

The `tikz-cd` package

Florêncio Neves*

Version 0.2a, of December 9, 2011

The general-purpose drawing package `TikZ` 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 `TikZ` 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

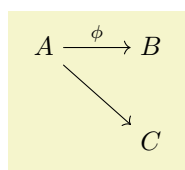
1	Basic usage	1
1.1	Inserting arrows	1
1.2	Changing arrow tips	2
2	Controlling the appearance of diagrams	3
2.1	General options	4
2.2	Options for arrows	5
2.3	Options for labels	6
3	Further possibilities	7
3.1	Internals of <code>tikzcd</code> and the arrow commands	7
3.2	Tweaking to paths	8
3.3	Drawing diagrams directly with <code>TikZ</code>	8
4	Computer Modern arrow tips	9
5	Change history	10

1 Basic usage

Commutative diagrams are created with the `tikzcd` environment. Its content describes a matrix, like the `\matrix` command in `TikZ` 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

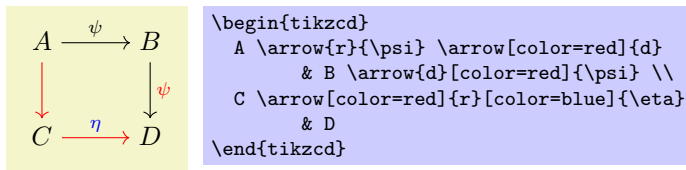
Inside the `tikzcd` environment, the command `\arrow` is provided to produce arrows. In its simplest form, it takes one argument, a string containing the characters `r`, `l`, `u` or `d`, standing for right, left, up and down, that determine the arrow target. A label can be placed on an arrow by providing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



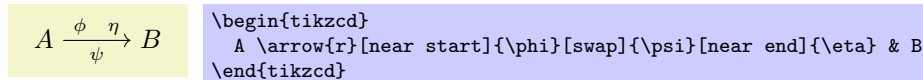
```
\begin{tikzcd}
A \arrow{rd} \arrow{r}{\phi} & B \\
& C
\end{tikzcd}
```

*E-mail: florencioneves@gmail.com

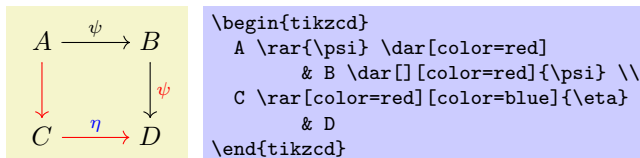
You can control the behavior of the arrow by placing an argument inside square braces before the direction parameter. It may contain any option that can be passed to TikZ's `\path` command. Similarly, a label can be modified by an argument in square braces right before it. It may contain anything that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



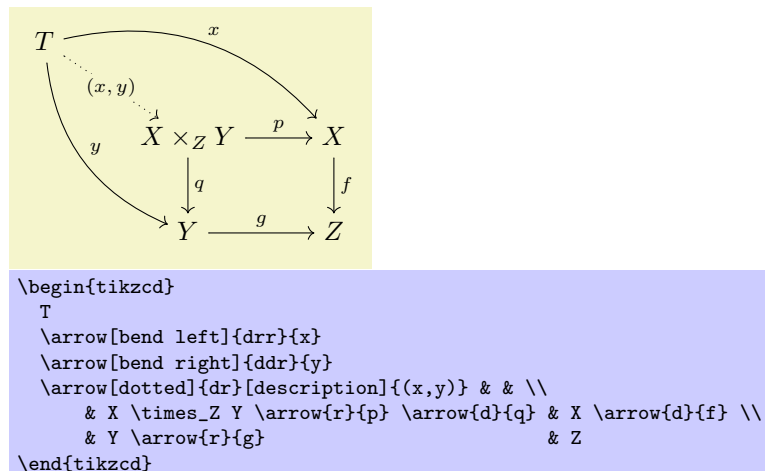
An arrow can actually have an arbitrary number of labels, each one specified by juxtaposing $\{\langle text \rangle\}$ or $[\langle options \rangle]\{\langle text \rangle\}$ to the `\arrow` command.



