {\l_tmpa_tl}
```

If the arrow ends after the last line of the environment, we raise an error (we recall that, after the construction of the `\halign`, `\g_@@_line_int` is the total number of lines of the environment). If the initial node or the final node doesn't exist, we also raise an error.[20]

```
      \int_compare:nNnTF \l_@@_final_int > \g_@@_line_int
         {\msg_error:nn {witharrows} {Too~few~lines~for~an~arrow}}
         {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-\int_use:N\l_@@_initial_int-l}
             { \msg_error:nn {witharrows} {A~PGF~node~doesn't~exist} }
             {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-\int_use:N\l_@@_final_int-l}
                 { \msg_error:nn {witharrows} {A~PGF~node~doesn't~exist} }
                 {\@@_draw_arrows_i:}}}
      \int_incr:N \l_@@_arrow_int
     }
   \group_end:
   }
```

The macro `\@@_draw_arrows_i:` is only for the lisibility of the code. This macro will draw the current arrow if the arrow is not impossible (that is to say if the Tikz node exists). The first `\group_begin:` is for the options of the arrow.

```
\cs_new:Nn \@@_draw_arrows_i:
    {\group_begin:
     \int_set:Nn \l_@@_previous_pos_arrows_int {-1}
     \int_set:Nn \l_@@_level_int 3
```

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.

```
      \prop_get:cnN {g_@@_arrow_\l_@@_prefix_str
                        _\int_use:N\l_@@_arrow_int _prop} {options} \l_tmpa_tl
      \exp_args:NNo \exp_args:No
          \@@_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`:

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

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

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

[20] This case occurs if the considered line has no ampersand. In fact, we raise an error if there is such a line in the `\halign`, but, nonetheless, we consider the case where the user goes on and we try to avoid other errors.

In case of option `i` (`\l_@@_pos_arrows_int` = 5), we have to compute the x-value of the arrow (which is vertical). The computed x-value is stored in `\g_@@_x_dim` (the same variable used when the option `group` or the option `groups` is used). This variable is global for technical reasons: we have to do assignments in a Tikz node.

```
\int_compare:nNnT \l_@@_pos_arrows_int = 5
      {
```

First, we calculate the initial value for `\g_@@_x_dim`. We use a Tikz command, but, in fact, nothing is drawn. We use this Tikz command only to read the abscissa of a Tikz node.

```
        \begin{tikzpicture} [@@_standard]
        \tikz@parse@node\pgfutil@firstofone (\int_use:N\l_@@_initial_int-l)
        \dim_gset:Nn \g_@@_x_dim \pgf@x
        \end{tikzpicture}
```

A global assignment is necessary because of Tikz.
Then, we will loop to determine the maximal length of the lines between the lines `\l_@@_initial_int` and `\l_@@_final_int`... but we have written a command dedicated to this work because it will also be used in `\@@_draw_arrows:`.

```
        \@@_update_x_value:nn \l_@@_initial_int \l_@@_final_int
      }
```

`\l_@@_initial_tl` contains the name of the Tikz node from which the arrow starts (in normal cases... because with the option `i`, `group` and `groups`, the point will perhaps have another x-value — but always the same y-value). Idem for `\l_@@_final_tl`.

```
\tl_set:Nx \l_@@_initial_tl
      {\int_use:N\l_@@_initial_int-\bool_if:NTF\l_@@_initial_r_bool rl .south}
\tl_set:Nx \l_@@_final_tl
      {\int_use:N\l_@@_final_int-\bool_if:NTF\l_@@_final_r_bool rl .north}
```

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

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

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

We have to compute the coordinates of the extremities of the arrow. We retrieve the coordinates in `\g_tmpa_tl` and `\g_tmpb_tl`. This extraction of the coordinates is necessary because we must give coordinates and not nodes (even anchored) to `\@@_draw_arrow:nnn` to have the `xshift` correct.

```
\int_compare:nNnTF \l_@@_pos_arrows_int < 5
   {\begin{tikzpicture} [@@_standard]
        \tikz@scan@one@point\pgfutil@firstofone(\l_@@_initial_tl)
        \tl_gset:Nx \g_tmpa_tl {\dim_use:N\pgf@x,\dim_use:N\pgf@y}
        \tikz@scan@one@point\pgfutil@firstofone(\l_@@_final_tl)
        \tl_gset:Nx \g_tmpb_tl {\dim_use:N\pgf@x,\dim_use:N\pgf@y}
    \end{tikzpicture}
}
```

