{tikzcd}

Commutative diagrams with Ti*k*Z

Version 0.9c October 20, 2014

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. The present 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.

Contents

0 Disclaimer

Before version 1.0 of this package is released, minor modifications and bug fixes may be performed. All documents making an orthodox use of this package will continue to compile and generate essentially the same output. However, if you have strict stability requirements (for instance, if you want to assure no page break changes will happen in your documents), keep a copy of the current version of the file `tikzlibrarycd.code.tex` in your document's directory.

1 Getting started

To invoke this package in LaTeX, type

```
\usepackage{tikz-cd}
```

or load Ti*k*Z and then type

```
\usetikzlibrary{cd}
```

1.1 Creating a diagram

The basic tool to create a commutative diagram is the following environment.

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

The environment contents should describe a matrix, as in LaTeX's familiar `{tabular}` environment. The optional argument ⟨*options*⟩ may be used to modify the appearance of the diagram. Any of the customization keys described in this manual, as well as those originally provided by Ti*k*Z, can be used here. Arrows between matrix entries can be created with the `\arrow` command described below.

Everything inside `{tikzcd}` is typeset in math mode, but you will probably want to use it inside an `{equation}` environment or `\[ ... \]`, so that the diagram is placed on a new line and centered.

It is important to note that DVI viewers will not display diagrams correctly. It is necessary to convert the DVI file to the PDF or PS format—or, even better, use `pdflatex` to obtain a PDF file directly.

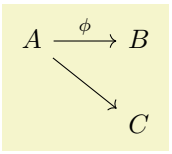
1.2 Inserting arrows

Inside the `{tikzcd}` environment, the following synonymous commands are provided to produce arrows.

```
\arrow[⟨options⟩]
\ar[⟨options⟩]
```

Here, ⟨*options*⟩ is a comma-separated list of options which can be used to specify the arrow target, add labels, change arrow tips, and perform additional modifications.

The arrow target can be specified by a direction parameter, which consists of a string of characters `r`, `l`, `d`, `u` (standing for right, left, down and up). Labels can be placed on an arrow by means of the quotes syntax, described in detail in the PGF manual [3, §17.10.4]. Notice the use of `"\phi"` in the example below.

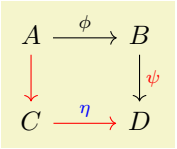



```
\begin{tikzcd}
  A \arrow[rd] \arrow[r, "\phi"] & B \\
                                 & C
\end{tikzcd}
```

To further modify the appearance of an arrow, note that ⟨*options*⟩ may contain any key that can be passed to Ti*k*Z's `\path` command. Similarly, a label can receive additional options via the syntax

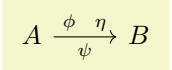
`"`⟨*label text*⟩`"`⟨*label options*⟩.

Both ⟨*label text*⟩ and ⟨*label options*⟩ need to be enclosed in curly braces if they contain commas.



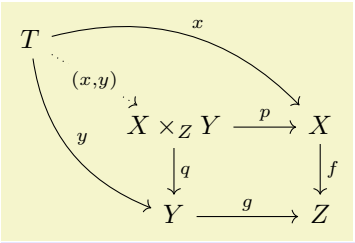

```
\begin{tikzcd}
  A \arrow[r, "\phi"] \arrow[d, red]
    & B \arrow[d, "\psi" red] \\
  C \arrow[r, red, "\eta" blue]
    & D
\end{tikzcd}
```

Arrows can have an arbitrary number of labels, by repeated use of arguments in quotes. The example below shows how to control the positioning of labels. Notice in particular that an apostrophe as ⟨*label option*⟩ causes the label to be placed on the opposite side of the arrow.

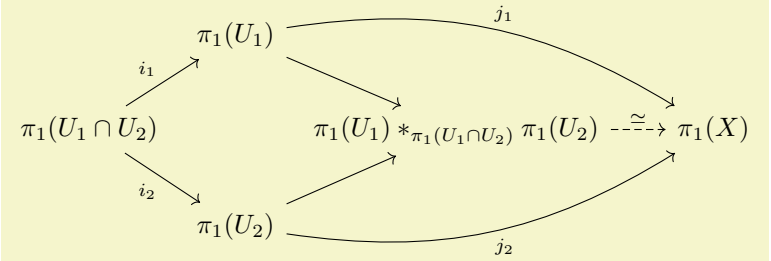



```
\begin{tikzcd}
  A \arrow[r, "\phi" near start, "\psi"', "\eta" near end] & B
\end{tikzcd}
```

We provide two real-life examples.