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 $\{\langle text \rangle\}$ or $[\langle options \rangle]\{\langle text \rangle\}$. Thus, the diagram above can be rewritten as follows.

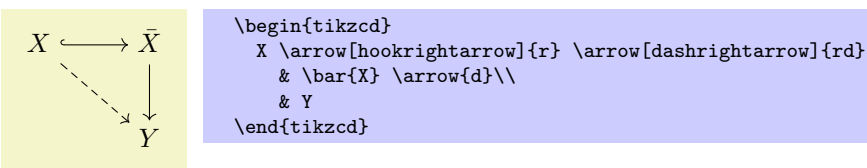


We provide one more example, reproduced from XY-pic user's guide.



1.2 Changing arrow tips

For each arrow-producing command present in $\mathcal{A}\mathcal{M}\mathcal{S}$ - \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.



Of course, a similar effect could be achieved by directly using TikZ's arrow tip selection scheme (e.g., using the option `right hook->`). However, the method more 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 arrow styles available.

<code>rightarrow</code>	yelds \longrightarrow
<code>Rightarrow</code>	yelds \Longrightarrow
<code>leftarrow</code>	yelds \longleftarrow
<code>Leftarrow</code>	yelds \Longleftarrow
<code>leftrightharrow</code>	yelds \longleftrightarrow
<code>Leftrightharrow</code>	yelds \Longleftrightarrow
<code>equals</code>	yelds \equiv
<code>mapsto</code> (or <code>maps to</code>)	yelds \longmapsto
<code>hookrightarrow</code> (or <code>hook</code>)	yelds \hookrightarrow
<code>hookleftarrow</code>	yelds \hookleftarrow
<code>rightharpoonup</code>	yelds \rightharpoonup
<code>rightharpoondown</code>	yelds \rightharpoondown
<code>leftharpoonup</code>	yelds \leftharpoonup
<code>leftharpoondown</code>	yelds \leftharpoondown
<code>dashrightarrow</code> (or <code>dashed</code>)	yelds \dashrightarrow
<code>dashleftarrow</code>	yelds \dashleftarrow
<code>rightarrowtail</code> (or <code>tail</code>)	yelds \rightarrowtail
<code>leftarrowtail</code>	yelds \leftarrowtail
<code>twoheadrightarrow</code> (or <code>two heads</code>)	yelds \twoheadrightarrow
<code>twoheadleftarrow</code>	yelds \twoheadleftarrow
<code>rightsquigarrow</code> (or <code>squiggly</code>)	yelds \rightsquigarrow
<code>leftsquigarrow</code>	yelds \leftsquigarrow
<code>leftrightsquigarrow</code>	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.

$A \dashrightarrowtail B$

```
\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 next.
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 *TikZ* 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`, which is what you would expect. 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/text height=1.5ex,
/tikz/text depth=0.25ex,
/tikz/baseline=0pt,
row sep=normal,
column sep=normal
```

The options `text height` and `text depth` are set to ensure arrows between nodes in the same row are drawn horizontally, as discussed in [2]. The `baseline=0pt` setting is used to make equation numbers be placed correctly.

`/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 TikZ’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 values of `<size>` initially available, and their values, are the following:

<code>tiny</code>	<code>small</code>	<code>scriptsize</code>	<code>normal</code>	<code>large</code>	<code>huge</code>
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

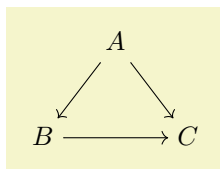
```
\tikzset{commutative diagrams/row sep/my size/.initial=1cm}.
```

`/tikz/commutative diagrams/column sep=<size>` (no default, initially `normal`)

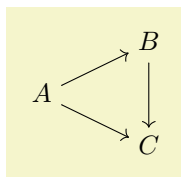
This works analogously to the `row sep` key above. The sizes available initially are the following:

<code>tiny</code>	<code>small</code>	<code>scriptsize</code>	<code>normal</code>	<code>large</code>	<code>huge</code>
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 & \\
B \arrow{rr} & & C \\
\end{tikzcd}
```



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

`/tikz/commutative diagrams/math mode=<boolean>` (no default, initially `true`)

This key 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 to be painted.

2.2 Options for arrows

Besides the options and styles provided by this package, there are several options defined by TikZ 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 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
```

Initially, the style `default arrow` is a synonym to `rightarrow`.

