The package witharrows[*]

F. Pantigny
`fpantigny@wanadoo.fr`

November 6, 2017

Abstract

The LaTeX package witharrows gives 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, footnote[1] and tikz. The following Tikz libraries are also required : calc, arrows.meta and bending.

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

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

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

The arrow has been drawn with the command `\Arrow` on the ligne from which it starts. The command `\Arrow` can be used anywhere on the line but the best way is to put it at the end.

1 Options for the shape of the arrows

The commande `\Arrow` has several options. These options can be put between square brackets, before, or after the mandatory argument.
The option `jump` gives the number of lines the arrow must jump (the default value is, of course, 1) [2]

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

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

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

[*]This document corresponds to the version 1.1 of witharrows, at the date of 2017/11/06.
[1]The package footnote is used to extract the notes from the environments `{WithArrows}`.
[2]It's not possible to give a non-positive value to `jump`. See below the way to draw an arrow which goes backwards.

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

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

The option **`xoffset`** shift 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.

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

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

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 blue thick arrow.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 a 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.

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

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

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

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.[3]

[3]They applies also to the nested environments `{WithArrows}` with the notable exception of `interline`.

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

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

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` :

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

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

The same example with the option `displaystyle` :

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

Almost all the options can also be set at the document level with the command `\WithArrowsOptions`. In this case, the scope of the declarations is the current TeX group (these declarations are "semi-global"). For example, if we want all the environments `{WithArrows}` composed in `\displaystyle` with blue arrows, we can write `\WithArrowsOptions{displaystyle,tikz=blue}`.[4]

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

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

[4]It's also possible to give the options directly when loading the package, *i.e.* with the command `\usepackage` in the preamble.

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.

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

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

It's possible to use directly the nodes created by {WithArrows} with explicit Tikz instructions (in order, for example, to draw something that can't be drawn with the command \Arrow). That's why a style for the tips of the arrows has be created : TipsOfWithArrows. By using this style, we will have homogeneous tips for the arrows of the document.
Therefore, if we want to modify the tips of the arrows of {WithArrows}, we have to modify the style TipsOfWithArrows.

```
\tikzset{TipsOfWithArrows/.style= { > = {Latex[scale=1.2,bend]}} }
```

The names of the Tikz nodes created by witharrows in the whole document are explained below.

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.[5]

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

The nodes of the left are at the end of each line of text. These nodes will be called *left nodes*. The nodes of the right side are aligned vertically on the right side of the array. These nodes will be called *right nodes*.
The names of the Tikz nodes created by {WithArrows} are wa-n-l and wa-n-r where n is the number of the line. It's possible to refer to these Tikz nodes after the environment (one should use

[5]The option shownodes can be used to materialize the nodes.

the options `remember picture` and `overlay` and also `TipsOfWithArrows` and `->` in order to have the same arrowheads).
By default, the arrows use the right nodes. We will say that they are in `rr` mode (*r* for *right*). These arrows are `vertical` (we will say that an arrow is *vertical* when its two ends have the same abscissa).

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

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

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

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

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

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

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

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

If desired, the option `group` can be given to the command `\WithArrowsOptions` so that it will become the default value.

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 lines 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).

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

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

In the environments of amsmath (or mathtools), the spacing between lines 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]

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

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

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

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

[6]In fact, it's possible to use the package witharrows without the environment amsmath.
[7]It's also possible to change `\jot` with the environment `{spreadlines}` of mathtools.

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

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

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

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

Like the environment `{aligned}`, `{WithArrows}` has an option of placement which can assume the values `t`, `c` or `b`. However, the 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.

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

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

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

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

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

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

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

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

4 Examples

4.1 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.

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

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

4.2 MoveEqLeft

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

```
$\begin{WithArrows}[jot=2mm]
\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}$
```

$$\begin{aligned} & \arccos(x) = \arcsin \tfrac45 + \arcsin \tfrac5{13} && \\ & && \big\downarrow \textit{ because both are in } [-\tfrac{\pi}{2},\tfrac{\pi}{2}] \\ & \quad \Leftrightarrow x = \sin\bigl(\arcsin\tfrac45 + \arcsin\tfrac5{13}\bigr) && \\ & \quad \Leftrightarrow x = \tfrac45\cos\arcsin\tfrac5{13} + \tfrac5{13}\cos\arcsin\tfrac45 && \\ & && \big\downarrow\ \forall x \in [-1,1], \cos(\arcsin x) = \sqrt{1-x^2} \\ & \quad \Leftrightarrow x = \tfrac45\sqrt{1-\bigl(\tfrac5{13}\bigr)^2} + \tfrac5{13}\sqrt{1-\bigl(\tfrac45\bigr)^2} && \end{aligned}$$

[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}`.