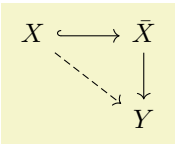

```
\begin{tikzcd}
  T
  \arrow[drr, bend left, "x"]
  \arrow[ddr, bend right, "y"]
  \arrow[dr, dotted, "{(x,y)}" description] & & \\
    & X \times_Z Y \arrow[r, "p"] \arrow[d, "q"]
      & X \arrow[d, "f"] \\
    & Y \arrow[r, "g"]
      & Z
\end{tikzcd}
```




```
\begin{tikzcd}[column sep=tiny]
    & \pi_1(U_1) \ar[dr] \ar[drr, "j_1", bend left=20]
      &
        &[1.5em] \\
  \pi_1(U_1\cap U_2) \ar[ur, "i_1"] \ar[dr, "i_2"']
    &
      & \pi_1(U_1) \ast_{ \pi_1(U_1\cap U_2)} \pi_1(U_2) \ar[r, dashed, "\simeq"]
        & \pi_1(X) \\
    & \pi_1(U_2) \ar[ur]\ar[urr, "j_2"', bend right=20]
      &
        &
\end{tikzcd}
```

1.3 Changing arrow tips

A set of `\arrow` options is provided to create different kinds of arrows. Some of these options have a short descriptive name, such as `hook`, and others are named after TEX arrow-producing commands (without a "\"), like `dashrightarrow`.




```
\begin{tikzcd}
  X \arrow[r, hook] \arrow[dr, dashrightarrow]
    & \bar{X} \arrow[d]\\
    & Y
\end{tikzcd}
```

The following list shows all available arrow types (each of them is a style key located in `/tikz/commutative diagrams`).

Basic arrows

`to head`	yields $\xrightarrow{}$
`rightarrow`	yields $\longrightarrow$
`leftarrow`	yields $\longleftarrow$
`leftrightarrow`	yields $\longleftrightarrow$
`Rightarrow`	yields $\Longrightarrow$
`Leftarrow`	yields $\Longleftarrow$
`Leftrightarrow`	yields $\Longleftrightarrow$

Arrows from bar

`maps to`	yields $\longmapsto$
`mapsto`	yields $\longmapsto$
`mapsfrom`	yields $\longmapsfrom$
`Mapsto`	yields $\Longmapsto$
`Mapsfrom`	yields $\Longmapsfrom$

Arrows with hook

`hook`	yields $\hookrightarrow$
`hook'`	yields $\hookleftarrow$
`hookrightarrow`	yields $\hookrightarrow$
`hookleftarrow`	yields $\hookleftarrow$

Arrows with tail

`tail`	yields $\rightarrowtail$
`rightarrowtail`	yields $\rightarrowtail$
`leftarrowtail`	yields $\leftarrowtail$

Two-headed arrows

`two heads`	yields $\twoheadrightarrow$
`twoheadrightarrow`	yields $\twoheadrightarrow$
`twoheadleftarrow`	yields $\twoheadleftarrow$

Harpoons

`harpoon`	yields $\rightharpoonup$
`harpoon'`	yields $\rightharpoondown$
`rightharpoonup`	yields $\rightharpoonup$
`rightharpoondown`	yields $\rightharpoondown$
`leftharpoonup`	yields $\leftharpoonup$
`leftharpoondown`	yields $\leftharpoondown$

Dashed arrows

`dashed`	yields $\dashrightarrow$
`dashrightarrow`	yields $\dashrightarrow$
`dashleftarrow`	yields $\dashleftarrow$

Squiggly arrows

`squiggly`	yields $\rightsquigarrow$
`rightsquigarrow`	yields $\rightsquigarrow$
`leftsquigarrow`	yields $\leftsquigarrow$
`leftrightsquigarrow`	yields $\leftrightsquigarrow$

Non-arrows

`no head`	yields $\nrightarrow$
`no tail`	yields $\not\rightarrow$
`dash`	yields $\dashrightarrow$
`equal`	yields $\Rightarrow$

A gray cross ($\times$) in the samples above indicates that the corresponding tip is kept unchanged. This allows several arrow styles to be superimposed.

$A \rightarrowtail \dashrightarrow \twoheadrightarrow B$

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

1.4 Alternative syntax for arrows

The following forms of the arrow command were used before the appearance of the quotes syntax for labels, and now may seem somewhat convoluted. They are nonetheless still available for backwards compatibility.

