# The `tikz-cd` package

Florêncio Neves*

Version 0.3a, of August 15, 2012

The general-purpose drawing package Ti*k*Z can be used to typeset commutative diagrams and other kinds of mathematical pictures, generating high-quality results (see for example [2] or [3]). This package facilitates the creation of such diagrams by providing a convenient set of macros and reasonable default settings. Familiarity with Ti*k*Z is helpful but not necessary, since the examples contained here cover the most common situations.

This package also includes an arrow tip library closely matching the Computer Modern arrow tips.

## Contents

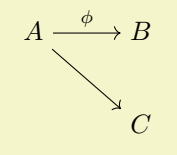


# 1 Basic usage

Commutative diagrams are created with the `tikzcd` environment. Its content describes a matrix, similarly to the `\matrix` command in Ti*k*Z or the `align` environment in LaTeX. Everything is typeset in math mode, but you will probably want use `tikzcd` inside `\[ ... \]` or an `equation` environment, so that the diagram is placed on a new line and centered.

## 1.1 Inserting arrows

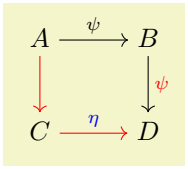
Inside the `tikzcd` environment, the command `\arrow` is provided to produce arrows. In its simplest form, it takes one argument, a string containing the characters `r`, `l`, `u` or `d`, standing for right, left, up and down, that determine the arrow target. A label can be placed on an arrow by providing a second argument.




```
\begin{tikzcd}
  A \arrow{rd} \arrow{r}{\phi} & B \\
                               & C
\end{tikzcd}
```

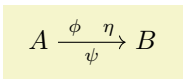
* *E-mail:* florencioneves@gmail.com

You can control the appearance of the arrow by placing an argument inside square braces before the direction parameter. It may contain any option that can be passed to Ti*k*Z's `\path` command. Similarly, a label can be modified by an argument in square braces right before it. It may contain anything that can be passed to Ti*k*Z's `node` operator.



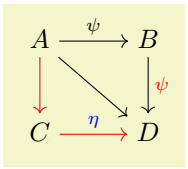
```
\begin{tikzcd}
  A \arrow{r}{\psi} \arrow[color=red]{d}
        & B \arrow{d}[color=red]{\psi} \\
  C \arrow[color=red]{r}[color=blue]{\eta}
        & D
\end{tikzcd}
```

Arrows can actually have an arbitrary number of labels, each one specified by juxtaposing `{`⟨*text*⟩`}` or `[`⟨*options*⟩`]{`⟨*text*⟩`}` to the `\arrow` command.



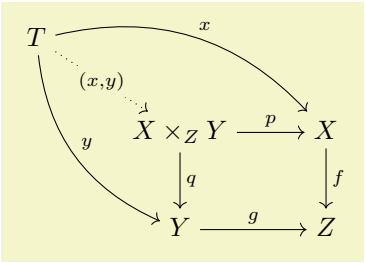
```
\begin{tikzcd}
  A \arrow{r}[near start]{\phi}[swap]{\psi}[near end]{\eta} & B
\end{tikzcd}
```

To save typing, the command `\ar` is provided as a shorthand to `\arrow`. There are also commands `\rar`, `\lar`, `\uar`, `\dar`, `\urar`, `\ular`, `\drar` and `\dlar` that act like `\arrow{r}`, `\arrow{l}`, ..., `\arrow{dl}`. They can take an optional argument in square braces to modify the arrow, followed by label specifiers of the form `{`⟨*text*⟩`}` or `[`⟨*options*⟩`]{`⟨*text*⟩`}`. Thus, the diagram above can be rewritten as follows.



```
\begin{tikzcd}
  A \drar \rar{\psi} \dar[color=red]
        & B \dar[]{\psi} \\
  C \rar[color=red][color=blue]{\eta}
        & D
\end{tikzcd}
```

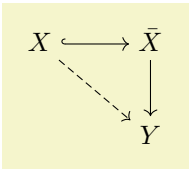
We provide one more example, reproduced from XY-pic user's guide.



```
\begin{tikzcd}
  T
  \arrow[bend left]{drr}{x}
  \arrow[bend right]{ddr}{y}
  \arrow[dotted]{dr}[description]{(x,y)} & & \\
      & X \times_Z Y \arrow{r}{p} \arrow{d}{q} & X \arrow{d}{f} \\
      & Y \arrow{r}{g}                         & Z
\end{tikzcd}
```