4.3 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 the notable exception of `interline`).

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

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

4.4 A loop flow

Here is an example with a loop flow.

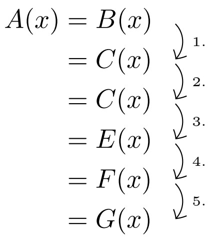
```
$\begin{WithArrows}[tikz={font={\tiny}}]
a.\;& f \text{ est continuous on } E
\Arrow{(1)}\Arrow[tikz=<-,jump=4,xoffset=1cm]{(5)}\\
b.\;& f \text{ est continuous in } 0
\Arrow{(2)}\\
c.\;& f \text{ is bounded on the unit sphere}
\Arrow{(3)}\\
d.\;& \exists K > 0\quad \forall x \in E\quad \|f(x)\| \le K \|x\|
\Arrow{(4)}\\
e.\;& f \text{ is lipschitzian}
\end{WithArrows}$
```

$$\begin{aligned} a.\;& f \text{ est continuous on } E && \Big\downarrow \ {\scriptstyle (1)} \qquad \Big\uparrow \\ b.\;& f \text{ est continuous in } 0 && \Big\downarrow \ {\scriptstyle (2)} \\ c.\;& f \text{ is bounded on the unit sphere} && \Big\downarrow \ {\scriptstyle (3)} \qquad \Big) \ {\scriptstyle (5)} \\ d.\;& \exists K>0 \quad \forall x\in E \quad \|f(x)\| \le K\|x\| && \Big\downarrow \ {\scriptstyle (4)} \\ e.\;& f \text{ is lipschitzian} \end{aligned}$$

4.5 Automatic numerotation

The option `font` of Tikz contains in fact a list of tokens which will be placed at the beginning of the text.
These tokens can be true commands for a change of font (like `\bfseries` or `\sffamily`) but can also be, in fact, any TeX command.
In the following example, the argument of `font` is the token list `\tiny\counter` where `\counter` is a command which increment a counter previously defined and display its new value. Thus, the arrows are automatically numbered.

```
\newcounter{MyCounter}
\newcommand{\counter}{\stepcounter{MyCounter}\theMyCounter.}
$\begin{WithArrows}[tikz={font={\tiny\counter}}]
A(x)
& = B(x) \Arrow{} \\
& = C(x) \Arrow{} \\
& = C(x) \Arrow{} \\
& = E(x) \Arrow{} \\
& = F(x) \Arrow{} \\
& = G(x)
\end{WithArrows}$
```

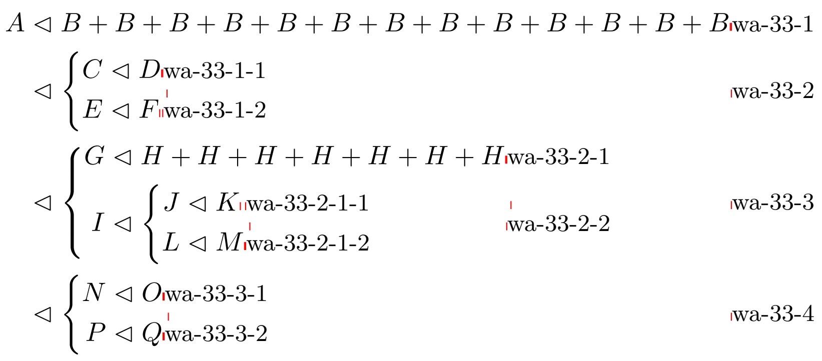


5 An technical remark about the names of the nodes

Environments {WithArrows} can be nested, and, therefore, we have a "nesting tree" for the environments {WithArrows} of the whole document. This nesting tree is used to give a unique name to each node in the document.

The Tikz name of a node created by witharrows is prefixed by `wa-`. Then, we have a list of numbers with give the position in the nesting tree and the number of the line 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]



The command `\WithArrowsLastEnv` gives the number of the last environment of level 0. For example, we can draw an arrow from the node `wa-33-1` to the node `wa-33-2-1` with the following Tikz command.[10]

```
\begin{tikzpicture}[remember picture,overlay,->,TipsOfWithArrows]
\draw (wa-\WithArrowsLastEnv-1-r) to (wa-\WithArrowsLastEnv-2-1-r) ;
\end{tikzpicture}
```

[9]There is an option `shownodenames` to show the names of these nodes.
[10]The command `\WithArrowsLastEnv` is *fully expandable* and thus, can be used directly in the name of a Tikz node.

