

# Guide to commutative diagram packages

J.S. Milne

October 19, 2009; October 25, 2009 (minor fixes)\*

Mathematicians have been using diagrams of objects and arrows to explain their work since at least 1945. Conventially, these are called commutative diagrams (even when they don't commute<sup>1</sup>). When publishers first began using  $\text{\TeX}$ , commutative diagrams caused them problems — I remember being asked by one publisher to turn a commutative triangle into a square by the addition of an equals sign. Fortunately, there are now several very capable packages for producing commutative diagrams. Despite their wide use by mathematicians and others, these packages are barely mentioned in the usual books on  $\text{\TeX}$ . In this guide, I show by means of examples what each package can do, and I provide enough information for you to begin using them.

In this version of the guide, I describe the packages

`array`, `amscd`, `diagrams`, `xymatrix`, `diagxy`, `pgf/tikz`.

and I have dropped `kuvio`, `DCpic`.<sup>2</sup>

Except where noted, each package is included in the standard large MikTeX installation and can be used without restriction.

To avoid conflicts between the packages, I produced this document (using `pdflatex`) in segments which I joined using Acrobat.

Corrections, comments, and tips that can be used to improve future versions of this guide are welcome, and can be sent to me at `tex@jmilne.org`. I thank Joel Friedman, Richard Lewis, José Carlos Santos, and Bob Tennent for their comments on earlier versions.

## Summary

It is possible to produce commutative diagrams as arrays, but they are ugly, so this method should only be used for drafts.

The package `amscd` is so easy to use that it should be the first choice for the simple diagrams it can handle.

The package `diagrams` uses a syntax similar to that of `array`, which makes it easy to use, and it is the best at making automatic adjustments. However, it doesn't support curved arrows, arrow heads don't match those in inline mathematics, and it has a quirky licence.

For complicated diagrams `Xy-pic/xymatrix` is probably the most popular package and it is the one recommended by the American Mathematical Society. However, its syntax

---

\*Available at <http://www.jmilne.org/not/CDGuide.pdf>.

<sup>1</sup>A diagram is commutative if one gets the same map when going from one point to another by different paths.

<sup>2</sup>The old version is available as `.../CDGuide06.pdf`.

is clumsy (e.g., `\ar@{^}{()->}` to produce a hook arrow), and it can take a good deal of fiddling to get its diagrams to come out correctly.

The package `diagxy` is a front end to `Xy-pic` that allows for a more precise control over positioning and spacing.

Finally `pgf/tikz` is a general purpose graphics package that can probably produce any diagram you need with whatever precision you desire, but the code can be complex. Probably it will be mainly of interest to those who already use it for other purposes.

## Using array

It is possible to produce simple diagrams very easily as arrays, but they are ugly:

$  \begin{array}{ccc}  A & \xrightarrow{a} & B \\  \downarrow b & & \downarrow c \\  C & \xrightarrow{d} & D  \end{array}  $	<pre> <math>\begin{array}[c]{ccc} A&amp;\stackrel{a}{\rightarrow}&amp;B\\ \downarrow\scriptstyle{b}&amp;&amp;\downarrow\scriptstyle{c}\\ C&amp;\stackrel{d}{\rightarrow}&amp;D \end{array}</math> </pre>
$  \begin{array}{ccccc}  A & \rightarrow & B & \leftarrow & C \\  \searrow & & \downarrow & & \swarrow \\  & & D & &   \end{array}  $	<pre> <math>\begin{array}[c]{ccccc} A&amp;\rightarrow&amp;B&amp;\leftarrow&amp;C\\ \searrow&amp;&amp;\downarrow&amp;&amp;\swarrow\\ &amp;&amp;D&amp;&amp; \end{array}</math> </pre>
$  \begin{array}{ccccc}  A \times A' & \rightarrow & R \\  \cup & & \cup & & \cup \\  B \times B' & \rightarrow & S.  \end{array}  $ <p>(rotatebox requires graphicx.)</p>	<pre> <math>\renewcommand{\arraystretch}{1.3}</math> <math>\begin{array}[c]{ccccc} A&amp;\times&amp;A^{\prime}&amp;\rightarrow&amp;R\\ \cup&amp;&amp;\cup&amp;&amp;\cup\\ B&amp;\times&amp;B^{\prime}&amp;\rightarrow&amp;S. \end{array}</math> </pre>

