The `tikz-cd` package

Florêncio Neves*

Version 0.2c, of July 17, 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, as the examples contained here cover the most often encountered situations.

This package also includes an arrow tip library that match closely the arrows present in the Computer Modern typeface.

PGF version 2.10 is required.

Contents

1 Basic usage **1**
- 1.1 Inserting arrows . . . 1
- 1.2 Changing arrow tips . . . 2

2 Controlling the appearance of diagrams **3**
- 2.1 General options . . . 4
- 2.2 Options for arrows . . . 5
- 2.3 Options for labels . . . 6

3 Further topics **7**
- 3.1 Internals of `tikzcd` and the arrow commands . . . 7
- 3.2 Tweaking `to` paths . . . 8
- 3.3 The `asymmetrical rectangle` shape . . . 8
- 3.4 Drawing diagrams directly with Ti*k*Z . . . 9

4 Computer Modern arrow tips **9**

5 Font arrow tips **10**

6 Change history **11**

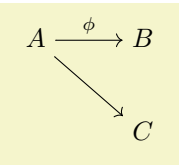
1 Basic usage

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 an `equation` environment or inside `\[ ... \]`, so that the diagram is placed on a new line and centered.

1.1 Inserting arrows

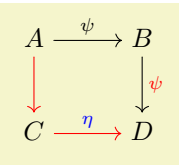
Inside the `tikzcd` environment, the command `\arrow` is provided to produce arrows. In its simplest form, it takes one argument, a string containing the characters `r`, `l`, `u` or `d`, standing for right, left, up and down, that determine the arrow target. A label can be placed on an arrow by providing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

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



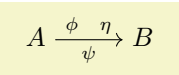
```
\begin{tikzcd}
  A \arrow{rd} \arrow{r}{\phi} & B \\
                               & C
\end{tikzcd}
```

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.



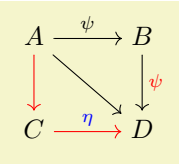
```
\begin{tikzcd}
  A \arrow{r}{\psi} \arrow[color=red]{d}
        & B \arrow{d}[color=red]{\psi} \\
  C \arrow[color=red]{r}[color=blue]{\eta}
        & D
\end{tikzcd}
```

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



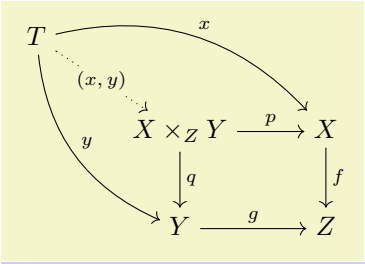
```
\begin{tikzcd}
  A \arrow{r}[near start]{\phi}[swap]{\psi}[near end]{\eta} & B
\end{tikzcd}
```

To save some 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{ur}`, and so forth. 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[][color=red]{\psi} \\
  C \rar[color=red][color=blue]{\eta}
        & D
\end{tikzcd}
```

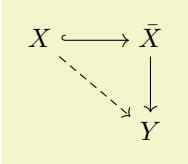
We provide one more example, reproduced from XY-pic user's guide.