6 Implementation

6.1 Some extensions are loaded

This package is written in expl3.

```
\RequirePackage{expl3}
```

The package xparse will be used to define the environment `{WithArrows}` and the document-level commands (`\Arrow`, `\WithArrowsOptions` ans `\WithArrowsLastEnv`).

```
\RequirePackage{xparse}
```

The arrows are drawn with Tikz and that's why tikz is required with some libraries.

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

The package footnote will be used to extract footnotes.

```
\RequirePackage{footnote}
```

6.2 Some technical definitions

We define a Tikz style `wa_node_style` for the nodes that will be created in the `\halign`.

```
\tikzstyle{@@_node_style}=[rectangle,
                           inner~sep = 0 pt,
                           minimum~height = 3 pt,
                           minimum~width = 0pt,
                           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).

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{TipsOfWithArrows/.style= { > = {Straight~Barb[scale=1.2,bend]}} }
```

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:Npn \spread@equation{\openup\jot
                                  \cs_set_eq:NN \spread@equation \prg_do_nothing}}
```

6.3 Variables

The following sequence is the position of the last environment `{WithArrows}` in the tree of the imbricated 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 of the `\halign`.

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

If the following flag is raised, then the user can use more than two columns.

```
\bool_new:N \l_@@_MoreColumns_bool
```

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_int
```

The following flag indicates if the beginning of the arrow use a `r` option (if not, it's a `l` option).

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

The following flag indicates if the end of the arrow use a `r` option (if not, it's a `l` option).

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

The following flag indicates if the arrow use a `i` option (in this case, the options `l` and `r` for the ends are meaningless).

```
\bool_new:N \l_@@_i_bool
```

The following dimension is the value of the translation of the whole arrow to the right (of course, it's a dimension and not a skip).

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

If the following flag is raised, the nodes will be drawn in red (useful for debugging).

```
\bool_new:N \l_@@_shownodes_bool
```

If the following flag is raised, the name of the "right nodes" will be shown in the document (useful for debugging).

```
\bool_new:N \l_@@_shownodenames_bool
```

If the following flag is raised, the elements of the `\halign` will be composed with `\displaystyle` :

```
\bool_new:N \l_@@_displaystyle_bool
```

The following token list variable will contains the `Tikz` options used to draw the arrows.

```
\tl_clear_new:N \l_@@_options_tikz_tl
```

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

The following flag will be raised when a key controling position of the arrow has been set in the same list of keys (this is useful in order to raise an error if two incompatibles keys are specified at the same level).

```
\bool_new:N \l_@@_position_key_already_set_bool
```

This flag is set to `false` each time we have to process a list of options.

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.[11]

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_@@_stack_counter_arrows_seq
\int_new:N \g_@@_arrow_int
\seq_new:N \g_@@_stack_counter_lines_seq
\int_new:N \g_@@_line_int
\seq_new:N \g_@@_stack_counter_lines_bis_seq
\int_new:N \g_@@_line_bis_int
```

6.4 The definition of the options

There are three levels where options can be set :

- with `\WithArrowsOptions{...}` : this level will be called *global* level;
- with `\begin{WithArrows}[...]` : this level will be called *env* level;
- with `\Arrow[...]` : this level will be called *local* level.

That's why there is three groups of keys named `global`, `env` and `local`.

Before each `\keys_set_groups:nnn`, we execute a `\keys_set_filter:nnn` with group `secondary`. Thus, an unknown key will raise an error. Furthermore, the *primary* keys (those which are not in the group `secondary`) will be set first : it's useful to raise an error if, for example, keys `i` and `group` are set at the same level (which is incoherent).

```
\keys_define:nn {WithArrows}
```

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 this key is the only key of a dedicated group also named `jump`.

```
    { jump  .groups:n   = {jump,secondary},
      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 options `t`, `c` and `b` indicate if we will create a `\vtop`, a `\vcenter` of a `\vbox`. This information is stored in the variable `\l_@@_pos_int`.

```
      t   .groups:n          = {global,env,secondary},
      t   .code:n            = {\int_set:Nn \l_@@_pos_int 0},
      t   .value_forbidden:n = true,
      c   .groups:n          = {global,env,secondary},
      c   .code:n            = {\int_set:Nn \l_@@_pos_int 1},
      c   .value_forbidden:n = true,
      b   .groups:n          = {global,env,secondary},
      b   .code:n            = {\int_set:Nn \l_@@_pos_int 2},
      b   .value_forbidden:n = true,
```

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 .groups:n          = {global,env,secondary},
      MoreColumns .bool_set:N        = \l_@@_MoreColumns_bool,
      MoreColumns .default:n         = true,
      MoreColumns .value_forbidden:n = true,
```

[11]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).

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

```
      CommandName .groups:n         = {global,env,secondary},
      CommandName .tl_set:N         = \l_@@_CommandName_tl,
      CommandName .initial:n        = {Arrow},
      CommandName .value_required:n = true,
```

With the option `displaystyle`, all the elements of the environment will be composed in `\displaystyle`.