If we use option `i` or `group` or `groups`, we use the abscissa specially computed in `\g_@@_x_dim`.

```
   {\begin{tikzpicture} [@@_standard]
        \tikz@scan@one@point\pgfutil@firstofone (\l_@@_initial_tl)
        \tl_gset:Nx \g_tmpa_tl {\dim_use:N \g_@@_x_dim , \dim_use:N \pgf@y}
        \tikz@scan@one@point\pgfutil@firstofone (\l_@@_final_tl)
        \tl_gset:Nx \g_tmpb_tl {\dim_use:N \g_@@_x_dim , \dim_use:N \pgf@y}
    \end{tikzpicture}}
```

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 `TikzCode`. 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*.[21]

```
\@@_draw_arrow:nno {\g_tmpa_tl} {\g_tmpb_tl} {\l_tmpa_tl}
```

We close the TeX group opened for the options given to `\Arrow[...]` (local level of the options).

```
\group_end: }
```

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

```
\cs_new_protected:Nn \@@_def_function_tmpa:n
    {\cs_set:Nn \@@_tmpa:nnn
          {\begin{tikzpicture}[@@_standard,every~path/.style = {WithArrows/arrow}]
              #1
           \end{tikzpicture}}}
```

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

```
\cs_new_protected:Nn \@@_draw_arrow:nnn
           {\exp_args:No \@@_def_function_tmpa:n \l_@@_tikz_code_tl
            \@@_tmpa:nnn {#1} {#2} {#3} }
\cs_generate_variant:Nn \@@_draw_arrow:nnn {nno}
```

The command `\@@_update_x_value:nn` will analyze the lines[22] between `#1` and `#2` in order to modify `\g_@@_x_dim` in consequence. More precisely, `\g_@@_x_dim` is increased if a line longer than the current value of `\g_@@_x_dim` is found. `\@@_update_x_value:nn` is used in `\@@_draw_arrows:` (for options `group` and `groups`) and in `\@@_draw_arrows:nn` (for option `i`).

```
\cs_new_protected:Nn \@@_update_x_value:nn
    {\int_step_inline:nnnn {#1} \c_one {#2}
       {\cs_if_exist:cT {pgf@sh@ns@wa-\l_@@_prefix_str-##1-l}
          {\begin{tikzpicture} [@@_standard]
           \tikz@scan@one@point\pgfutil@firstofone (##1-l)
           \dim_gset:Nn \g_@@_x_dim {\dim_max:nn \g_@@_x_dim \pgf@x}
           \end{tikzpicture} } } }
```

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 sens of the nested environments). This macro is fully expandable and, thus, can be used directly in the name of a Tikz node.

```
\cs_new:Npn \WithArrowsLastEnv {\int_use:N \g_@@_last_env_int}
```

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

[22] If a line has no ampersand, this line is ignored. In fact, we raise an error if there is a line without ampersand but, nonetheless, we consider the case where the user goes on and we try to avoid other errors.

9.10 The command Arrow in CodeAfter

The option `CodeAfter` is an option of the environment `{WithArrows}` (this option is only available at the environment level). In the option `CodeAfter`, 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/CodeAfter`.

```
\keys_define:nn {WithArrows/CodeAfter}
     {tikz     .code:n            = \tikzset {WithArrows/arrow/.append~style = {#1}} ,
      tikz     .value_required:n  = true,
      rr       .value_forbidden:n = true,
      rr       .code:n            = \@@_analyze_option_position:n 0 ,
      ll       .value_forbidden:n = true,
      ll       .code:n            = \@@_analyze_option_position:n 1 ,
      rl       .value_forbidden:n = true,
      rl       .code:n            = \@@_analyze_option_position:n 2 ,
      lr       .value_forbidden:n = true,
      lr       .code:n            = \@@_analyze_option_position:n 3 ,
      v        .value_forbidden:n = true,
      v        .code:n            = \@@_analyze_option_position:n 4 ,
      TikzCode .tl_set:N          = \l_@@_tikz_code_tl,
      TikzCode .value_required:n  = true,
      xoffset  .dim_set:N         = \l_@@_xoffset_dim,
      xoffset  .value_required:n  = true}


\NewDocumentCommand \@@_Arrow_code_after {O{} mmm O{}}
     {\int_set:Nn \l_@@_pos_arrows_int 1
      \int_set:Nn \l_@@_previous_pos_arrows_int {-1}
      \group_begin:
```