It is possible to improve the quality a little by using various tricks,

$$\begin{array}{ccccc}
 A & \xrightarrow{\text{label}} & B & \xleftarrow{\text{label}} & C \\
 & \searrow & \downarrow \alpha & \swarrow & \\
 & & D & & 
 \end{array}$$

```

 $\renewcommand{\arraystretch}{1.5}$ 
 $\begin{array}[c]{ccccc}
A&\xrightarrow{\text{label}}&B&\xleftarrow{\text{label}}&C\\
&\searrow&\downarrow\scriptstyle{\alpha}&\swarrow&\\
&&D&&
\end{array}$ 

```

but there is not much point since there are better alternatives.

## The amscd package

The American Mathematical Society's package `amscd` can produce only rectangular diagrams (no diagonal arrows) and supports only plain labelled arrows and equal signs, but its arrows do stretch to match labels and it is easy to use. Load it with the command `\usepackage{amsmath,amscd}`

Its syntax is illustrated by the following example:

$$\begin{array}{ccc}
 A & \xrightarrow{a} & B \\
 \downarrow b & & \downarrow c \\
 C & \xrightarrow{d} & D
 \end{array}$$

```

\begin{CD}
A @>a>> B \\
@VVbV @VVcV \\
C @>d>> D
\end{CD}

```

Rows with horizontal arrows must alternate with those with vertical arrows, and each row except the last must end with `\\`.

The possible arrows (or their replacements) are:

`@<<<` left arrow      `@>>>` right arrow  
`@AAA` up arrow      `@=` horizontal equals  
`@VVV` down arrow    `@|` vertical equals  
`@.` empty arrow.

Their use is illustrated by:

$$\begin{array}{ccccc}
 A & \longleftarrow & B & \longrightarrow & C \\
 & & \parallel & & \uparrow \\
 & & D & \longlongequal & E
 \end{array}$$

```

$\begin{CD}
A @<<< B @>>> C \\
@. @| @AAA \\
@. D @= E
\end{CD}$

```

Items inserted into the code for the arrows will appear in scriptstyle (the size of sub/superscripts) as labels on the arrows, as illustrated by:

$$\begin{array}{ccc}
 A & \xrightarrow{a} & B \\
 \downarrow r & & \uparrow r \\
 C & \xleftarrow{a} & D
 \end{array}$$

```

$\begin{CD}
A @>a>> B \\
@VrVV @AaA \\
C @<a<< D
\end{CD}$

```

Arrows stretch to match long labels:

$$\begin{array}{ccccc}
 A & \longrightarrow & B & \xrightarrow{\text{very long label}} & C \\
 \downarrow & & \downarrow & & \downarrow \\
 D & \longrightarrow & E & \longrightarrow & F
 \end{array}$$

```

$\begin{CD}
A @>>> B @>\text{very long label}>> C \\
@VVV @VVV @VVV \\
D @>>> E @>>> F
\end{CD}$

```

Notice that the lower arrow doesn't stretch to match the upper arrow. To fix this, add a "phantom" label:

$A \longrightarrow B \xrightarrow{\text{very long label}} C$	<code>\begin{CD}</code>
$\downarrow \qquad \downarrow \qquad \downarrow$	<code>A @&gt;&gt; B @&gt;{\text{very long label}}&gt;&gt; C \\</code>
$D \longrightarrow E \longrightarrow F$	<code>@VVV @VVV @VVV \\</code>
	<code>D @&gt;&gt; E @&gt;{\phantom{\text{very long label}}}&gt;&gt; F \\</code>
	<code>\end{CD}</code>

To get a shorter label centred on the lower arrow, use the following code:

`@>{\rlap{\scriptstyle\ \ \ \text{shorter}}}\phantom{\text{very long label}}>>`

$A \longrightarrow B \xrightarrow{\text{very long label}} C$
$\downarrow \qquad \downarrow \qquad \downarrow$
$D \longrightarrow E \xrightarrow{\text{shorter}} F$

Alternatively, increase the minimum length of a horizontal arrow by replacing the code with `[\minCDarrowwidth55pt\begin{CD}...\end{CD}]`.