`\arrow[`⟨*options*⟩`]{`⟨*direction*⟩`}`⟨*labels*⟩

Here, ⟨*direction*⟩ is a string containing the characters `r`, `l`, `d`, `u` and is used to determine the arrow target. Alternatively, you can specify an explicit matrix cell by replacing ⟨*direction*⟩ with something of the form ⟨*row number*⟩`-`⟨*column number*⟩, or the name of a node. The trailing ⟨*labels*⟩ can be the empty string or of the form

`[`⟨*label options*⟩`]{`⟨*label text*⟩`}`⟨*more labels*⟩.

The equivalent command `\ar` can also be used in this form. Here is an example.

$$\begin{array}{ccc} A & \xrightarrow{\phi \quad \psi} & B \\ \downarrow & & \downarrow{\scriptstyle \xi} \\ C & \xrightarrow{\eta} & D \end{array}$$

```
\begin{tikzcd}
  A \arrow{d} \arrow{r}[near start]{\phi}[near end]{\psi}
    & B \arrow[red]{d}{\xi} \\
  C \arrow[red]{r}[blue]{\eta}
    & D
\end{tikzcd}
```

There are further shortened forms:

`\rar[`⟨*options*⟩`]`⟨*labels*⟩
`\lar[`⟨*options*⟩`]`⟨*labels*⟩
`\dar[`⟨*options*⟩`]`⟨*labels*⟩
`\uar[`⟨*options*⟩`]`⟨*labels*⟩
`\drar[`⟨*options*⟩`]`⟨*labels*⟩

`\urar[`⟨*options*⟩`]`⟨*labels*⟩
`\dlar[`⟨*options*⟩`]`⟨*labels*⟩
`\ular[`⟨*options*⟩`]`⟨*labels*⟩

The first one is equivalent to

```
\arrow[⟨options⟩]{r}⟨labels⟩
```

and the other ones work analogously.

1.5 Usage in plain TEX

To use this software in plain TEX, load Ti*k*Z and the `cd` library by saying

```
\input tikz.tex
\usetikzlibrary{cd}
```

The `{tikzcd}` environment should then be replaced by the following:

`\tikzcd[`⟨*options*⟩`]`
⟨*environment contents*⟩
`\endtikzcd`

All other functions of this library work as described in this manual without change.

1.6 Usage in ConTEXt

To use this software in ConTEXt, load Ti*k*Z and then the `cd` library by saying

```
\usemodule[tikz]
\usetikzlibrary[cd]
```

The `{tikzcd}` environment should then be replaced by the following:

`\starttikzcd[`⟨*options*⟩`]`
⟨*environment contents*⟩
`\stoptikzcd`

All other functions of this library work as described in this manual without change.

2 Controlling the appearance of diagrams

This section describes a number of customization keys defined by this package. All keys are located in the path `/tikz/commutative diagrams`. Options passed to `{tikzcd}` or `\arrow` are searched for in that path, and, if not found there, in `/tikz`. To set options globally, it is convenient to use the following command.

`\tikzcdset{`⟨*options*⟩`}`
Executes ⟨*options*⟩ in the path `/tikz/commutative diagrams`.

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 cell`, `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}`.

We also point out that besides the options and styles provided by this package, several keys defined by Ti*k*Z are useful for arrows. Some examples are `dashed`, `dotted`, and its relatives, `line width=`⟨*dimension*⟩, `color=`⟨*color*⟩, `bend right`, `bend left`, `in=`⟨*angle*⟩, `out=`⟨*angle*⟩, `loop`, etc. See the PGF manual [3, §15.3.1 and §70]. Likewise, Ti*k*Z provides several keys that are useful for labels, such as `above`, `below`, `left`, `right`, `swap` (which makes the label be placed on the right side of the arrow, relative to its direction), `sloped`, `pos=`⟨*fraction*⟩, `near start`, `near end`, `inner sep=`⟨*dimension*⟩, `font=`⟨*font command*⟩, `text width=`⟨*dimension*⟩, etc. See the PGF manual [3, §17, esp. §17.8].

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:

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

The baseline=0pt setting is used to make equation numbers be placed correctly (as an exception, one-row diagrams are anchored at their matrix base, which is exactly what you want).

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

This key appends ⟨*options*⟩ to the style every diagram.

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

This style is applied to the Ti*k*Z matrix created internally by {tikzcd}. Initially, it contains the following:

```
/tikz/inner sep=0pt
```

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

This style is applied to every Ti*k*Z matrix cell created by {tikzcd}. Initially, it contains the following:

```
/tikz/shape=asymmetrical rectangle,
/tikz/inner xsep=1ex,
/tikz/inner ysep=0.85ex
```

The asymmetrical rectangle shape is described in §4.1. The inner xsep, inner ysep options determine the spacing between a diagram entry and any arrows reaching it.