Even if `\Arrow` in `CodeAfter` has its own set of options (`WithArrows/CodeAfter`), we set the level of the options to 3 (as with the classical command `\Arrow`) because of the error messages.

```
           \int_set:Nn \l_@@_level_int 3
           \keys_set:nn {WithArrows/CodeAfter}
                        {#1,#5,tikz={xshift = \l_@@_xoffset_dim}}
           \bool_set_false:N \l_@@_initial_r_bool
           \bool_set_false:N \l_@@_final_r_bool
           \int_case:nn \l_@@_pos_arrows_int
                  {0 {\bool_set_true:N \l_@@_initial_r_bool
                      \bool_set_true:N \l_@@_final_r_bool}
                   2 {\bool_set_true:N \l_@@_initial_r_bool}
                   3 {\bool_set_true:N \l_@@_final_r_bool}}
```

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

```
           \cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-#2-l}
             {\msg_error:nnx {witharrows} {Wrong~line~specification~in~Arrow} {#2}}
             {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-#3-l}
                {\msg_error:nnx {witharrows} {Wrong~line~specification~in~Arrow} {#3}}
                {\int_compare:nNnTF \l_@@_pos_arrows_int = 4
                    {\begin{tikzpicture} [@@_standard]
                        \tikz@scan@one@point\pgfutil@firstofone(#2-l.south)
                        \dim_set_eq:NN \l_tmpa_dim \pgf@x
                        \dim_set_eq:NN \l_tmpb_dim \pgf@y
                        \tikz@scan@one@point\pgfutil@firstofone(#3-l.north)
                        \dim_set:Nn \l_tmpa_dim {\dim_max:nn \l_tmpa_dim \pgf@x}
                        \tl_gset:Nx \g_tmpa_tl {\dim_use:N \l_tmpa_dim , \dim_use:N \l_tmpb_dim}
                        \tl_gset:Nx \g_tmpb_tl {\dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y}
                     \end{tikzpicture} }
                    {\begin{tikzpicture} [@@_standard]
                        \tikz@scan@one@point\pgfutil@firstofone
                                            (#2-\bool_if:NTF\l_@@_initial_r_bool rl .south)
                        \tl_gset:Nx \g_tmpa_tl {\dim_use:N \pgf@x , \dim_use:N \pgf@y}
                        \tikz@scan@one@point\pgfutil@firstofone
                                            (#3-\bool_if:NTF\l_@@_final_r_bool rl .north)
```

```
                        \tl_gset:Nx \g_tmpb_tl {\dim_use:N \pgf@x , \dim_use:N \pgf@y}
                    \end{tikzpicture}}
                \@@_draw_arrow:nnn {\g_tmpa_tl} {\g_tmpb_tl} {#4} }}
    \group_end:
    }
```

9.11 MultiArrow

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

```
\cs_new_protected:Nn \@@_MultiArrow:nn
  {
```

The user of the command `\MultiArrow` (in `CodeAfter`) will be able to specify the list of lines with the same syntax as the loop `\foreach` of pgffor. 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 construct the list in `\g_tmpa_clist`.

```
    \foreach \x in {#1} {\cs_if_free:cTF {pgf@sh@ns@wa-\l_@@_prefix_str-\x-l}
                          {\msg_error:nnx {witharrows}
                                          {Wrong~line~specification~in~MultiArrow}
                                          {\x}}
                          {\clist_gput_right:Nx \g_tmpa_clist {\x}}}
```

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

```
    \int_compare:nNnTF {\clist_count:N \g_tmpa_clist} < 2
      {\msg_error:nn {witharrows} {Too~small~specification~for~MultiArrow}}
      {\clist_sort:Nn \g_tmpa_clist
                      {\int_compare:nNnTF {##1} > {##2}
                            {\sort_return_swapped:}
                            {\sort_return_same:}}
```

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

```
       \clist_pop:NN \g_tmpa_clist \l_tmpa_tl
```

We extract the maximum in `\l_tmpb_tl`. The remaining list (in `\g_tmpa_clist` will be sorted in decreasing order but never mind...).

```
       \clist_reverse:N \g_tmpa_clist
       \clist_pop:NN \g_tmpa_clist \l_tmpb_tl
```

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

```
       \exp_args:Nx \@@_MultiArrow_i:n {\g_tmpa_clist}
```

