The package \texttt{witharrows} for plain-TeX and \LaTeX\*

F. Pantigny
fpantigny@wanadoo.fr
August 24, 2023

Abstract

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

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

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

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

The arrows are drawn with Tikz and that’s why \texttt{several compilations may be necessary}.\footnote{If you use Overleaf, Overleaf will do automatically a number compilations sufficient (by using \texttt{latexmk}).}

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

\begin{verbatim}
$\begin{WithArrows}
A \quad \quad & = (a+1)^2 \quad \Arrow{\text{we expand}} \quad \\
& = a^2 + 2a + 1 \quad \text{\textless------ don’t put \textless here}
\end{WithArrows}$
\end{verbatim}

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

The arrow has been drawn with the command \texttt{\Arrow} on the row from which it starts. The command \texttt{\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 \texttt{WithArrows} bears similarities with the environment \texttt{aligned} of \texttt{amsmath} (and \texttt{mathtools}). The extension \texttt{witharrows} also provides an environment \texttt{DispWithArrows} which is similar to the environment \texttt{align} of \texttt{amsmath}: cf. p. 17.

\footnote{This document corresponds to the version 2.8b of \texttt{witharrows}, at the date of 2023/08/24.}
1 Options for the shape of the arrows

The command \texttt{Arrow} has several options. These options can be put between square brackets, before, or after the mandatory argument. The option \texttt{jump} gives the number\textsuperscript{2} of rows the arrow must jump (the default value is, of course, 1).

\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \Arrow\{we expand\} \\\n  & = (a+b)^2 + 2(a+b) + 1 \\\n  & = a^2 + 2ab + b^2 + 2a + 2b + 1
\end{WithArrows}\\n\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \\\n  & = (a+b)^2 + 2(a+b) + 1 \quad \text{we expand}
\end{WithArrows}\\
\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \\\n  & = (a+b)^2 + 2(a+b) + 1 \quad \text{with xoffset=1cm}
\end{WithArrows}\\

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

\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \Arrow{}\Arrow{}\{jump=2\} \\\n  & = (a+b)^2 + 2(a+b) + 1
\end{WithArrows}\\
\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \\\n  & = (a+b)^2 + 2(a+b) + 1
\end{WithArrows}

The option \texttt{xoffset} shifts the arrow to the right (we usually don’t want the arrows to be stucked on the text). The initial value of \texttt{xoffset} is 3 mm.

\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \\text{\texttt{xoffset=1cm}}\{with \texttt{xoffset=1cm}\} \\\n  & = (a+b)^2 + 2(a+b) + 1
\end{WithArrows}\\
\begin{WithArrows}
A & = \bigl((a+b)+1\bigr)^2 \\\n  & = (a+b)^2 + 2(a+b) + 1 \quad \text{with xoffset=1cm}
\end{WithArrows}

The arrows are drawn with Tikz. That’s why the command \texttt{Arrow} has an option \texttt{tikz} which can be used to give to the arrow (in fact, the command \texttt{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\{we expand\} \\\n  & = a^2 + 2a + 1
\end{WithArrows}\\
\begin{WithArrows}
A & = (a+1)^2 \\text{\texttt{tikz=thick}}\{we expand\} \\\n  & = a^2 + 2a + 1
\end{WithArrows}

It’s also possible to change the arrowheads. For example, we can draw an arrow which goes backwards with the Tikz option \texttt{<--}.

\textsuperscript{2}It’s not possible to give a non-positive value to \texttt{jump}. See below (p. 2) the way to draw an arrow which goes backwards.
\begin{WithArrows}
A &= (a+1)^2 \text{\arrowleft we factorize} \\
&= a^2 + 2a + 1
\end{WithArrows}$

It’s also possible to suppress both tips of the arrow with the Tikz option “\textasciitilde”.

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

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

\begin{WithArrows}
A &= (a+1)^2 \text{\arrowneg\textwidth=5.3cm} \text{\arrowright we expand} \\
&= a^2 + 2a + 1
\end{WithArrows}$

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

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

\begin{WithArrows}
A &= \left((a+b)+1\right)^2 \text{\arrowleft \textwidth=5.3cm} \text{\arrowright we expand} \\
&= (a+b)^2 + 2(a+b) +1
\end{WithArrows}$

\begin{DispWithArrows}
\begin{DispWithArrows*}
A &= (a+b)^2 + 2(a+b) +1 \\
&= a^2 + 2ab + b^2 + 2a + 2b +1
\end{DispWithArrows*}
\end{DispWithArrows*}$

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

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

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

It’s possible to put commands `\` in the text to force new lines\(^3\). However, if we put a `\`, a command of font placed in the beginning of the text will have effect only until the first command `\` (like in an environment `tabular`). That’s why Tikz gives an option `font` to modify the font of the whole text. Nevertheless, if we use the option `tikz={font={\bfseries}}`, the default specification of `\small` and `\itshape` will be overwritten.

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

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

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

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

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

\[
A = ((a + b) + 1)^2
= (a + b)^2 + 2(a + b) + 1 \quad \text{first expansion.}
= a^2 + 2ab + b^2 + 2a + 2b + 1 \quad \text{second expansion.}
\]

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

Without the option `displaystyle`:

\[
\begin{WithArrows}
\int_0^1 (x+1)^2 \, dx \\
& = \int_0^1 (x^2+2x+1) \, dx \Arrow\{\text{linearity of integration}\} \\& = \int_0^1 x^2 \, dx + 2 \int_0^1 x \, dx + \int_0^1 1 \, dx \\& = \frac{x^3}{3} + 2 \frac{x^2}{2} + 1 \\& = \frac{x^3}{3} + 2 + 1 \\
\end{WithArrows}
\]

\[
\int_0^1 (x + 1)^2 \, dx = \int_0^1 (x^2 + 2x + 1) \, dx
= \int_0^1 x^2 \, dx + 2 \int_0^1 x \, dx + \int_0^1 1 \, dx \quad \text{linearity of integration}
= \frac{1}{3} + 2 \frac{1}{2} + 1
= \frac{7}{3}
\]

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

\(^4\)They also apply to the nested environments `{WithArrows}` (with the logical exceptions of `interline`, `code-before` and `code-after`).
The same example with the option displaystyle:

\[\int_0^1 (x+1)^2 dx = \int_0^1 (x^2 + 2x + 1) dx = \int_0^1 x^2 dx + 2 \int_0^1 x dx + \int_0^1 dx\]

linearity of integration

\[= \frac{1}{3} + \frac{2}{2} + 1 = \frac{7}{3}\]

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}.

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

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

\begin{align*}
\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 \text{by linearity}
\end{align*}

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

\$\begin{WithArrows}\[\text{\texttt{\textbackslash n}}\text{\texttt{\textbackslash Color}}=\text{\texttt{\textcolor{blue}}}\]
A & = (a+b)^2 \Arrow{we expand} \\
& = a^2 + 2ab + b^2
\end{WithArrows}\$

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

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

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

\[ A = (a + b)^2 = a^2 + 2ab + b^2 \]

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

2 Numbers of columns

So far, we have used the environment `{WithArrows}` with two columns. However, it’s possible to use the environment with an arbitrary number of columns with the option `format`. The value given to this option is like the preamble of an environment `{array}`, that is to say a sequence of letters `r`, `c` and `1`, but also `R`, `C` and `L`.

The letters `R`, `C` and `L` add empty groups `{}` which provide correct spaces when these columns contain symbols with the type `\mathrel` (such as `=`, `≤`, etc.) or `\mathbin` (such as `+`, `×`, etc.). This system is inspired by the environment `{IEEEeqnarray}` of the package `IEEEtrantools`.

The initial value of the parameter `format` is, in fact, `rL`.

For exemple, if we want only one column left-aligned, we use the option `format=1`.

\begin{WithArrows}[format = 1]
\begin{align*}
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{align*}
\end{WithArrows}

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

\begin{DispWithArrows*}[format = cCcCc, wrap-lines, interline=1mm]
k \le t \le k+1 \\
\frac{1}{k+1} \le \frac{1}{t} \le \frac{1}{k} \\
\int_k^{k+1} \frac{dt}{t+1} \le \int_k^{k+1} \frac{dt}{t} \le \int_k^{k+1} \frac{dt}{t+1} \\
\frac{1}{k+1} \le \ln(k+1) - \ln(k) \le \frac{1}{k}
\end{DispWithArrows*}
3 Precise positioning of the arrows

The environment `{WithArrows}` defines, during the composition of the array, two series of nodes materialized in red in the following example.\footnote{The option `show-nodes` can be used to materialize the nodes. The nodes are in fact Tikz nodes of shape “rectangle”, but with zero width. An arrow between two nodes starts at the `south` anchor of the first node and arrives at the `north` anchor of the second node.}

\[
I = \int_\frac{\pi}{4}^0 \ln \left( 1 + \tan \left( \frac{\pi}{4} - u \right) \right) (-du) \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( 1 + \tan \left( \frac{\pi}{4} - u \right) \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( \frac{1 + \tan u}{1 + \tan u} \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( \frac{1 + \tan u + 1 - \tan u}{1 + \tan u} \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( \frac{2}{1 + \tan u} \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \left( \ln 2 - \ln(1 + \tan u) \right) du \quad .
\]

\[
= \frac{\pi}{4} \ln 2 - \int_0^\frac{\pi}{4} \ln(1 + \tan u) du \quad .
\]

\[
= \frac{\pi}{4} \ln 2 - I \quad .
\]

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

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

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

Therefore \( I = \int_0^\frac{\pi}{4} \ln \left( 1 + \tan \left( \frac{\pi}{4} - u \right) \right) (-du) \) \hspace{1cm} \text{This arrow uses the lr option.}

\[
= \int_0^\frac{\pi}{4} \ln \left( 1 + \tan \left( \frac{\pi}{4} - u \right) \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( \frac{1 + \tan u + 1 - \tan u}{1 + \tan u} \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \ln \left( \frac{2}{1 + \tan u} \right) du \quad .
\]

\[
= \int_0^\frac{\pi}{4} \left( \ln 2 - \ln(1 + \tan u) \right) du \quad .
\]

\[
= \frac{\pi}{4} \ln 2 - \int_0^\frac{\pi}{4} \ln(1 + \tan u) du \quad .
\]

\[
= \frac{\pi}{4} \ln 2 - I \quad .
\]

There is also an option called i (i for intermediate). With this option, the arrow is vertical and at the leftmost position.
\( (a+b)(a+ib)(a-b)(a-ib) \)
\[
\begin{align*}
&= (a+b)(a-b) \cdot (a+ib)(a-ib) \\
&= (a^2-b^2)(a^2+b^2) \\
&= a^4-b^4
\end{align*}
\]

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.

\[
\begin{align*}
2xy' - 3y &= \sqrt{x} \\
&\iff 2x(K'y_0 + Ky_0') - 3Ky_0 = \sqrt{x} \\
&\iff 2xK'y_0 + K(2xy_0' - 3y_0) = \sqrt{x} \\
&\iff 2xK'y_0 = \sqrt{x} \\
&\iff K' = \frac{1}{2\sqrt{x}} \\
&\iff K = -\frac{1}{2\sqrt{x}}
\end{align*}
\]

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

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

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

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

\(^8\)Such arrow will be called independent in the technical documentation.
In the following example, we have used the option `groups` for the environment and the option `new-group` for the last arrow (that’s why the last arrow is not aligned with the others).

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

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

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

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

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

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

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

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

\[
= \frac{1}{\cos^n x} \frac{\sin(n+1)x}{\sin x}
\]

### 4 The option “o” for individual arrows

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

We will say that the arrow `a` covers the arrow `b` when `i_a ≤ i_b ≤ f_b ≤ f_a`. We will also say that the arrow `a` is over the arrow `b`.

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

\[
A = B
\]

\[
= C
\]

\[
= D
\]

\[
= E
\]

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

An arrow of type `o` is drawn with an horizontal shift (such as those set by `xoffset`) automatically computed by taking into account the arrows covered by our arrow.\footnote{Among the covered arrows, the independent ones (that is to say with an explicit key `rr`, `lr`, `rl`, `i`, `up` or `down`) are not taken into account in the computation of the value of `xoffset`.}

\[
\begin{align*}
A & = B \quad \text{\texttt{\begin{WithArrows}[groups]}} \quad \text{\texttt{\begin{WithArrows}[o, jump=3]{direct} \}}}
& = C + C \quad \text{\texttt{\begin{WithArrows}[o, jump=3]{direct} \}}}
& = D + D + D \quad \text{\texttt{\begin{WithArrows}[o, jump=3]{direct} \}}}
& = E + E \quad \text{\texttt{\begin{WithArrows}[o, jump=3]{direct} \}}}
& = F + F \quad \text{\texttt{\begin{WithArrows}[o, jump=3]{direct} \}}}
\end{WithArrows}}\end{align*}
\]

\]
Arrows of type \texttt{o} may themselves be covered by other arrows of type \texttt{o}.

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

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

\section{The options \texttt{“up”} and \texttt{“down”} for individual arrows}

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

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

The options \texttt{up} and \texttt{down} require the Tikz library \texttt{calc}. It it has not been previously loaded by the user, an error will be raised.

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

- The key \texttt{radius} is the radius of the rounded corner of the arrow.\footnote{The initial value of this parameter is 4 pt, which is the default value of the \texttt{“rounded corners”} of Tikz.}

- The key \texttt{width} is the width of the (horizontal part of) the arrow:
  - with the value \texttt{max}, the width of the arrow is adjusted with respect of the position of the nodes (that’s the behaviour by default of the arrows \texttt{up} and \texttt{down} as shown in the previous example);
– with a numerical value, the width of the arrow is directly fixed to that numerical value;
– with the value \texttt{min}, the width of the arrow is adjusted with respect to the contents of the label of the arrow.

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

we try
\[ A = B \]
\[ = C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C \]

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

we try
\[ A = B \]
\[ = C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C + C \]

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

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

\section{Comparison with the environment \{aligned\}}
\{WithArrows\} bears similarities with the environment \{aligned\} of the extension \texttt{amsmath}. These are only similarities because \{WithArrows\} has not been written upon the environment \{aligned\}.

As in the environments of \texttt{amsmath}, it’s possible to change the spacing between two given rows with the option of the command \texttt{\\\textbackslash} of end of line (it’s also possible to use \texttt{\\\textbackslash*} but it has exactly the same effect as \texttt{\\\textbackslash} since an environment \{WithArrows\} is always unbreakable). This option is designed to be used with positive values only.

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

we expand

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

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

we expand

we go on

\( F = \frac{1}{2}G \)
\[ = H + \frac{1}{2}K \]
\[ = K \]

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

\begin{verbatim}
\begin{WithArrows}[jot=2ex]
\varphi(x,y) = 0 & \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0 \\
& \Leftrightarrow \left\{ \begin{array}{l}
x+y = 0 \\
x+2y = 0
\end{array} \right. \quad \text{x and y are real}
\end{WithArrows}
\end{verbatim}

\( \varphi(x,y) = 0 \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0 \)
\[ \Leftrightarrow \left\{ \begin{array}{l}
x+y = 0 \\
x+2y = 0
\end{array} \right. \quad \text{x and y are real} \]

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

\begin{verbatim}
\begin{WithArrows}[interline=2ex]
\varphi(x,y) = 0 & \Leftarrowrightarrow (x+y)^2 + (x+2y)^2 = 0 \\
& \Leftarrowrightarrow \left\{ \begin{array}{l}
x+y = 0 \\
x+2y = 0
\end{array} \right. \quad \text{x and y are real}
\end{WithArrows}
\end{verbatim}

\( \varphi(x,y) = 0 \Leftrightarrow (x+y)^2 + (x+2y)^2 = 0 \)
\[ \Leftrightarrow \left\{ \begin{array}{l}
x+y = 0 \\
x+2y = 0
\end{array} \right. \quad \text{x and y are real} \]
\[ \varphi(x, y) = 0 \iff (x + y)^2 + (x + 2y)^2 = 0 \]
\[ \iff \begin{cases} 
x + y = 0 \\
x + 2y = 0 
\end{cases} \quad \text{x and y are real} \]

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

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

So \( A = (a + 1)^2 \) \( \iff \) \( a^2 + 2a + 1 \) \( \iff \) \text{we expand}

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

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

\text{Let’s set} \begin{align*}
f(x) &= 3x^3 + 2x^2 - x + 4 \\
g(x) &= 5x^2 - 5x + 6
\end{align*} \text{both are polynoms}

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

The following example is composed with \texttt{aligned}:
\[
\begin{aligned}
\sum_{i=1}^{n} (x_i + 1)^2 &= \sum_{i=1}^{n} (x_i^2 + 2x_i + 1) \\
&= \sum_{i=1}^{n} x_i^2 + 2 \sum_{i=1}^{n} x_i + n
\end{aligned}
\]

The following is composed with \texttt{WithArrows}[c,displaystyle]. The results are strictly identical.\textsuperscript{13}
\[
\begin{aligned}
\sum_{i=1}^{n} (x_i + 1)^2 &= \sum_{i=1}^{n} (x_i^2 + 2x_i + 1) \\
&= \sum_{i=1}^{n} x_i^2 + 2 \sum_{i=1}^{n} x_i + n
\end{aligned}
\]

\textsuperscript{13}\text{In versions of amsmath older than the 5 nov. 2016, a thin space was added on the left of an environment \texttt{aligned}. The new versions do not add this space and neither do \texttt{WithArrows}.}
The environments \{WithArrows\} can be nested. In this case, the options given to the encompassing environment applies also to the inner ones (with logical exceptions for interline, code-before and code-after). The command \texttt{Arrow} can be used as usual in each environment \{WithArrows\}.

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

\begin{WithArrows}
\varphi(x,y)=0 & \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \\
& \Leftrightarrow \left\{
\begin{array}{c}
x+2y & = 0 \text{ the same equation} \\
x+2y & = 0
\end{array}
\right.
\end{WithArrows}

\begin{WithArrows}
\varphi(x,y)=0 & \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \\
& \Leftrightarrow \left\{
\begin{array}{c}
x+2y & = 0 \\
x+2y & = 0 \text{ division by } 2
\end{array}
\right.
\end{WithArrows}

\begin{WithArrows}
\varphi(x,y)=0 & \Leftrightarrow (x+2y)^2+(2x+4y)^2 = 0 \\
& \Leftrightarrow \left\{
\begin{array}{c}
x+2y & = 0 \\
x+2y & = 0
\end{array}
\right.
\end{WithArrows}

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

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

\begin{WithArrows}
\varphi(x,y) = 0 \Leftrightarrow (x + 2y)^2 + (2x + 4y)^2 = 0

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

A command \texttt{\Arrow} in code-after takes three arguments:

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

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

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

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

14
\[\varphi(x,y) = 0 \iff (x + 2y)^2 + (2x + 4y)^2 = 0 \]
\[\iff \begin{cases} 
  x + 2y = 0 \\
  2x + 4y = 0 
\end{cases} \text{ environment number 1} \]
\[\iff \begin{cases} 
  x + 2y = 0 \\
  x + 2y = 0 \end{cases} \text{ division by 2} \]
\[\iff x + 2y = 0 \]

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

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

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

With the option \texttt{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 \texttt{i}. Currently, the option \texttt{i} is not available for the command \texttt{\Arrow} in \texttt{code-after}. However, it’s always possible to translate an arrow with \texttt{xoffset} (or \texttt{xshift} of Tikz).

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

The package \texttt{witharrows} gives also another command available only in \texttt{code-after}: the command \texttt{\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 \texttt{\MultiArrow}. This list is given with the syntax of the list in a \texttt{foreach} command of \texttt{pgffor}.

\begin{verbatim}
\begin{WithArrows}[tikz = rounded corners, code-after = \texttt{\MultiArrow{1,...,4}{text}}]
A & = B \\
& = C \\
\end{WithArrows}
\end{verbatim}
As of now, there is no option available for the command \texttt{MultiArrow} (maybe in a future release).

8 Arrows from outside environments \{WithArrows\}

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

The Tikz name of a node created by \texttt{witharrows} is prefixed by \texttt{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 suffix \texttt{l} for a “left node” and \texttt{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.\footnote{There is an option \texttt{show-node-names} to show the names of these nodes.}

\begin{align*}
A & \iff B + B + B + B + B + B + B + B + B + B + B + B + \texttt{wa-45-1-r} \\
& \iff C \iff \texttt{wa-45-1-1-r} \\
& \iff D \iff \texttt{wa-45-1-2-r} \\
& \iff E \iff \texttt{wa-45-2-r} \\
& \iff F \iff \texttt{wa-45-3-r} \\
& \iff G \iff H + H + H + H + H + H + \texttt{wa-45-2-1-r} \\
& \iff I \iff \texttt{wa-45-2-2-r} \\
& \iff J \iff K \iff \texttt{wa-45-2-1-1-r} \\
& \iff L \iff M \iff \texttt{wa-45-2-1-2-r} \\
& \iff N \iff \texttt{wa-45-3-1-r} \\
& \iff O \iff \texttt{wa-45-3-2-r} \\
& \iff P \iff \texttt{wa-45-4-r} \\
& \iff Q
\end{align*}

The package \texttt{witharrows} provides some tools facilitating the use of these nodes:

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

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

- the Tikz style \texttt{\WithArrows/arrow} is the style used by \texttt{witharrows} when drawing an arrow\footnote{More precisely, this style is given to the Tikz option “\texttt{every path}” before drawing the arrow with the code of the option \texttt{tikz-code}. This style is modified (in TeX scopes) by the option \texttt{tikz} of \texttt{witharrows}.};

- the Tikz style \texttt{\WithArrows/arrow/tips} is the style for the tip of the arrow (loaded by \texttt{\WithArrows/arrow}).