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

This key appends ⟨*options*⟩ to the style every cell.

/tikz/commutative diagrams/row sep=⟨*size*⟩ (no default)

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 sizes, and their values, are the following:

tiny	small	scriptsize	normal	large	huge
0.45 em	0.9 em	1.35 em	1.8 em	2.7 em	3.6 em

Notice that setting, say, row sep=1cm globally with \tikzcdset 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

```
\tikzcdset{row sep/normal=1cm}.
```

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

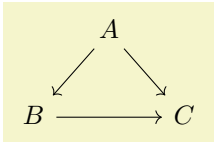
```
\tikzcdset{row sep/my size/.initial=1ex}.
```

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

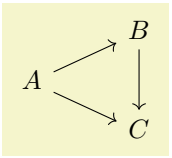
This works analogously to the row sep key above. The sizes available initially are the following:

tiny	small	scriptsize	normal	large	huge
0.6 em	1.2 em	1.8 em	2.4 em	3.6 em	4.8 em

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}
```

Section 20.3.2 of the PGF manual [3] contains further details on the spacing of matrix cells.

`/tikz/commutative diagrams/math mode=`⟨*boolean*⟩ (default `true`)

This key determines whether or not the contents of a diagram are typeset in math mode. 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 Global options for arrows

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

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

```
/tikz/draw,
/tikz/line width=rule_thickness,
rightarrow
```

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

This key appends ⟨*options*⟩ to the style `every arrow`.

`/tikz/commutative diagrams/arrow style=`⟨*style*⟩ (no default)

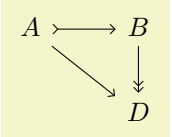
This key determines which collection of arrow tips is used by the arrow tip selection styles listed in §1.3. The initial setting is suitable for documents using the Computer Modern font at any size. The available choices for ⟨*style*⟩ are:

`Latin Modern` A small variant of the initial settings, intended for documents using the Latin Modern font at any size.

`math font` This setting uses the `Glyph` meta arrow tip described in §4.4.

`tikz` This setting uses the arrow tips defined in Ti*k*Z's `arrows.meta` library. It honors the option `/tikz/>`.

If you are using a font different from Computer Modern or Latin Modern, you may find the best results by selecting the `math font` style. As detailed in §4.4, this is not guaranteed to work perfectly with all fonts, but gives good results in many cases. If the `math font` style gives unsatisfactory results, you can try selecting the `tikz` style, and setting `/tikz/>` to the value that best matches your font (among those shown in [3, §16.5]).




```
\tikzcdset{
  arrow style=tikz,
  diagrams={>={Straight Barb[scale=0.8]}}
}

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

2.3 Absolute placement of arrows

The usual behavior of `\arrow` is to produce an arrow starting at the diagram entry where the command appears, and ending at an entry whose location is specified relative to that. The following keys override this behavior, allowing source and target to be selected explicitly.

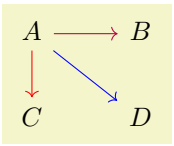
`/tikz/commutative diagrams/from`=⟨*argument*⟩ (no default)

If ⟨*argument*⟩ is of the form ⟨*row number*⟩-⟨*column number*⟩, or if it is a string of characters `r`, `l`, `d`, `u`, this key sets the arrow source to be the corresponding cell in the diagram matrix. Otherwise, it assumes the argument is the name of a node and sets the arrow source to ⟨*argument*⟩.

`/tikz/commutative diagrams/to`=⟨*argument*⟩ (no default)

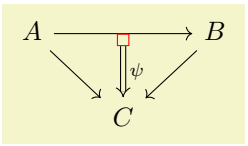
Similar to `from`, but refers to the arrow target.

Recall that it is possible to give a specific entry of a Ti*k*Z matrix a name by using the `|[`⟨*options*⟩`]|` syntax, as done for entry C in the example below. You must be careful not to create nodes whose name contains only the characters `l`, `r`, `u`, `d` if you want to refer to them using `from` or `to`.