```
\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 each arrow-producing command present in $\mathcal{AMS}$-LaTeX, 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}
```

Of course, a similar effect could be achieved by directly using Ti*k*Z options (e.g., using the option `right hook->`). However, the method 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 probably also better from the mnemonic standpoint.

The following list shows all available arrows.

`rightarrow`	yelds $\longrightarrow$
`Rightarrow`	yelds $\Longrightarrow$
`leftarrow`	yelds $\longleftarrow$
`Leftarrow`	yelds $\Longleftarrow$
`leftrightarrow`	yelds $\longleftrightarrow$
`Leftrightarrow`	yelds $\Longleftrightarrow$
`equals`	yelds $=$
`mapsto` (or `maps to`)	yelds $\longmapsto$
`hookrightarrow` (or `hook`)	yelds $\lhook\joinrel\longrightarrow$
`hookleftarrow`	yelds $\longleftarrow\joinrel\rhook$
`rightharpoonup`	yelds $\rightharpoonup$
`rightharpoondown`	yelds $\rightharpoondown$
`leftharpoonup`	yelds $\leftharpoonup$
`leftharpoondown`	yelds $\leftharpoondown$
`dashrightarrow` (or `dashed`)	yelds $\dashrightarrow$
`dashleftarrow`	yelds $\dashleftarrow$
`rightarrowtail` (or `tail`)	yelds $\rightarrowtail$
`leftarrowtail`	yelds $\leftarrowtail$
`twoheadrightarrow` (or `two heads`)	yelds $\twoheadrightarrow$
`twoheadleftarrow`	yelds $\twoheadleftarrow$
`rightsquigarrow` (or `squiggly`)	yelds $\rightsquigarrow$
`leftsquigarrow`	yelds $\leftsquigarrow$
`leftrightsquigarrow`	yelds $\leftrightsquigarrow$

Some of the styles above have two names. In these cases, the second one behaves a little differently from the first, in that it can be superimposed with other arrow styles.

$A \rightarrowtail\!\!\!\!\dashrightarrow\!\!\!\!\twoheadrightarrow B$

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

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 the current 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). Thus you can, for instance, use the `execute at end picuture` option in this situation to have arbitrary Ti*k*Z code executed after a diagram is drawn.
3. For the current arrow or label, by placing options in one of the optional arguments of `\arrow`.

Of course, not all options make sense in all contexts. For example, setting `row sep=1cm` globally with `\tikzset` will have no effect on diagrams, since the `row sep` option is re-set at the beginning of each diagram. To make all diagrams have `row sep` set to 1 cm, you can use

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

as detailed below.

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*⟩`}`.

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}}},
row sep=normal,
column sep=normal,
/tikz/baseline=0pt
```

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 §3.3.

/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

To change, say, the normal size (which is applied by default) to 1 cm, you can use the code

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

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

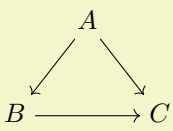
```
\tikzset{commutative diagrams/row sep/my size/.initial=1cm}.
```

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

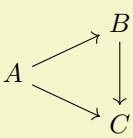
This works analogously to the row sep key above. The sizes available initially are the following:

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.

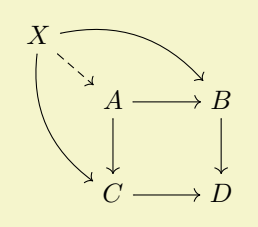
\colsep{⟨*size*⟩}

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

`\rowsep{`⟨*size*⟩`}`

Returns the value stored in 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 is 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 affect 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 options 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,
/tikz/commutative diagrams/default arrow
```

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

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

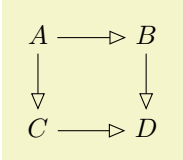
This key appends ⟨*options*⟩ to the style `/tikz/commutative diagrams/every arrow`.

`/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 (re)defined. In particular, it makes the style `/tikz/commutative diagrams/default arrow`, which is automatically applied to every arrow, to be defined. Currently, ⟨*style*⟩ can be one of `computer modern` or `tikz`. The `tikz` style uses the arrow tips defined in Ti*k*Z's `arrows` library, and it honors the option `/tikz/>`.

The `computer modern` style sets the parameter `/tikz/line width` in accordance to the current font size. The `tikz` arrow style, however, does not change it, allowing you to directly control the thickness of arrows.

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, 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=\tikzcdrulethickness}}

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

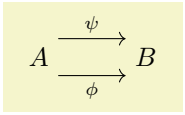
/tikz/commutative diagrams/start anchor=⟨*anchor*⟩ (default empty)

This style specifies at which anchor of the current node an arrow should start. The default behavior is not to specify an anchor, causing the arrow to start at the point in the boundary of the current node closest to its target.

/tikz/commutative diagrams/end anchor=⟨*anchor*⟩ (default empty)

This style works analogously, but refers to the target node of the arrow.

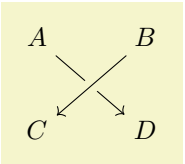
See the picture on §3.3 for some of the possible values for ⟨*anchor*⟩. The picture below illustrates the use of `end anchor` and `start anchor`.