```
      displaystyle .groups:n          = {global,env,secondary},
      displaystyle .bool_set:N        = \l_@@_displaystyle_bool,
```

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

```
      shownodes .groups:n          = {global,env,secondary},
      shownodes .bool_set:N        = \l_@@_shownodes_bool,
      shownodes .default:n         = true,
```

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

```
      shownodenames .groups:n          = {global,env,secondary},
      shownodenames .bool_set:N        = \l_@@_shownodenames_bool,
      shownodenames .default:n         = true,
```

The option `jot` can be used to change the value of the LaTeX parameter `\jot`. If we put this option in the `global` group, the use of this option in `\WithArrowsOptions` will change `\jot` for the whole document (at least the current TeX group) and not only for the `{WithArrows}` environments. This is certainly not what the user wants. That's why the option `jot` is not in the group `global`. It's interesting to note that `\jot` is a dimension and not a skip (=glue).

```
      jot       .groups:n          = {env,secondary},
      jot       .dim_set:N         = \jot,
      jot       .value_required:n  = true,
```

The option `interline` gives the vertical skip (=glue) inserted between two lines (independently of `\jot`). By design, this option has a particular behaviour : it applies only to an environment and doesn't apply to the nested environments.

```
      interline   .groups:n         = {env,secondary},
      interline   .skip_set:N       = \l_@@_interline_skip,
      interline   .initial:n        = \c_zero_skip,
      interline   .value_required:n = true,
```

The option `xoffset` change the x-offset of the arrows (towards the right). It's a dimension and not a skip.

```
      xoffset  .groups:n          = {global,env,local,secondary},
      xoffset  .dim_set:N         = \l_@@_xoffset_dim,
      xoffset  .value_required:n  = true,
```

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).

```
      tikz     .groups:n          = {global,env,local,secondary},
      tikz     .code:n            = {\tl_put_right:Nn \l_@@_options_tikz_tl {,#1}},
      tikz     .value_required:n  = true,
```

The following options can be used to specify the position of the arrows. We control that a previous specification of position has not been set at the same level of option. This is done with the help of the flag `\l_@@_position_key_already_set_bool`.

```
      ll       .groups:n          = {global,env,local,secondary},
      ll       .value_forbidden:n = true,
      ll       .default:n         = true,
      ll       .code:n            = {\bool_if:NT \l_@@_position_key_already_set_bool
```

```
                                    {\msg_error:nn {witharrows}
                                                   {Two~options~are~incompatible}}
                                  \bool_if:NT \l_@@_group_bool
                                    {\msg_error:nn {witharrows}
                                                   {Option~incompatible~with~group}}
                                  \bool_set_true:N \l_@@_position_key_already_set_bool
                                  \bool_set_false:N \l_@@_initial_r_bool
                                  \bool_set_false:N \l_@@_final_r_bool
                                  \bool_set_false:N \l_@@_i_bool },
         lr       .groups:n          = {global,env,local,secondary},
         lr       .value_forbidden:n = true,
         lr       .default:n         = true,
         lr       .code:n            = {\bool_if:NT \l_@@_position_key_already_set_bool
                                    {\msg_error:nn {witharrows}
                                                   {Two~options~are~incompatible}}
                                  \bool_if:NT \l_@@_group_bool
                                    {\msg_error:nn {witharrows}
                                                   {Option~incompatible~with~group}}
                                  \bool_set_true:N \l_@@_position_key_already_set_bool
                                  \bool_set_false:N \l_@@_initial_r_bool
                                  \bool_set_true:N  \l_@@_final_r_bool
                                  \bool_set_false:N \l_@@_i_bool},
         rl       .groups:n          = {global,env,local,secondary},
         rl       .value_forbidden:n = true,
         rl       .default:n         = true,
         rl       .code:n            = {\bool_if:NT \l_@@_position_key_already_set_bool
                                    {\msg_error:nn {witharrows}
                                                   {Two~options~are~incompatible}}
                                  \bool_if:NT \l_@@_group_bool
                                    {\msg_error:nn {witharrows}
                                                   {Option~incompatible~with~group}}
                                  \bool_set_true:N \l_@@_position_key_already_set_bool
                                  \bool_set_true:N  \l_@@_initial_r_bool
                                  \bool_set_false:N \l_@@_final_r_bool
                                  \bool_set_false:N \l_@@_i_bool},
         rr       .groups:n          = {global,env,local,secondary},
         rr       .value_forbidden:n = true,
         rr       .default:n         = true,
         rr       .code:n            = {\bool_if:NT \l_@@_position_key_already_set_bool
                                    {\msg_error:nn {witharrows}
                                                   {Two~options~are~incompatible}}
                                  \bool_if:NT \l_@@_group_bool
                                    {\msg_error:nn {witharrows}
                                                   {Option~incompatible~with~group}}
                                  \bool_set_true:N \l_@@_position_key_already_set_bool
                                  \bool_set_true:N  \l_@@_initial_r_bool
                                  \bool_set_true:N  \l_@@_final_r_bool
                                  \bool_set_false:N  \l_@@_i_bool},
```

With option `i` (for *intermediate*), the arrow will be drawn on the leftmost position compatible with all the lines between the starting line and the final line of the arrow.