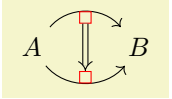



```
\begin{tikzcd}
  A \arrow[to=Z, red] \arrow[to=2-2, blue]
    & B \\
  |[alias=Z]| C
    & D
  \arrow[from=ul, to=1-2, purple]
\end{tikzcd}
```

In the next examples, empty labels are used to create nodes for later reference. The `draw=red` option is used to show where these empty nodes are located, but of course you want to remove that when using this technique.




```
\begin{tikzcd}[column sep=scriptsize]
  A \arrow[dr] \arrow[rr, ""{name=U, below, draw=red}]{}
    & & B \arrow[dl] \\
    & C \arrow[Rightarrow, from=U, "\psi"]
\end{tikzcd}
```




```
\begin{tikzcd}
  A \arrow[r, bend left=50, ""{name=U, below, draw=red}]
    \arrow[r, bend right=50, ""{name=D, draw=red}]
    & B
  \arrow[Rightarrow, from=U, to=D]
\end{tikzcd}
```

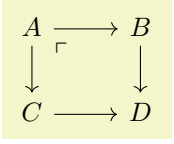
2.4 Phantom arrows

Sometimes it is necessary to insert a symbol outside the grid subjacent to the diagram. The easiest way to achieve this is as a label to an invisible arrow.

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

Creates an invisible arrow. Labels to this arrow are not invisible. They will be anchored at their center and typeset in full size (i.e., with `\textstyle`). To get smaller labels, as in ordinary arrows, use the `\scriptstyle` command.

In the picture below, the arrow containing the `phantom` option goes from A to D, and the `\ulcorner` symbol (⌜) is inserted closer to the starting point A.




```
\begin{tikzcd}
  A \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near start]
    & B \arrow[d] \\
  C \arrow[r]
    & D
\end{tikzcd}
```

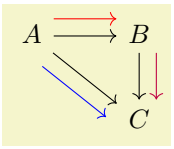
2.5 Fine-tuning the placement of arrows

`/tikz/commutative diagrams/shift left=`⟨*dimension*⟩ (default `0.56ex`)

Shifts arrows by ⟨*dimension*⟩ to the left, relative to the arrow direction. A dimensionless argument causes that multiple of the default value to be used.

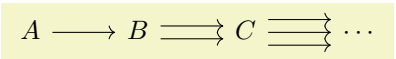
`/tikz/commutative diagrams/shift right=`⟨*dimension*⟩ (default `1`)

A shortcut to `shift left=-`⟨*dimension*⟩.



```
\begin{tikzcd}
  A \arrow[r, red, shift left=1.5ex] \arrow[r]
    \arrow[dr, blue, shift right=1.5ex] \arrow[dr]
    & B \arrow[d, purple, shift left=1.5ex] \arrow[d]\\
    & C
\end{tikzcd}
```

The default values of `shift left` and `shift right` are appropriate for a pair of parallel arrows, and dimensionless arguments are useful to create sets of multiple parallel arrows.



```
\begin{tikzcd}
  A \arrow[r]
    & B \arrow[r, shift left]
        \arrow[r, shift right]
      & C \arrow[r]
          \arrow[r, shift left=2]
          \arrow[r, shift right=2]
        & \cdots
\end{tikzcd}
```

`/tikz/commutative diagrams/shift=`{⟨*coordinate*⟩} (no default)

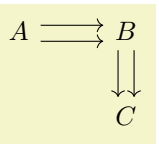
Shifts arrows by ⟨*coordinate*⟩.

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

Shifts arrows right by ⟨*dimension*⟩.

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

Shifts arrows up by ⟨*dimension*⟩.



```
\begin{tikzcd}
  A \arrow[r, yshift=0.7ex] \arrow[r, yshift=-0.7ex]
    & B \arrow[d, xshift=0.7ex] \arrow[d, xshift=-0.7ex] \\
    & C
\end{tikzcd}
```

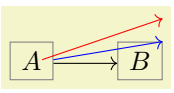
`/tikz/commutative diagrams/start anchor=`{[⟨*coordinate transformations*⟩]⟨*anchor*⟩} (no default)

This key specifies at which anchor of the source node the arrow should start. Optionally, additional coordinate transformations can be supplied. An empty ⟨*anchor*⟩ argument causes no anchor to be specified, which is the usual behavior.

`/tikz/commutative diagrams/end anchor=`{[⟨*coordinate transformations*⟩]⟨*anchor*⟩} (no default)

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

See the picture on §4.1 for some of the possible values for ⟨*anchor*⟩.

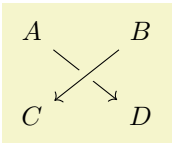


```
\begin{tikzcd}[cells={nodes={draw=gray}}]
  A \arrow[r, black]
    \arrow[r, blue, end anchor=north east]
    \arrow[r,
           red,
           start anchor={[xshift=-1ex]},
           end anchor={[yshift=2ex]north east}]
    & B
\end{tikzcd}
```

2.6 Three-dimensional diagrams

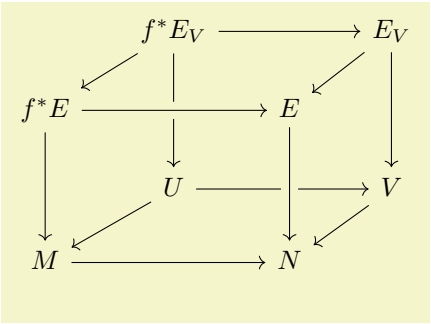
/tikz/commutative diagrams/crossing over (style, no value)

This style makes a thicker line, with color 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[dl, crossing over] \\
  C            & D
\end{tikzcd}
```

Note that, since arrows are drawn in the order they are read, it may be necessary to defer the drawing of certain arrows to achieve the desired result. This can be done using the from key, as shown in the following picture.