## 1.2 Changing arrow tips

For some LaTeX arrow-producing commands, there is a corresponding option (without a "\") that can be passed to `\arrow` to produce that kind of arrow. Notice the use of `hookrightarrow` and `dashrightarrow` in the example below.



```
\begin{tikzcd}
  X \arrow[hookrightarrow]{r} \arrow[dashrightarrow]{rd}
    & \bar{X} \arrow{d}\\
    & Y
\end{tikzcd}
```

The following list shows all available arrows.

| | |
|---|---|
| `rightarrow` | yields $\longrightarrow$ |
| `leftarrow` | yields $\longleftarrow$ |
| `leftrightarrow` | yields $\longleftrightarrow$ |
| `dash` | yields $\dashrightarrow$ |
| `Rightarrow` | yields $\Longrightarrow$ |
| `Leftarrow` | yields $\Longleftarrow$ |
| `Leftrightarrow` | yields $\Longleftrightarrow$ |
| `equal` | yields $=$ |
| `mapsto` (or `maps to`) | yields $\longmapsto$ |
| `mapsfrom` | yields $\longmapsfrom$ |
| `hookrightarrow` (or `hook`) | yields $\hookrightarrow$ |
| `hookleftarrow` | yields $\hookleftarrow$ |
| `rightharpoonup` | yields $\rightharpoonup$ |
| `rightharpoondown` | yields $\rightharpoondown$ |
| `leftharpoonup` | yields $\leftharpoonup$ |
| `leftharpoondown` | yields $\leftharpoondown$ |
| `dashrightarrow` (or `dashed`) | yields $\dashrightarrow$ |
| `dashleftarrow` | yields $\dashleftarrow$ |
| `rightarrowtail` (or `tail`) | yields $\rightarrowtail$ |
| `leftarrowtail` | yields $\leftarrowtail$ |
| `twoheadrightarrow` (or `two heads`) | yields $\twoheadrightarrow$ |
| `twoheadleftarrow` | yields $\twoheadleftarrow$ |
| `rightsquigarrow` (or `squiggly`) | yields $\rightsquigarrow$ |
| `leftsquigarrow` | yields $\leftsquigarrow$ |
| `leftrightsquigarrow` | yields $\leftrightsquigarrow$ |

Some of the arrows above have an alternate name shown in parenthesis. These forms behave a little differently from the original ones, in that they can be superimposed with other arrows.

$A \rightarrowtail B$

```
\begin{tikzcd}
  A \arrow[tail, two heads, dashed]{r} & B
\end{tikzcd}
```

It is also possible to select arrow tips using the Ti*k*Z `arrow` option directly (e.g., by saying `right hook->`). However, the method described here above is more appropriate, as it abstracts the choice of an actual arrow tip design (see also the key `commutative diagrams/arrow style` below), and is arguably better from the mnemonic standpoint.

## 2 Controlling the appearance of diagrams

This section lists all customization keys defined by this package. Options can be made take effect in different scopes:

1. Globally, using the command `\tikzset`. This can be done in the document preamble, or in the body, to affect all diagrams appearing thereafter.
2. For a particular diagram, by placing options in the optional argument to the `tikzcd` environment. Such options are applied to the `tikzpicture` environment generated by `tikzcd` (cf. §3.1).
3. For a particular arrow or label, by placing options in one of the optional arguments of `\arrow`.

Of course, not all options make sense in all contexts.

All keys provided by this package are located in the path `/tikz/commutative diagrams`. Methods 2 and 3 above will search in this path by default; if a key is not found there, an attempt is made to find it in `/tikz`. When using method 1, it is convenient to change the default search path by using

`\tikzset{commutative diagrams/.cd,` ⟨*options*⟩`}`.

Besides the keys described in this manual, numerous Ti*k*Z parameters can affect the appearance of a diagram. However, only a few of them (namely those appearing in `every diagram`, `every arrow` and `every label` below) are reinitialized when `tikzcd` is called. This means that modifying a certain Ti*k*Z parameter globally may or may not affect the output of `tikzcd`.

## 2.1 General options

/tikz/commutative diagrams/every diagram (style, no value)

This style is applied to every `tikzcd` environment. Initially, it contains the following:

```
/tikz/commutative diagrams/.cd,
/tikz/cells={nodes={shape={asymmetrical rectangle}}},
/tikz/baseline=0pt,
row sep=normal,
column sep=normal
```

The `baseline=0pt` setting is used to make equation numbers be placed correctly. The `asymmetrical rectangle` shape, used in the matrix nodes, is described in §4.1.