Now, we draw the rest of the structure.

```
       \begin{tikzpicture}[@@_standard,every~path/.style={WithArrows/arrow}]
         \draw [<->] ($(\l_tmpa_tl-r.south)+(\l_@@_xoffset_dim,0)$)
                     -- ++(5mm,0)
                     -- node {#2} ($(\l_tmpb_tl-r.south)+(\l_@@_xoffset_dim+5mm,0)$)
                     -- ($(\l_tmpb_tl-r.south)+(\l_@@_xoffset_dim,0)$)  ;
       \end{tikzpicture} } }

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

9.12 The error messages of the package

```
\msg_new:nnn {witharrows}
             {Third~column~in~an~environment~{WithArrows}}
             {By~default,~an~environment~\{WithArrows\}~can~only~have~two~columns.~
              Maybe~you~have~forgotten~a~\str_use:N \c_backslash_str
              \str_use:N \c_backslash_str.~If~you~really~want~more~than~two~columns,~
              you~should~use~the~option~"MoreColumns"~at~a~global~level~or~for~
              an~environment.~However,~you~can~go~one~for~this~time.}
\msg_new:nnn {witharrows}
             {Third~column~in~an~environment~{DispWithArrows}}
             {An~environment~\{DispWithArrows\}~or~\{DispWithArrows*\}~can~only~have~two~columns.~
              If~you~go~on,~you~may~have~an~incorrect~output.}
\msg_new:nnn {witharrows}
             {The~option~"jump"~must~be~non~negative}
             {You~can't~use~a~strictly~negative~value~for~the~option~"jump"~of~command~
              \token_to_str:N\Arrow.~ You~can~create~an~arrow~going~backwards~with~
              the~option~"<-"~of~Tikz.}
\msg_new:nnn {witharrows}
             {Too~few~lines~for~an~arrow}
             {An~arrow~specified~in~line~\int_use:N \l_@@_initial_int\ can't~be~drawn~
              because~it~arrives~after~the~last~line~of~the~environment~(remind~that~
              the~command~\token_to_str:N\Arrow\ must~be~in~the~*start*~line~
              of~the~arrow).~If~you~go~on,~this~arrow~will~be~ignored.}
\msg_new:nnn {witharrows}
             {{WithArrows}~used~outside~math~mode}
             {The~environment~\{WithArrows\}~should~be~used~only~in~math~mode.~
              Nevertheless,~you~can~go~on.}
\msg_new:nnn {witharrows}
             {{DispWithArrows}~used~in~math~mode}
             {The~environment~\{DispWithArrows\}~should~be~used~only~outside~math~mode.~
              If~you~go~on,~you~will~have~other~errors.}
\msg_new:nnn {witharrows}
             {Two~options~are~incompatible}
             {You~try~to~use~the~option~"\tl_use:N\l_keys_key_tl"~but~
             this~option~is~incompatible~or~redundant~with~the~option~"
             \int_case:nn\l_@@_previous_pos_arrows_int
                 {0 {rr}
                  1 {ll}
                  2 {rl}
                  3 {lr}
                  4 {v}
                  5 {i}
                  6 {group}
                  7 {groups}}"~
             previously~set~in~the~same~
             \int_case:nn\l_@@_level_int
                  {1 {command~\token_to_str:N\WithArrowsOptions}
                   2 {declaration~of~options~of~the~environment~\{WithArrows\}}
                   3 {command~\token_to_str:N\Arrow}}.~
             If~you~go~on,~I~will~overwrite~the~first~option.}
\msg_new:nnnn {witharrows}
              {All~lines~must~have~an~ampersand}
              {All~lines~of~an~environment~\{WithArrows\}~should~have~an~second~column~
               (because~the~nodes~are~created~in~the~second~column).~However,~you~can~go~but~you~will~
               have~an~error~if~one~of~your~arrows~needs~an~PGF~node~absent~by~lack~of~ampersand.~
               If~you~don't~want~to~see~this~message~again,~you~can~use~the~option~
               AllowLineWithoutAmpersand.}
              {Moreover, the~ampersand~can~be~implicit~
               (e.g.~if~you~use~\token_to_str:N\MoveEqLeft\ of~mathtools).}
\msg_new:nnn {witharrows}
```