```
\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[rr, crossing over] \arrow[dd] & & E \\
  & U \arrow[dl] \arrow[rr] &  & V \arrow[dl] \\
  M \arrow[rr] & & N \arrow[from=uu, crossing over]\\
\end{tikzcd}
```

/tikz/commutative diagrams/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.

2.7 Options for labels

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

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

```
/tikz/auto,
/tikz/font=⟨something⟩,
/tikz/inner sep=0.5ex
```

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

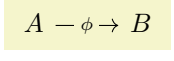
The key /tikz/auto makes the label be placed on the left side of the arrow, relative to its direction. The key /tikz/inner sep controls the distance between a label and the corresponding arrow.

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

This key appends ⟨*options*⟩ to 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.




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

3 Advanced usage

This section provides further details on the functioning of this package, with the aim of allowing the advanced user to make a more or less arbitrary use of other Ti*k*Z features within `{tikzcd}`.

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} [⟨options⟩]
    \matrix [matrix of nodes] {
        ⟨contents⟩ \\
    };
    ⟨paths⟩
\end{tikzpicture}
```

Not shown above are a number of initialization procedures, such as defining `\arrow` and its relatives, as well as applying the default settings specified by `every diagram` and its relatives. Note that the next-row command `\\` for the last row is inserted by `{tikzcd}`, and therefore should not be present in ⟨*contents*⟩. 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.

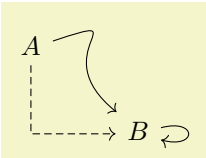
Initially, ⟨*paths*⟩ is the empty string. A command `\arrow[`⟨*options*⟩`]` does nothing at the point it is inserted, and causes the following code to be appended to ⟨*paths*⟩:

```
\path[⟨options⟩] (⟨current node⟩) to (⟨target node⟩);
```

Here, ⟨*current node*⟩ is the node corresponding to the matrix cell where the command `\arrow` is present. A special `.unknown` key handler is set up to parse direction arguments in ⟨*options*⟩ and set ⟨*target node*⟩ accordingly.

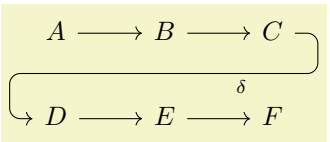
3.2 Tweaking `to` paths

Recall that the `to` path operation used in the paths created by `\arrow` can take a number of options, as described in §70 of the PGF manual [3]. 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[dr, controls={+(1.5,0.5) and +(-1,0.8)}]
    \arrow[dr, dashed, to path=|- (\tikztotarget)]
    & \\
    & B \arrow[loop right]
\end{tikzcd}
```

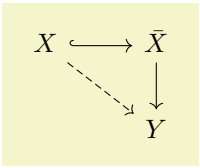
The following example shows how to produce a "snake" map. The arrow with the `phantom` option (going from B to E) has the sole purpose of creating a coordinate, named `Z`, lying halfway between these two cells. The arrow starting at C has target D, so the macros `\tikztostart` and `\tikztotarget` will expand to the nodes corresponding to these two cells in the argument of `to path`. Notice also the use of `\tikztonodes` at the point where we want the label to be inserted.



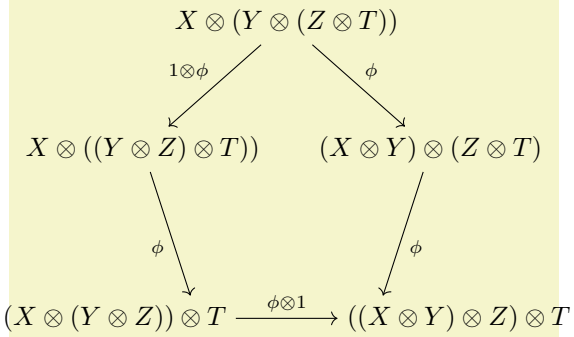
```
\begin{tikzcd}
  A \arrow[r]
    & B \arrow[r]
        \arrow[d, phantom, ""{coordinate, name=Z}]
      & C \arrow[dll,
                "\delta",
                rounded corners,
                to path={ -- ([xshift=2ex]\tikztostart.east)
                          |- (Z) [near end]\tikztonodes
                          -| ([xshift=-2ex]\tikztotarget.west)
                          -- (\tikztotarget)}] \\
   D \arrow[r]
    & E \arrow[r]
      & F
\end{tikzcd}
```

3.3 Drawing diagrams directly with Ti*k*Z

If you find that this package is not flexible enough for some particular application, you can use the methods described in [1], [2] and draw diagrams directly with Ti*k*Z. In this case, you can still use the styles provided here to obtain pictures with a uniform appearance throughout your document. The pictures below show how this can be done (the second one is adapted from [2]).