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

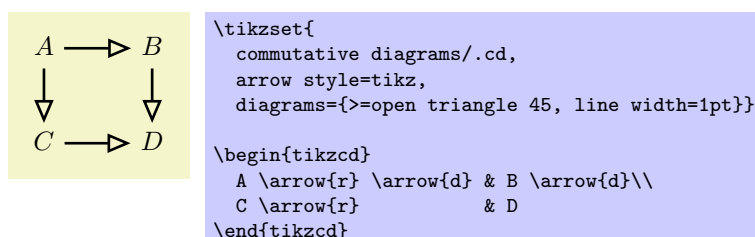
This key appends `⟨options⟩` to the style `/tikz/commutative diagrams/every arrow`.

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

Setting this key causes the several arrow-tip-specifying styles listed on §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 TikZ'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.



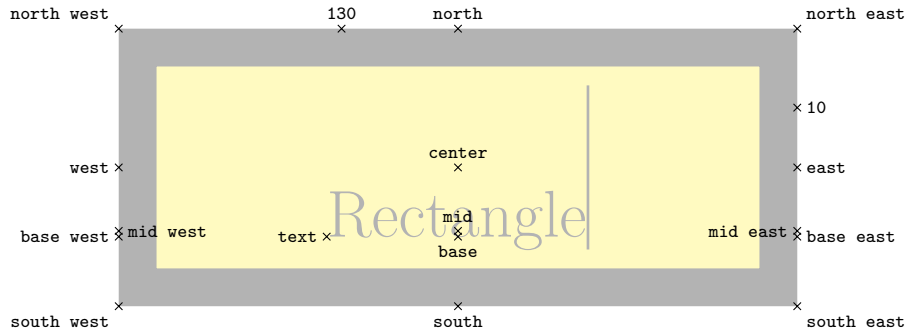
`/tikz/commutative diagrams/start anchor=⟨anchor⟩` (default empty)

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 the destination, as explained in PGF manual [4].

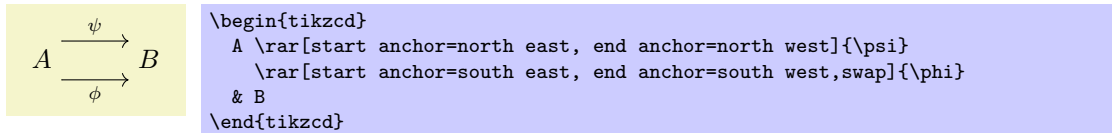
`/tikz/commutative diagrams/end anchor=⟨anchor⟩` (default empty)

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

Some of the possible values for `⟨anchor⟩` are shown below.



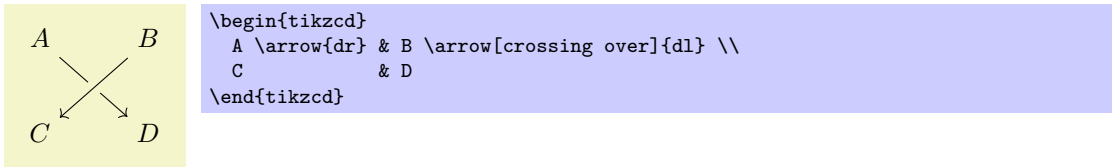
The picture below illustrates the use of `end anchor` and `start anchor`.