For example, we can draw an arrow from \texttt{wa-45-2-1-2-r.south} to \texttt{wa-45-3-2-r.north} with the following Tikz command.
In this case, it would be easier to use a command \Arrow in code-after but this is an example to explain how the Tikz nodes created by witharrows can be used.

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

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

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

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

9 The environment \{DispWithArrows\}

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

The environment \{DispWithArrows\} must be used outside math mode. Like \{align\}, it should be used in horizontal mode.
\begin{DispWithArrows}
A &= (a+1)^2 \Arrow{we expand} \\
&= a^2 + 2a + 1
\end{DispWithArrows}
\[A = (a+1)^2 = a^2 + 2a + 1\]
\[\text{we expand}\]
\[\text{(1)}\]
\[\text{(2)}\]

It’s possible to use the command \texttt{\notag} (or \texttt{\nonumber}) to suppress a tag.

It’s possible to use the command \texttt{\tag} to put a special tag (e.g. \texttt{\*}).

It’s also possible to put a label to the line of an equation with the command \texttt{\label}.

These commands must be in the second column of the environment.

\begin{DispWithArrows}
A &= (a+1)^2 \Arrow{we expand} \texttt{\notag} \\
&= a^2 + 2a + 1 \texttt{\tag{\texttt{\*}}} \texttt{\label{my-equation}}
\end{DispWithArrows}
\[A = (a+1)^2 = a^2 + 2a + 1\]
\[\text{we expand}\]
\[\text{(\*)}\]

A link to the equation (\*).

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

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

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

\begin{DispWithArrows*}
A &= (a+1)^2 \Arrow{we expand} \texttt{\notag} \\
&= a^2 + 2a + 1
\end{DispWithArrows*}
\[A = (a+1)^2 = a^2 + 2a + 1\]
\[\text{we expand}\]

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

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

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

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

\[\text{Even in this case, it’s possible to put a “manual tag” with the command \texttt{\tag}.}\]
\begin{DispWithArrows}[fleqn,mathindent = 1cm]
A & = (a+1)^2 \ Arrow{we expand} \\
& = a^2 + 2a + 1
\end{DispWithArrows}

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

If the option \texttt{leqno} is used as a class option, the labels will be composed on the left also for the environments \texttt{DispWithArrows} and \texttt{DispWithArrows*}.\footnote{The package \texttt{amsmath} has an option \texttt{leqno} but \texttt{witharrows}, of course, is not aware of that option: \texttt{witharrows} only checks the option \texttt{leqno} of the document class.}

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

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

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

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

If the option \texttt{leqno} is used as a class option, the labels will be composed on the left also for the environments \texttt{DispWithArrows} and \texttt{DispWithArrows*}.\footnote{The package \texttt{amsmath} has an option \texttt{leqno} but \texttt{witharrows}, of course, is not aware of that option: \texttt{witharrows} only checks the option \texttt{leqno} of the document class.}

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

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

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

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

If the option \texttt{leqno} is used as a class option, the labels will be composed on the left also for the environments \texttt{DispWithArrows} and \texttt{DispWithArrows*}.\footnote{The package \texttt{amsmath} has an option \texttt{leqno} but \texttt{witharrows}, of course, is not aware of that option: \texttt{witharrows} only checks the option \texttt{leqno} of the document class.}

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

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

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

Remark: By design, the option \texttt{fleqn} of \texttt{witharrows} is independent of the option \texttt{fleqn} of LaTeX. Indeed, since the environments of \texttt{witharrows} are meant to be used with arrows on the right side, the user may want to use \texttt{witharrows} with the option \texttt{fleqn} (in order to have more space on the right of the equations for the arrows) while still centering the classical equations.
If there is not enough space to put the tag at the end of a line, there is no automatic positioning of the label on the next line (as in the environments of amsmath). However, in \{DispWithArrows\}, the user can use the command \texttt{\textbackslash tagnextline} to manually require the composition of the tag on the following line.

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

The environments \{DispWithArrows\} and \{DispWithArrows*\} provide an option \texttt{wrap-lines}. With this option, the lines of the label are automatically wrapped on the right.\footnote{Note: This feature is not directly shown in the text but is implied by the example.}

\begin{DispWithArrows*}[displaystyle,wrap-lines]
S_n \quad \begin{aligned}
&= \frac{1}{n} \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac{1}{n} \Re \left( \frac{1 - (e^{i \frac{\pi}{2n}})^n}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{aligned}
\end{DispWithArrows*}

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

We have said that the environments \{DispWithArrows\} and \{DispWithArrows*\} should be used in horizontal mode and not in vertical mode. However, there is an exception. These environments can...
be used directly after a \item of a LaTeX list. In this case, no vertical space is added before the environment.\footnote{It’s possible to disable this feature with the option \texttt{standard-behaviour-with-items}.}

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

\begin{enumerate}
\item \begin{DispWithArrows}\
\[\text{displaystyle, wrap-lines, tagged-lines = last, fleqn, mathindent = 0 pt}\]
\begin{align*}
S_n &= \frac1n \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac1n \Re \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac1n \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{align*}
\end{DispWithArrows}
\end{enumerate}

1. \begin{align*}
S_n &= \frac1n \Re \left( \sum_{k=0}^{n-1} \left( e^{i \frac{\pi}{2n}} \right)^k \right) \\
&= \frac1n \Re \left( \frac{1 - \left( e^{i \frac{\pi}{2n}} \right)^n}{1 - e^{i \frac{\pi}{2n}}} \right) \\
&= \frac1n \Re \left( \frac{1 - i}{1 - e^{i \frac{\pi}{2n}}} \right)
\end{align*} \tag{13}

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

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

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

Concerning the references, the package \texttt{witharrows} is compatible with the extensions \texttt{autonum}, \texttt{cleveref}, \texttt{fancyref}, \texttt{hyperref}, \texttt{listlbls}, \texttt{prettyref}, \texttt{refcheck}, \texttt{restyle}, \texttt{showlabels}, \texttt{smartref}, \texttt{typedref} and \texttt{varpiref}, and with the options \texttt{showonlyrefs} and \texttt{showmanualtags} of mathtools.\footnote{We recall that \texttt{varpiref}, \texttt{hyperref}, \texttt{cleveref} and \texttt{autonum} must be loaded in this order. The package \texttt{witharrows} can be loaded anywhere.}

It is not compatible with \texttt{showkeys} (not all the labels are shown).
9.1 The option $<...>$ of DispWithArrows

The environment \{DispWithArrows\} provides an option left-brace. When present, the value of this option is composed on the left, followed by a curly brace (hence the name) and the body of the environment.\footnote{The option left-brace can also be used without value: in this case, only the brace is drawn...}

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

The following code is an example of multi-case equations.\footnote{The environment \{cases\} of amsmath is a way to compose such multi-cases equations. However, it’s not possible to use the automatic numbering of equations with this environment. The environment \{numcases\} of the extension cases (written by Donald Arseneau) provides this possibility but, of course, it’s not possible to draw arrows with this extension.}

\begin{DispWithArrows}
\begin{cases}
\binom{n}{p} = & \begin{array}{ll}
0 & \quad \text{if } p > n \\
\frac{n(n-1)\cdots(n-p+1)}{p!} & \quad \text{if } 0 \leq p \leq n \\
0 & \quad \text{if } p < 0
\end{array}
\end{cases}
\end{DispWithArrows}

\begin{align}
\binom{n}{p} &= \begin{cases}
0 & \quad \text{if } p > n \\
\frac{n(n-1)\cdots(n-p+1)}{p!} & \quad \text{if } 0 \leq p \leq n \\
0 & \quad \text{if } p < 0
\end{cases} \\
&\quad \text{if fact, it’s a special case of the following one} \\
&\quad \text{of the following one}
\end{align}

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

\begin{DispWithArrows}
\begin{align*}
\label{system} &\begin{array}{l}
x+y+z = -3 \\
x+y+z = -2 \\
xyz = -15
\end{array} \Leftrightarrow \\
&\begin{cases}
x + y + z = -3 \\
x y + x z + y z = -2 \\
xyz = -15
\end{cases} \text{3 equations}
\end{align*}
\end{DispWithArrows}

The whole system is the equation (17) (this reference has been coded by \ref{system}) whereas the last equation is the equation (17c) (this reference has been coded by \ref{last-equation}). The command \ref* used in the code above is a variant of the command \ref which does not create interactive link (even when hyperref is loaded).

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

10.1 Use with plain-TeX

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

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

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

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

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

The option tikz-code allows the user to change the shape of the arrows.\footnote{If the option wrap-lines is used in an environment \{DispWithArrows\} or \{DispWithArrows\*}, the option tikz-code will have no effect for the arrows of this environment but only for the arrows in the nested environments \{WithArrows\}.}

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

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

By default, the value is the following:

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

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

\begin{WithArrows}
\begin{verbatim}
format=c,ygap=5pt,interline=4mm,
tikz-code = \{\draw[rounded corners]
  (#1) -- ([xshift=5mm]#1)
  -- node[circle, draw, auto = false, fill = gray!50, inner sep = 1pt] {	iny #3}
  ([xshift=5mm]#2) \}
\end{verbatim}
\end{WithArrows}

\begin{itemize}
\item $3 (2x+4) = 6 \Arrow{$\div 3$}$ \\
\item $2x+4 = 2 \Arrow{$-4$}$ \\
\item $2x = -2 \Arrow{$\div 2$}$ \\
\item $x = -1$
\end{itemize}
The environments \{DispWithArrows\} and its starred version \{DispWithArrows*\} provide a command \WithArrowsRightX which can be used in a definition of tikz-code. This command gives the x-value of the right side of the composition box (taking into account the eventual tags of the equations). For an example of use, see p. 30.

10.3 The command \WithArrowsNewStyle

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

The command \WithArrowsNewStyle takes two mandatory arguments. The first is the name of the style and the second is a list of key-value pairs. The scope of the definition done by \WithArrowsNewStyle is the current TeX scope.\footnote{We recall that, in particular, every LaTeX environment is a TeX group.}

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

For an example of use, see p. 30.

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

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

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

10.4 The key right-overlap

\textbf{New 2.8}

The key right-overlap is a boolean key whose initial value is true. It deals with the environments \{WithArrows\} only.

When the key right-overlap is in force, the arrows (and their labels) are drawn in an overlapping position and are not relevant for the computation of the dimensions of the TeX box containing the environment \{WithArrows\}.

When the key right-overlap is set to false (with \WithArrowsOptions or within an individual environment \{WithArrows\}), the overlapping on the right is taken into account in the dimensions of the encompassing box.

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

\$\begin{WithArrows}[c,format = rCrCl]
2x & + & 3y & = & 5 \Arrow(we add $L_1$ to $L_2$)\
-2x & - & 5y & = & 2
\end{WithArrows}$

\$\\$
2x + 3y = 5 \quad \text{we add } L_1 \text{ to } L_2 \quad \begin{cases} 2x + 3y = 5 \\ -2y = 7 \end{cases}

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

10.5 Vertical positioning of the arrows

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

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

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

\[(\cos x + \sin x)^2 = \cos^2 x + 2 \cos x \sin x + \sin^2 x \quad \\text{ystart}\]
\[= \cos^2 x + \sin^2 x + 2 \sin x \cos x \quad \\text{ygap}\]
\[= 1 + \sin(2x)\]

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

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

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

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

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

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

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

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

\footnote{We call length of an arrow the difference between the y-value of its start point and the y value of its end point.}
\[
\begin{WithArrows}
max-length-of-arrow = 1.5cm
\]
\[
A
\begin{vmatrix}
1 & a & a^2 & a^3 & a^4 \\
1 & b & b^2 & b^3 & b^4 \\
1 & c & c^2 & c^3 & c^4 \\
1 & d & d^2 & d^3 & d^4 \\
1 & e & e^2 & e^3 & e^4
\end{vmatrix}
\Arrow{
L_2 \gets L_2-L_1 \\
L_3 \gets L_3-L_1 \\
L_4 \gets L_4-L_1 \\
L_5 \gets L_5-L_1$
}
\]
\[
\begin{vmatrix}
1 & a & a^2 & a^3 & a^4 \\
0 & b-a & b^2-a^2 & b^3-a^3 & b^4-a^4 \\
0 & c-a & c^2-a^2 & c^3-a^3 & c^4-a^4 \\
0 & d-a & d^2-a^2 & d^3-a^3 & d^4-a^4 \\
0 & e-a & e^2-a^2 & e^3-a^3 & e^4-a^4
\end{vmatrix}
\end{WithArrows}
\]
\[
L_2 \gets L_2-L_1 \\
L_3 \gets L_3-L_1 \\
L_4 \gets L_4-L_1 \\
L_5 \gets L_5-L_1$
\]
\[
A
\begin{vmatrix}
1 & a & a^2 & a^3 & a^4 \\
1 & b & b^2 & b^3 & b^4 \\
1 & c & c^2 & c^3 & c^4 \\
1 & d & d^2 & d^3 & d^4 \\
1 & e & e^2 & e^3 & e^4
\end{vmatrix}
\]

10.6 Footnotes in the environments of witharrows

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

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

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

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

Caution: The packages \footnote and \footnotehyper are incompatible. The package \footnotehyper is the successor of the package \footnote and should be used preferentially. 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:
We expand
\[
A = (a + b)^2 \\
= a^2 + b^2 + 2ab
\]

\[\text{Footnote.}\]

10.7 Option no-arrows

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

10.8 Note for the users of AUCTeX

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

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

\texttt{M-x customize > [Text] > [TeX] > [Font LaTeX]}

10.9 Note for developers

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

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

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

\texttt{\NewDocumentEnvironment {DWA} {} {\DispWithArrows}{\endDispWithArrows}}

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

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

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

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

11 Examples

11.1 \texttt{\MoveEqLeft}

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

---

\footnote{A footnote.}
\[ \begin{align*}
\arccos(x) &= \arcsin \frac{4}{5} + \arcsin \frac{5}{13} \\
& \quad \text{because both are in } [-\frac{\pi}{2}, \frac{\pi}{2}] \\
& \Leftrightarrow x = \sin \left( \arcsin \frac{4}{5} + \arcsin \frac{5}{13} \right) \\
& \quad \text{by } \forall x \in [-1,1], \cos(\arcsin x) = \sqrt{1-x^2} \\
& \Leftrightarrow x = \frac{4}{5} \cos \arcsin \frac{5}{13} + \frac{5}{13} \cos \arcsin \frac{4}{5} \\
& \quad \forall x \in [-1,1], \cos(\arcsin x) = \sqrt{1-x^2}
\end{align*} \]

11.2 A command \DoubleArrow

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

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

Example of use:

\$\begin{WithArrows}[groups]
A & = (a+b)^2 \DoubleArrow[format = c, interline = 4mm, tikz = {every node/.style = {circle, draw, auto = false, fill = gray!50, inner sep = 1pt, font = \tiny}]{expansion}{factorization} \div 3 \\phantom{2x+4} \\
2x+4 &= 2 \Div 3 \\
2x &= -2 \Div 2 \\
2x &= -1
\end{WithArrows}\$

11.3 Modifying the shape of the nodes

It’s possible to change the shape of the labels, which are Tikz nodes, by modifying the key “every node” of Tikz.
11.4 Examples with the option \texttt{tikz-code}

We recall that the option \texttt{tikz-code} is the Tikz code used by \texttt{witharrows} to draw the arrows.\footnote{If an environment \texttt{DispWithArrows} or \texttt{DispWithArrows*} is used with the option \texttt{wrap-lines}, the value of the option \texttt{tikz-code} is not used for this environment (but is used for the environments nested inside).}

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

11.4.1 Example 1

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

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

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

\begin{itemize}
\item $\cos x = \Re(e^{ix})$
\item $\Re(z + z') = \Re(z) + \Re(z')$
\item $\exp$ is a morphism for $\times$ and $+$
\item sum of terms of a geometric progression of ratio $e^{i \frac{\pi}{2}}$
\end{itemize}
11.4.2 Example 2

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

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

begin{DispWithArrows}[MyStyle]
\begin{align*}
S_n &= \frac{1}{n} \sum_{k=0}^{n-1} \cos\left(\frac{\pi}{2} \cdot \frac{k}{n}\right) \\
&= \frac{1}{n} \sum_{k=0}^{n-1} \Re\left(e^{i\frac{2\pi k}{n}}\right) \quad \text{(18)} \\
&= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} e^{i\frac{2\pi k}{n}}\right) \quad \text{(19)} \\
&= \frac{1}{n} \Re\left(\sum_{k=0}^{n-1} (e^{i\frac{2\pi}{n}})^k\right) \quad \text{(20)} \\
&= \frac{1}{n} \Re\left(\frac{1 - (e^{i\frac{2\pi}{n}})^n}{1 - e^{i\frac{2\pi}{n}}}\right) \quad \text{(21)} \\
&= \frac{1}{n} \Re\left(\frac{1 - i}{1 - e^{i\frac{2\pi}{n}}}\right) \quad \text{(22)} \\
\end{align*}

11.4.3 Example 3

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

\begin{WithArrows}[ll,interline=5mm,xoffset=5mm,
\begin{align*}
\text{tikz-code} &= \{\draw[rounded corners,
\quad \text{every node/.style = {circle,
draw,}
\quad \text{auto = false,}
\end{align*}
let \( p_1 = (#1) \),
\( p_2 = (#2) \)
in \( \text{ifdim } x_1 > x_2 \)
\( (p_1) -- node {#3} (x_1,y_2) -- (p_2) \)
\( \text{else} \)
\( (p_1) -- (x_2,y_1) -- node {#3} (p_2) \)
\fi ;}

\begin{WithArrows}
E \& \Longleftarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\begin{tabular}{l}
\arrow{$\times 15$}\n\arrow{$\div 20$}\\
\arrow{$-29$}\\
\arrow{$\equiv$}\\
\arrow{$\equiv$}\\
\arrow{$\equiv$}\\
\end{tabular}
 E \& \Longleftarrow \frac{(x+4)}{3} + \frac{5x+3}{5} = 7
\arrow{$\times 15$}\n\arrow{$\div 20$}\\
\arrow{$-29$}\\
\arrow{$\equiv$}\\
\arrow{$\equiv$}\\
\arrow{$\equiv$}\\
\begin{WithArrows}

11.5 Automatic numbered loop

Assume we want to draw a loop of numbered arrows. In this purpose, it’s possible to write a dedicated command \texttt{\textbackslash NumberedLoop} which will do the job when used in \texttt{code-after}. In the following example, we write this command with \texttt{\textbackslash NewDocumentCommand} (of \LaTeX) and \texttt{\textbackslash foreach} of \texttt{pgffor} (which is loaded when \texttt{witharrows} is loaded).

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

The command \texttt{\WithArrowsNbLines} is a command available in \texttt{code-after} which gives the total number of lines (=rows) of the current environment (it’s a command and not a counter).
e. \& f \text{ is lipschitzian}
\end{WithArrows}

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

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

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

\begin{itemize}
    \item \texttt{1}
    \item \texttt{2}
    \item \texttt{3}
    \item \texttt{4}
    \item \texttt{5}
\end{itemize}

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

\section{Implementation}

\subsection{Declaration of the package and extensions loaded}

The prefix \texttt{witharrows} has been registered for this extension.

See: \url{http://mirrors.ctan.org/macros/latex/contrib/l3kernel/l3prefixes.pdf}

\begin{verbatim}
\input tikz.tex
\input expl3-generic.tex
\usetikzlibrary{arrows.meta}
\usepgfmodule{bending}% https://texnique.fr/osqa/questions/12199
\end{verbatim}

Then, \texttt{tikz} and some Tikz libraries are loaded before the \texttt{\ProvidesExplPackage}. They are loaded this way because \texttt{\usetikzlibrary} in L3 code fails.\footnote{cf. \url{tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails}}

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

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

\begin{verbatim}
\{LaTeX\}
\RequirePackage{13keys2e}
\ProvidesExplPackage
\{witharrows\}
\{myfiledate\}
\{myfileversion\}
\{Draws arrows for explanations on the right\}
\end{verbatim}
12.2 The packages footnote and footnotehyper

A few options can be given to the package witharrows when it is loaded (with \usepackage, \RequirePackage or \PassOptionsToPackage). Currently (version 2.8b), there are two such options: \texttt{footnote} and \texttt{footnotehyper}. With the option \texttt{footnote}, witharrows loads \texttt{footnote} and uses it to extract the footnotes from the environments \texttt{WithArrows}. Idem for the option \texttt{footnotehyper}.