```
\begin{tikzpicture}[commutative diagrams/every diagram]
  \matrix[matrix of math nodes, name=m, commutative diagrams/every cell] {
    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 (90:2.3cm) {$X\otimes (Y\otimes (Z\otimes T))$};
  \node (P1) at (90+72:2cm) {$X\otimes ((Y\otimes Z)\otimes T))$} ;
  \node (P2) at (90+2*72:2cm) {\makebox[5ex][r]{$(X\otimes (Y\otimes Z))\otimes T$}};
  \node (P3) at (90+3*72:2cm) {\makebox[5ex][l]{$((X\otimes Y)\otimes Z)\otimes T$}};
  \node (P4) at (90+4*72:2cm) {$(X\otimes Y)\otimes (Z\otimes T)$};

  \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}
```

3.4 Issues with active characters

By default, Ti*k*Z makes the character `&` active inside matrices, and this causes the error message

```
! Package pgfbasematrix Error: Single ampersand used with wrong catcode.
```

when `{tikzcd}` is used inside the argument to a macro such as a Beamer frame or a footnote. One solution to this problem is to call `{tikzcd}` with the option `ampersand replacement=\&`, and replace all occurrences

of & with \& in the diagram. This procedure is also needed if you want to use matrices in a diagram cell or label.

$$A \oplus B \xrightarrow{\begin{pmatrix} e & f \\ g & h \end{pmatrix}} C \oplus D$$

```
\begin{tikzcd}[ampersand replacement=\&]
  A \oplus B \ar[r, "{\begin{pmatrix}  e & f \\ g & h \end{pmatrix}}"]
    \& C \oplus D
\end{tikzcd}
```

An alternative fix to this issue that does not require replacing & with a different column separator consists in adding the following line to your document after all packages have been loaded:

```
\def\temp{&} \catcode`&=\active \let&=\temp
```

However, this may interfere in unexpected ways with other packages. Use this trick at your own risk.

A different but related issue is that some packages, notably babel, modify the catcodes of certain characters in a way that may upset Ti*k*Z's parser. To fix this, add