```
\begin{tikzcd}
  A \rar[start anchor=north east, end anchor=north west]{\psi}
    \rar[start anchor=south east, end anchor=south west, swap]{\phi}
  & B
\end{tikzcd}
```

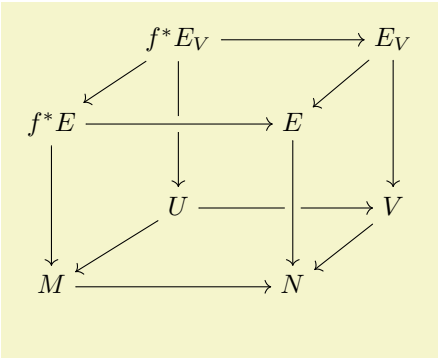
/tikz/commutative diagrams/crossing over (style, no value)

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 `6pt`)

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

\tikzcdrulethickness

This macro returns the value (in ex) of the math font default rule thickness.

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 the following:

```
/tikz/commutative diagrams/.cd,
/tikz/auto,
/tikz/font=\scriptsize,
/tikz/inner sep=0.5ex
```

The style `auto` makes the label be placed to the left of the arrow, considering "front" as the direction the arrow points at. The dimension `inner sep` controls the distance between an arrow and its labels.

/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 on the arrow, with the background filled.

$A \mathrel{-\phi\to} B$

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

/tikz/commutative diagrams/description clearance=⟨*dimension*⟩ (no default, initially `1.5pt`)

This key determines the size of the border around a label when the `description` style above is applied. Its value is used to set the key `/tikz/inner sep`.

3 Further topics

This sections provides further details on the functioning of this package, with the aim of showing how a more or less arbitrary use of Ti*k*Z features can be made inside it.

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}
```

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`. Also, not shown above are a number of initialization procedures, such as defining the commands `\arrow`, `\ar`, `\rar`, `\lar`, `\dar`, `\uar`, `\urar`, `\ular`, `\drar`, `\dlar`.

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. The abbreviated commands `\rar`, `\dar`, etc., generate the same kind of code.

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`.

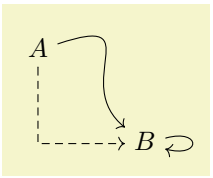
For the abbreviated form `\rar` (and similarly for `\dar`, `\lar`, etc.), the following are valid usages:

`\rar{`⟨*label text*⟩`}`⟨*more labels*⟩,
`\rar[`⟨*path options*⟩`]{`⟨*label text*⟩`}`⟨*more labels*⟩, or
`\rar[`⟨*path options*⟩`][`⟨*label options*⟩`]{`⟨*label text*⟩`}`⟨*more labels*⟩.

Their effect is entirely analogous to the above.

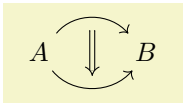
3.2 Tweaking `to` paths

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 option `/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}
```

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.



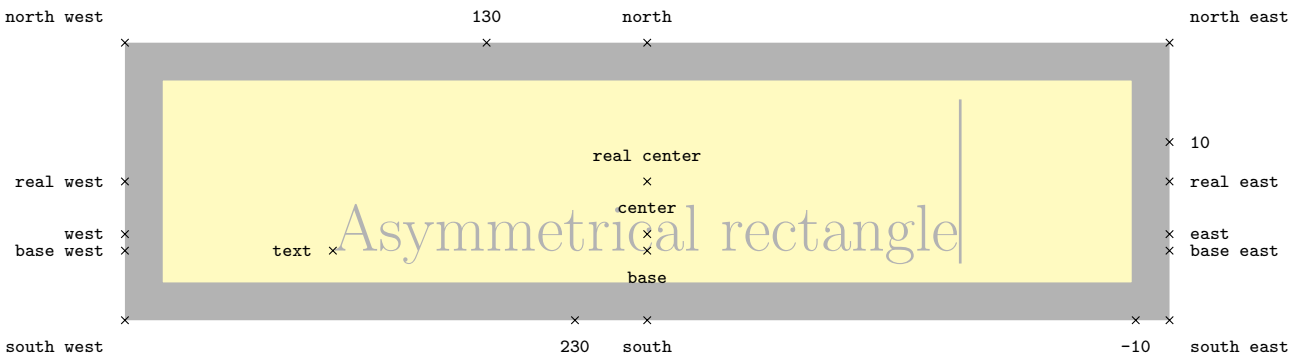