The boolean \texttt{\c@@footnotehyper_bool} will indicate if the option \texttt{footnotehyper} is used.

\begin{verbatim}
\bool_new:N \c@@footnotehyper_bool
\end{verbatim}

The boolean \texttt{\c@@footnote_bool} will indicate if the option \texttt{footnote} is used, but quickly, it will also be set to \texttt{true} if the option \texttt{footnotehyper} is used.

\begin{verbatim}
\bool_new:N \c@@footnote_bool
\end{verbatim}

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

\begin{verbatim}
\cs_new_protected:Npn \@@error_or_warning:n { \bool_if:NTF \c@@messages_for_Overleaf_bool \@@error:n \@@warning:n }
\end{verbatim}

\begin{verbatim}
\keys_define:nn { WithArrows / package }
{ \texttt{footnote} .bool_set:N = \c@@footnote_bool , \texttt{footnotehyper} .bool_set:N = \c@@footnotehyper_bool , \texttt{unknown} .code:n = \texttt{\c@@fatal:n} { \texttt{Option-unknown-for-package} }
}
\end{verbatim}

33
We process the options when the package is loaded (with \usepackage).}
\ProcessKeysOptions { WithArrows / package }

\bool_if:NT \c_@@_footnote_bool 
{ \begin{warning}

The class beamer has its own system to extract footnotes and that’s why we have nothing to do if \texttt{beamer} is used.
\end{warning}
\bool_set_true:N \c_@@_footnote_bool
}

The flag \c_@@_footnote_bool is raised and so, we will only have to test \c_@@_footnote_bool in order to know if we have to insert an environment \texttt{\begin{savenotes}} (the \texttt{\begin{savenotes}} is in \texttt{\@\pre_halign:n} and \texttt{\end{savenotes}} at the end of the environments \texttt{\WithArrows} and \texttt{\DispWithArrows}).
12.3 The class option leqno

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

\begin{verbatim}
\bool_new:N \c@@leqno_bool
\DeclareOption { leqno } { \bool_set_true:N \c@@leqno_bool }
\DeclareOption* { }
\ProcessOptions*
\end{verbatim}

12.4 Collecting options

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

\textit{Exemple}:
\begin{verbatim}
\@@_collect_options:n \{ \F \} \{ x=a,y=b \} \{ z=c,t=d \} \{ arg \}
\end{verbatim}
will be transformed in:
\begin{verbatim}
\F{x=a,y=b,z=c,t=d}{arg}
\end{verbatim}
Therefore, by writing: \begin{verbatim}
\def\G{\@@_collect_options:n{\F}}
\end{verbatim},
the command \G takes in an arbitrary number of optional arguments between square brackets.

\begin{verbatim}
\cs_new_protected:Npn \@@_collect_options:n #1
{ \peek_meaning:NTF [ 
{ \@@_collect_options:nw { #1 } }
{ #1 { } }
] }
\end{verbatim}

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

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

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

\begin{verbatim}
\cs_new_protected:Npn \@@_collect_options:nnw #1#2[#3]
{ \@@_collect_options:nn { #1 } { #2 , #3 } }
\end{verbatim}

12.5 Some technical definitions

\begin{verbatim}
\cs_generate_variant:Nn \seq_set_split:Nnn { N x x }
\end{verbatim}

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

\begin{verbatim}
\cs_new_protected:Npn \@@_sort_seq:N #1
{ \seq_sort:Nn #1 #1 }
\end{verbatim}
The following command creates a sequence of strings (str) from a clist.

\cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
{ \seq_set_from_clist:Nn #1 { #2 } \seq_set_map_x:NNn #1 #1 { \tl_to_str:n { ##1 } } \sort_return_same: \sort_return_swapped: }

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

\cs_new_protected:Npn \@@_save:N #1
{ \seq_set_split:Nxx \l_tmpa_seq { \char_generate:nn { `_ } { 12 } } { \cs_to_str:N #1 } \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl \str_set:Nx \l_tmpa_str { \seq_item:Nn \l_tmpa_seq { -1 } } \use:c { \l_tmpa_str_if_exist:cF } { g_{\seq_use:Nnnn \l_tmpa_seq _ _ _} } \use:c { \l_tmpa_str_new:c } { g_{\seq_use:Nnnn \l_tmpa_seq _ _ _} } \use:c { \l_tmpa_str_gset_eq:cN } #1 }

The string \l_{tmpa\_str} will contain the \textit{type} of the variable.

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

We define a Tikz style \@@_node_style for the l-nodes and r-nodes that will be created in the \halign. These nodes are Tikz nodes of shape “rectangle” but with zero width. An arrow between two nodes starts from the \textit{south} anchor of the first node and arrives at the \textit{north} anchor of the second node.

\tikzset
{ \@@_node_style / .style =
  { above = \l_{\@_ystart_dim} ,
    inner-sep = \c_zero_dim ,
    minimum-width = \c_zero_dim ,
    minimum-height = \l_{\@_ygap_dim} } }
If the user uses the option `show-nodes` (it’s a l3keys option), the Tikz options `draw` and `red` will be appended to this style. This feature may be useful for debugging.\footnote{The \texttt{v}-nodes, created near the end of line in \texttt{DispWithArrows} and \texttt{DispWithArrows*} are not shown with the option \texttt{show-nodes}.}

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

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

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

\begin{verbatim}
\sys_if_engine_xetex:T\n{\n  \tikzset\n  {\n    @@_standard_arrow / .append-style =\n      { every-node / .append-style = \{ text = . \} }\n  }\n}
\end{verbatim}

We also define a style for the tips of arrow. The final user of the extension \texttt{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 \texttt{\{WithArrows\}} in the \texttt{\{halign\}}). This style is documented in the documentation of \texttt{witharrows}.

\begin{verbatim}
\tikzset\n{\n  WithArrows / arrow / tips / .style =\n    { > = \{ Straight-Barb [ scale = 1.2 , bend ] \} }\n}
\end{verbatim}

The style \texttt{WithArrows/arrow} will be used to draw the arrows (more precisely, it will be passed to \texttt{every-path}). This style is documented in the documentation of \texttt{witharrows}.

\begin{verbatim}
\tikzset\n{\n  WithArrows / arrow / .style =\n    {\n      align = flush-left ,\n    }\n}
\end{verbatim}

Before the version 2.7, it was \texttt{align = left}.

\begin{verbatim}[@LaTeX]\n  \texttt{auto = left} ,\n\end{verbatim}

\begin{verbatim}[@LaTeX]\n  \{\texttt{\{WithArrows / arrow / tips ,}\n  \texttt{\{ WithArrows / arrow / tips ,}\n  \texttt{\{ bend-left = 45 ,}\n  \texttt{\} ->}\n\end{verbatim}

\footnote{The \texttt{v}-nodes, created near the end of line in \texttt{DispWithArrows} and \texttt{DispWithArrows*} are not shown with the option \texttt{show-nodes}.}
The option `subequations` is an option which uses the environment `{subequations}` of `amsmath`. That’s why, if `amsmath` is loaded, we add the key `subequations` to the list of the keys available in `{WithArrowsOptions}` and `{DispWithArrows}`.

In order to increase the interline in the environments `{WithArrows}`, `{DispWithArrows}`, etc., we will use the command \spread@equation of `amsmath`. When used, this command becomes no-op (in the current TeX group). Therefore, it will be possible to use the environments of `amsmath` (e.g. `{aligned}`) in an environment `{WithArrows}`. Nevertheless, we want the extension `witharrows` available without `amsmath`. That’s why we give a definition of `\spread@equation` if `amsmath` is not loaded.

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

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

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}`.

The following integer indicates the position of the box that will be created for an environment `{WithArrows}` (not an environment `{DispWithArrows}`): 0 (=t=\vtop), 1 (=c=\vcenter) or 2 (=b=\vbox).
The integer $\_@@\_pos\_arrow\_int$ indicates the position of the arrow with the following code (the option $v$ is accessible only for the arrows in code-after where the options $i$, group and groups are not available).

<table>
<thead>
<tr>
<th>option</th>
<th>lr</th>
<th>ll</th>
<th>rl</th>
<th>rr</th>
<th>v</th>
<th>i</th>
<th>groups</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>$_@@_pos_arrow_int$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

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

\begin{verbatim}
\int_new:N \l_@@_pos_arrow_int
\int_set:Nn \l_@@_pos_arrow_int 3
\end{verbatim}

In the \halign of an environment \{WithArrows\} or \{DispWithArrows\}, we will have to use four counters:

- $\g_@@_arrow\_int$ to count the arrows created in the environment;
- $\g_@@_line\_int$ to count the lines of the \halign;
- $\g_@@_col\_int$ to count the columns of the \halign.

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

\begin{verbatim}
\seq_new:N \g_@@_arrow_int_seq
\int_new:N \g_@@_arrow_int
\seq_new:N \g_@@_line_int_seq
\int_new:N \g_@@_line_int
\seq_new:N \g_@@_col_int_seq
\int_new:N \g_@@_col_int
\end{verbatim}

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

\begin{verbatim}
\seq_new:N \g_@@_static_col_int_seq
\int_new:N \g_@@_static_col_int
\end{verbatim}

For the environment \{DispWithArrows\}, the comma list $\l_@@\_tags\_clist$ will be the list of the numbers of lines to be tagged (with the counter equation of LaTeX). In fact, $\l_@@\_tags\_clist$ may contain non negative integers but also three special values: first, last and all.

\begin{verbatim}
⟨∗LaTeX⟩
\clist_new:N \l_@@_tags_clist
\clist_set:Nn \l_@@_tags_clist { all }
⟨/LaTeX⟩
\end{verbatim}

During the execution of an environment \{DispWithArrows\}, if a row must be tagged, the (local) value of $\l_@@\_tags\_clist$ will be put (by convention) to all.

\begin{verbatim}
\cs_new_protected:Npn \@@_test_if_to_tag:
{ \clist_if_in:NVT \l_@@_tags_clist \g_@@_line_int
\{ \clist_set:Nn \l_@@_tags_clist { all } \}
}
⟨/LaTeX⟩
\end{verbatim}

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

\begin{verbatim}
\str_new:N \l_@@_command_name_str
\str_set:Nn \l_@@_command_name_str { Arrow }
\end{verbatim}
The string \_@@\_string\_Arrow\_for\_msg\_str is only a string that will be displayed in some error messages. For example, if \texttt{command-name} is defined to be \texttt{Explanation}, this string will contain “\texttt{Arrow alias Explanation}”.

\begin{verbatim}
\str_new:N \_@@\_string\_Arrow\_for\_msg\_str
\str_set:Nx \_@@\_string\_Arrow\_for\_msg\_str { \token_to_str:N \Arrow }
\end{verbatim}

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

\begin{verbatim}
\seq_new:N \_@@\_names_seq
\end{verbatim}

The boolean \_@@\_sbwi\_bool corresponds to the option \texttt{standard-behaviour-with-items}. Since the version 1.16 of \texttt{witharrows}, no vertical space is added between an \texttt{item} of a \LaTeX{} list and an environment \texttt{(DispWithArrows)}. With the option \texttt{standard-behaviour-with-items}, it’s possible to restore the previous behaviour (which corresponds to the standard behaviour of \texttt{align} of \texttt{amsmath}). \_@@\_sbwi\_bool is the boolean corresponding to this option.

\begin{verbatim}
\bool_new:N \_@@\_sbwi\_bool
\end{verbatim}

The string \_@@\_format\_str will contain the \texttt{format} of the array which is a succession of letters \texttt{r}, \texttt{c} and \texttt{l} specifying the type of the columns of the \texttt{halign} (except the column for the labels of the equations in the environment \texttt{(DispWithArrows)}).

\begin{verbatim}
\str_new:N \_@@\_format\_str
\end{verbatim}

The option \_@@\_subequations\_bool corresponds to the option \texttt{subequations}.

\begin{verbatim}
\bool_new:N \_@@\_subequations\_bool
\end{verbatim}

The dimension \_@@\_arrow\_width\_dim is only for the arrows of type \texttt{up} and \texttt{down}. A value of \texttt{\c\_max\_dim} means that the arrow has the maximal possible width. A value of \texttt{0 pt} means that the arrow has a width adjusted to the content of the node.

\begin{verbatim}
\dim_new:N \_@@\_arrow\_width\_dim
\dim_set_eq:NN \_@@\_arrow\_width\_dim \c\_max\_dim
\end{verbatim}

The parameter \_@@\_up\_and\_down\_radius\_dim corresponds to option \texttt{radius_for_up_and_down}.

\begin{verbatim}
\dim_new:N \_@@\_up\_and\_down\_radius\_dim
\dim_set:Nn \_@@\_up\_and\_down\_radius\_dim { 4 pt }
\end{verbatim}

The sequence \_@@\_o\_arrows\_seq will be used to store the numbers of the arrows which are of type \texttt{o} (for \texttt{over}) (they are drawn \texttt{after} the other arrows).

\begin{verbatim}
\seq_new:N \_@@\_o\_arrows\_seq
\end{verbatim}
The dimension \_\_\_\_xoffset_for_o_arrows_dim is the xoffset added when drawing an arrow of type o (for over).

\dim_new:N \_\_\_\_xoffset_for_o_arrows_dim
\dim_set:Nn \_\_\_\_xoffset_for_o_arrows_dim { 2 \text{ mm} }

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

\bool_new:N \_\_\_\_right_overlap_bool
\bool_set_true:N \_\_\_\_right_overlap_bool

12.7 The definition of the options

There are four levels where options can be set:

- with \usepackage{witharrows}: this level will be called package level;
- with \WithArrowsOptions{...}: this level will be called global level\(^{31}\);
- with \begin{WithArrows}{...}: this level will be called environment level;
- with \Arrow{...} (included in code-after): this level will be called local level.

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

\cs_new_protected:Npn \_\_\_eval_if_allowed:n #1

\cs_new_protected:Npn \_\_\_fix_pos_option:n #1

First a set of keys that will be used at the global or environment level of options.

\keys_define:nn { WithArrows / Global }

\cs_new_protected:Npn \_\_\_eval_if_allowed:n #1

31 This level is called global level but the settings done by \WithArrowsOptions are local in the TeX sense: their scope corresponds to the current TeX group.
\str_set:Nn \l_@@_command_name_str { #1 }
\str_set:Nx \l_@@_string_Arrow_for_msg_str { \c_backslash_str Arrow\~alias\~\c_backslash_str #1 },
command-name .value_required:n = true,
tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
tikz-code .initial:n = \draw-(#1)-to-node(#3)-(#2); ,
tikz-code .value_required:n = true,
displaystyle .bool_set:N = \l_@@_displaystyle_bool ,
displaystyle .default:n = true,
show-nodes .code:n = \tikzset { @@_node_style / .append\~style = { draw , red } },
show-node-names .bool_set:N = \l_@@_show_node_names_bool ,
show-node-names .default:n = true,
group .code:n = \str_if_empty:NTF \l_@@_previous_key_str
{ \str_set:Nn \l_@@_previous_key_str { group }
 \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
 \int_set:Nn \l_@@_pos_arrow_int 7
}
\{ \@@_error:n { Incompatible\~options } },
group .value_forbidden:n = true ,
groups .code:n = \str_if_empty:NTF \l_@@_previous_key_str
{ \str_set:Nn \l_@@_previous_key_str { groups }
 \seq_if_in:NnF \l_@@_options_Arrow_seq { new\~group }
 \seq_put_right:Nn \l_@@_options_Arrow_seq { new\~group }
 \seq_remove_all:Nn \l_@@_options_Arrow_seq { xoffset }
 \int_set:Nn \l_@@_pos_arrow_int 6
}
\{ \@@_error:n { Incompatible\~options } },
groups .value_forbidden:n = true ,
tikz .code:n = \tikzset { WithArrows / arrow / .append\~style = { #1 } },
tikz .initial:n = \c_empty_tl ,
tikz .value_required:n = true ,
rr .code:n = \@@_fix_pos_option:n 3 ,
rr .value_forbidden:n = true ,
ll .code:n = \@@_fix_pos_option:n 1 ,
ll .value_forbidden:n = true ,
rl .code:n = \@@_fix_pos_option:n 2 ,
rl .value_forbidden:n = true ,
lr .code:n = \@@_fix_pos_option:n 0 ,
lr .value_forbidden:n = true ,
i .code:n = \@@_fix_pos_option:n 5 ,
i .value_forbidden:n = true ,
offset .dim_set:N = \l_@@_offset_dim ,
offset .value_required:n = true ,
offset .initial:n = 3 mm ,
jot .dim_set:N = \jot ,
jot .value_required:n = true ,
interline .skip_set:N = \l_@@_interline_skip ,
start-adjust .dim_set:N = \l_@@_start_adjust_dim ,
start-adjust .initial:n = 0.4 ex ,
start-adjust .value_required:n = true ,
end-adjust .dim_set:N = \l_@@_end_adjust_dim ,
end-adjust .initial:n = 0.4 ex ,
end-adjust .value_required:n = true ,
adjust .meta:n = { start-adjust = #1 , end-adjust = #1 } ,
adjust .value_required:n = true ,
up-and-down .code:n = \keys_set:nn { WithArrows / up\~and\~down } { #1 } ,
up-and-down .value_required:n = true ,
With the option no-arrows, the arrows won’t be drawn. However, the “first pass” of the arrows is done and some errors may be detected. The nullification of \@@_draw_arrows:nn is for the standard arrows and the nullification of \@@_draw_arrow:nn is for “Arrow in code-after”.

\begin{verbatim}
no-arrows .code:n = \cs_set_eq:NN \@@_draw_arrows:nn \use_none:nn \cs_set_eq:NN \@@_draw_arrow:nn \use_none:nn ,
no-arrows .value_forbidden:n = true }
\end{verbatim}

Now a set of keys specific to the environments \{WithArrows\} (and not \{DispWithArrow\}). Despite its name, this set of keys will also be used in \WithArrowsOptions.

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

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

\begin{verbatim}
\clist_new:N \c_@@_ext_delimiters_clist \clist_set:Nn \c_@@_ext_delimiters_clist { , \{, (, \[, \lbrace, \lbrack, \lgroup, \langle, \lmoustache, \lceil, \lfloor
\} \AtBeginDocument
\begin{verbatim}
{ \bool_set_false:N \l_tmpa_bool \IfPackageLoadedTF { amsmath } { \bool_set_true:N \l_tmpa_bool } { }
\IfPackageLoadedTF { unicode-math } { \bool_set_true:N \l_tmpa_bool } { }
\bool_if:NT \l_tmpa_bool
\{ \clist_put_right:Nn \c_@@_ext_delimiters_clist { \lvert, \lVert } \}
\end{verbatim}
\end{verbatim}

Now a set of keys specific to the environments \{DispWithArrows\} and \{DispWithArrows*\} (and not \{WithArrows\}). Despite its name, this set of keys will also be used in \WithArrowsOptions.

\begin{verbatim}
\keys_define:nn { WithArrows / DispWithArrowsSpecific }
{ fleqn .bool_set:N = \l_@@_fleqn_bool ,
fleqn .default:n = true ,
mathindent .skip_set:N = \l_@@_mathindent_skip ,
mathindent .initial:n = 25 pt ,
mathindent .value_required:n = true ,
\end{verbatim}

Since the option subequations is an option which insert the environment \{DispWithArrows\} in an environment \{subequations\} of \amsmath, we must test whether the package \amsmath is loaded.

\begin{verbatim}
subequations .code:n = \IfPackageLoadedTF { \amsmath }
\end{verbatim}

43
\bool_set_true:N \l_@@_subequations_bool
\@@_error:n { amsmath~not~loaded }
\group_begin:
\globaldefs = 1
\@@_msg_redirect_name:nn { amsmath~not~loaded } { info }
\group_end:
\subequations .default:n = true,
\subequations .value_forbidden:n = true,
\nonumber .meta:n = notag,
\allow-multiple-labels .code:n =
\@@_msg_redirect_name:nn { Multiple~labels } { none },
\allow-multiple-labels .value_forbidden:n = true,
\tagged-lines .code:n =
\clist_set:Nn \l_@@_tags_clist { #1 }
\clist_if_in:NnT \l_@@_tags_clist { first }
{ \clist_remove_all:Nn \l_@@_tags_clist { first }
\clist_put_left:Nn \l_@@_tags_clist 1 }
\tagged-lines .value_required:n = true,
\langle/LaTeX\rangle
\wrap-lines .bool_set:N = \l_@@_wrap_lines_bool,
\wrap-lines .default:n = true,
\replace-left-brace-by .code:n =
\tl_set:Nx \l_tmpa_tl { \tl_head:n { #1 } }
\clist_if_in:NVTF \c_@@_ext_delimiters_clist \l_tmpa_tl
{ \@@_error:n { Bad~value~for~replace~brace~by } }
\replace-left-brace-by .initial:n = \lbrace,
\langle/LaTeX\rangle
\standard-behaviour-with-items .bool_set:N = \l_@@_sbwi_bool,
\standard-behaviour-with-items .default:n = true
\langle/LaTeX\rangle
\keys_define:nn { WithArrows / Env }
{ \name .code:n =
\str_set:Nn \l_tmpa_str { #1 }
\seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
{ \@@_error:n { Duplicate~name } }
{ \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
\str_set_eq:NN \l_@@_name_str \l_tmpa_str,
\name .value_required:n = true,
\code-before .code:n = \tl_put_right:Nn \l_@@_code_before_tl { #1 },
\code-before .value_required:n = true,
\CodeBefore .meta:n = { code-before = #1 },
\code-after .code:n = \tl_put_right:Nn \l_@@_code_after_tl { #1 },
\code-after .value_required:n = true,
\langle/LaTeX\rangle
\name .code:n =
\str_set:Nn \l_tmpa_str { #1 }
\seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
{ \@@_error:n { Duplicate~name } }
{ \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
\str_set_eq:NN \l_@@_name_str \l_tmpa_str,
\name .value_required:n = true,
CodeAfter .meta:n = { code-after = #1 },
format .code:n =
\tl_if_empty:nTF { #1 }
{ \@@_error:n { Invalid-option-format } }
{ \regex_match:nnTF { \A[rclRCL]*\Z } { #1 }
{ \tl_set:Nn \l_@@_format_str { #1 } }
{ \@@_error:n { Invalid-option-format } }
},
format .value_required:n = true
)

Now, we begin the construction of the major sets of keys, named “WithArrows / WithArrows”,
“WithArrows / DispWithArrows” and “WithArrows / WithArrowsOptions”. Each of these sets of
keys will be completed after.
\keys_define:nn { WithArrows }
{ WithArrows .inherit:n =
{ WithArrows / Global ,
WithArrows / WithArrowsSpecific ,
WithArrows / Env }
},
WithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
DispWithArrows .inherit:n =
{ WithArrows / DispWithArrowsSpecific ,
WithArrows / Global ,
WithArrows / Env ,
},
DispWithArrows / up-and-down .inherit:n = WithArrows / up-and-down ,
WithArrowsOptions .inherit:n =
{ WithArrows / Global ,
WithArrows / WithArrowsSpecific ,
WithArrows / DispWithArrowsSpecific ,
},
WithArrowsOptions / up-and-down .inherit:n = WithArrows / up-and-down
}

A sequence of str for the options available in {WithArrows}. This sequence will be used in the error
messages and can be modified dynamically.
\seq_new:N \l_@@_options_WithArrows_seq
\@@_set_seq_of_str_from_clist:Nn \l_@@_options_WithArrows_seq
{ adjust, b, c, code-after, code-before, command-name,
right-overlap, displaystyle, end-adjust,
format, group, groups, i,
interline, jot, ll,
lr, max-length-of-arrow, more-columns, name,
no-arrows, rl, rr, up-and-down,
show-node-names, show-nodes, start-adjust,
t, tikz, tikz-code,
oxoffset, ygap, ystart
}

\keys_define:nn { WithArrows / WithArrows }
{ unknown .code:n =
\@@_sort_seq:N \l_@@_options_WithArrows_seq
\@@_error:n { Unknown-option-WithArrows }
A sequence of the options available in \texttt{DispWithArrows}. This sequence will be used in the error messages and can be modified dynamically.