```
         i        .groups:n          = {global,env,local,secondary},
         i        .code:n            = {\bool_if:NT \l_@@_position_key_already_set_bool
                                    {\msg_error:nn {witharrows}
                                                   {Two~options~are~incompatible}}
                                  \bool_if:NT \l_@@_group_bool
                                    {\msg_error:nn {witharrows}
                                                   {Option~incompatible~with~group}}
                                  \bool_set_true:N \l_@@_position_key_already_set_bool
                                  \bool_set_true:N \l_@@_i_bool},
         i        .value_forbidden:n = true,
         i        .default:n         = true,
```

With the option `group`, *all* the arrows of the environment are grouped on a same vertical line and at a leftmost position. Of course, this option is not available at the local level.
This key is not in the `secondary` group. Thus, it will be set first during the `\keys_set_filter:nnn` and an incompatibility like, for example, `[i,group]` will be detected (with a non fatal error).

```
      group        .groups:n          = {global,env},
      group        .bool_set:N        = \l_@@_group_bool,
      group        .value_forbidden:n = true
}
```

We process the options when the package is loaded (with `\usepackage`) but we recommend to use `\WithArrowOptions` instead.

```
\ProcessKeysOptions {WithArrows}
```

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

```
\NewDocumentCommand \WithArrowsOptions {m}
    {\bool_set_false:N \l_@@_position_key_already_set_bool
     \keys_set_filter:nnn {WithArrows} {secondary} {#1}
     \keys_set_groups:nnn {WithArrows} {global} {#1}}
```

6.5 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{}}
         {\tl_if_eq:noF {WithArrows} {\@currenvir}
                  {\msg_error:nn {witharrows} {Arrow~used~outside~{WithArrows}~environment}}
```

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 group `jump` with only one key (the key `jump`).

```
          \int_zero_new:N \l_@@_jump_int
          \int_set:Nn \l_@@_jump_int 1
          \keys_set_groups:nnn {WithArrows} {jump} {#1,#3}
```

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

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_tl _\int_use:N\g_@@_arrow_int _prop}
             \prop_gset_eq:cN
                   {g_@@_arrow_\l_@@_prefix_tl _\int_use:N\g_@@_arrow_int _prop}
                   \l_tmpa_prop
             }
```

6.6 The environnement {WithArrows}

The environment `{WithArrows}` starts with the initialisation of the three counters `\g_@@_arrow_int`, `\g_@@_line_int` dans `\g_@@_line_bis_int`. However, we have to save their previous values with the three stacks created for this end.

```
\NewDocumentEnvironment {WithArrows} {O{}}
          { \seq_gput_right:NV \g_@@_stack_counter_arrows_seq \g_@@_arrow_int
            \int_gzero:N \g_@@_arrow_int
            \seq_gput_right:NV \g_@@_stack_counter_lines_seq \g_@@_line_int
            \int_gzero:N \g_@@_line_int
            \seq_gput_right:NV \g_@@_stack_counter_lines_bis_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
            \tl_clear_new:N \l_@@_prefix_tl
            \tl_set:Nx \l_@@_prefix_tl {\seq_use:Nnnn \l_tmpa_seq {-} {-} {-}}
```

The environment `{WithArrows}` must be used in math mode.

```
            \reverse_if:N \if_mode_math:
                              \msg_error:nn {witharrows} {{WithArrows}~used~outside~math~mode}
                          \fi
```

We extract the footnotes of the environments `{WithArrows}` with the pair `\savenotes-\spewnotes` of the extension `footnote` (of course, we have put a `\spewnotes` at the end of the environment).

```
            \savenotes
```

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

```
            \cs_set_eq:NN \\ \@@_cr:
            \mathsurround = \c_zero_dim
```

These three 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 flag `\l_@@_position_key_already_set_bool` is set to false before the treatment of the options of the environment. It will be raised if a key indicating the position of the arrows is found. Thus, we can detect incompatible keys.