/tikz/commutative diagrams/diagrams=⟨*options*⟩ (no default)

This key appends ⟨*options*⟩ to the style `/tikz/commutative diagrams/every diagram`.

/tikz/commutative diagrams/row sep=⟨*size*⟩ (no default, initially `normal`)

This key acts as a "frontend" to Ti*k*Z's `/tikz/row sep` key. If the key

`/tikz/commutative diagrams/row sep/`⟨*size*⟩

stores a ⟨*value*⟩, then it is read and `/tikz/row sep`=⟨*value*⟩ is set. If the key above is not initialized, then ⟨*size*⟩ is presumably a dimension, and `/tikz/row sep`=⟨*size*⟩ is set.

The initially available ⟨*size*⟩'s, and their values, are the following:

| `tiny` | `small` | `scriptsize` | `normal` | `large` | `huge` |
|---|---|---|---|---|---|
| 1.25 ex | 2.5 ex | 3.75 ex | 5 ex | 7.5 ex | 10 ex |

Notice that setting, say, `row sep=1cm` globally with `\tikzset` will have no effect, since the `row sep` option is re-set at the beginning of each diagram. To make all diagrams have `row sep` equal to 1 cm, you can modify the meaning of `normal` by saying

`\tikzset{commutative diagrams/row sep/normal=1cm}`.

You can also create new sizes, but PGF requires new keys to be initialized explicitly. For example, to create a size `my size`, meaning 1 ex, you should use

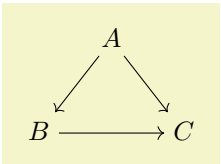
`\tikzset{commutative diagrams/row sep/my size/.initial=1ex}`.

/tikz/commutative diagrams/column sep=⟨*size*⟩ (no default, initially `normal`)

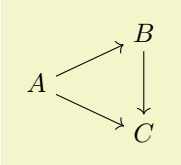
This works analogously to the `row sep` key above. The sizes available initially are the following:

| `tiny` | `small` | `scriptsize` | `normal` | `large` | `huge` |
|---|---|---|---|---|---|
| 1.5 ex | 3 ex | 4.5 ex | 6 ex | 9 ex | 12 ex |

In the examples below, the triangular diagrams would look too wide or too tall if the column or row separation were not set appropriately.




```
\begin{tikzcd}[column sep=small]
                & A \arrow{dl}\arrow{dr} & \\
  B \arrow{rr} &                        & C
\end{tikzcd}
```




```
\begin{tikzcd}[row sep=tiny]
                          & B \arrow{dd} \\
  A \arrow{ur}\arrow{dr} &              \\
                          & C
\end{tikzcd}
```

The commands below are available only inside a `tikzcd` and are intended to be used as arguments to `&` and `\\` when spacing adjustments are necessary (see also [4, §17.3.2]).

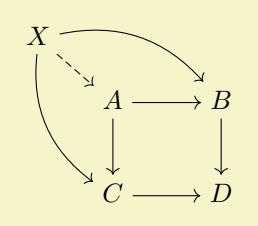
`\colsep{`⟨*size*⟩`}`

Returns the value stored by the key `/tikz/commutative diagrams/column sep/`⟨*size*⟩.

`\rowsep{`⟨*size*⟩`}`

Returns the value stored by the key `/tikz/commutative diagrams/row sep/`⟨*size*⟩.

In the following example, $X$ would appear too far from the square without any spacing adjustment.




```
\begin{tikzcd}
  X \ar[bend right]{rdd} \ar[bend left]{rrd} \ar[dashed]{dr}
    &[\colsep{small}-\colsep{normal}]  & \\[\rowsep{small}-\rowsep{normal}]
    & A \ar{r}\ar{d}                   & B \ar{d} \\
    & C \ar{r}                         & D
\end{tikzcd}
```

`/tikz/commutative diagrams/math mode=`⟨*boolean*⟩ (no default, initially `true`)

This keys determines whether the contents of a diagram are typeset in math mode or not. If set globally or diagram-wise, it affects both the diagram entries and arrow labels. If used with `\arrow`, it affects only its labels.