```
\begin{tikzcd}
  A \arrow[bend left=50]{r}[name=U,below]{}
    \arrow[bend right=50]{r}[name=D]{} &
  B \arrow[Rightarrow,to path=(U) -- (D)]{}
\end{tikzcd}
```

3.3 The `asymmetrical rectangle` shape

The `asymmetrical rectangle` shape, defined by this package, is very similar to the `rectangle` shape, except that its `center` anchor is not located at the geometric center of the shape, but rather right above the `base` anchor, at a distance that can be specified by the following key, which is initially set to `\tikzcdaxisheight`.

`/tikz/commutative diagrams/center shift=`⟨*dimension*⟩ (no default)

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`.



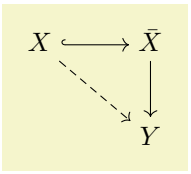
This shape is used in the matrix cells inside the `tikzcd` environment to ensure arrows between nodes in the same row are drawn horizontally. When Ti*k*Z draws a path between two nodes, it chooses endpoints lying on their borders in such a way that the path points towards the `center` of these nodes. 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.

`\tikzcdaxisheight`

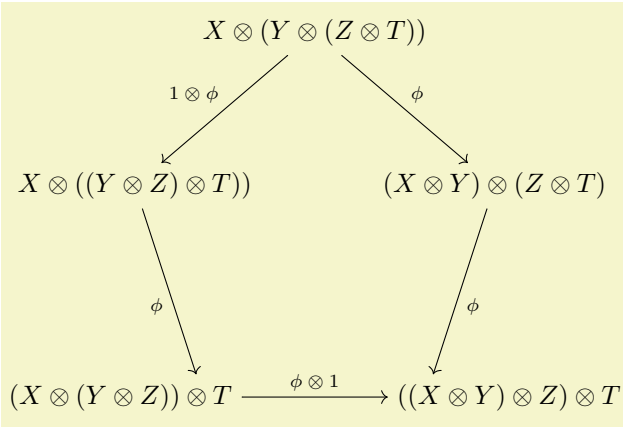
This macro returns the value (in ex) of the math font axis height.

3.4 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 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 macro `\tikzcdrulethickness`.

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` yelds ⟷
`cm to reversed` yelds ⟻⟼
`cm bold to` yelds ⟷ (with a line 50% thicker)
`cm |` yelds ⟼⟻
`cm o` yelds ⟛
`cm *` yelds ⧟
`cm cap` yelds —

Arrow tips for double lines

`cm implies` yelds ⟸⟹
`cm implies cap` yelds ═

Hooks

`cm left hook` yelds ↩
`cm right hook` yelds ↪

Double arrow tips

`cm double to` yelds ⇷
`cm double to reversed` yelds ⇸

Partial arrow tips

`cm left to` yelds ↞
`cm left to reversed` yelds ↜
`cm right to` yelds ↠
`cm right to reversed` yelds ↝

5 Font arrow tips

This is an experimental feature. It may be modified, moved elsewhere or even removed in the future, and it may conflict with future versions of PGF.

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

`\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.

The above option 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 currently selected font size, so you will probably want to set `line width=\tikzcdrulethickness` 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 ⟶ Other TEX engines ⟶

I don't know of any automatic way of determining the distance between the lines in a ⇒, 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.

6 Change history

Version 0.2c (July 2012) Added `\tikzcdrulethickness`, `\tikzcdaxisheight`, `\colsep`, `\rowsep`. Added experimental font arrows.

Version 0.2b (March 2012) Added `asymetrical rectangle` shape.

Version 0.2a (December 2011) Option `/tikz/commutative diagrams/path operation` removed in favor of direct use of `/tikz/to path`. Double lines with `tikz` arrow style now use `double equal sign distance`.

Version 0.2 (October 2011) Several changes.

Version 0.1 (September 2011) Preliminary release.

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.