```
            \bool_set_false:N \l_@@_position_key_already_set_bool
```

The value corresponding to the key `interline` is put to zero before the treatment of the options of the environment.[12]

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

The following code in order to raise an error if a key is unknown.

```
            \keys_set_filter:nnn {WithArrows} {secondary} {#1}
```

We process the options given at the *env* level, that is to say in the option of the `{WithArrows}` environment.

```
            \keys_set_groups:nnn {WithArrows} {env} {#1}
```

If the user has given a value for the option `CommandName` (at the global or at the *env* 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
```

The environment begins with a `\vtop`, a `\vcenter` or a `\vbox`[13] depending of the value of `\l_@@_pos_int` (usually fixed by the options `t`, `c` or `b`). The environment `{WithArrows}` must be used in math mode[14] and therefore, we can use `\vcenter`.

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

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

```
            \spread@equation
```

We begin the `\halign` and the preamble.

```
            \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
            \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).

```
            \tikz[remember~picture]
                  \node [@@_node_style]
                  (wa-\tl_use:N\l_@@_prefix_tl-\int_use:N\g_@@_line_int-l) {} ;
            \hfil
```

[12] It's recalled that, by design, the option `interline` of a environment doesn't apply in the nested environments.
[13] Notice that the use of `\vtop` seems color-safe here...
[14] An error is raised if the environment is used outside math mode.

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).

```
            \tikz[remember~picture,label~position=right]
                  \node [@@_node_style]
                  (wa-\tl_use:N\l_@@_prefix_tl-\int_use:N\g_@@_line_int-r) {} ;
            \bool_if:NT \l_@@_shownodenames_bool
                 {\hbox_overlap_right:n {\small wa-\tl_use:N\l_@@_prefix_tl
                                                    -\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`.

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

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 with `\@@_draw_arrows:` (a special macro has been written for lisibility of the code).

```
             \int_compare:nNnT \g_@@_arrow_int > 0
                     \@@_draw_arrows:
```

We use `\spewnotes` of `footnote` to spew the footnotes of the environment (a `\savenotes` has been put at the beginning of the environment).

```
            \spewnotes
```

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 fonction `\WithArrowsLastEnv`.

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

Finally, we restore the previous values of the 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_@@_stack_counter_arrows_seq {\l_tmpa_tl}
            \int_gset:Nn \g_@@_arrow_int {\l_tmpa_tl}
            \seq_gpop_right:NN \g_@@_stack_counter_lines_seq \l_tmpa_tl
            \int_gset:Nn \g_@@_line_int {\l_tmpa_tl}
            \seq_gpop_right:NN \g_@@_stack_counter_lines_bis_seq \l_tmpa_tl
            \int_gset:Nn \g_@@_line_bis_int {\l_tmpa_tl}
            }
```

That's the end of the environment `{WithArrows}`.

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

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

6.7 We draw the arrows

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

```
\cs_generate_variant:Nn \keys_set_groups:nnn {nno}
\cs_new_protected:Nn \keys_set_groups_WithArrows_local:
                     {\keys_set_groups:nno {WithArrows} {local}}
```

```
\cs_generate_variant:Nn \keys_set_filter:nnn {nno}
\cs_new_protected:Nn \keys_set_filter_WithArrows_secondary:
          {\keys_set_filter:nno {WithArrows} {secondary}}
```

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

If the option `group` is used (for the whole document — with `\WithArrowsOptions` — or for the current environment), we have to compute the x-value common to all the arrows. This work is done by the command `\@@_x_computation_for_option_group:` and the computed x-value is store in `\g_@@_x_dim` (we use a global variable for technical reasons).

```
   \bool_if:NT \l_@@_group_bool
        \@@_x_computation_for_option_group:
```

We begin a loop over the arrows of the environment. The variable `\l_@@_arrow_int` (local in the environment `{WithArrows}`) will be used as index for the loop. The number of arrows in the environment is `\g_@@_arrow_int`. This variable was a counter incremented when an arrows is encountered during the construction of the `\halign`. After the end of the `\halign`, `g_@@_arrow_int` is the number of arrows in the environment.

```
   \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.