`/tikz/commutative diagrams/background color=`⟨*color*⟩ (no default, initially `white`)

This key stores the name of a color, and is read by styles that fill the background, such as `description` and `crossing over`. It does not cause the background of diagrams to be filled.

## 2.2 Options for arrows

Besides the options and styles provided by this package, there are several keys defined by Ti*k*Z that are useful for arrows, such as `dashed`, `dotted`, and its relatives, `line width=`⟨*dimension*⟩, `color=`⟨*color*⟩, `bend right`, `bend left`, `in=`⟨*angle*⟩, `out=`⟨*angle*⟩, `loop`, etc. See the PGF manual [4].

`/tikz/commutative diagrams/every arrow` (style, no value)

This style is applied to every `\arrow`. Initially, it contains the following:

```
/tikz/commutative diagrams/.cd,
/tikz/draw,
default arrow
```

The `default arrow` style is similar to `rightarrow`, but may perform some additional tasks, such as setting the line width.

`/tikz/commutative diagrams/arrows=`⟨*options*⟩ (no default)

This key appends ⟨*options*⟩ to the style `/tikz/commutative diagrams/every arrow`.

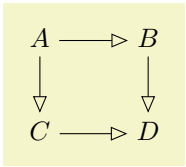
`/tikz/commutative diagrams/arrow style=`⟨*style*⟩ (no default, initially `computer modern`)

Setting this key causes the several arrow-tip-specifying styles listed in §1.2 to be redefined. In particular, it makes the style `/tikz/commutative diagrams/default arrow`, which is automatically applied to every arrow, to be redefined. Currently, ⟨*style*⟩ can be one of the following:

`computer modern` This is the initial setting and is suitable for documents using Computer Modern or Latin Modern fonts.

`tikz` This style uses the arrow tips defined in Ti*k*Z's `arrows` library (which has to be loaded before setting this option). It honors the option `/tikz/>`, and it keeps the parameter `/tikz/line width` untouched.

If you are using a font different from Computer Modern, you may find better results by selecting the `tikz` arrow style, setting `/tikz/>` to the value that best matches your font (among those shown in [4, §23]), and adjusting `/tikz/line width` if necessary. The following example, if not particularly elegant, should be instructive.



```
\usetikzlibrary{arrows}

\tikzset{
  commutative diagrams/.cd,
  arrow style=tikz,
  diagrams={>=open triangle 45, line width=tikzcdrule}}

\begin{tikzcd}
  A \arrow{r} \arrow{d} & B \arrow{d}\\
  C \arrow{r}           & D
\end{tikzcd}
```

/tikz/commutative diagrams/start anchor=⟨*anchor*⟩ (no default)

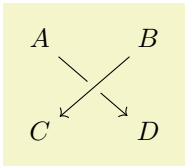
This key specifies at which anchor of the current node the arrow should start.

/tikz/commutative diagrams/end anchor=⟨*anchor*⟩ (no default)

This key specifies at which anchor of the target node the arrow should arrive.

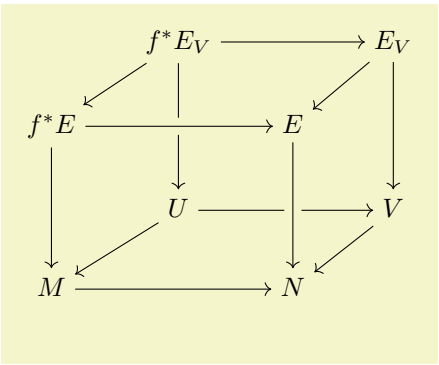
/tikz/commutative diagrams/crossing over (style, no value)

This style makes a thicker line, with color `/tikz/commutative diagrams/background color`, to be drawn under the current arrow, simulating the effect of its passing over other arrows.



```
\begin{tikzcd}
  A \arrow{dr} & B \arrow[crossing over]{dl} \\
  C            & D
\end{tikzcd}
```

Note that, since arrows are drawn in the order they are read, some arrows may need to run "backwards" to achieve the desired result. The following picture, adapted from [2], illustrates this.



```
\begin{tikzcd}[row sep=scriptsize, column sep=scriptsize]
  & f^* E_V \arrow{dl}\arrow{rr}\arrow{dd} & & E_V \arrow{dl}\arrow{dd} \\
  f^* E \arrow[crossing over]{rr}\arrow{dd} & & E \\
  & U \arrow{dl}\arrow{rr} &  & V \arrow{dl} \\
  M \arrow{rr} & & N\arrow[crossing over, leftarrow]{uu} \\
\end{tikzcd}
```

/tikz/commutative diagram/crossing over clearance=⟨*dimension*⟩ (no default, initially `1.5ex`)

This key specifies the width of the background-colored line drawn under a `crossing over` arrow.