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

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

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

The command \texttt{\_\_\_set_independent:} is a command without argument that will be used to specify that the arrow will be “independent” (of the potential groups of the option \texttt{group} or \texttt{groups}). This information will be stored in the field “status” of the arrow. Another possible value of the field “status” is “new-group”.

\begin{verbatim}
\cs_new_protected:Npn \_\_\_set_independent: 
\{ 
  \str_if_eq:VnF \l_keys_value_tl { NoValue } 
  \{ \_\_\_error:n \{ Value-for-a-key \} \}
\} /*LaTeX*/
\end{verbatim}
The command \@@_set_independent_bis: is the same as \@@_set_independent: except that the key may be used with a value.

\cs_new_protected:Npn \@@_set_independent_bis:n {
  \str_if_empty:NTF \l_@@_previous_key_str {
    \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
    \str_set:Nn \l_@@_status_arrow_str { independent }
  }
  \@@_error:n { Incompatible-options-inArrow }
}

The options of an individual arrow are parsed twice. The first pass is when the command \Arrow is read. The second pass is when the arrows are drawn (after the end of the environment \WithArrows or \DispWithArrows). Now, we present the set of keys for the first pass. The main goal is to extract informations which will be necessary during the scan of the arrows. For instance, we have to know if some arrows are “independent” or use the option “new-group”.

\keys_define:nn { WithArrows / Arrow / FirstPass } {
  \jump .code:n = \int_compare:nTF { #1 > 0 } {
    \int_set:Nn \l_@@_jump_int { #1 } }
  \jump .value_required:n = true,
  \rr .code:n = \@@_set_independent: ,
  \ll .code:n = \@@_set_independent: ,
  \rl .code:n = \@@_set_independent: ,
  \lr .code:n = \@@_set_independent: ,
  \i .code:n = \@@_set_independent: ,
  \rr .default:n = NoValue ,
  \ll .default:n = NoValue ,
  \rl .default:n = NoValue ,
  \lr .default:n = NoValue ,
  \i .default:n = NoValue ,
  \new-group .value_forbidden:n = true ,
  \new-group .code:n = \int_compare:nTF { \l_@@_pos_arrow_int = 6 } {
    \str_set:Nn \l_@@_status_arrow_str { new-group }
  } { \@@_error:n { new-group~without~groups } },
  \o .code:n = \str_if_empty:NTF \l_@@_previous_key_str {
    \int_compare:nNnTF \l_@@_pos_arrow_int < 6 {
      \@@_error:n { invalid-key-o }
    } {
      \str_set:Nn \l_@@_status_arrow_str { over }
      \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
    }
  } { \@@_error:n { Incompatible-options-inArrow } },
  \tikz-code .code:n = \prg_do_nothing: ,
  \tikz-code .value_required:n = true ,
  \tikz .code:n = \prg_do_nothing: ,
  \tikz .value_required:n = true ,
  \start-adjust .code:n = \prg_do_nothing: ,
  \start-adjust .value_required:n = true ,
  \end-adjust .code:n = \prg_do_nothing: ,
  \end-adjust .value_required:n = true ,
}

The other keys don’t give any information necessary during the scan of the arrows. However, you try to detect errors and that’s why all the keys are listed in this keys set. An unknown key will be detected at the point of the command \Arrow and not at the end of the environment.
adjust .code:n = \prg_do_nothing: ,
adjust .value_required:n = true ,
oxoffset .code:n = ,
unknown .code:n = \@@_sort_seq:N \l_@@_options Arrow_seq
\seq_if_in:NVTF \l_@@_options WithArrows_seq \l_keys_key_str
{ \str_set:Nn \l_tmpa_str
  { -However,-this-key-can-be-used-in-the-options-of-{WithArrows}. }
}
{ \str_clear:N \l_tmpa_str }
\@@_error:n { Unknown-option-in-Arrow }
\seq_new:N \l_@@_options Arrow_seq
\@@_set_seq_of_str_from_clist:Nn \l_@@_options Arrow_seq
{ adjust, end-adjust, i, jump, ll, lr, o, rl, rr, start-adjust, tikz, tikz-code, xoffset }
\cs_new_protected:Npn \@@_fix_pos_arrow:n #1
{ \str_if_empty:NT \l_@@_previous_key_str
  { \str_set_eq:NN \l_@@_previous_key_str \l_keys_key_str
    \int_set:Nn \l_@@_pos_arrow_int { #1 }
  }
}
The options of the individual commands \Arrows are scanned twice. The second pass is just before
the drawing of the arrow. In this set of keys, we don’t put an item for the unknown keys because an
unknown key would have been already detected during the first pass.
\keys_define:nn {WithArrows / Arrow / SecondPass }
{ \tikz-code .tl_set:N = \l_@@_tikz_code_tl ,
  \tikz-code .initial:n = \draw(#1)-to-node(#3)-(#2)-; ,
  \tikz .code:n = \tikzset { WithArrows / arrow / .append-style = { \l_@@_tikz_code_tl } } ,
  \rr .code:n = \@@_fix_pos_arrow:n 3 ,
  \ll .code:n = \@@_fix_pos_arrow:n 1 ,
  \rl .code:n = \@@_fix_pos_arrow:n 2 ,
  \lr .code:n = \@@_fix_pos_arrow:n 0 ,
  \i .code:n = \@@_fix_pos_arrow:n 5 ,
  \o .code:n = \str_set:Nn \l_@@_previous_key_str { o } ,
}
The option xoffset is not allowed when the option group or the option groups is used except, if the
arrow is independent or if there is only one arrow.
oxoffset .code:n = \bool_if:nTF
{ \int_compare_p:nNn \g_@@_arrow_int > 1
  \int_compare_p:nNn \l_@@_pos_arrow_int > 5
  \str_if_eq_p:Vn \l_@@_status_arrow_str { independent }
}
{ \@@_error:n { Option-xoffset-forbidden } }
{ \dim_set:Nn \l_@@_xoffset_dim { #1 } },
The command \Arrow

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

\WithArrowsOptions is the command of the witharrows package to fix options at the document level. It’s possible to fix in \WithArrowsOptions some options specific to \{WithArrows\} (in contrast with \{DispWithArrows\}) or specific to \{DispWithArrows\} (in contrast with \{WithArrows\}). That’s why we have constructed a set of keys specific to \WithArrowsOptions.

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

\int_gincr:N \g_@@_arrow_int

We will construct a global property list to store the informations of the considered arrow. The six fields of this property list are “initial”, “final”, “status”, “options”, “label” and “input-line”. In order to compute the value of “final” (the destination row of the arrow), we have to take into account
a potential option jump. In order to compute the value of the field “status”, we have to take into account options ll, rl, rr, lr, etc. or new-group.
We will do that job with a first analyze of the options of the command \Arrow with a dedicated set of keys called WithArrows/Arrow/FirstPass.

\str_clear_new:N \l_@@_previous_key_str
\keys_set:nn { WithArrows / Arrow / FirstPass } { #1 , #3 }

We construct now a global property list to store the informations of the considered arrow with the six fields “initial”, “final”, “status”, “options”, “label” and “input-line”.

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

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

2. The row where the arrow ends (that’s why it was necessary to analyze the key jump):

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

3. The “status” of the arrow, with 4 possible values: empty, independent, new-group or over.

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

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

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

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

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

6. The number of the line where the command \Arrow is issued in the TeX source (as of now, this is only useful for some error messages).

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

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

\prop_put:Nnn \l_tmpa_prop { width } { 0 pt }

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

\prop_gclear_new:c
{ \g_@@_arrow_ \l_@@_prefix_str_ \int_use:N \g_@@_arrow_int_ prop }
\prop_gset_eq:cN
{ \g_@@_arrow_ \l_@@_prefix_str_ \int_use:N \g_@@_arrow_int_ prop }
\l_tmpa_prop

The command \Arrow (or the corresponding command with a name given by the user with the option command-name) will be available only in the last column of the environments \{WithArrows\} and \{DispWithArrows\}. In the other columns, the command will be linked to the following command \@@_Arrow_first_columns: which will raise an error.

\cs_new_protected:Npn \@@_Arrow_first_columns:
{ \@@_error:n { Arrow~not~in~last~column } \@@_Arrow }
12.9 The environments \texttt{\{WithArrows\}} and \texttt{\{DispWithArrows\}}

12.9.1 Code before the \texttt{\textbackslash{halign}}

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

First, the initialization of \texttt{\l_@@\_type\_env\_str} which is the name of the encompassing environment. In fact, this token list is used only in the error messages.

We deactivate the potential externalization of Tikz. The Tikz elements created by \texttt{\witharrows} can’t be externalized since they are created in Tikz pictures with \texttt{overlay} and \texttt{remember picture}.

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

The parameter \texttt{\l_@@\_status\_arrow\_str} will be used to store the “status” of an individual arrow. It will be used to fill the field “status” in the property list describing an arrow.

The dimension \texttt{\l_@@\_x\_dim} will be used to compute the \texttt{x}-value for some vertical arrows when one of the options \texttt{i}, \texttt{group} and \texttt{groups} (values 5, 6 and 7 of \texttt{\l_@@\_pos\_arrow\_int}) is used.

The variable \texttt{\l_@@\_input\_line\_str} will be used only to store, for each command \texttt{\Arrow} the line (in the TeX file) where the command is issued. This information will be stored in the field “input-line” of the arrow. As of now, this information is used only in some error messages.

Initialization of \texttt{\g_@@arrow\_int}, \texttt{\g@@line\_int}, \texttt{\g_@@col\_int} and \texttt{\g_@@static\_col\_int}. However, we have to save their previous values with the stacks created for this end.

In the preamble of the \texttt{\textbackslash{halign}}, there will be two counters of the columns. The aim of this program-\texttt{\textbackslash{}mation is to detect the use of a command \texttt{\textbackslash{}omit} in a cell of the \texttt{\textbackslash{}halign} (it should be forbidden). For example, in the part of the preamble concerning the third column (if there is a third column in the environment), we will have the following instructions:

\begin{verbatim}
\int_gincr:N \g_\textbackslash{}col\_int
\int_set:NN \g_\textbackslash{}static\_col\_int 3
\end{verbatim}
The counter \g_@@_col_int is incremented dynamically and the second is static. If the user has used a command \omit, the dynamic incrementation is not done in the cell and, at the end of the row, the difference between the counters may infer the presence of \omit at least once.

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

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

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

These counters will be used later as variables.

The counter \l_@@_jump_int corresponds to the option jump. Now, we set the initial value for this option.

The string \l_@@_format_str corresponds to the option format. Now, we set the initial value for this option.

In (the last column of) \{DispWithArrows\}, it’s possible to put several labels (for the same number of equation). That’s why these labels will be stored in a sequence \l_@@_labels_seq.

The value corresponding to the key interline is put to zero before the treatment of the options of the environment.\footnote{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 code-before is put to nil before the treatment of the options of the environment, because, of course, we don’t want the code executed at the beginning of all the nested environments \{WithArrows\}. Idem for code-after.
We process the options given to the environment \{WithArrows\} or \{DispWithArrows\}.

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

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

\begin{verbatim}
\bool_if:NF \l_@@_right_overlap_bool
  \bool_if:NT \l_@@_in_WithArrows_bool
    \dim_gzero_new:N \g@@overlap_x_dim
    \dim_zero_new:N \l@@delta_x_dim
  \end{verbatim}

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

\begin{verbatim}
\cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_first_columns:
\end{verbatim}

It’s only in the last column of the environment that it will be linked to the command \@@_Arrow:. The counter \l@@nb_cols_int is the number of columns in the \halign (excepted the column for the labels of equations in {DispWithArrows} and excepted eventuals other columns in \{WithArrows\} allowed by the option more-columns).

\begin{verbatim}
\int_set:Nn \l_@@_nb_cols_int { \str_count:N \l_@@_format_str }
\end{verbatim}

Be careful! The following counter \g@@col_int will be used for two usages:

- during, the construction of the preamble of the \halign, it will be used as counter for the number of the column under construction in the preamble (since the preamble is constructed backwards, \g@@col_int will go decreasing from \l@@nb_cols_int to 1);
- once the preamble constructed, the primitive \halign is executed, and, in each row of the \halign, the counter \g@@col_int will be increased from column to column.

\begin{verbatim}
\int_gset_eq:NN \g@@col_int \l@@nb_cols_int
\end{verbatim}

We convert the format in a sequence because we use it as a stack (with the top of the stack at the end of the sequence) in the construction of the preamble.

\begin{verbatim}
\seq_clear_new:N \l@@format_seq
\seq_set_split:NnV \l@@format_seq { } \l@@format_str
\end{verbatim}

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).

\begin{verbatim}
\{\LaTeX\}
\bool_if:NT \c@@footnote_bool { \begin { savenotes } }
\end{verbatim}

\begin{verbatim}
\LaTeX\}
\end{verbatim}

We execute the code \l@@code_before_tl of the option code-before of the environment after the potential \begin{savenotes} and, symetrically, we will execute the \l@@code_after_tl before the potential \end{savenotes} (we have a good reason for the last point: we want to extract the footnotes of the arrows executed in the code-after).

\begin{verbatim}
\l@@code_before_tl
\end{verbatim}
12.9.2 The construction of the preamble of the \halign

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

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

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

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

We begin the construction of a generic column.

We redefine the command \Arrow (or the name given to the corresponding command by the option \command-name) in each cell of the last column. The braces around \l_@@_command_name_str are mandatory because \l_@@_command_name_str will be expanded by the \use:e and the command \cs_set_eq:cN must still be efficient during the execution of the \halign.
The command \@@_test_if_to_tag: (which is protected and, thus, will not be expanded during the construction of the preamble) will test, at each row, whether the current row must be tagged (and the tag will be put in the very last column).

\@@_test_if_to_tag:

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

\IfPackageLoadedTF { amsmath } { \@@_set_qedhere: } { }

⟨\LaTeX⟩

\str_if_eq:VnT \l_@@_type_col_str { c } \hfil
\str_if_eq:VnT \l_@@_type_col_str { C } \hfil
\str_if_eq:VnT \l_@@_type_col_str { r } \hfill
\str_if_eq:VnT \l_@@_type_col_str { R } \hfill
\int_gincr:N \g_@@_col_int
\int_gset:Nn \g_@@_static_col_int { \int_use:N \g_@@_col_int }
\c_math_toggle_token
\str_if_eq:VnT \l_@@_type_col_str { C } { { } }
\str_if_eq:VnT \l_@@_type_col_str { L } { { } }
\bool_if:NT \l_@@_displaystyle_bool \displaystyle
\str_if_eq:VnT \l_@@_type_col_str { C } { { } }
\str_if_eq:VnT \l_@@_type_col_str { R } { { } }
\c_math_toggle_token
\int_compare:nNnTF \g_@@_col_int = \l_@@_nb_cols_int
\@@_construct_nodes:
{

The following glue (\hfil) will be added only if we are not in the last cell because, in the last cell, a glue (=skip) is added between the nodes (in \@@_construct_nodes:).

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

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

\bool_if:NTF \l_@@_in_WithArrows_bool
{
\ialign
\bgroup

\halign to \l_@@_linewidth_dim
\bgroup
\bool_if:NT \l_@@_fleqn_bool { \skip_horizontal:N \l_@@_mathindent_skip }

\int_gincr:N \g_@@_line_int
\int_gzero:N \g_@@_col_int
\tl_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
{ \skip_horizontal:n \box_wd:N \l_@@_left_brace_box + \l_@@_delim_wd_dim }
& }
\strut
}
}

{55}
The command \_construct_nodes: is only for the lisibility of the code because, in fact, it is used only once. It constructs the “left node” and the “right node” at the end of each row of the arrow.

\cs_new_protected:Npn \_construct_nodes:
  {
  
  We create the “left node” of the line (when using macros in Tikz node names, the macros have to be fully expandable: here, \_int_use:N is fully expandable).

\tikz [remember-picture, overlay]
\node [
  node-contents = { },
  \_node_style,
  name = wa - \_l_\_prefix_str - \_int_use:N \_g_\_line_int - l ,
]
\hfil

Now, after the \hfil, we create the “right node” and, if the option show-node-names is raised, the name of the node is written in the document (useful for debugging).

\tikz [remember-picture, overlay]
\node [
  node-contents = { },
  \_node_style,
  name = wa - \_l_\_prefix_str - \_int_use:N \_g_\_line_int - r ,
]
\str_if_empty:NF \_l_\_name_str
  {
  \pgfpicture
  \pgfnodealias { \_l_\_name_str - \_int_use:N \_g_\_line_int - l }
  \pgfnodealias { wa - \_l_\_prefix_str - \_int_use:N \_g_\_line_int - l }
  \pgfnodealias { \_l_\_name_str - \_int_use:N \_g_\_line_int - r }
  \pgfnodealias { wa - \_l_\_prefix_str - \_int_use:N \_g_\_line_int - r }
  \endpgfpicture
  \bool_if:NT \_l_\_show_node_names_bool
  {
  \hbox_overlap_right:n
  \small wa - \_l_\_prefix_str - \_int_use:N \_g_\_line_int - r }
  \}

12.9.3 The environment \{WithArrows\}

(\LaTeX)
\NewDocumentEnvironment { WithArrows } { ! O { } } { }
(/\LaTeX)
(plain-\TeX)
\cs_new_protected:Npn \WithArrows
  {
  \group_begin:
  \peek_meaning:NTF [
    { \WithArrows_i }]
  \WithArrows_i [ ]
  }
\cs_new_protected:Npn \WithArrows_i [ #1 ]
(/plain-\TeX)
  {
  \bool_set_true:N \_l_\_in_WithArrows_bool

56
The environment begins with a \vtop, a \vcenter or a \vbox\[^{33}\] 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\[^{34}\] and therefore, we can use \vcenter.

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

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).

\begin{verbatim}
\spread@equation
\end{verbatim}

We begin the \halign and the preamble. During the construction of the preamble, \l_tmpa_int will be incremented during each column constructed.

\begin{verbatim}
\@@_construct_halign:
\end{verbatim}

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

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

\begin{verbatim}
\@@_error:n { Too-much-columns-in-WithArrows }
\c_math_toggle_token
\bool_if:NT \l_@@_displaystyle_bool \displaystyle
{ ## }
\c_math_toggle_token
\cr
\egroup
\end{verbatim}

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

\begin{verbatim}
\cs_new_protected:Npn \endWithArrows
\end{verbatim}

\begin{verbatim}
\bool_if:NTF \l_@@_right_overlap_bool \endWithArrows
\end{verbatim}

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

\begin{verbatim}
\bool_if:NF \l_@@_right_overlap_bool
\end{verbatim}

\[^{33}\]Notice that the use of \vtop seems color-safe here...

\[^{34}\]An error is raised if the environment is used outside math mode.
If the option \texttt{footnote} or the option \texttt{footnotehyper} is used, then we extract the footnotes with an environment \texttt{\{footnote\}} (of the package \texttt{footnote} or the package \texttt{footnotehyper}).

\begin{verbatim}
\bool_if:NT \c_@@_footnote_bool \{ \end { savenotes } \}
\end{verbatim}

This is the end of the environment \texttt{\{WithArrows\}}.