If the arrow ends after the last line of the environment, we raise an error.

```
    \prop_get:cnN {g_@@_arrow_\l_@@_prefix_tl _\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_tl _\int_use:N\l_@@_arrow_int _prop}
                  {final} \l_tmpa_tl
    \int_set:Nn \l_@@_final_int {\l_tmpa_tl}
    \int_compare:nNnT \l_@@_final_int > \g_@@_line_int
           {\msg_error:nn {witharrows} {Too~few~lines~for~an~arrow}}
```

We begin an TeX group which will contains the options local to the current arrow. We declare “undefined” the key `group` because it’s not possible to have a `group` key for an individual arrow.

```
        \group_begin:
        \keys_define:nn {WithArrows} {group .undefine:}
        \bool_set_false:N \l_@@_position_key_already_set_bool
```

We process the options of the current arrow. The third argument de `\keys_set_groups:nnn` 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_tl
                         _\int_use:N\l_@@_arrow_int _prop} {options} \l_tmpa_tl
        \exp_args:NNo \exp_args:No \keys_set_filter_WithArrows_secondary: {\l_tmpa_tl}
        \exp_args:NNo \exp_args:No \keys_set_groups_WithArrows_local: {\l_tmpa_tl}
```

In case of option `i`, we have to compute the x-value of the arrow (which is vertical). This work is done by the command `\@@_x_computation_for_option_i:` and the computed x-value is stored in `g_@@_x_dim` (the same variable used when the option `group` is used).

```
        \bool_if:NT \l_@@_i_bool
            \@@_x_computation_for_option_i:
```

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

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

We use “`. 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).

We can now draw the arrow in a `{tikzpicture}` :

```
        \begin{tikzpicture}[remember~picture,
                            overlay,
                            align=left,
                            auto=left,
                            font = {\small\itshape},
                            TipsOfWithArrows,
                            ->,
                            looseness=1,
                            bend~left=45]
```

Of course, the arrow is drawn with the command `\draw` of Tikz. The syntax for this command is :

`\draw` (x_1, y_1) `to node` $(name)$ `{`*contents*`}` (x_2, y_2)

The surprising aspect of this syntax is the position of *contents* which is the label of the arrow.
We give a name to the node (*name* in the previous syntax) but, in fact, we don't use it in the extension `witharrows`.
`\p1` and `\p2` are the two ends of the arrow (in fact, if the option `i` or the option `group` is used, it's not exactly the two ends of the arrow because, in this case, this abscissa used is the value previously calculated in `g_@@_x_dim`).
The ability to define `\p1` and `\p2` is given by the library `calc` of Tikz. When `\p1` and `\p2` are defined, the x-value and y-value of these two points can be read in `\x1`, `\x2`, `\y1` and `\y2`. This is the way to have the coordinates of a node defined in Tikz.

```
              \prop_get:cnN {g_@@_arrow_\l_@@_prefix_tl _\int_use:N\l_@@_arrow_int _prop}
                            {label} \l_tmpa_tl
              \draw \exp_after:wN [\l_@@_options_tikz_tl]
                    let \p1 = (\tl_use:N \l_@@_initial_tl),
                        \p2 = (\tl_use:N \l_@@_final_tl) in
                      (\bool_if:nTF {\l_@@_group_bool || \l_@@_i_bool}
                        {\dim_use:N \g_@@_x_dim + \dim_use:N \l_@@_xoffset_dim, \y1}
                        {\x1 + \dim_use:N \l_@@_xoffset_dim, \y1} )
```

There are two ways to give the content of the node : the classical way, with curly braces, and the option "node contents". However, both are not strictly equivalent : when `\usetikzlibrary{babel}` is used, the tokens of the contents are rescanned in the first way but not in the second. We don't want the tokens to be rescanned (because this would lead to an error due of the characters `_` and `:` of the expl3 syntax) and that's why we use the second method. [15]

```
                      to node [node~contents = {\tl_use:N \l_tmpa_tl}] {}
                      (\bool_if:nTF {\l_@@_group_bool || \l_@@_i_bool}
                        {\dim_use:N \g_@@_x_dim + \dim_use:N \l_@@_xoffset_dim, \y2}
                        {\x2 + \dim_use:N \l_@@_xoffset_dim, \y2} ) ;
              \end{tikzpicture}
```

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

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

We want to compute the x-value for the current arrow which has option `i` (and therefore is vertical). This value will be computed in `\g_@@_x_dim` (which is global for technical reasons : we have to do assignments in a Tikz command).

```
\cs_new_protected:Nn \@@_x_computation_for_option_i:
  {\dim_gzero_new:N \g_@@_x_dim
```

First, we calculate the initial value for `\g_@@_x_dim`. In this loop, 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.