## 2.3 Options for labels

Besides the options provided by this package and listed below, there are many options provided by Ti*k*Z that are useful for labels, such as `above`, `below`, `left`, `right`, `swap` (which makes the label be placed in the opposite side to the default), `sloped`, `pos=`⟨*fraction*⟩, `near start`, `near end`, `inner sep=`⟨*dimension*⟩, `font=`⟨*font command*⟩, `text width=`⟨*dimension*⟩, etc. See the PGF manual [4].

/tikz/commutative diagrams/every label (style, no value)

This style is applied to every label produced with `\arrow`. It is initially set to

```
/tikz/commutative diagrams/.cd,
/tikz/auto,
/tikz/font=⟨something⟩,
/tikz/inner sep=0.5ex
```

where ⟨*something*⟩ is something that makes `\scriptstyle` be applied to labels in math mode.

The key `/tikz/auto` makes the label be placed to the left of the arrow, considering "front" as the direction the arrow points at. The key `/tikz/inner sep` controls the distance between a label and the corresponding arrow.

`/tikz/commutative diagrams/labels=`⟨*options*⟩ (no default)

This key appends ⟨*options*⟩ to the style `/tikz/commutative diagrams/every label`.

`/tikz/commutative diagrams/description` (style, no value)

This style causes the label to be placed over the arrow, with the background filled. The clearance around the label is determined by `/tikz/inner sep`.

$A \xrightarrow{\phi} B$

```
\begin{tikzcd}
  A \arrow{r}[description]{\phi} & B
\end{tikzcd}
```

# 3 Advanced usage

This section provides further details on the functioning of this package, with the aim of showing how a more or less arbitrary interaction with other Ti*k*Z features is possible.

## 3.1 Internals of `tikzcd` and the arrow commands

The `tikzcd` environment works by substituting code of the form

```
\begin{tikzcd}[⟨options⟩] ⟨contents⟩ \end{tikzcd}
```

with roughly the following:

```
\begin{tikzpicture}[commutative diagrams/every diagram, ⟨options⟩]
  \matrix[matrix of [math] nodes] {
    ⟨contents⟩ \\
  };
  ⟨paths⟩
\end{tikzpicture}
```

Not shown above are a number of initialization procedures, such as defining `\arrow` and its relatives. Note that the next-row command `\\` for the last row is inserted by `tikzcd`, and therefore does not need to be present in ⟨*contents*⟩. The `\matrix` is supplied the option `matrix of nodes` or `matrix of math nodes` as specified by the option `commutative diagrams/math mode`. Notice also that you can use the key `execute at end picuture` in ⟨*options*⟩ to have arbitrary Ti*k*Z code executed after a diagram is drawn.

Initially, ⟨*paths*⟩ is empty. The command `\arrow[`⟨*path options*⟩`]{`⟨*direction*⟩`}` does nothing at the point it is inserted, and causes the following code to be added to ⟨*paths*⟩:

```
\path[commutative diagrams/every arrow, ⟨path options⟩] (⟨current node⟩) to ⟨labels⟩ (⟨target node⟩);
```

Here, ⟨*current node*⟩ is the node corresponding to the matrix cell where the command `\arrow` is present, and ⟨*target node*⟩ is determined by ⟨*direction*⟩ as explained in §1.1.

Initially, ⟨*labels*⟩ is the empty string. A label specifier of the form `{`⟨*label text*⟩`}` or `[`⟨*label options*⟩`]{`⟨*label text*⟩`}` immediately following the ⟨*direction*⟩ argument or a previous label specifier causes the string

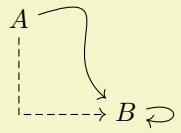
```
node [commutative diagrams/every label, ⟨label options⟩] {[$]⟨label text⟩[$]}
```

to be appended to ⟨*labels*⟩. Dollars signs surround ⟨*label text*⟩ depending on the setting `commutative diagrams/math mode`.

The abbreviated forms `\rar`, `\dar`, ..., have entirely analogous effect.