```
\usetikzlibrary{babel}
```

to your document preamble. Characters inside {tikzcd} cells will be assigned their standard (non-babel) catcode.

4 Additional goodies

This package provides some general PGF infrastructure to achieve its goals. These additional goodies are documented in this section.

4.1 The asymmetrical rectangle shape

The following shape is used inside {tikzcd} to ensure that arrows between nodes in the same row are perfectly horizontal, even if the nodes contain text with different heights and depths.

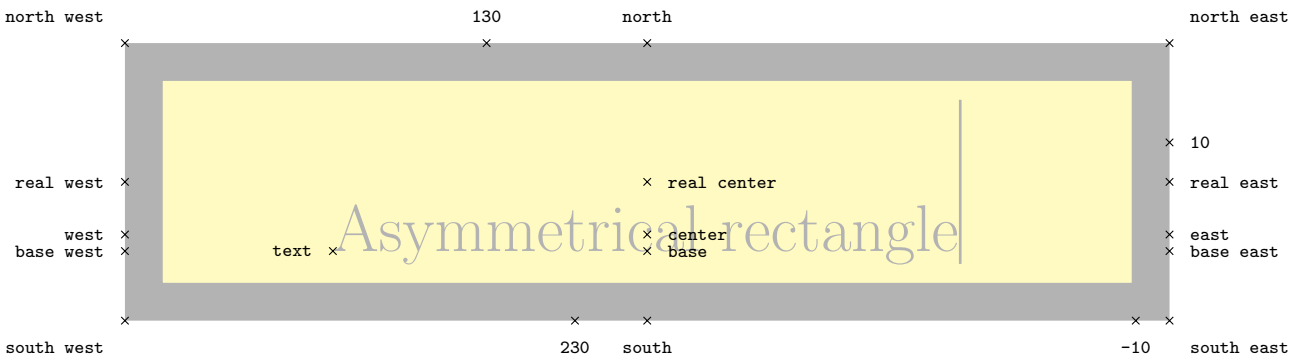
Shape asymmetrical rectangle

This shape is similar to the rectangle shape, but its center is located at a fixed distance of the base, as determined by the center yshift key, rather than lying at the shape's geometric center. The numerical anchors, as well as east and west, are modified accordingly, and there are anchors called real center, real east, and real west matching rectangle's original definitions. All other anchors provided by rectangle are available and remain unmodified.

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

Determines the distance between asymmetrical rectangle's base and center anchors.

The picture below shows some of the available anchors.



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.

`axis_height`

Returns the axis height parameter (a.k.a. σ_{22}) of the document's math font.

`rule_thickness`

Returns the fraction rule thickness (a.k.a. ξ_8) of the document's math font.

4.3 Computer Modern arrow tips

The following arrow tips mimic the Computer Modern designs. It is useful to know that at size 10 pt, the Computer Modern arrow stems are 0.4 pt thick; for other font sizes, scale this parameter accordingly, or set `line width=rule_thickness`.

Notice that by using the mechanism explained in §1.3, it is not necessary, and in fact not advisable, to directly refer to the arrow tips listed in this section inside a `{tikzcd}`.

`cm to`	yields ⟷	`cm bar`	yields ⊢⊣
`cm implies`	yields ⟸⟹	`cm left to`	yields ↚
`cm bold to`	yields ⟷	`cm right to`	yields ↛
`cm double to`	yields ⇷⇸	`cm left hook`	yields ↩
`cm to reversed`	yields ⟵	`cm right hook`	yields ↪

4.4 Glyph arrow tips

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) font glyphs carefully placed at the endpoints of the path. To activate it in `{tikzcd}` diagrams, refer to the `arrow style` key.

Arrow Tip Kind `Glyph`

An arrow tip made from a piece of text. It accepts the following parameters.

`/pgf/arrow keys/glyph math command=`⟨*name*⟩ (no default)

The name of a command (to be used inside `$\csname ... \endcsname$`) producing the desired glyph.

`/pgf/arrow keys/glyph length=`⟨*dimension*⟩ (no default, initially `1ex`)

The length of the portion of the glyph not clipped away. Also used to set the 'tip end' parameter.

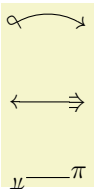
`/pgf/arrow keys/glyph axis=`⟨*dimension*⟩ (no default, initially `axis_height`)

A vertical displacement applied to the glyph in order to make the glyph's central axis (typically an arrow stem) aligned with the path.

`/pgf/arrow keys/glyph shorten=`⟨*dimension*⟩ (no default, initially `-0.1ex`)

An additional amount by which the end of the path is shortened. This is used to compensate for the gap that usually exists between the tip end and the glyph's bounding box.

Below are some usage examples. Notice that glyph arrow tips do not scale with PGF line width but their size depends on the current font size, so you will probably want to set `line width=rule_thickness` when using them. Also, contrarily to the arrow parameters defined by the `arrows.meta` library, the parameters described above are evaluated only at the time the arrow tip is drawn, so they can (and should) be given in the units em or ex.



```
\tikzset{
  math to/.tip={Glyph[glyph math command=rightarrow]},
  loop/.tip={Glyph[glyph math command=looparrowleft, swap]},
  weird/.tip={Glyph[glyph math command=Rrightarrow, glyph length=1.5ex]},
  pi/.tip={Glyph[glyph math command=pi, glyph length=1.5ex, glyph axis=0pt]},
}

\begin{tikzpicture}[line width=rule_thickness]
  \draw[loop-math to, bend left] (0,2) to (1,2);
  \draw[math to-weird] (0,1) to (1,1);
  \draw[pi-pi] (0,0) to (1,0);
\end{tikzpicture}
```

It is important to be aware of some drawbacks of this feature. First, the transition between a line and the arrow tip may become visible with some printers (especially in low resolutions or draft mode) and document viewers, as you may be able to see in the samples above. Second, these rather long tips may (--→) or may not (---→) fit nicely with dashed or curved lines. Finally, the method used to place the arrow tip at the end of a stroked path and clip away the arrow stem makes certain assumptions about the font design and could fail in cases where unusual design choices are made.

References

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

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

[3] Till Tantau, *The TikZ and* PGF *packages: Manual for version 3.0.0*. Available at http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf.

Index

This index only contains automatically generated entries. A good index should also contain carefully selected keywords. This index is not a good index.