```
773             {Option~incompatible~with~"group(s)"}
774             {You~try~to~use~the~option~"\tl_use:N\l_keys_key_tl"~while~
775              you~are~using~the~option~"
776              \int_compare:nNnTF \l_@@_pos_arrows_int = 5
777                   {group}
778                   {groups}".~
779              It's~incompatible.~You~can~go~on~ignoring~this~option~
780              "\tl_use:N\l_keys_key_tl"~but~you~should~correct~your~code.}
781 \msg_new:nnn {witharrows}
782             {Option~will~be~ignored}
783             {The~option~"\tl_use:N\l_keys_key_tl"~can't~be~used~here.~
784              If~you~go~on,~it~will~be~ignored.}
785 \msg_new:nnn {witharrows}
786             {Option~unknown}
787             {The~option~"\tl_use:N\l_keys_key_tl"~is~unknown~or~
788              meaningless~in~the~context.~If~you~go~on,~it~will~be~ignored.}
789 \msg_new:nnn {witharrows}
790             {Arrow~in~first~column}
791             {You~should~not~use~the~command~\token_to_str:N\Arrow\
792              \tl_if_eq:NNF \l_@@_CommandName_tl \l_tmpa_tl
793                    {(renamed~in~\str_use:N \c_backslash_str
794                     \tl_use:N \l_@@_CommandName_tl)~}
795              ~in~the~first~column~but~only~in~the~second~column.~
796              This~is~a~restriction~of~the~version~1.3~of~the~
797              package~witharrows~(in~the~aim~of~developping~further~
798              ~a~new~functionality~with~\token_to_str:N\Arrow\ in~the~
799              first~column).\\
800              However~you~can~go~on~for~this~time.}
801 \msg_new:nnn {witharrows}
802             {Wrong~line~specification~in~Arrow}
803             {The~specification~of~line~"#1"~you~use~in~\token_to_str:N\Arrow\
804              ~doesn't~exist.\\
805              If~you~go~on,~the~arrow~will~be~ignored.}
806 \msg_new:nnn {witharrows}
807             {Wrong~line~specification~in~MultiArrow}
808             {The~specification~of~line~"#1"~doesn't~exist.\\
809              If~you~go~on,~it~will~be~ignored~for~\token_to_str:N \MultiArrow.}
810 \msg_new:nnn {witharrows}
811             {Too~small~specification~for~MultiArrow}
812             {The~specification~of~lines~you~gave~to~\token_to_str:N \MultiArrow\
813              is~too~small:~we~need~at~least~two~lines.~If~you~go~on,~the~
814              command~\token_to_str:N\MultiArrow\ ~will~be~ignored.}
815 \msg_new:nnn {witharrows}
816             {A~PGF~node~doesn't~exist}
817             {A~PGF~node~necessary~to~draw~an~arrow~doesn't~exist~
818              because~you~didn't~put~an~ampersand~in~the~corresponding~line.~
819              If~you~go~on,~the~arrow~will~be~ignored.}
820 \msg_new:nnn {witharrows}
821             {tag*~without~amsmath}
822             {We~can't~use~\token_to_str:N\tag*~because~you~haven't~load~amsmath~
823              (or~mathtools).~If~you~go~on,~the~command~\token_to_str:N\tag\
824              will~be~used~instead.}
825 \msg_new:nnn {witharrows}
826             {Command~not~allowed~in~{DispWithArrows}}
827             {The~command~\token_to_str:N #1
828              is~not~allowed~in~the~first~column~of~\{DispWithArrows\}~but~
829              only~in~the~second~column~(and,~of~course,~in~the~
830              environments~of~amsmath).~If~you~go~on,~this~command~will~be~ignored.}
831 \msg_new:nnn {witharrows}
```

```
832             {Command~not~allowed~in~{WithArrows}}
833             {The~command~\token_to_str:N #1
834              is~not~allowed~in~\{WithArrows\}~but~is~allowed~in~the~second~
835              column~of~\{DispWithArrows\}~(and,~of~course,~in~the~
836              environments~of~amsmath).~If~you~go~on,~this~command~will~be~ignored.}
```

10 History

10.1 Changes between versions 1.0 and 1.1

Option for the command \\ and option `interline`
Compatibility with `\usetikzlibrary{babel}`
Possibility of nested environments `{WithArrows}`
Better error messages
Creation of a DTX file

10.2 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*)
Better error messages

10.3 Changes between versions 1.2 and 1.3

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

10.4 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.

10.5 Changes between versions 1.4 and 1.5

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

10.6 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}`.

10.6.1 Changes between 1.6 and 1.6.1

Correction of a bug that leads to incompatibility with `\usetikzlibrary{babel}`.

10.7 Changes between 1.6.1 and 1.7

New environment `{DispWithArrows}`.