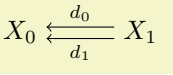
## 3.2 Tweaking `to` paths

Recall that the `to` path operation used in the paths created by `\arrow` can take a number of options, as described in §14.14 and §51 of the PGF manual [4]. In particular, the key `/tikz/to path` determines the path that is actually drawn, and can be used to do all sorts of fiddling.




```
\begin{tikzcd}
  A \arrow[dashed, to path=|- (\tikztotarget)]{dr}
    \arrow[to path={..controls +(1.5,0.5) and +(-1,0.8).. (\tikztotarget)}]{dr}
    & \\
    & B \arrow[loop right]{}
\end{tikzcd}
```

A common use case is to produce parallel arrows.




```
\begin{tikzcd}
  X_0
    \rar[to-,
         to path={
           ([yshift=0.5ex]\tikztostart.east) --
           ([yshift=0.5ex]\tikztotarget.west) \tikztonodes}]{d_0}
    \rar[to-,
         to path={
           ([yshift=-0.5ex]\tikztostart.east) --
           ([yshift=-0.5ex]\tikztotarget.west) \tikztonodes}][swap]{d_1}
  & X_1
\end{tikzcd}
```

Whenever one starts using long options repeatedly, it might be a good idea to define a new style. In the example below, this is done with the statement `shift up/.style={ ... }` inside `\tikzset`. Styles can take one argument, which replaces all occurrences of `#1` in its definition.


```
\tikzset{
  shift up/.style={
    to path={([yshift=#1]\tikztostart.east) -- ([yshift=#1]\tikztotarget.west) \tikztonodes}
  }
}

\begin{tikzcd}
  X_0
    \rar[to-, shift up=0.5ex]
    \rar[to-, shift up=-0.5ex] &
  X_1
    \rar[to-, shift up=1ex]
    \rar[to-]
    \rar[to-, shift up=-1ex] &
  X_2
    \rar[to-, shift up=1.5ex]
    \rar[draw=none][anchor=base,yshift=-1ex]{\vdots}
    \rar[to-, shift up=-1.5ex] &
  \cdots
\end{tikzcd}
```

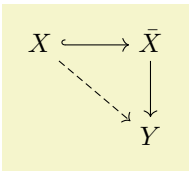
In the next example, empty labels are used to create nodes for later reference, and in the third `\arrow` command the `to path` key is used in a way that, in the terminology of the previous section, ⟨*current node*⟩ and ⟨*target node*⟩ are completely ignored.


```
\begin{tikzcd}
  A \arrow[bend left=50]{r}[name=U,below]{}
    \arrow[bend right=50]{r}[name=D]{} &
  B \arrow[Rightarrow,to path=(U) -- (D)]{}
\end{tikzcd}
```

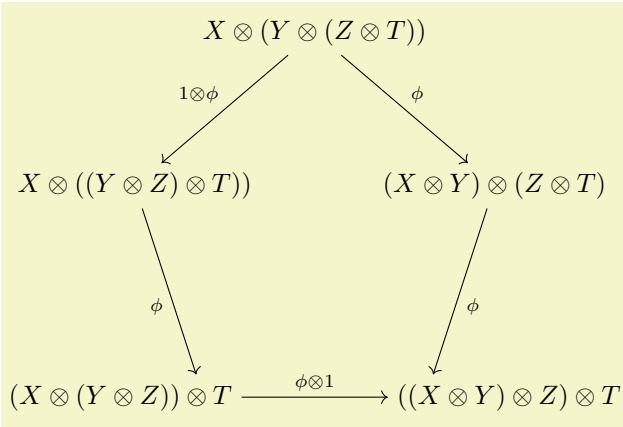
## 3.3 Drawing diagrams directly with Ti*k*Z

If the tools provided by this package prove not flexible enough for some application, you can use the methods described in [2] and [3] to draw diagrams directly with Ti*k*Z. In this case, you can still use the styles provided