\subsection*{12.9.4 After the construction of the \texttt{\halign}}

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

\begin{verbatim}
\cs_new_protected:Npn \@@_post_halign: 
\end{verbatim}

The command \texttt{\WithArrowsRightX} is not used by \texttt{witharrows}. It’s only a convenience given to the user.

\begin{verbatim}
\cs_set:Npn \WithArrowsRightX { \g_@@_right_x_dim }
\end{verbatim}

We use \texttt{\normalbaselines} of plain-TeX because we have used \texttt{\spread@equation} (of \texttt{amsmath} or defined directly if \texttt{amsmath} is not loaded) and you don’t want \texttt{\spread@equation} to have effects in the labels of the arrows.

\begin{verbatim}
\normalbaselines
\end{verbatim}

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

\begin{verbatim}
\int_compare:nNnT \g_@@_arrow_int > 0 
\end{verbatim}

If there is only one arrow, the options \texttt{group} and \texttt{groups} do not really make sense and it will be quicker to act as if we were in option \texttt{i} (moreover, it allows the option \texttt{xoffset} for the unique arrow).

\begin{verbatim}
\int_compare:nNnT \g_@@_arrow_int = 1 
\end{verbatim}

\begin{verbatim}
\int_compare:nNnT \l_@@_pos_arrow_int > 5 
\end{verbatim}

\begin{verbatim}
\int_set:Nn \l_@@_pos_arrow_int 5 
\end{verbatim}

\begin{verbatim}
\@@_scan_arrows: 
\end{verbatim}

We will execute the code specified in the option \texttt{code-after}, after some settings.

\begin{verbatim}
\group_begin: 
\tikzset { every-picture / .style = @@_standard }
\end{verbatim}

The command \texttt{\WithArrowsNbLines} is not used by \texttt{witharrows}. It’s only a convenience given to the user.

\begin{verbatim}
\cs_set:Npn \WithArrowsNbLines \{ \int_use:N \g_@@_line_int \} 
\end{verbatim}
The command \MultiArrow is available in code-after, and we have a special version of \Arrow, called “\Arrow in code-after” in the documentation.\footnote{As of now, \MultiArrow has no option, and that’s why its internal name is a name of L3 with the signature :nn whereas \Arrow in code-after provides options and has the name of a function defined with \NewDocumentCommand.}

\begin{verbatim}
\cs_set_eq:NN \MultiArrow \@@_MultiArrow:nn
\cs_set_eq:cN \l_@@_command_name_str \@@_Arrow_code_after
\bool_set_true:N \l_@@_in_code_after_bool
\l_@@_code_after_tl\group_end:
\end{verbatim}

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

\begin{verbatim}
\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 } }
\end{verbatim}

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

\begin{verbatim}
\int_compare:nNnT { \seq_count:N \g_@@_position_in_the_tree_seq } = 1
{ \int_gincr:N \g_@@_last_env_int }
\end{verbatim}

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

\begin{verbatim}
\seq_gpop_right:NN \g_@@_arrow_int_seq \l_tmpa_tl
\int_gset:Nn \g_@@_arrow_int \l_tmpa_tl
\seq_gpop_right:NN \g_@@_line_int_seq \l_tmpa_tl
\int_gset:Nn \g_@@_line_int \l_tmpa_tl
\seq_gpop_right:NN \g_@@_col_int_seq \l_tmpa_tl
\int_gset:Nn \g_@@_col_int \l_tmpa_tl
\seq_gpop_right:NN \g_@@_static_col_int_seq \l_tmpa_tl
\int_gset:Nn \g_@@_static_col_int \l_tmpa_tl
\end{verbatim}

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

\subsection{The command of end of row}

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

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

\begin{verbatim}
\cs_new_protected:Npn \@@_cr:
{ \scan_stop:
  \int_compare:nNnF \g_@@_col_int = \g_@@_static_col_int
  { \@@_error:n { omit~probably~used } }
  \prg_replicate:nn { \l_@@_nb_cols_int - \g_@@_static_col_int } { & { } }
  \group_align_safe_begin:
  \peek_meaning_remove:NTF \@@_cr_i: \@@_cr_i:
  }
\end{verbatim}
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.

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

The analyse of the argument of the token \end must be after the \group_align_safe_end: which is the beginning of \@@_cr_iii:n.

For the environment \{DispWithArrows\}, the behaviour of \ is different because we add the last column which is the column for the tag (number of the equation). Even if there is no tag, this column is used for the v-nodes.\footnote{The v-nodes are used to compute the abscissa of the right margin, used by the option \texttt{wrap-lines}.}

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

Here, we can’t use \texttt{\refstepcounter{equation}} because if the user has issued a \texttt{\tag} command, we have to use \texttt{\l_@@_tag_tl} and not \texttt{\theequation}. That’s why we have to do the job done by \texttt{\refstepcounter} manually.

First, the incrementation of the counter (potentially).

We store in \texttt{\g_tmpa_tl} the tag we will have to compose at the end of the line. We use a global variable because we will use it in the \texttt{next} cell (after the &).

\footnote{The v-nodes are used to compute the abscissa of the right margin, used by the option \texttt{wrap-lines}.}
It’s possible to put several labels for the same line (it’s not possible in the environments of amsmath). That’s why the different labels of a same line are stored in a sequence \l_@@_labels_seq.

\seq_if_empty:NF \l_@@_labels_seq
\{

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

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

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

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

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

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

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

Now, we can issue the command \label (some packages may have redefined \label, for example typedref) for each item in the sequence of the labels (it’s possible with witharrows to put several labels to the same line and that’s why the labels are in the sequence \l_@@_labels_seq).

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

We save the booleans \l_@@_tag_star_bool and \l_@@_qedhere_bool because they will be used in the next cell (after the &). We recall that the cells of a \halign are TeX groups.

\@@_save:N \l_@@_tag_star_bool
\@@_save:N \l_@@_qedhere_bool
\bool_if:NT \l_@@_tag_next_line_bool
{ \openup -\jot
\bool_set_false:N \l_@@_tag_next_line_bool
\notag \& }

&
\@@_restore:N \l_@@_tag_star_bool
\@@_restore:N \l_@@_qedhere_bool
\bool_if:NT \l_@@_qedhere_bool
{ \hbox_overlap_left:n \@@_qedhere_i: }
\cs_set_eq:NN \theequation \g_tmpa_tl
\bool_if:NT \l_@@_tag_star_bool
{ \cs_set_eq:NN \tagform@ \prg_do_nothing: }
We use \texttt{\@eqnnum} (we recall that there are two definitions of \texttt{\@eqnnum}, a standard definition and another, loaded if the class option \texttt{leqno} is used). However, of course, the position of the \texttt{v}-node is not the same whether the option \texttt{leqno} is used or not. That’s here that we use the flag \texttt{\_\_\_leqno\_bool}.

\begin{verbatim}
\hbox_overlap_left:n
  { \bool_if:NF \c_@@_leqno_bool
    { \pgfpicture
      \pgfrememberpicturepositiononpagetrue
      \pgfcoordinate{ wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
      \pgfpointorigin
      \endpgfpicture
    }
  \quad \@eqnnum
}
\bool_if:NT \c_@@_leqno_bool
  { \pgfpicture
    \pgfrememberpicturepositiononpagetrue
    \pgfcoordinate{ wa - \l_@@_prefix_str - \int_use:N \g_@@_line_int - v }
    \pgfpointorigin
    \endpgfpicture
  }
\endverbatim

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

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

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

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

\subsection*{12.9.6 The environment \texttt{DispWithArrows}}

For the environment \texttt{DispWithArrows}, the general form of the construction is of the type:
\[
\begin{vmatrix}
\halign to \displaywidth { ... } \\
\end{vmatrix}
\]

The purpose of the vtop is to have an environment unbreakable.
However, if we are just after an item of a LaTeX list or at the beginning of a \texttt{minipage}, the construction is slightly different:
\[
\begin{vmatrix}
\halign to \linewidth { ... } \\
\end{vmatrix}
\]

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

\begin{verbatim}
\bool_new:N \l_@@_in_label_or_minipage_bool
\NewDocumentEnvironment { DispWithArrows } { ! d < > ! O { } } 
\cs_new_protected:Npn \DispWithArrows 
\{
\group_begin:
\peek_meaning:NTF <
\{
\DispWithArrows_i 
\}{\DispWithArrows_i < \c_novalue_tl > }
\}
\cs_new_protected:Npn \DispWithArrows_i < #1 >
\{
\peek_meaning:NTF [ 
\{
\DispWithArrows_ii < #1 > 
\}{\DispWithArrows_ii < #1 > [ ] }
\}
\cs_new_protected:Npn \DispWithArrows_ii < #1 > [ #2 ]
\{
\bool_set_true:N \l_@@_in DispWithArrows_bool
\str_set_new:N \l_@@_type_env_str
\str_set:Nn \l_@@_type_env_str { DispWithArrows }
\end{verbatim}

63
Since the version 1.16 of witharrows, no space is added between an \item of a LaTeX list and an environment \begin{DispWithArrows} except with the option standard-behaviour-with-items stored in the boolean \l_@@_sbwi_bool. We have to know if we are just after an \item and this information will be stored in \l_@@_in_label_or_minipage_bool. We have to do this test quickly after the beginning of the environment (in particular, because it must be done before the execution of the code-before\textsuperscript{37}).

\begin{LaTeX}
\bool_if:NF \l_@@_sbwi_bool
\{ \\
\legacy_if:nT { @inlabel } \{ \bool_set_true:N \l_@@_in_label_or_minipage_bool \}
\legacy_if:nT { @minipage } \{ \bool_set_true:N \l_@@_in_label_or_minipage_bool \}
\}
\end{LaTeX}

If mathtools has been loaded with the option showonlyrefs, we disable the code of mathtools for the option showonlyrefs with the command \MT_showonlyrefs_false: (it will be reactivated at the end of the environment).

\begin{LaTeX}
\IfPackageLoadedTF { mathtools }
\{ \\
\MH_if_boolean:nT { show_only_refs } \{ \\
\MT_showonlyrefs_false: \}
\}
\end{LaTeX}

However, we have to re-raise the flag \{show_only_refs\} of mhsetup because it has been switched off by \MT_showonlyrefs_false: and we will use it in the code of the new version of \label.

\begin{LaTeX}
\MH_set_boolean_T:n { show_only_refs }
\end{LaTeX}

An action done by typedref in its redefinition of \refstepcounter. The command \sr@name is a prefix added to the name of the label by the redefinition of \label done by typedref.

\begin{LaTeX}
\IfPackageLoadedTF { typedref }
\{ \\
{ \str_set:Nn \sr@name { equation } } 
\}
\end{LaTeX}

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

\begin{LaTeX}
\IfPackageLoadedTF { amsmath } { \intertext@ } \{ \\
\end{LaTeX}

\exp_args:No \tl_if_novalue:nF \{ \#1 \} \{ \tl_set:Nn \l_@@_left_brace_tl \{ \#1 \} \}
\@_pre_halign:n \{ \#2 \}

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

\begin{LaTeX}
\bool_if:NT \l_@@_subequations_bool \{ \begin { subequations } \}
\end{LaTeX}

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

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

\begin{LaTeX}
\hbox_set:Nn \l_tma_box
\end{LaTeX}

\textsuperscript{37}The code-before is not meant to contain typesetting material. However, it may contain, for example, a \{tikzpicture\} with options overlay and remember picture in order to draw nodes under some elements of the environment \begin{DispWithArrows}.  

64
Even if the default value of `\nulldelimiterspace` is 1.2 pt, we take it into account.

```latex
\dim_zero:N \nulldelimiterspace
\c_math_toggle_token
\left \@@_replace_left_brace_by_tl \vcenter to 1 cm { } \right.
\c_math_toggle_token
\group_end:
\dim_zero_new:N \l_@@_delim_wd_dim
\dim_set:Nn \l_@@_delim_wd_dim { \box_wd:N \l_tmpa_box }
\box_clear_new:N \l_@@_left_brace_box
\hbox_set:Nn \l_@@_left_brace_box
{    
\group_begin:    
\cs_set_eq:NN \label \@@_old_label
\c_math_toggle_token
\bool_if:NT \l_@@_displaystyle_bool \displaystyle
\l_@@_left_brace_tl
{ }
\c_math_toggle_token
\group_end:
}\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_qedhere_bool
\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_tag_star_bool
\if_mode_math:
\@@_fatal:n { DispWithArrows~in~math~mode }
\fi:
\if_mode_vertical:
\nointerlineskip
\hbox_to_wd:nn { .6 \linewidth } { }
\fi:
\c_math_toggle_token \c_math_toggle_token
\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_qedhere_bool
\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_tag_star_bool
\if_mode_math:
\@@_fatal:n { DispWithArrows~in~math~mode }
\fi:
\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_qedhere_bool
\tl_clear_new:N \l_@@_tag_tl
\bool_set_false:N \l_@@_tag_star_bool
```

The token list \l_@@_tag_tl will contain the argument of the command \texttt{\tag}.

We don't use \texttt{\LaTeX} of \LaTeX because some extensions, like \texttt{autonum}, do a redefinition of \texttt{\LaTeX}. However, we put the following lines which are in the definition of \texttt{\LaTeX} even though they are in case of misuse.
The command \texttt{\spread@equation} is the command used by \texttt{amsmath} in the beginning of an alignment to fix the interline. When used, it becomes no-op. However, it’s possible to use \texttt{witharrows} without \texttt{amsmath} since we have redefined \texttt{\spread@equation} (if it is not defined yet).

If the user tries to use more columns than the length of the format, we have to raise an error. However, the error won’t be in the next column which is the columns for the labels of the equations. The error will be after... and it must be after. That means that we must not have an error in the next column simply because we are not in math mode. That’s why this column, even if it is for the labels, is in math mode.

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

The following \texttt{\egroup} is for the \texttt{\halign}.

\begin{verbatim}
\egroup
\unskip \unpenalty \unskip \unpenalty
\box_set_to_last:N \l_tmpa_box
\nointerlineskip
\box_use:N \l_tmpa_box
\dim_gzero_new:N \g_@@_alignment_dim
\dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_tmpa_box }
\box_clear_new:N \l_@@_new_box
\hbox_set:Nn \l_@@_new_box { \hbox_unpack_drop:N \l_tmpa_box }
\dim_compare:nNnT { \box_wd:N \l_@@_new_box } < \g_@@_alignment_dim
{ \dim_gset:Nn \g_@@_alignment_dim { \box_wd:N \l_@@_new_box } }
\end{verbatim}
The \egroup is for the box \l_@@_halign_box.

\begin{verbatim}
\l_if_eq:NNTF \l_@@_left_brace_tl \c_novalue_tl
{ \box_use_drop:N \l_@@_halign_box }
\{ \box_to wd:nn \l_@@_linewidth_dim 
{ \bool_if:NTF \l_@@_fleqn_bool 
{ \skip_horizontal:N \l_@@_mathindent_skip }
\hfil
\box_to wd:nn \g_@@_alignment_dim 
{ \box_use_drop:N \l_@@_left_brace_box }
\endverbatim

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

\begin{verbatim}
\dim_set:Nn \l_tmpa_dim
{ \box_ht:N \l_@@_halign_box + \box_dp:N \l_@@_halign_box }
\dim_gzero_new:N \g_@@_right_x_dim
\dim_gset_eq:NN \g_@@_right_x_dim \c_max_dim
\pgfpicture
\pgfrememberpicturepositiononpagetrue
\int_step_variable:nNn \g_@@_line_int \l_tmpa_int
\bottomskip
\cs_if_free:cTF
{ pgf @ sh @ ns @ wa - \l_@@_prefix_str - \l_tmpa_int - v }
\{ \@@_fatal:n { Inexistent-v-node } }
\{ \pgfpointanchor
{ wa - \l_@@_prefix_str - \l_tmpa_int - v }
{ center }
\dim_compare:nNnT \pgf@x < \g_@@_right_x_dim
\{ \dim_gset_eq:NN \g_@@_right_x_dim \pgf@x \}
\endverbatim

We compute the dimension \g_@@_right_x_dim. As a first approximation, \g_@@_right_x_dim is the x-value of the right side of the current composition box. In fact, we must take into account the potential labels of the equations. That's why we compute \g_@@_right_x_dim with the v-nodes of each row specifically built in this goal. \g_@@_right_x_dim is the minimal value of the x-value of these nodes.

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

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

\@@_post_halign:
If \texttt{mathtools} has been loaded with the option \texttt{showonlyrefs}, we reactivate the code of \texttt{mathtools} for the option \texttt{showonlyrefs} with the command \texttt{\MT\_showonlyrefs\_true:} (it has been deactivated in the beginning of the environment).

\begin{verbatim}
\IfPackageLoadedTF { mathtools }
{ \MH_if_boolean:nT { show_only.refs } \MT\_showonlyrefs\_true: }
\end{verbatim}

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

\begin{verbatim}
\IfPackageLoadedTF { \footnote }
{ \ bool_if:NT \c_@@\_footnote_bool \{ \end \{ \savenotes \} \} }
\end{verbatim}

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

\begin{verbatim}
\cs_new_protected:Npn \@@_if_in_last_col_of_disp:Nn #1 #2
{ \bool_if:NTF \l_@@_in_WithArrows_bool
{ \@@_error:nn { Not\_allowed\_in\_WithArrows } { #1 } }
{ \int_compare:nNnTF \g_@@\_col_int < \l_@@\_nb\_cols\_int
{ \@@_error:nn { Not\_allowed\_in\_DispWithArrows } { #1 } }
}
\end{verbatim}

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

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

\begin{verbatim}
\cs_new_protected:Npn \@@\_if\_in\_last\_col\_of\_disp:NN \#1 \#2
{ \bool_if:NTF \l_@@\_in\_DispWithArrows_bool
{ \\@@\_error:nn { Not\_allowed\_in\_DispWithArrows } { \#1 } }
{ \\int_compare:nNnTF \g_@@\_col\_int < \l_@@\_nb\_cols\_int
{ \\@@\_error:nn { Not\_allowed\_in\_DispWithArrows } { \#1 } }
}
\end{verbatim}
The command \@@_notag: will be linked to the command \notag in the environments {WithArrows} and {DispWithArrows}.

\LaTeX
\cs_new_protected:Npn \@@_notag:
\{ \@@_if_in_last_col_of_disp:Nn \notag \{ \clist_clear:N \l_@@_tags_clist \} \}

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

\cs_new_protected:Npn \@@_nonumber:
\{ \@@_if_in_last_col_of_disp:Nn \nonumber \{ \clist_clear:N \l_@@_tags_clist \} \}

The command \@@_tag will be linked to \tag in {WithArrows} and {DispWithArrows}. We do the definition with \NewDocumentCommand because this command has a starred version.

\NewDocumentCommand \@@_tag { s m }
\{ \@@_if_in_last_col_of_disp:Nn \tag \{
\tl_if_empty:NF \l_@@_tag_tl
\{ \@@_error:nn { Multiple~tags } \{ #2 \} \}
\clist_set:Nn \l_@@_tags_clist \{ all \}
\IfPackageLoadedTF { mathtools }\{ 
\MH_if_boolean:nT \show_manual_tags
\{ \MH_if_boolean:nF \show_only_refs
\{ \clist_clear:N \l_@@_tags_clist \}
\}
\}
\\tl_set:Nn \l_@@_tag_tl \{ #2 \}
\bool_set:Nn \l_@@_tag_star_bool \{ #1 \}
\}

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

\bool_if:nT \l_@@_tag_star_bool
\{ #1 \}
\IfPackageLoadedTF \amsmath
\{ \}
\@@_error:n \{ tag*~without~amsmath \}
\}
}

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

\cs_new_protected:Npn \@@_label:n #1
\{ \@@_if_in_last_col_of_disp:Nn \label \{
\seq_if_empty:NF \l_@@_labels_seq
\{ 

39There are two versions of \@eqnnum, a standard version and a version for the option leqno.
The command \@@_tagnextline: will be linked to \tagnextline in {DispWithArrows}.
\cs_new_protected:Npn \@@_tagnextline: {
    \@@_if_in_last_col_of_disp:Nn \tagnextline {
        \bool_set_true:N \l_@@_tag_next_line_bool
    }
}

The environments \{DispWithArrows\} and \{DispWithArrows*\} are compliant with the command \texttt{\textbar{qedhere}} of \texttt{amsthm}. However, this compatibility requires a special version of \texttt{\textbar{qedhere}}.
This special version is called \texttt{\textbar{qedhere}$_{\text{special}}$} and will be linked with \texttt{\textbar{qedhere}} in the last column of the environment \{DispWithArrows\} (only if the package \texttt{amsthm} has been loaded). \texttt{\textbar{qedhere}$_{\text{special}}$} raises the boolean \texttt{\textbar{\l_\@\textbar{qedhere}_\textbar{bool}}$_{\text{special}}$}.
\cs_new_protected:Npn \@@_qedhere: {
    \bool_set_true:N \l_@@_qedhere_bool
}
\cs_new_protected:Npn \@@_set_qedhere: {
    \cs_set_eq:NN \texttt{\textbar{qedhere}} \@@_qedhere: 
}\QED@stack \relax \relax

The line \texttt{\cs_set_eq:NN \texttt{\textbar{qed}@elt} \texttt{\textbar{setQED}@elt}} is a preparation for an action on the QED stack. Despite its form, the instruction \texttt{\textbar{QED}@stack} executes an operation on the stack. This operation prints the QED symbol and nullify the top of the stack.
\cs_set_eq:NN \texttt{\textbar{qed}@elt} \texttt{\textbar{setQED}@elt}
QED@stack \relax \relax
\group_end:
⟨/LaTeX⟩

12.11 We draw the arrows

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

- If the option \texttt{group} or the option \texttt{groups} is used, all the arrows of a group are drawn on a same vertical at an abscissa of \texttt{\l_\@\textbar{x}_\textbar{dim}}.
• For aesthetic reasons, the starting point of all the starting arrows of a group is raised upwards by the value \l_@@_start_adjust_dim. Idem for the ending arrows.

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

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

\cs_new_protected:Npn \@@_scan_arrows_i:
\l_@@_first_arrow_of_group_int will be the first arrow of the current group.
\l_@@_first_line_of_group_int will be the first line involved in the group of arrows (equal to the initial line of the first arrow of the group because the option jump is always positive).
\l_@@_first_arrows_seq will be the list of the arrows of the group starting at the first line of the group (we may have several arrows starting from the same line). We have to know all these arrows because of the adjustment by \l_@@_start_adjust_dim.
\l_@@_last_line_of_group_int will be the last line involved in the group (impossible to guess in advance).
\l_@@_last_arrows_seq will be the list of all the arrows of the group ending at the last line of the group (impossible to guess in advance).