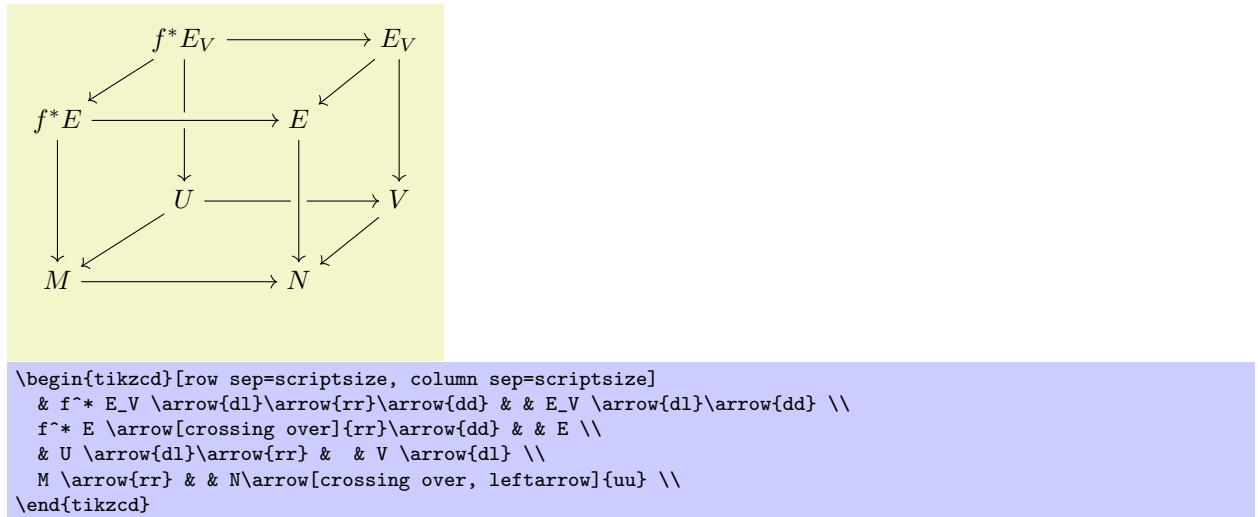
`/tikz/commutative diagrams/crossing over`

(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.



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, from [2], illustrates this.



`/tikz/commutative diagram/crossing over clearance= $\langle dimension \rangle$`

(no default, initially 6pt)

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 `TikZ` 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= $\langle fraction \rangle$` , `near start`, `near end`, `inner sep= $\langle dimension \rangle$` , `font= $\langle font command \rangle$` , `text width= $\langle dimension \rangle$` , etc. See PGF manual [4].

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

This style works as if it were applied to the optional argument corresponding to each label attached to an arrow. Initially is set to the following:

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

The style `/tikz/auto` makes the label be placed to the left of the arrow, understanding “front” as the direction the arrow points at. The dimension `/tikz/inner sep` controls the distance between the arrow and the label. The options `text height` and `text depth` are reset here because they are set picture-wise by the `every diagram` style.

`/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.

$A - \phi \rightarrow B$

```
\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 possibilities

This sections provides further details on the functioning of this package, with the aim of showing how a more or less arbitrary use of TikZ 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`.

The command `\arrow[<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, <options>] (<current node>) to <labels> (<target node>);
```

Here, $\langle current\ node \rangle$ is the node corresponding to the matrix cell where the command `\arrow` is, and $\langle target\ node \rangle$ is determined by $\langle direction \rangle$ as explained in §1.1. The abbreviated commands `\rar`, `\dar`, etc., generate the same kind of code.