```
   \tikz[remember~picture]
      \path let \p1 = (wa-\tl_use:N\l_@@_prefix_tl-\int_use:N\l_@@_initial_int-l)
            in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\x1}} ;
```

A global assignment is necessary because of Tikz.
Then, we will loop to determine the maximal length of all 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 `\@@_x_computation_for_option_group:`

```
   \@@_x_computation_analyze_lines_between:
   }
```

[15]cf. : `tex.stackexchange.com/questions/298177/how-to-get-around-a-problem-with-usetikzlibrarybabel`

The command `\@@_x_computation_analyze_lines_between:` will analyse the lines between between `\l_@@_initial_int` and `\l_@@_final_int` in order to modify `\g_@@_x_dim` in consequence. More precisely, we will increase `\g_@@_x_dim` if we find a line longer than the current value of `\g_@@_x_dim`.

```
\cs_new_protected:Nn \@@_x_computation_analyze_lines_between:
    {\int_compare:nNnT \l_@@_final_int > \g_@@_line_int
             {\msg_error:nn {witharrows} {Too~few~lines~for~an~arrow}}
```

We begin a loop with `\l_tmpa_int` as index. In this loop, 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.

```
     \int_set:Nn \l_tmpa_int \l_@@_initial_int
     \int_until_do:nNnn \l_tmpa_int > \l_@@_final_int
         {\tikz[remember~picture]
             \path let \p1 = (wa-\tl_use:N\l_@@_prefix_tl-\int_use:N\l_tmpa_int-l)
                   in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\dim_max:nn \g_@@_x_dim {\x1}}} ;
          \int_incr:N \l_tmpa_int
         }
    }
```

We want to compute the x-value for the current environment which has option `group` (and therefore all arrows are vertical at the same abscissa). Once again, the value will be computed in `\g_@@_x_dim`.

```
\cs_new_protected:Nn \@@_x_computation_for_option_group:
  {\dim_gzero_new:N \g_@@_x_dim
```

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

```
    \prop_get:cnN {g_@@_arrow_\l_@@_prefix_tl _1_prop} {initial} \l_tmpa_tl
    \tikz[remember~picture]
       \path let \p1 = (wa-\tl_use:N\l_@@_prefix_tl-\tl_use:N\l_tmpa_tl-l)
             in \pgfextra {\dim_gset:Nn \g_@@_x_dim {\x1}} ;
```

A global assignment is necessary because of Tikz.
Then, we loop to determine the maximal length of all the lines concerned by the arrows of the environment.

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

`\l_@@_initial_int` is the line number from which the arrow starts and `\l_@@_final_int` is the line number to which the arrow ends.

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

`\@@_x_computation_analyze_lines_between:` will compute between lines `\l_@@_initial_int` and `\l_@@_final_int`.

```
         \@@_x_computation_analyze_lines_between:
         \int_incr:N \l_@@_arrow_int
       }
    }
\cs_generate_variant:Nn \int_compare:nNnT {cNnT}
\cs_generate_variant:Nn \tl_if_eq:nnF {noF}
```

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

```
\NewDocumentCommand \WithArrowsLastEnv {}
     {\int_use:N \g_@@_last_env_int}
```

6.8 The error messages of the package

```
\msg_new:nnn {witharrows}
             {Third~column~in~a~{WithArrows}~environment}
             {By~default,~a~\{WithArrows\}~environment~can~only~have~two~columns.~
              Maybe~you~have~forgotten~a~newline~symbol.~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}
             {Arrow~used~outside~{WithArrows}~environment}
             {The~command~\string\Arrow\space~should~be~used~only~directly~
              in~\{WithArrows\}~environment~and~not~in~a~subenvironment.~However,~you~
              can~go~on.}
\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~
              \string\Arrow.~ You~can~create~an~arrow~going~backwards~with~
              the~option~"<-"~of~Tikz.}
\msg_new:nnn {witharrows}
             {Too~few~lines~for~an~arrow}
             {There~is~at~least~an~arrow~that~can't~be~drawn~because~it~arrives~after~the~
              last~line~of~the~environment.}
\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}
             {Option~incompatible~with~group}
             {You~try~to~use~the~option~"\tl_use:N\l_keys_key_tl"~while~
              you~are~using~the~option~"group".~It's~incompatible.~You~can~go~on~ignoring~
              this~option~"\tl_use:N\l_keys_key_tl"~and~"group"~will~be~used.}
\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~with~an~option~previously~set.~
             If~you~go~on,~I~will~overwrite~the~previous~option.}
\msg_new:nnnn {witharrows}
              {All~lines~must~have~an~ampersand}
              {All~lines~of~an~environment~\{WithArrows\}~must~have~an~second~column~
               (because~the~nodes~are~created~in~the~second~column).~You~can~go~on~but~maybe~
               you~will~have~an~pgf~error~for~an~undefined~shape.}
              {The~ampersand~can~be~implicit~
               (e.g.~if~you~use~\string\MoveEqLeft\space~of~mathtools).}
```

7 History

7.1 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