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

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

\bool_set_true:N \l_@@_new_group_bool

We begin a loop over all the arrows of the environment. Inside this loop, if a group is finished, we will draw the arrows of that group.

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

We extract from the property list of the current arrow the fields “initial”, “final”, “status” and “input-line”. For the two former, we have to do conversions to integers.

\prop_get:cnN
\l_@@_prefix_str \l_@@_arrow_int \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
\l_@@_prefix_str \l_@@_arrow_int \l_@@_prefix_str _ \int_use:N \l_@@_arrow_int _ prop
\{ final \} \l_tmpa_tl
\int_set:Nn \l_@@_final_int \l_tmpa_tl
We recall that, after the construction of the \halign, \g@@line_int is the total number of lines of the environment. Therefore, the conditional \l@@final_int > \g@@line_int tests whether an arrow arrives after the last line of the environment. In this case, we raise an error (except in the second step of treatment for the option group). The arrow will be completely ignored, even for the computation of \l@@x_dim.

\int_compare:nNnTF \l@@final_int > \g@@line_int
{ \int_compare:nNnF \l@@pos_arrow_int = 8
{ \@@error:n { Too~few~lines~for~an~arrow } }
}
\@@treat_an_arrow_in_scan:

Incrementation of the index of the loop (and end of the loop).
\int_incr:N \l@@arrow_int

After the last arrow of the environment, we have to draw the last group of arrows. If we are in option group and in the first step of treatment (\l@@pos_arrow_int = 7), we don’t draw because, in the first step, we don’t draw anything. If there is no arrow in the group, we don’t draw (this situation occurs when all the arrows of the potential group arrive after the last line of the environment).

\bool_if:nT
{ ! \int_compare_p:nNn \l@@pos_arrow_int = 7
 &
 \int_compare_p:nNn \l@@first_arrow_of_group_int > 0
}
{ \@@draw_arrows:nn \l@@first_arrow_of_group_int \g@@arrow_int }

The following command is only for the lisibility of the code. It’s used only once. Its name may be misleading. Indeed, it treats an arrow in the scan but it may trigger the construction of all arrows of a group if it detects that a group has just been completed (with \@@draw_arrows:nn)
\cs_new_protected:Npn \@@treat_an_arrow_in_scan:
{ We test whether the previous arrow was in fact the last arrow of a group. In this case, we have to draw all the arrows of that group, except if we are with the option group and in the first step of treatment (\l@@pos_arrow_int = 7).
\bool_lazy_and:nnT
{ \int_compare_p:nNn \l@@arrow_int > 1 }
{ \bool_lazy_or_p:nn
{ \bool_lazy_and_p:nn
{ \int_compare_p:nNn \l@@initial_int > \l@@last_line_of_group_int
 }
{ \bool_not_p:n { \int_compare_p:nNn \l@@pos_arrow_int = 7 } }
 }
{ \str_if_eq_p:Vn \l@@status_arrow_str { new-group } }
}
}
\int_compare:nNnF \l_@@_first_arrow_of_group_int = \c_zero_int
{
    \@@_draw_arrows:nn
    \l_@@_first_arrow_of_group_int
    \l_@@_arrow_int - 1
}
\bool_set_true:N \l_@@_new_group_bool
}

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

\bool_if:nTF \l_@@_new_group_bool
{
    \bool_set_false:N \l_@@_new_group_bool
    \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
    \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
    \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
    \seq_clear:N \l_@@_first_arrows_seq
    \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
    \seq_clear:N \l_@@_last_arrows_seq
    \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
    \int_compare:nNnF \l_@@_pos_arrow_int = \l_@@_pos_arrow_int
    \dim_set:Nn \l_@@_x_dim { - \c_max_dim }
} If we are in option group and in the second step of treatment (\l_@@_pos_arrow_int = 8), we don’t initialize \l_@@_x_dim because we want to use the same value of \l_@@_x_dim (computed during the first step) for all the groups.

\bool_if:nTF \l_@@_new_group_bool
{
    \bool_set_false:N \l_@@_new_group_bool
    \int_set_eq:NN \l_@@_first_arrow_of_group_int \l_@@_arrow_int
    \int_set_eq:NN \l_@@_first_line_of_group_int \l_@@_initial_int
    \int_set_eq:NN \l_@@_last_line_of_group_int \l_@@_final_int
    \seq_clear:N \l_@@_first_arrows_seq
    \seq_put_left:NV \l_@@_first_arrows_seq \l_@@_arrow_int
    \seq_clear:N \l_@@_last_arrows_seq
    \seq_put_left:NV \l_@@_last_arrows_seq \l_@@_arrow_int
} If we are not at the beginning of a new group.

\str_if_eq:VnF \l_@@_status_arrow_str { independent }
{
If the arrow is independent, we don’t take into account that arrow for the detection of the end of the group.

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

} If the arrow is not independent, the arrow belongs to the current group and we have to take it into account in some variables.
If the arrow is not independent, we update the current x-value (in \l_@@_x_dim) with the dedicated command \@@_update_x:nn. If we are in option group and in the second step of treatment (\l_@@_pos_arrow_int = 8), we don’t initialize \l_@@_x_dim because we want to use the same value of \l_@@_x_dim (computed during the first step) for all the groups.

\begin{verbatim}
\str_if_eq:VnF \l_@@_status_arrow_str { independent }
  \int_compare:nNnF \l_@@_pos_arrow_int = 8
    \@@_update_x:nn \l_@@_initial_int \l_@@_final_int }
\end{verbatim}

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

\begin{verbatim}
\cs_generate_variant:Nn \keys_set:nn { n o }
\cs_new_protected:Npn \@@_keys_set:
  \keys_set_known:no { WithArrows / Arrow / SecondPass }
\end{verbatim}

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

\begin{verbatim}
\cs_new_protected:Npn \@@_draw_arrows:nn #1 #2
  \group_begin:
  \int_zero_new:N \l_@@_first_arrow_int
  \int_set:Nn \l_@@_first_arrow_int { #1 }
  \int_zero_new:N \l_@@_last_arrow_int
  \int_set:Nn \l_@@_last_arrow_int { #2 }
  \prop_get:cnN
    \int_compare:nNnF \l_@@_final_int > \g_@@_line_int
    \str_if_eq:VnTF \l_@@_status_arrow_str { over }
      \seq_put_right:NV \l_@@_o_arrows_seq \l_@@_arrow_int }
\end{verbatim}

The following code is necessary because we will have to expand an argument exactly 3 times.
The first \group_begin: is for the options of the arrows (but we remind that the options ll, rr, rl, lr, i and jump have already been extracted and are not present in the field options of the property list of the arrow).

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.

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:

<table>
<thead>
<tr>
<th>option</th>
<th>lr</th>
<th>ll</th>
<th>rl</th>
<th>rr</th>
<th>v</th>
<th>i</th>
<th>groups</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>\l_@@_pos_arrow_int</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

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

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

\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.
The label of the arrow will be stored in \l_tmpa_tl.
\prop_get:cnN \l_@@_arrow_int \prop \l_@@_arrow_int \l_tmpa_tl

Now, we have to know if the arrow starts at the first line of the group and/or ends at the last line of the group. That’s the reason why we have stored in \l_@@_first_arrows_seq the list of all the arrows starting at the first line of the group and in \l_@@_last_arrows_seq the list of all the arrows ending at the last line of the group. We compute these values in the booleans \l_tmpa_bool and \l_tmbp_bool. These computations can’t be done in the following {tikzpicture} because of the command \seq_if_in:NnTF which is not expandable.
\seq_if_in:NxTF \l_@@_first_arrows_seq \l_@@_arrow_int \bool_set_true:N \l_tmpa_bool \bool_set_false:N \l_tmpa_bool \seq_if_in:NxTF \l_@@_last_arrows_seq \l_@@_arrow_int \bool_set_true:N \l_tmbp_bool \bool_set_false:N \l_tmbp_bool \int_compare:nNnT \l_@@_pos_arrow_int = 5 \bool_set_true:N \l_tmpa_bool \bool_set_true:N \l_tmbp_bool

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

- Concerning the $x$-values, the abscissa computed in \l_@@_x_dim will be used if the option of position is i, group or groups.

- Concerning the $y$-values, an adjustment is done for each arrow starting at the first line of the group and each arrow ending at the last line of the group (with the values of \l_@@_start_adjust_dim and \l_@@_end_adjust_dim).
\dim_gzero_new:N \g_@@_x_initial_dim \dim_gzero_new:N \g_@@_x_final_dim \dim_gzero_new:N \g_@@_y_initial_dim \dim_gzero_new:N \g_@@_y_final_dim \pgfpicture \pgfrememberpicturepositiononpagetrue \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_initial_tl } { south } \dim_gset:Nn \g_@@_x_initial_dim \pgf@x \dim_gset:Nn \g_@@_y_initial_dim \pgf@y \pgfpointanchor { wa - \l_@@_prefix_str - \l_@@_final_tl } { north } \dim_gset:Nn \g_@@_x_final_dim \pgf@x \dim_gset:Nn \g_@@_y_final_dim \pgf@y \endpgfpicture \bool_lazy_and:nnTF \dim_compare_p:nNn { \g_@@_y_initial_dim - \g_@@_y_final_dim } \l_@@_max_length_of_arrow_dim \int_compare_p:nNnT \l_@@_final_int - \l_@@_initial_int = 1 \tl_gset:Nx \g_tmpa_tl \int_compare_p:nNnT \l_@@_pos_arrow_int < 5 \dim_use:N \g_@@_x_initial_dim \dim_use:N \l_@@_x_dim \dim_eval:n
The dimension \l_@@_delta_x_dim is the difference of abscissa between the right side of the alignment (\halign) and the left side of the arrow.

Eventually, we can draw the arrow with the code in \l_@@_tikz_code_tl. We recall that the value by default for this token list is: \draw (#1) to node {#3} (#2);. This value can be modified with the option tikz-code. We use the variant \@@_draw_arrow:nno of the macro \@@_draw_arrow:nnn.
because of the characters underscore in the name \_l\_tmpa\_tl: if the user uses the Tikz library \texttt{babel}, the third argument of the command \texttt{@\_draw\_arrow:nno} will be rescanned because this third argument will be in the argument of a command \texttt{node} of an instruction \texttt{draw} of Tikz... and we will have an error because of the characters underscore.\footnote{There were other solutions: use another name without underscore (like \texttt{\l\_tmpatl}) or use the package \texttt{underscore} (with this package, the characters underscore will be rescanned without errors, even in text mode).}

\begin{verbatim}
\@\_draw\_arrow:nno \g\_tmpa\_tl \g\_tmpb\_tl \l\_tmpa\_tl
\end{verbatim}

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

\begin{verbatim}
\group_end:
\end{verbatim}

The function \texttt{@@\_tmpa:nnn} will draw the arrow. It’s merely an environment \{tikzpicture\}. However, the Tikz instruction in this environment must be inserted from \texttt{\_l\_@@\_tikz\_code\_tl} with the markers \#1, \#2 and \#3. That’s why we create a function \texttt{@@\_def\_function\_tmpa:n} which will create the function \texttt{@@\_tmpa:nnn}.\footnote{There were other solutions: use another name without underscore (like \texttt{\l\_tmpatl}) or use the package \texttt{underscore} (with this package, the characters underscore will be rescanned without errors, even in text mode).}

\begin{verbatim}
\cs_new_protected:Npn \@@\_def\_function\_tmpa:n #1
\{
 \cs_set:Npn \@@\_tmpa:nnn ##1 ##2 ##3
\{
 \pgf@relevantforpicturesizetrue
 #1
 \dim_compare:nNnTF \pgf@picminx = { 16000 pt } { \dim_zero:N \l\_tmpa\_dim } { \dim_set:Nn \l\_tmpa\_dim { \pgf@picmaxx - \pgf@picminx } }
 \dim_add:Nn \l\_tmpa\_dim \l\_@@\_xoffset\_dim
 \prop_gput:cnV { g\_@@\_arrow \_l\_@@\_prefix\_str \_ \int\_use:N \l\_@@\_arrow\_int \_ \prop } { width } \l\_tmpa\_dim
\}
\}
\pgfresetboundingbox
\bool_if:NF \l\_@@\right\_overlap\_bool \l\_@@\_in\_With\_Arrows\_bool
\{
 \bool_if:NT \l\_@@\_in\_With\_Arrows\_bool
\{
 \dim_gset:Nn \g\_@@\_overlap\_x\_dim \l\_@@\_overlap\_x\_dim
 \dim_max:nn \g\_@@\_overlap\_x\_dim \l\_tmpa\_dim - \l\_@@\_delta\_x\_dim
\}
\}
\end{verbatim}

You keep track of the bounding box because we want to compute the total width of the arrow (with the label) for the arrows of type \texttt{over} and also for the actualization of \texttt{\g\_@@\_overlap\_x\_dim}.\footnote{There were other solutions: use another name without underscore (like \texttt{\l\_tmpatl}) or use the package \texttt{underscore} (with this package, the characters underscore will be rescanned without errors, even in text mode).}
When we draw the arrow (with \_draw\_arrow:nnn), we first create the function \_tmpa:nnn and, then, we use the function \_tmpa:nnn:

\cs_new_protected:Npn \_draw\_arrow:nnn #1 #2 #3

If the option \_wrap\_lines is used, we have to use a special version of \_tikz\_code:tl (which corresponds to the option tikz-code).

\bool_lazy_and:nnT \_wrap\_lines\_bool \_in\_Disp\_With\_Arrows\_bool

\tl_set_eq:NN \_tikz\_code\_tl \c_\_tikz\_code\_wrap\_lines\_tl

Now, the main lines of this function \_draw\_arrow:nnn.

\exp_args:NV \_def\_function\_tmpa:n \_tikz\_code\_tl

\_tmpa:nnn { #1 } { #2 } { #3 }

\cs_generate_variant:Nn \_draw\_arrow:nnn { n n o }

If the option \_wrap\_lines is used, we have to use a special version of \_tikz\_code:tl (which corresponds to the option tikz-code).

\tl_const:Nn \c_\_tikz\_code\_wrap\_lines\_tl

First, we draw the arrow without the label.

\draw ( #1 ) to node ( \_label ) { }

We retrieve in \_x the abscissa of the left-side of the label we will put.

\_pointanchor \_wa\_\_prefix\_str \_label \_west

We compute in \_dim the maximal width possible for the label. Here is the use of \_right\_x\_dim which has been computed previously with the v-nodes.

\dim_set:Nn \_dim \_right\_x\_dim \_pgf\_x - 0.3333 ex

We retrieve in \_tm\_tl the current value of the Tikz parameter “text width”.

\dim_compare:nNnT \_dim < \_dim

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

\dim_compare:nNnT \_dim > \c_zero

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

\dim_compare:nNnT \_dim > \c_zero

\path \_node \_anchor \_west

\begin\{\minipage\} \_dim \\tikz@text@action

\_key\_get\_value \_tikz\_\_node\_halign\_header \_dim\_\_align\_\_header

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

40
12.11.1 The command \@@_update_x:nn

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

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

12.11.2 We draw the arrows of type o

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

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

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

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

The second point ensures the expected output in situations such as in the following example:
\begin{WithArrows}
A = B \Arrow[\text{o},\text{jump}=3]{\text{one}}
& = C \Arrow[\text{o},\text{jump}=2]{\text{two}}
& = D \Arrow{\text{three}}
& = E + E
\end{WithArrows}

\seq_sort:Nn \l_@@_o_arrows_seq
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##2_ prop }
\int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
\sort_return_same:
\int_compare:nNnTF \l_tmpa_tl > \l_tmpb_tl
\sort_return_swapped:
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##2_ prop }
\int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
\sort_return_swapped:
\sort_return_same:

Now, we can draw the arrows of type o of the group in the order of the sequence.
\seq_map_inline:Nn \l_@@_o_arrows_seq
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##2_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##2_ prop }
\int_compare:nNnTF \l_tmpa_tl < \l_tmpb_tl
\sort_return_swapped:
\sort_return_same:

We retrieve the initial row and the final row of the arrow.
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\int_set:Nn \l_@@_initial_int \l_tmpa_tl
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\int_set:Nn \l_@@_final_int \l_tmpa_tl

The string \l_@@_input_line_str will be used only in some error messages.
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\prop_get:cnN { g_@@_arrow_ \l_@@_prefix_str_ ##1_ prop }
\input-line \l_@@_input_line_str

We have to compute the maximal width of all the arrows (with their labels) which are covered by our arrow. We will compute that dimension in \g_tmpa_dim. We need a global dimension because we will have to exit a pgfpicture.
\dim_gzero:N \g_tmpa_dim
We will raise the boolean \texttt{\g_tmpa_bool} if we find an arrow “under” our arrow (we should find at least once since you are drawing an arrow of type \texttt{o}; if not, we will raise an error\footnote{Maybe we will change that in future versions}).

\begin{verbatim}
\bool_set_false:N \g_tmpa_bool
\pgfpicture
\pgfrememberpicturepositiononpagetrue
\int_step_inline:nnn \l_@@_first_arrow_int \l_@@_last_arrow_int
\prop_get:cnN \g_@@_arrow_{\l_@@_prefix_str}_{\l_@@_index_1}_{prop}\l_tmpa_tl
\prop_get:cnN \g_@@_arrow_{\l_@@_prefix_str}_{\l_@@_index_1}_{prop}\l_tmpb_tl
\prop_get:cnN \g_@@_arrow_{\l_@@_prefix_str}_{\l_@@_index_1}_{prop}\l_@@_status_arrow_str
\bool_if:nT
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\int_compare_p:n { \l_@@_index_1 = \l_@@_index_1}
\bool_gset_true:N \g_tmpa_bool
\prop_get:cnN \g_@@_arrow_{\l_@@_prefix_str}_{\l_@@_index_1}_{prop}\l_tmpa_tl
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\endpgfpicture
\bool_if:NTF \g_tmpa_bool
\int_set:Nn \l_@@_arrow_int { \l_@@_index_1}
\dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
\dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
\@@_draw_arrow:
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\bool_if:NTF \g_tmpa_bool
\int_set:Nn \l_@@_arrow_int { \l_@@_index_1}
\dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
\dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
\@@_draw_arrow:
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\endpgfpicture
\endverbatim

The boolean \texttt{\g_tmpa_bool} is raised if at least one arrow has been found “under” our arrow (it should be the case since we are drawing an arrow of type \texttt{o}).

\begin{verbatim}
\bool_gset_true:N \g_tmpa_bool
\prop_get:cnN \g_@@_arrow_{\l_@@_prefix_str}_{\l_@@_index_1}_{prop}\l_tmpa_tl
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\bool_if:NTF \g_tmpa_bool
\int_set:Nn \l_@@_arrow_int { \l_@@_index_1}
\dim_set_eq:NN \l_@@_xoffset_dim \g_tmpa_dim
\dim_add:Nn \l_@@_xoffset_dim \l_@@_xoffset_for_o_arrows_dim
\@@_draw_arrow:
\prop_set:cnN \l_@@_status_arrow_str { \l_@@_status_arrow_str\and\l_@@_initial_int \leq \l_tmpa_tl\and\\l_tmpb_tl \leq \l_@@_final_int}
\endverbatim

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

\footnote{\texttt{\LaTeX}}
12.12 The command \Arrow in code-after

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

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

A sequence of the options available in \Arrow in code-after. This sequence will be used in the error messages and can be modified dynamically.
\seq_new:N \l_@@_options_Arrow_code_after_seq
\@@_set_seq_of_str_from_clist:Nn \l_@@_options_Arrow_code_after_seq { ll, lr, rl, rr, tikz, tikz-code, v, x, offset }

\NewDocumentCommand \@@_Arrow_code_after { O { } m m m ! O { } }
\cs_new_protected:Npn \@@_Arrow_code_after
{ \peek_meaning:NTF [ \@@_Arrow_code_after_i ] { \@@_Arrow_code_after_i [ ] } }
\int_set:Nn \l_@@_pos_arrow_int 1
\str_clear_new:N \l_@@_previous_key_str
\group_begin:
\keys_set:nn { WithArrows / Arrow / code-after }
{ #1, #5, tikz = { xshift = \l_@@_offset_dim } }
\bool_set_false:N \l_@@_initial_r_bool
\bool_set_false:N \l_@@_final_r_bool
\int_case:nn \l_@@_pos_arrow_int
{ 0
 { \bool_set_true:N \l_@@_initial_r_bool
  \bool_set_true:N \l_@@_final_r_bool }
 2 { \bool_set_true:N \l_@@_initial_r_bool }
 3 { \bool_set_true:N \l_@@_final_r_bool }
\group_end:
\tl_if_eq:nnTF { #2 } { #3 }
{ \@@_error:nn { Both~lines~are~equal } { #2 } }
\tl_gset:Nx \g_tmpa_tl { \dim_use:N \l_tmpa_dim , \dim_use:N \pgf@y }
\tl_gset:Nx \g_tmpb_tl { \dim_use:N \l_tmpb_dim , \dim_use:N \pgf@y }
\endpgfpicture
}\pgfpicture
\pgfpointanchor { wa - \l_@@_prefix_str - #2 - l }
\tl_gset:Nx \g_tmpa_tl { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
\pgfpointanchor { wa - \l_@@_prefix_str - #3 - l }
\tl_gset:Nx \g_tmpb_tl { \dim_use:N \l_tmpb_dim , \dim_use:N \pgf@y }
\endpgfpicture
\pgfpicture
\pgfpointanchor 
\pgfpointanchor { wa - \l_@@_prefix_str - #2 - l }
{ south }
\dim_set_eq:NN \l_tmpa_dim \pgf@x
\dim_set_eq:NN \l_tmpb_dim \pgf@y
\pgfpointanchor { wa - \l_@@_prefix_str - #3 - l }
{ north }
\dim_set:Nn \l_tmpa_dim
\dim_set:Nn \l_tmpb_dim
\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 \l_tmpb_dim }
\endpgfpicture
\pgfpicture
\pgfpointanchor 
\pgfpointanchor { wa - \l_@@_prefix_str - #2 - \bool_if:NTF \l_@@_initial_r_bool r l }
{ south }
\tl_gset:Nx \g_tmpa_tl { \dim_use:N \pgf@x , \dim_use:N \pgf@y }
\pgfpointanchor { wa - \l_@@_prefix_str -
12.13 The command \MultiArrow in code-after

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

The user of the command \MultiArrow (in code-after) will be able to specify the list of lines with the same syntax as the loop \foreach of pgf\texttt{for}. First, we test with a regular expression whether the format of the list of lines is correct.