here in order to achieve uniformity with diagrams drawn with `tikzcd`. The pictures below show how this can be done (the second one is adapted from [3]).



```
\begin{tikzpicture}[commutative diagrams/every diagram]
  \matrix[matrix of math nodes, name=m] {
    X & \bar X \\
      & Y       \\};
  \path[commutative diagrams/.cd, every arrow, every label]
    (m-1-1) edge[commutative diagrams/hook] (m-1-2)
            edge[commutative diagrams/dashed] (m-2-2)
    (m-1-2) edge (m-2-2);
\end{tikzpicture}
```



```
\begin{tikzpicture}[commutative diagrams/every diagram]
  \node (P0) at (90:2.8cm) {$X\otimes (Y\otimes (Z\otimes T))$};
  \node (P1) at (90+72:2.5cm) {$X\otimes ((Y\otimes Z)\otimes T))$} ;
  \node (P2) at (90+2*72:2.5cm) {\makebox[5ex][r]{$(X\otimes (Y\otimes Z))\otimes T$}};
  \node (P3) at (90+3*72:2.5cm) {\makebox[5ex][l]{$((X\otimes Y)\otimes Z)\otimes T$}};
  \node (P4) at (90+4*72:2.5cm) {$(X\otimes Y)\otimes (Z\otimes T)$};

  \path[commutative diagrams/.cd, every arrow, every label]
    (P0) edge node[swap] {$1\otimes\phi$} (P1)
    (P1) edge node[swap] {$\phi$} (P2)
    (P2) edge node {$\phi\otimes 1$} (P3)
    (P4) edge node {$\phi$} (P3)
    (P0) edge node {$\phi$} (P4);
\end{tikzpicture}
```

# 4 Additional goodies

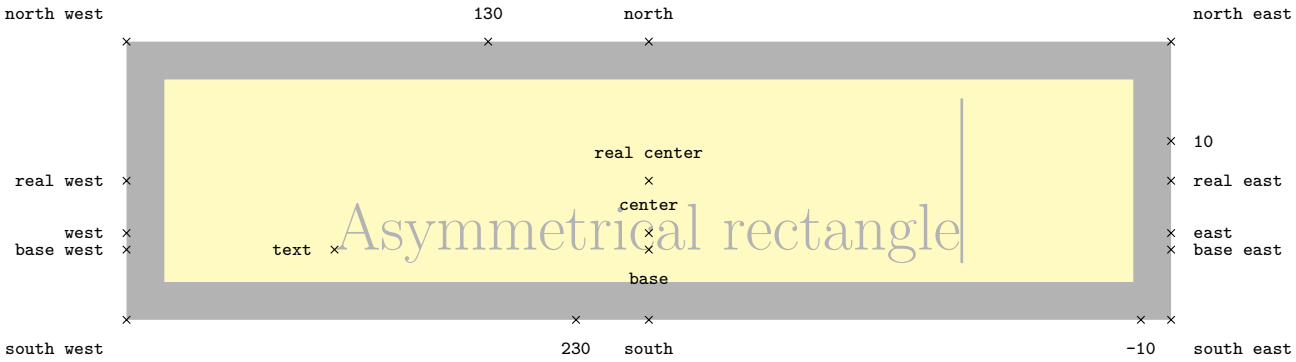
This package provides some general PGF infrastructure to meet its ends, as described in this section.

## 4.1 The `asymmetrical rectangle` shape

The `asymmetrical rectangle` shape is similar to the `rectangle` shape, except that the `center` anchor is not located at its geometric center, but rather right above the `base` anchor, at a distance that can be specified by the following key.

`/tikz/commutative diagrams/center yshift=`⟨*dimension*⟩ (no default, initially `tikzcdaxis`)

The picture below shows some of the available anchors. All anchors provided by `rectangle` are available and behave exactly the same, except for `center`, `west`, `east`, and the numerical anchors. The anchors `real center`, `real west` and `real east` agree with `rectangle`'s `center`, `west` and `east`.




This shape is used in `tikzcd`'s matrix cells to ensure arrows between nodes in the same row are drawn horizontally. When Ti*k*Z draws a path between two nodes, it chooses endpoints lying on their borders in such a way that the path points towards the `center` of the nodes. With the `rectangle` shape, the anchor `center` lies halfway between two nonadjacent vertices and therefore its position depends on the height and depth of the text, potentially causing slanted lines to be draw. The picture above, for instance, has a humongous ascender (namely, the rule after the word "rectangle"), so its `real center` lies well above its `center`.

## 4.2 Reading font parameters

The following are `pgfmath` functions used to access relevant math font parameters. They take no arguments, but the result depends of the currently selected font size.

`tikzcdaxis`

Returns the "axis height" parameter of the document's math font.

`tikzcdrule`

Returns the fraction rule thickness of the document's math font.