Initially, $\langle labels \rangle$ is the empty string. A label specifier of the form $\{\langle label\ text \rangle\}$ or $[\langle label\ options \rangle]\{\langle label\ text \rangle\}$ immediately following the $\langle direction \rangle$ argument or a previous label specifier causes the string

`node [commutative diagrams/every label, $\langle label\ options \rangle$] $\{[\$]\langle label\ text \rangle[\$]\}$`

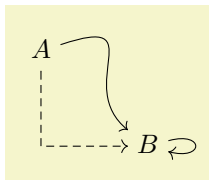
to be appended to $\langle labels \rangle$. Dollars signs surround $\langle label\ text \rangle$ depending on the setting `commutative diagrams/math` mode. For the abbreviated form `\rar` (and similarly for `\dar`, `\lar`, etc.), the following are valid usages:

`\rar{\langle label\ text \rangle}\langle more\ labels \rangle`,
`\rar[\langle options \rangle]\{\langle label\ text \rangle\}\langle more\ labels \rangle`, or
`\rar[\langle options \rangle][\langle label\ options \rangle]\{\langle label\ text \rangle\}\langle more\ labels \rangle`.

Their effects are entirely analogous.

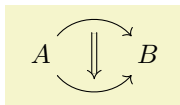
3.2 Tweaking to paths

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 tweaking.



```
\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}
```

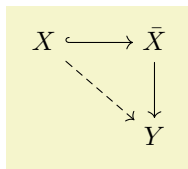
In the next example, empty labels are used to create nodes for later reference, and in the third `\arrow` command the `to path` key is used in a way that, in the terminology of the previous section, $\langle current\ node \rangle$ and $\langle target\ node \rangle$ are completely ignored.



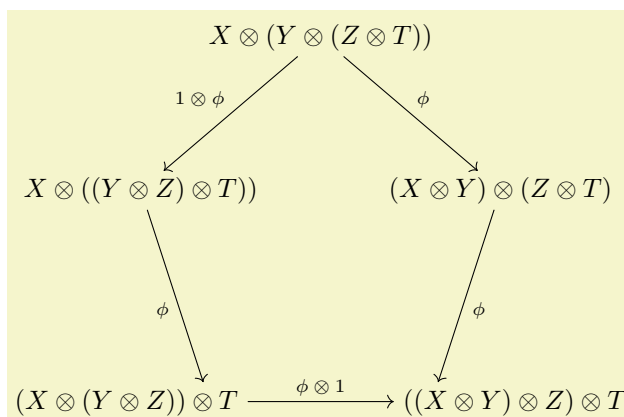
```
\begin{tikzcd}
A \arrow[bend left=50]{r}[name=U,below]{}
\arrow[bend right=50]{r}[name=D]{} & \\
B \arrow[Rightarrow, to path=(U) -- (D)]{}
\end{tikzcd}
```

3.3 Drawing diagrams directly with TikZ

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 TikZ. 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 & \bar{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 PGF manual [4]. To match the Computer Modern typeface at size 10pt, line width should be set to 0.4pt; for larger font sizes, scale this parameter accordingly, or use the esoteric figure 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, TikZ's original `to` arrow tip seems to be based on the pre-1992 version of Computer Modern which, in spite of the author's wish [1], can still be found in many systems. T_EXLive, for instance, distributed the old version up until 2007 or 2008. Therefore, an up-to-date T_EX distribution may be necessary to get matching arrows in formulas and diagrams.

Basic arrow tips

<code>cm to</code>	yields \longleftrightarrow
<code>cm to reversed</code>	yields \rightrightarrows
<code>cm bold to</code>	yields \longleftrightarrow (with a line 50% thicker)
<code>cm </code>	yields \vdash
<code>cm o</code>	yields \circ
<code>cm *</code>	yields \bullet
<code>cm cap</code>	yields \cap

Arrow tips for double lines

<code>cm implies</code>	yields \Longrightarrow
<code>cm implies cap</code>	yields \Longrightarrow





Hooks

<code>cm left hook</code>	yields \hookleftarrow
<code>cm right hook</code>	yields \hookrightarrow

Double arrow tips

<code>cm double to</code>	yields \longleftrightarrow
<code>cm double to reversed</code>	yields \rightrightarrows

Partial arrow tips

<code>cm left to</code>	yelds 
<code>cm left to reversed</code>	yelds 
<code>cm right to</code>	yelds 
<code>cm right to reversed</code>	yelds 

5 Change history

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 T_EX*. 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>.