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

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

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

We extract the maximum in \l_tmpb_tl. The remaining list (in \g_tmpa_clist) will be sorted in decreasing order but never mind...
We draw the teeth of the rak (except the first one and the last one) with the auxiliary function \@\_MultiArrow_i:n. This auxiliary function is necessary to expand the specification of the list in the \foreach loop. The first and the last teeth of the rak can’t be drawn the same way as the others (think, for example, to the case of the option “rounded corners” is used).

\exp_args:NV \@\_MultiArrow_i:n \g_tampa_clist

Now, we draw the rest of the structure.

\begin{tikzpicture}
\foreach \k in { #1 }
{
\draw [ <- ] ([xshift = \l_@@_xoffset_dim]\l_\_\_offset_dim \k-r.south) -- ++(5mm,0) ;
}
\end{tikzpicture}

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

\begin{tikzpicture}
\foreach \k in { #1 }
{
\draw [ <- ] ([xshift = \l_@@_xoffset_dim]k-r.south) -- ++(5mm,0) ;
}
\end{tikzpicture}
12.14 The error messages of the package

\bool_if:NTF \c_@@_messages_for_Overleaf_bool
{ \str_const:Nn \c_@@_available_keys_str { } }
{ \str_const:Nn \c_@@_available_keys_str
{ For-a-list-of-the-available-keys,-type-H<-return>. }
}
\str_new:N \l_witharrows_body_str
The following commands must not be protected since they will be used in error messages.

\cs_new:Npn \@@_potential_body_i:
{ \str_if_empty:NF \l_witharrows_body_str
{ \\\ If-you-want-to-see-the-body-of-the-environment,-type-H<-return>. }
}
\cs_new:Npn \@@_potential_body_ii:
{ \str_if_empty:NTF \l_witharrows_body_str
{ No-further-help-available }
{ The-body-of-your-environment-was:\
 \l_witharrows_body_str }
}
\str_const:Nn \c_@@_option_ignored_str
{ If-you-go-on,-this-option-will-be-ignored. }
\str_const:Nn \c_@@_command_ignored_str
{ If-you-go-on,-this-command-will-be-ignored. }

\@@_msg_new:nn { amsmath~not~loaded }
{ amsmath~not~loaded.\ You~can\textquotesingle;t~use~the~option\textquotesingle;\l_keys_key_str\textquotesingle;~because~the~package\textquotesingle;amsmath\textquotesingle;~has~not~been~loaded.\ If~you~go~on,-this~option-will-be-ignored~in~the~rest~of~the~document. }

\@@_msg_new:nn { option~of~cr~negative }
{ Incorrect-value.\ Bad-value-for-the-option\textquotesingle;\l_keys_key_str\textquotesingle;~-The~value~must~begin~with-an-extensible-left-delimiter.~The~possible~values~are:-,\token_to_str:N \{,\(,\[,\lbrace,\token_to_str:N \lbrack,\token_to_str:N \lgroup,\token_to_str:N \langle,\token_to_str:N \lmoustache,\token_to_str:N \lfloor\ and\token_to_str:N \lceil\ (and\token_to_str:N \lvert\ and\token_to_str:N \lVert\~\text{if~amsmath~or~unicode-math~is~loaded~in-LaTeX}).\ If~amsmath~or~unicode-math~is~loaded~in-LaTeX}.\ }

\@@_msg_new:nn { Bad-value-for-replace-brace-by }
{ Incorrect-value.\ Bad-value-for-the-option\textquotesingle;\l_keys_key_str\textquotesingle;~-The~value~must~begin~with-an-extensible-left-delimiter.~The~possible~values~are:-,\token_to_str:N \{,\(,\[,\lbrace,\token_to_str:N \lbrack,\token_to_str:N \lgroup,\token_to_str:N \langle,\token_to_str:N \lmoustache,\token_to_str:N \lfloor\ and\token_to_str:N \lceil\ (and\token_to_str:N \lvert\ and\token_to_str:N \lVert\~\text{if~amsmath~or~unicode-math~is~loaded~in-LaTeX}).\ }

\@@_msg_new:nn { Bad-value.\ Bad-value-for-the-option\textquotesingle;\l_keys_key_str\textquotesingle;~-The~value~must~begin~with-an-extensible-left-delimiter.~The~possible~values~are:-,\token_to_str:N \{,\(,\[,\lbrace,\token_to_str:N \lbrack,\token_to_str:N \lgroup,\token_to_str:N \langle,\token_to_str:N \lmoustache,\token_to_str:N \lfloor\ and\token_to_str:N \lceil\ (and\token_to_str:N \lvert\ and\token_to_str:N \lVert\~\text{if~amsmath~or~unicode-math~is~loaded~in-LaTeX}).\ }

\@@_msg_new:nn { Bad-value.\ Bad-value-for-the-option\textquotesingle;\l_keys_key_str\textquotesingle;~The~value~must~begin~with-an-extensible-left-delimiter.~The~possible~values~are:-,\token_to_str:N \{,\(,\[,\lbrace,\token_to_str:N \lbrack,\token_to_str:N \lgroup,\token_to_str:N \langle,\token_to_str:N \lmoustache,\token_to_str:N \lfloor\ and\token_to_str:N \lceil\ (and\token_to_str:N \lvert\ and\token_to_str:N \lVert\~\text{if~amsmath~or~unicode-math~is~loaded~in-LaTeX}).\ }

87
The argument of the command \token_to_str:N should be positive in the row \int_use:N \g_@@_line_int \of your environment \{ \l_@@_type_env_str \}.\c_@@_option_ignored_str

\@@_msg_new:nn { omit-probably-used }
{
  Strange problem.\\ Maybe you have used a command-
  \token_to_str:N \omit \ in-the-line \int_use:N \g_@@_line_int \( or another line \) of your environment \{ \l_@@_type_env_str \}.\\ You can go on but you may have others errors.
}

\langle *LaTeX\rangle
\@@_msg_new:nnn { newline-at-the-end-of-env }
{
  Incorrect end.\\ The environments of witharrows- \{ WithArrows \} and-
  \{ DispWithArrows \} should not end by \token_to_str:N \.\\ However, you can go on for this time. No similar error will be-
  raised in this document.
\@@_potential_body_i:
}
\{ \@@_potential_body_ii: \}
\langle /LaTeX\rangle
\@@_msg_new:nnn { Invalid-option-format }
{
  Invalid value.\\ The key 'format' should contain only letters r, c and l and-
  must not be empty.\\ \c_@@_option_ignored_str
\@@_potential_body_i:
}
\{ \@@_potential_body_ii: \}
\@@_msg_new:nnn { invalid-key-o }
{
  Invalid use of a key.\\ The key 'o' for individual arrows can be used only in mode-
  'group' or in mode 'groups'.\\ \c_@@_option_ignored_str
\@@_potential_body_i:
}
\{ \@@_potential_body_ii: \}
\@@_msg_new:nnn { Value-for-a-key }
{
  Misuse of a key.\\ The key '\l_keys_key_str' should be used without value. \\ However, you can go on for this time.
\@@_potential_body_i:
}
\{ \@@_potential_body_ii: \}
\@@_msg_new:nnn { Unknown-option-in-Arrow }
{
  Unknown option.\\ The key '\l_keys_key_str' is unknown for the command-
  \l_@@_string_Arrow_for_msg_str in the row-
  \int_use:N \g_@@_line_int \of your environment-
  \{ \l_@@_type_env_str \}. \l_tmpa_str \\ \c_@@_option_ignored_str \\ \c_@@_available_keys_str
}

88
The available keys are (in alphabetic order):
\seq_use:Nnnn \l_@@_options_Arrow_seq {~and~} {,~} {~and~}.

\@@_msg_new:nnn { Unknown-option-WithArrows }
{ Unknown-option."
The key-'\l_keys_key_str'-is-unknown-in-\{\l_@@_type_env_str\}. \\
\c_@@_option_ignored_str \\
\c_@@_available_keys_str
}
{ The available keys are (in alphabetic order):-
\seq_use:Nnnn \l_@@_options_withArrows_seq {~and~} {,~} {~and~}.
}

\@@_msg_new:nnn { Unknown-option-DispWithArrows }
{ Unknown-option."
The key-'\l_keys_key_str'-is-unknown-in-\{\l_@@_type_env_str\}. \\
\c_@@_option_ignored_str \\
\c_@@_available_keys_str
}
{ The available keys are (in alphabetic order):-
\seq_use:Nnnn \l_@@_options_dispWithArrows_seq {~and~} {,~} {~and~}.
}

\@@_msg_new:nnn { Unknown-option-DispWithArrowsOptions }
{ Unknown-option."
The key-'\l_keys_key_str'-is-unknown-in-
\token_to_str:N \WithArrowsOptions. \\
\c_@@_option_ignored_str \\
\c_@@_available_keys_str
}
{ The available keys are (in alphabetic order):-
\seq_use:Nnnn \l_@@_options_withArrowsOptions_seq {~and~} {,~} {~and~}.
}

\@@_msg_new:nnn { Unknown-option-Arrow-in-code-after }
{ Unknown-option."
The key-'\l_keys_key_str'-is-unknown-in-
\token_to_str:N \Arrow\ in-code-after. \\
\c_@@_option_ignored_str \\
\c_@@_available_keys_str
}
{ The available keys are (in alphabetic order):-
\seq_use:Nnnn \l_@@_options_Arrow_code_after_seq {~and~} {,~} {~and~}.
}

\@@_msg_new:nnn { Too much columns in WithArrows }
{ Too much columns."
Your environment-\{\l_@@_type_env_str\}-has-\int_use:N \l_@@_nb_cols_int columns and you try to use one more. -
Maybe you have forgotten a-\backslash\backslash \c_backslash_str .-
If you really want to use more columns (after the arrows) you should use-
the option 'more-columns' at a global level or for an environment. \\
However, you can go one for this time.
\@@_potential_body_i:
{ \@@_potential_body_ii: }
\@@_msg_new:n { Too-much-columns-in-DispWithArrows }
{
Too-much-columns.
\l_@@_type_env_str\-has-\int_use:NN\l_@@_nb_cols_int\ columns-and-you-try-to-use-one-more.\-
Maybe-you-have-forgotten-a-\c_backslash_str\c_backslash_str at-the-end-of-row-\int_use:N \g_@@_line_int. \-
This-error-is-fatal.
\@@_potential_body_i: }
{
\@@_potential_body_ii: }
\@@_msg_new:n { Negative-jump }
{
Incorrect-value.\-
You-can't-use-a-negative-value-for-the-option-'jump'-of-command-
\l_@@_string_Arrow_for_msg_str in-the-row-\int_use:N \l_@@_initial_int\ of-your-environment-\l_@@_type_env_str-.\-
You-can-create-an-arrow-going-backwards-with-the-option-'<-'-of-Tikz. \-
\c_@@_option_ignored_str }
\@@_msg_new:n { new-group-without-groups }
{
Misuse-of-a-key.\-
You-can't-use-the-option-'new-group'-for-the-command-
\l_@@_string_Arrow_for_msg_str because-you-are-not-in-'groups'-mode.-Try-to-use-the-option-
'groups'-in-your-environment-\l_@@_type_env_str. \-
\c_@@_option_ignored_str }
\@@_msg_new:n { Too-few-lines-for-an-arrow }
{
Impossible-arrow.\-
Line-\l_@@_input_line_str\-
\l_@@_initial_int\ of-your-environment-\l_@@_type_env_str-can't-be-drawn-\-
because-it-arrives-after-the-last-row-of-the-environment. \-
If-you-go-on,-this-arrow-will-be-ignored.
\@@_potential_body_i: }
{
\@@_potential_body_ii: }
\@@_msg_new:n { o-arrow-with-no-arrow-under }
{
Problem-with-the-key-'o'.\-
Line-\l_@@_input_line_str\-
\l_@@_initial_int\ of-your-environment-\l_@@_type_env_str-can't-be-drawn-\-
because-it-arrives-after-the-last-row-of-the-environment. \-
If-you-go-on,-this-arrow-won't-be-drawn. }
\@@_msg_new:n { WithArrows-outside-math-mode }
{
You-are-outside-math-mode.\-
The-environment-\l_@@_type_env_str-should-be-used-only-in-math-mode-
like-the-environment-{aligned}-of-amsmath. \-
Nevertheless,-you-can-go-on.
\@@_potential_body_i: }
{
\@@_potential_body_ii: }
\@@_msg_new:n { DispWithArrows-in-math-mode }
{
You-are-in-math-mode.\-

The environment \{\l_@@_type_env_str\} should be used only outside math mode-like the environments \{align\}-and\{-align\}-of-amsmath. This error is fatal.
\\@ potential_body_i:
\{
\\@ potential_body_ii: 
\}\n\@_msg_new:nn { Incompatible-options-in-Arrow }
{ Incompatible-options.\}
You try to use the option '\l_keys_key_str'-but-
this option is incompatible or redundant with the option-
'\l_@@_previous_key_str'-set in the same command-
\l_@@_string_Arrow_for_msg_str. \}
c_@@_option_ignored_str
\}\n\@_msg_new:nn { Incompatible-options a}
{ Incompatible-options.\}
You try to use the option '\l_keys_key_str'-but-
this option is incompatible or redundant with the option-
'\l_@@_previous_key_str'-set in the same command-
\bool_if:NT \l_@@_in_code_after_bool 
{ \l_@@_string_Arrow_for_msg_str
in the code after of your environment-\{\l_@@_type_env_str\}
}. \}
c_@@_option_ignored_str
\}\n\@_msg_new:nnn { Arrow-not-in-last-column }
{ Bad use of \l_@@_string_Arrow_for_msg_str.\}
You should use the command \l_@@_string_Arrow_for_msg_str
only in the last column-(column\int_use:N\l_@@_nb_cols_int)-
in the row \int_use:N \g_@@_line_int
of your environment-\{\l_@@_type_env_str\}. \}
However you can go on for this time.
\\@ potential_body_i:
\{
\\@ potential_body_ii: 
\}\n\@_msg_new:nn { Wrong-line-in-Arrow }
{ Wrong-line.\}
The specification of line '#1'-you use in the command-
\l_@@_string_Arrow_for_msg_str
in the 'code-after' of-\{\l_@@_type_env_str\}-doesn't exist. \}
c_@@_option_ignored_str
\}\n\@_msg_new:nn { Both-lines-are-equal }
{ Both-lines-are-equal.\}
In the 'code-after' of-\{\l_@@_type_env_str\}-you try to-
draw an arrow going to itself from the line '#1'.-This is not possible. \}
c_@@_option_ignored_str
\}\n\@_msg_new:nn { Wrong-line-specification-in-MultiArrow }
{ Wrong-line-specification.\}
The specification of line '#1'-doesn't exist. \}
If you go on,-it will be ignored for \token_to_str:N \MultiArrow.
}\n\@_msg_new:nn { Too-small-specification-for-MultiArrow }
91
Too-small-specification.\ \ The-specification-of-lines-you-gave-to-MultiArrow\ is-too-small:-you-need-at-least-two-lines. \ \}
\c@@_command_ignored_str
\@@_msg_new:nn { Not-allowed-in-DispWithArrows }
\{ Forbidden-command.\ \ The-command-\token_to_str:N \#1\ is-allowed-only-in-the-last-column-\ (column-\int_use:N\l_@@_nb_cols_int)-of-\{\l_@@_type_env_str\}. \ \}
\c@@_option_ignored_str
\@@_msg_new:nn { Not-allowed-in-WithArrows }
\{ Forbidden-command.\ \ The-command-\token_to_str:N \#1\ is-not-allowed-in-\{\l_@@_type_env_str\}- (it's-allowed-in-the-last-column-of-\{DispWithArrows\}). \ \}
\c@@_option_ignored_str

(*LaTeX*)
\@@_msg_new:nn { tag*-without-amsmath }
\{ amsmath-not-loaded.\ \ We-can't-use-\token_to_str:N \tag*-because-you-haven't-loaded-amsmath- \ (or-mathtools). \ \ If-you-go-on,-the-command-\token_to_str:N \tag\ will-be-used-instead. \ }
\@@_msg_new:nn { Multiple-tags }
\{ Multiple-tags.\ \ You-can't-use-twice-the-command-\token_to_str:N \tag\ in-a-line-of-the-environment-\{\l_@@_type_env_str\}. \ \ If-you-go-on,-the-tag-'\#1'-will-be-used. \ }
\@@_msg_new:nn { Multiple-labels }
\{ Multiple-labels.\ \ Normally,-we-can't-use-the-command-\token_to_str:N \label\ twice-in-a-line-of-the-environment-\{\l_@@_type_env_str\}. \ \ However,-you-can-go-on.- \ \IfPackageLoadedTF { showlabels } \ { However,-only-the-last-label-will-be-shown-by-showlabels.- } { } \ If-you-don't-want-to-see-this-message-again,-you-can-use-the-option- \ 'allow-multiple-labels'-at-the-global-or-environment-level. \ }
\@@_msg_new:nn { Multiple-labels-with-cleveref }
\{ Multiple-labels.\ \ Since-you-use-cleveref,-you-can't-use-the-command-\token_to_str:N \label\ twice-in-a-line-of-the-environment-\{\l_@@_type_env_str\}. \ \ If-you-go-on,-you-may-have-undefined-references. \ }
\(/LaTeX/)
\@@_msg_new:nn { Inexistent-v-node }
\{ There-is-a-problem.\ \ Maybe-you-have-put-a-command-\token_to_str:N \cr\ instead-of-a-command-\token_to_str:N \at-the-end-of-
The following error when the user tries to use the option \texttt{xoffset} in mode \texttt{group} or \texttt{groups} (in fact, it's possible to use the option \texttt{xoffset} if there is only one arrow: of course, the option \texttt{group} and \texttt{groups} do not make sense in this case but, maybe, the option was set in a \texttt{\WithArrowsOptions}).

\begin{verbatim}
\@\_msg_new:nn { Option-xoffset-forbidden }
  { Incorrect-key.\\ }
  \texttt{You-can't-use-the-option-'xoffset'-in-the-command-}
  \texttt{\land_string\texttt{Arrow_for_msg_str} in-the-row\texttt{\int_use:N \ge_0_line_int}
  of-your-environment-\{\land_0_type_env_str\}-.\\ }
  \texttt{because-you-are-using-the-option-}
  '\texttt{int_compare:nNnTF \l_0_pos_arrow_int = 7}
  \texttt{group }\{ \texttt{groups }\texttt{'.-It's-possible-for-an-independent-arrow-or-if-there-is-}
  \texttt{only-one-arrow. }\\ }
  \texttt{\c_0\_option_ignored_str}
\end{verbatim}

\begin{verbatim}
\@\_msg_new:nnn { Duplicate-name }
  { Duplicate-name.\\ }
  \texttt{The-name-}'\texttt{\keys_value_tl}'-is-already-used-and-you-shouldn't-use-
  \texttt{the-same-environment-name-twice.-You-can-go-on,-but,-}
  \texttt{maybe,-you-will-have-incorrect-results. }\\ }
  \texttt{For-a-list-of-the-names-already-used,-type-N<return>. }\\ }
  \texttt{If-you-don't-want-to-see-this-message-again,-use-the-option-}
  \texttt{allow-duplicate-names'.}
\end{verbatim}

\begin{verbatim}
\@\_msg_new:nn { Invalid-specification-for-MultiArrow }
  { Invalid-specification.\\ }
  \texttt{The-specification-of-rows-for-\texttt{\token_to_str:N\MultiArrow}\ }
  \texttt{(i.e.-#1) is-invalid. }\\ }
  \texttt{\c_0\_command_ignored_str}
\end{verbatim}

\subsection{The command \texttt{\WithArrowsNewStyle}}

A new key defined with \texttt{\WithArrowsNewStyle} will not be available at the local level.
When we will consider that \texttt{\keys_precompile:nnN} (introduced in LaTeX on 2022-03-09) is widely available, we will delete that test and keep only the first version.