## 4.3 Computer Modern arrow tips

The naming scheme of the Computer Modern-like arrow tips provided by this package parallels that of PGF's `arrows` library, documented in §23 of the PGF manual [4]. To match the Computer Modern typeface at size 10 pt, line width should be set to 0.4 pt; for larger font sizes, scale this parameter accordingly, or use the value 0.0929 ex.

Notice that by using the mechanism explained in §1.2, it is not necessary, and in fact not advisable, to directly use the arrow tips listed in this section when creating diagrams with `tikzcd`.

Incidentally, Ti*k*Z's original `to` arrow tip seems to be based on the pre-1992 version of Computer Modern which, in spite of its author's wish [1], can still be found in many systems. TEXLive, for instance, distributed the old version up until 2007 or 2008. Therefore, an up-to-date TEX distribution may be necessary to get matching arrows in formulas and diagrams.

**Basic arrow tips**

`cm to` yields ⟷
`cm to reversed` yields ⟵
`cm bold to` yields ⟷ (with a line 50% thicker)
`cm |` yields ⟼
`cm o` yields ⟛
`cm *` yields ●—●
`cm cap` yields —

**Arrow tips for double lines**

`cm implies` yields ⟹
`cm implies cap` yields ═

**Hooks**

`cm left hook` yields ↪
`cm right hook` yields ↩

**Double arrow tips**

`cm double to` yields $\twoheadleftarrow\!\!\!\!\!\!\rightarrow$

`cm double to reversed` yields $\twoheadrightarrow\!\!\!\!\!\!\leftarrow$

**Partial arrow tips**

`cm left to` yields $\leftarrow\!\!\!\!\!\!\rightarrow$

`cm left to reversed` yields $\leftarrow\!\!\!\!\!\!\rightarrow$

`cm right to` yields $\leftarrow\!\!\!\!\!\!\rightarrow$

`cm right to reversed` yields $\leftarrow\!\!\!\!\!\!\rightarrow$

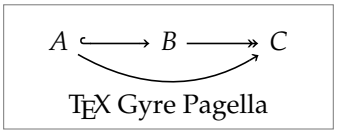
## 4.4 Font arrow tips

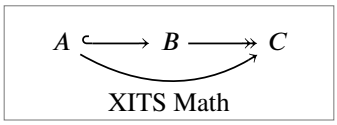
*This is an experimental feature. It may be modified, moved elsewhere or even removed in the future.*

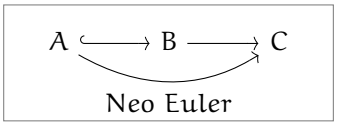
As an attempt to provide a general solution to the problem of having matching arrow tips in text and pictures, this feature produces arrow tips that consist of (pieces of) characters carefully placed at the endpoints of the path. To activate it in `tikzcd` diagrams, say

`\tikzset{commutative diagrams/arrow style=math font}`.

It is also necessary to load `amssymb` or some other package defining the symbols corresponding to the arrow tips you want to use. Here are some samples:




Setting the key above also makes arrow tips named `math to`, `math to reversed`, `math cap`, `math |`, `math o`, `math implies`, `math implies cap`, `math left hook`, `math right hook`, `math double to`, `math left to`, and `math right to` available for use in Ti*k*Z pictures in the usual way. These tips do not scale with the line width, but their size depends on the current font size, so you will probably want to set `line width=tikzcdrule` when using them. It is possible to define more arrows or tweak the existing ones. Look into the source code of this package if you are interested. (And let me know what you think!)

The transition between a line and the arrow tip may be visible with some document viewers and printers, as you may notice from the picture below. When using pdfTEX (or LuaTEX) with PDF output, a measure is taken to alleviate this.

pdfTEX $\longleftarrow$ Other TEX engines $\longleftarrow$

I don't know of any automatic way of determining the distance between the lines in a $\Rightarrow$, so this parameter has to be entered manually via the key `/tikz/commutative diagrams/font arrows/double distance` if you want to use double lines.

# References


[1] Donald Knuth, *Important message to all users of TEX*. Available at http://www-cs-staff.stanford.edu/~uno/cm.html

[2] Felix Lenders, *Commutative diagrams using TikZ*. Available at http://www.felixl.de/commu.pdf.

[3] James Milne, *Guide to commutative diagrams*. Available at http://www.jmilne.org/not/CDGuide.html.

[4] Till Tantau, *The TikZ and PGF packages: Manual for version 2.10*. Available at http://sourceforge.net/projects/pgf.