12.16 The options up and down

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

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

- The key \texttt{radius} is the radius of the rounded corner of the arrow.
- The key \texttt{width} is the width of the horizontal part of the arrow. The corresponding dimension is \texttt{\l_@@\_arrow\_width\_dim}. By convention, a value of 0 pt for \texttt{\l_@@\_arrow\_width\_dim} means that the option \texttt{width} has been used with the special value \texttt{min} and a value of \texttt{\c\_max\_dim} means that it has been used with the value \texttt{max}.

\begin{verbatim}
\keys_define:nn { WithArrows / up-and-down }
  { radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
    radius .value_required:n = true ,
    width .code:n =
      \str_case:nnF { #1 }
      { { min } { \dim_zero:N \l_@@_arrow_width_dim }
        { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c\_max\_dim } }
      \dim_set:Nn \l_@@_arrow_width_dim { #1 } ,
    width .value_required:n = true ,
    unknown .code:n = \@@_error:n { Option-unknown-for-up-and-down } }
\@@_msg_new:nn { Option-unknown-for-up-and-down }
  { Unknown-option.\ 
    The-option-'^\l\_keys_key_str'^-is-unknown.-\c\_@@\_option\_ignored_str }
\end{verbatim}

The token list \texttt{\c\_@@\_tikz\_code\_up\_tl} is the value of \texttt{tikz-code} which will be used for an option \texttt{up}.

\begin{verbatim}
\keys_define:nn { WithArrows / up-and-down }
  { radius .dim_set:N = \l_@@_up_and_down_radius_dim ,
    radius .value_required:n = true ,
    width .code:n =
      \str_case:nnF { #1 }
      { { min } { \dim_zero:N \l_@@_arrow_width_dim }
        { max } { \dim_set_eq:NN \l_@@_arrow_width_dim \c\_max\_dim } }
      \dim_set:Nn \l_@@_arrow_width_dim { #1 } ,
    width .value_required:n = true ,
    unknown .code:n = \@@_error:n { Option-unknown-for-up-and-down } }
\@@_msg_new:nn { Option-unknown-for-up-and-down }
  { Unknown-option.\ 
    The-option-'^\l\_keys_key_str'^-is-unknown.-\c\_@@\_option\_ignored_str }
\end{verbatim}

\begin{verbatim}
\tl_const:Nn \c\_@@\_tikz\_code\_up\_tl
  { First the case when the key \texttt{up} is used with \texttt{width=max} (that’s the default behaviour).
    \begin{verbatim}
    \dim_compare:nNnTF \l_@@_arrow_width_dim = \c\_max\_dim
      { \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
        let \p1 = ( #1 ) , \p2 = ( #2 )
          in \p1 -- node
            { \dim_set:Nn \l\_tma\_dim { \x2 - \x1 }
              \begin { varwidth } \l\_tma\_dim \end { varwidth }
              \narrowragged
            }
          \p2 ;
      }
    a \texttt{narrowragged} is a command of the package \texttt{varwidth}.
    \end{verbatim}

    Now the case where the key \texttt{up} is used with \texttt{width=value} with \texttt{value} equal to \texttt{min} or a numeric value. The instruction \texttt{\path} doesn’t draw anything: its aim is to compute the natural width of the label of the arrow. We can’t use \texttt{\pgfextra} here because of the \texttt{\hbox\_set\_Nn}.
      { }
\end{verbatim}
\path
\let \p1 = (#1), \p2 = (#2)
\in node
{

The length $\l_{tmpa\ dim}$ will be the maximal width of the box composed by the environment \verb|\varwidth|.

\begin{verbatim}
\dim_set:Nn \l_{tmpa\ dim}
{ \x_2 - \x_1 - \l_@@up_and_down_radius_dim }
\dim_compare:nNnF \l_@@arrow_width_dim = \c_zero_dim
{ \dim_set:Nn \l_{tmpa\ dim}
{ \dim_min:nn \l_{tmpa\ dim} \l_@@arrow_width_dim }
}
\end{verbatim}

Now, the length $\l_{tmpa\ dim}$ is computed. We can compose the label in the box $\g_{tmpa\ box}$. We have to do a global affectation to be able to exit the node.

\begin{verbatim}
\hbox_gset:Nn \g_{tmpa\ box}
{ \begin{\varwidth} \l_{tmpa\ dim}
\arrowragged #3
\end{\varwidth}
}
\dim_compare:nNnTF \l_@@arrow_width_dim > \c_zero_dim
{ \dim_gset_eq:NN \g_{tmpa\ dim} \l_@@arrow_width_dim }
{ \dim_gset:Nn \g_{tmpa\ dim} { \box_wd:N \g_{tmpa\ box} }
\dim_gadd:Nn \g_{tmpa\ dim} \l_@@up_and_down_radius_dim }
; \draw
\let \p1 = (#1), \p2 = (#2)
\in (\x_2-\g_{tmpa\ dim},\y_1)
\begin{verbatim}
\path node
{ \hbox_gset:Nn \g_{tmpa\ box} { #3 }
\dim_gset:Nn \g_{tmpa\ dim}
{ \box_wd:N \g_{tmpa\ box} + \l_@@up_and_down_radius_dim }
}
\end{verbatim}
\draw
\let \p1 = (#1), \p2 = (#2)
\in (\x_2-\g_{tmpa\ dim},\y_1)
\begin{verbatim}
\path node
{ \hbox_gset:Nn \g_{tmpa\ box} { #3 }
\dim_gset:Nn \g_{tmpa\ dim}
{ \box_wd:N \g_{tmpa\ box} + \l_@@up_and_down_radius_dim }
}
\end{verbatim}
\end{verbatim}

\begin{verbatim}
\tl_const:Nn \c_@@tikz_code_up_tl
{ \dim_case:nnF \l_@@arrow_width_dim
{ \c_max_dim
\begin{verbatim}
\draw [ rounded-corners = \l_@@up_and_down_radius_dim ]
\let \p1 = (#1), \p2 = (#2)
\in (\p1) -- node { #3 } (\x_2,\y_1) -- (\p2)
\end{verbatim}
\end{verbatim}

\begin{verbatim}
\c_zero_dim
\begin{verbatim}
\path node
{ \hbox_gset:Nn \g_{tmpa\ box} { #3 }
\dim_gset:Nn \g_{tmpa\ dim}
{ \box_wd:N \g_{tmpa\ box} + \l_@@up_and_down_radius_dim }
}
\end{verbatim}
\draw
\let \p1 = (#1), \p2 = (#2)
\in (\x_2-\g_{tmpa\ dim},\y_1)
\begin{verbatim}
\path node
{ \hbox_gset:Nn \g_{tmpa\ box} { #3 }
\dim_gset:Nn \g_{tmpa\ dim}
{ \box_wd:N \g_{tmpa\ box} + \l_@@up_and_down_radius_dim }
}
\end{verbatim}
\end{verbatim}

\end{verbatim}

\end{verbatim}

\end{verbatim}
The code for an arrow of type down is similar to the previous code (for an arrow of type up).

\begin{verbatim}
\tl_const:Nn \c_@@_tikz_code_down_tl
\{
   \dim_compare:nNnTF \l_@@_arrow_width_dim = \c_max_dim
   { \draw [ rounded-corners = \l_@@_up_and_down_radius_dim ]
     \let \p1 = ( #1 ) , \p2 = ( #2 )
     \in \langle \begin{varwidth} \l_tmpa_dim
       \hspace{\l_tmpa_dim}
       \narrowragged
       \#3
       \end{varwidth}
     \rangle
     \dim_compare:nNnTF \l_@@_arrow_width_dim > \c_zero_dim
     { \dim_gset_eq:NN \g_tmpa_dim \l_@@_arrow_width_dim }
     { \dim_gset:Nn \g_tmpa_dim { \box_wd:N \g_tmpa_box } }
     \dim_gadd:Nn \g_tmpa_dim \l_@@_up_and_down_radius_dim
   }
   \path
     \let \p1 = ( #1 ) , \p2 = ( #2 )
     \in node
     { \hbox_gset:Nn \g_tmpa_box
       \dim_set:Nn \l_tmpa_dim { \x1 - \x2 - \l_@@_up_and_down_radius_dim - 2 mm }
       \begin { varwidth } \l_tmpa_dim
         \narrowragged
         \#3
       \end { varwidth }
     }
     \begin { varwidth } \l_tmpa_dim
       \hspace{\l_tmpa_dim}
       \narrowragged
       \#3
     \end { varwidth }
   }
   \draw
   \let \p1 = ( #1 ) , \p2 = ( #2 )
   \in \langle \begin{varwidth} \l_tmpa_dim
     \hspace{\l_tmpa_dim}
     \narrowragged
     \#3
     \end{varwidth}
   \rangle
   \box_use:N \g_tmpa_box \langle \begin{varwidth} \l_tmpa_dim
     \hspace{\l_tmpa_dim}
     \narrowragged
     \#3
   \end{varwidth}
   \rangle
\}
\end{verbatim}

The 2 mm are for the tip of the arrow. We don’t want the label of the arrow too close to the tip of arrow (we assume that to the tip of the arrow has its standard position, that is at the end of the arrow.).
We recall that the options of the individual arrows are scanned twice. First, when are scanned when the command \texttt{\textbackslash Arrow} occurs (we try to know whether the arrow is “individual”, etc.). That’s the first pass.

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

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

\begin{verbatim}
\keys_define:nn { WithArrows / Arrow / SecondPass }
{  
  up .code:n =  
    \str_if_empty:NT \l_@@_previous_key_str
\end{verbatim}
We have to set \l_@@_wrap_lines_bool to false because, otherwise, if the option \texttt{wrap_lines} is used at a higher level (global or environment), we will have a special affectation to \texttt{tikz-code} that will overwrite our affectation.

The main action occurs now. We change the value of the \texttt{tikz-code}.

\begin{verbatim}
\tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_up_tl
{ \@@_error:n { calc-not-loaded } }
\end{verbatim}

\begin{verbatim}
down .code:n =
\str_if_empty:NT \l_@@_previous_key_str
{ \str_set:Nn \l_@@_previous_key_str { down }
\cs_if_exist:cTF { tikz@library@calc@loaded }
{ \keys_set:nV { WithArrows / up-and-down } \l_keys_value_tl
\int_set:Nn \l_@@_pos_arrow_int 1
\bool_set_false:N \l_@@_wrap_lines_bool
\tl_set_eq:NN \l_@@_tikz_code_tl \c_@@_tikz_code_down_tl
}
{ \@@_error:n { calc-not-loaded } }
}
\seq_put_right:Nn \l_@@_options_Arrow_seq { down }
\seq_put_right:Nn \l_@@_options_Arrow_seq { up }
\@@_msg_new:nn { calc-not-loaded }
{ calc-not-loaded.\}
\end{verbatim}

\begin{verbatim}
\c_@@_option_ignored_str
\end{verbatim}

\section{History}

\subsection*{Changes between 2.7 and 2.8}

New key right-overlap

\subsection*{Changes between 2.6b and 2.7}

Correction of a bug: when the key \texttt{wrap-lines} was in force, the content of the annotations was not “flush left” by default as it should be (but justified).

\subsection*{Changes between 2.6 and 2.6a (and 2.6b)}

Replacement of \texttt{\hbox_unpack_clear:N} by \texttt{\hbox_unpack_drop:N} since \texttt{\hbox_unpack_clear:N} is now deprecated in L3.

Version 2.6d: correction of a bug (cf. question 628461 on TeX StackExchange).
Changes between 2.5 and 2.5.1
Correction of the erroneous programmation of the nodes aliases.

Changes between 2.4 and 2.5
Arrows of type $o$ which are over other arrows.
witharrows now requires and loads varwidth

Changes between 2.3 and 2.4
Correction of a bug with \{DispWithArrows\}: cf. question 535989 on TeX StackExchange.

Changes between 2.2 and 2.3
Two options for the arrows of type up and down: width and radius.

Changes between 2.1 and 2.2
Addition of $\texttt{normalbaselines}$ at the beginning of $\backslash\texttt{\_post\_halign}$.
The warning for an environment ending by $\backslash\backslash$ has been transformed in error.

Changes between 2.0 and 2.1
Option max-length-of-arrow.
Validation with regular expression for the first argument of $\texttt{MultiArrow}$.

Changes between 1.18 and 2.0
A version of witharrows is available for plain-TeX.

Changes between 1.17 and 1.18
New option $\langle \ldots \rangle$ for \{DispWithArrows\}.
Option subequations.
Warning when \{WithArrows\} or \{DispWithArrows\} ends by $\backslash\backslash$.
No space before an environment \{DispWithArrows\} if we are at the beginning of a \{minipage\}.

Changes between 1.16 and 1.17
Option format.

Changes between 1.15 and 1.16
Option no-arrows
The behaviour of \{DispWithArrows\} after an $\texttt{item}$ of a LaTeX list has been changed: no vertical is added. The previous behaviour can be restored with the option standard-behaviour-with-items.
A given name can no longer be used for two distinct environments. However, it’s possible to deactivate this control with the option allow-duplicate-names.

Changes between 1.14 and 1.15
Option new-group to start a new group of arrows (only available when the environment is composed with the option groups).
Tikz externalization is now deactivated in the environments of the extension witharrows.\footnote{Before this version, there was an error when using witharrows with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by witharrows because they use the options overlay and remember picture.}
Changes between 1.13 and 1.14
New options up and down for the arrows.
Replacement of some options $0 \{ \}$ in commands and environments defined with \xparse by $! 0 \{ \}$
(a recent version of \xparse introduced the specifier $!$ and modified the default behaviour of the last optional arguments: \url{www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end}).
Modification of the code of \WithArrowsNewStyle following a correction of a bug in \l3keys in the version of \l3kernel of 2019/01/28.
New error message Inexistent-v-node to avoid a \pgf error.
The error Option incompatible with 'group(s)¹' was suppressed in the version 1.12 but this was a mistake since this error is used with the option xoffset at the local level. The error is put back.

Changes between 1.12 and 1.13
Options start-adjust, end-adjust and adjust.
This version is not strictly compatible with previous ones. To restore the behaviour of the previous versions, one has to use the option adjust with the value 0 pt:

\WithArrowsOptions{adjust = 0pt}

Changes between 1.11 and 1.12
New command \tagnextline.
New option tagged-lines.
An option of position (ll, lr, rl, rr or i) is now allowed at the local level even if the option group or the option groups is used at the global or environment level.
Compatibility of \DispWithArrows with \qedhere of \amsthm.
Compatibility with the packages refcheck, showlabels and listlbls.
The option \AllowLineWithoutAmpersand is deprecated because lines without ampersands are now always allowed.

Changes between 1.10 and 1.11
New commands \WithArrowsNewStyle and \WithArrowsRightX.

Changes between 1.9 and 1.10
If the option wrap-lines is used, the option “text width” of Ticz is still active: if the value given to “text width” is lower than the width computed by wrap-lines, this value is used to wrap the lines.
The option wrap-lines is now fully compatible with the class option leqno.
Correction of a bug: \nointerlineskip and \makebox[.6\linewidth]{ } should be inserted in \DispWithArrows only in vertical mode.

Changes between 1.8 and 1.9
New option wrap-lines for the environments \DispWithArrows and \DispWithArrows*.

Changes between 1.7 and 1.8
The numbers and tags of the environment \DispWithArrows are now compatible with all the major \LaTeX packages concerning references (autonum, cleveref, fancyref, hyperref, prettyref, refstyle, typedref and varioref) and with the options showonlyrefs and showmanualtags of mathtools.

Changes between 1.6 and 1.7
New environments \DispWithArrows and \DispWithArrows*.
Changes between versions 1.5 and 1.6
The code has been improved to be faster and the Tikz library calc is no longer required.
A new option name is available for the environments {WithArrows}.

Changes between versions 1.4 and 1.5
The Tikz code used to draw the arrows can be changed with the option tikz-code.
Two new options code-before and code-after have been added at the environment level.
A special version of \Arrow is available in code-after in order to draw arrows in nested environments.
A command \MultiArrow is available in code-after to draw arrows of other shapes.

Changes between versions 1.3 and 1.4
The package footnote is no longer loaded by default. Instead, two options footnote and footnotehyper have been added. In particular, witharrows becomes compatible with beamer.

Changes between versions 1.2 and 1.3
New options ygap and ystart for fine tuning.

Changes between versions 1.1 and 1.2
The package witharrows can now be loaded without having loaded previously tikz and the libraries arrow.meta and bending (this extension and these libraries are loaded silently by witharrows).
New option groups (with a s)

Changes between versions 1.0 and 1.1
Option for the command \ and option interline
Compatibility with \usetikzlibrary{babel}
Possibility of nested environments {WithArrows}

Contents

1 Options for the shape of the arrows 2
2 Numbers of columns 6
3 Precise positioning of the arrows 7
4 The option 'o' for individual arrows 9
5 The options 'up' and 'down' for individual arrows 10
6 Comparison with the environment {aligned} 11
7 Arrows in nested environments 14
8 Arrows from outside environments {WithArrows} 16
9 The environment {DispWithArrows} 17
  9.1 The option <...> of DispWithArrows 22
## Advanced features

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Use with plain-TeX</td>
<td>23</td>
</tr>
<tr>
<td>10.2</td>
<td>The option tikz-code: how to change the shape of the arrows</td>
<td>23</td>
</tr>
<tr>
<td>10.3</td>
<td>The command <code>\WithArrowsNewStyle</code></td>
<td>24</td>
</tr>
<tr>
<td>10.4</td>
<td>The key right-overlap</td>
<td>24</td>
</tr>
<tr>
<td>10.5</td>
<td>Vertical positioning of the arrows</td>
<td>25</td>
</tr>
<tr>
<td>10.6</td>
<td>Footnotes in the environments of witharrows</td>
<td>26</td>
</tr>
<tr>
<td>10.7</td>
<td>Option no-arrows</td>
<td>27</td>
</tr>
<tr>
<td>10.8</td>
<td>Note for the users of AUCTeX</td>
<td>27</td>
</tr>
<tr>
<td>10.9</td>
<td>Note for developers</td>
<td>27</td>
</tr>
</tbody>
</table>

## Examples

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td><code>\MoveEqLeft</code></td>
<td>27</td>
</tr>
<tr>
<td>11.2</td>
<td>A command <code>\DoubleArrow</code></td>
<td>28</td>
</tr>
<tr>
<td>11.3</td>
<td>Modifying the shape of the nodes</td>
<td>28</td>
</tr>
<tr>
<td>11.4</td>
<td>Examples with the option tikz-code</td>
<td>29</td>
</tr>
<tr>
<td>11.4.1</td>
<td>Example 1</td>
<td>29</td>
</tr>
<tr>
<td>11.4.2</td>
<td>Example 2</td>
<td>30</td>
</tr>
<tr>
<td>11.4.3</td>
<td>Example 3</td>
<td>30</td>
</tr>
<tr>
<td>11.5</td>
<td>Automatic numbered loop</td>
<td>31</td>
</tr>
</tbody>
</table>

## Implementation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>Declaration of the package and extensions loaded</td>
<td>32</td>
</tr>
<tr>
<td>12.2</td>
<td>The packages footnote and footnotehyper</td>
<td>33</td>
</tr>
<tr>
<td>12.3</td>
<td>The class option leqno</td>
<td>35</td>
</tr>
<tr>
<td>12.4</td>
<td>Collecting options</td>
<td>35</td>
</tr>
<tr>
<td>12.5</td>
<td>Some technical definitions</td>
<td>35</td>
</tr>
<tr>
<td>12.6</td>
<td>Variables</td>
<td>38</td>
</tr>
<tr>
<td>12.7</td>
<td>The definition of the options</td>
<td>41</td>
</tr>
<tr>
<td>12.8</td>
<td>The command <code>\Arrow</code></td>
<td>49</td>
</tr>
<tr>
<td>12.9</td>
<td>The environments <code>{WithArrows}</code> and <code>{DispWithArrows}</code></td>
<td>51</td>
</tr>
<tr>
<td>12.9.1</td>
<td>Code before the <code>\halign</code></td>
<td>51</td>
</tr>
<tr>
<td>12.9.2</td>
<td>The construction of the preamble of the <code>\halign</code></td>
<td>54</td>
</tr>
<tr>
<td>12.9.3</td>
<td>The environment <code>{WithArrows}</code></td>
<td>56</td>
</tr>
<tr>
<td>12.9.4</td>
<td>After the construction of the <code>\halign</code></td>
<td>58</td>
</tr>
<tr>
<td>12.9.5</td>
<td>The command of end of row</td>
<td>59</td>
</tr>
<tr>
<td>12.9.6</td>
<td>The environment <code>{DispWithArrows}</code></td>
<td>63</td>
</tr>
<tr>
<td>12.10</td>
<td>The commands <code>\tag</code>, <code>\notag</code>, <code>\label</code>, <code>\tagnextline</code> and <code>\qedhere</code> for <code>{DispWithArrows}</code></td>
<td>68</td>
</tr>
<tr>
<td>12.11</td>
<td>We draw the arrows</td>
<td>70</td>
</tr>
<tr>
<td>12.11.1</td>
<td>The command <code>update_x</code></td>
<td>80</td>
</tr>
<tr>
<td>12.11.2</td>
<td>We draw the arrows of type o</td>
<td>80</td>
</tr>
<tr>
<td>12.12</td>
<td>The command <code>\Arrow</code> in code-after</td>
<td>83</td>
</tr>
<tr>
<td>12.13</td>
<td>The command <code>\MultiArrow</code> in code-after</td>
<td>85</td>
</tr>
<tr>
<td>12.14</td>
<td>The error messages of the package</td>
<td>87</td>
</tr>
<tr>
<td>12.15</td>
<td>The command <code>\WithArrowsNewStyle</code></td>
<td>93</td>
</tr>
<tr>
<td>12.16</td>
<td>The options up and down</td>
<td>94</td>
</tr>
</tbody>
</table>

## History

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td></td>
<td>99</td>
</tr>
</tbody>
</table>