$A \longrightarrow B \xrightarrow{\text{very long label}} C$
$\downarrow \qquad \downarrow \qquad \downarrow$
$D \longrightarrow E \xrightarrow{\text{shorter}} F$

This trick also allows you to shorten arrows if necessary to fit a long diagram onto a page.

Here is how `amscd` handles large objects:

$A \times A \times A \times A \times A$	$\xrightarrow{a}$	$B$
$\downarrow b$		$\downarrow c$
$C$	$\xrightarrow{d}$	$D$

## The diagrams package

If this is not already on your computer, you can download it from <http://www.paultaylor.eu/diagrams/>. To install it, simply rename the file `diagrams.sty` and put it somewhere that  $\TeX$  can find it. A manual dated 15 June 1997 is available on the same site.

The program supports a great variety of arrows, which stretch to match their labels, and produces diagrams of high quality. However, unlike `xymatrix` for example, it doesn't curve arrows. It allows a large number of options. For this file, I loaded it using

```
\usepackage[small,nohug,heads=vee]{diagrams}
\diagramstyle[labelstyle=\scriptstyle]
```

The “small” reduces the distances between objects, “nohug” stops the labels rotating with their arrows (mostly), and “heads=vee” determines the size and shape of the heads on the arrows (alternatives `LaTeX`, `littlevee`, `triangle`, ...). The second row makes the labels print in `scriptstyle`.

The syntax is similar to that of `array`, as illustrated by:

$\begin{array}{ccc} A & \xrightarrow{a} & B \\ \downarrow b & & \downarrow c \\ C & \xrightarrow{d} & D \end{array}$	<pre>\begin{diagram} A &amp; \rTo^{a} &amp; B \\ \downarrow b &amp; &amp; \downarrow c \\ C &amp; \rTo^{d} &amp; D \end{diagram}</pre>
--	--

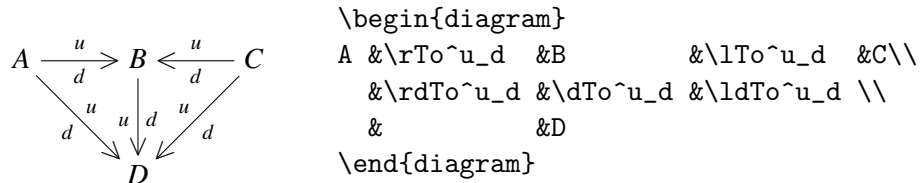
Arrows are specified by a one- or two-letter prefix describing the direction, and a suffix describing the body of the arrow. For example:

$\begin{array}{ccccc} & \text{lu} & & \text{u} & & \text{ru} \\ & \swarrow & & \uparrow & & \searrow \\ & \text{l} & \leftarrow & & \rightarrow & \text{r} \\ & \swarrow & & \downarrow & & \searrow \\ & \text{ld} & & \text{d} & & \text{rd} \end{array}$	<pre>\rightarrow To \mapsto Mapsto \line\ Line \into Into \onto Onto \dotsto Dotsto \dashrightarrow Dashto \implies Implies</pre>
---	---

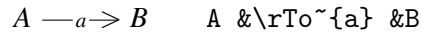
To invoke an arrow, combine the two, as illustrated by:

$\begin{array}{ccccc} A & \xrightarrow{\quad} & B & \xleftarrow{\quad} & C \\ & \searrow & \vdots & \swarrow & \\ & & D & & \end{array}$	<pre>\begin{diagram} A &amp; \rInto &amp; B &amp; &amp; \lInto &amp; C \\ &amp; \rdOnto &amp; \dDotsto &amp; &amp; \ldOnto \\ &amp; &amp; &amp; &amp; D \end{diagram}</pre>
--	---

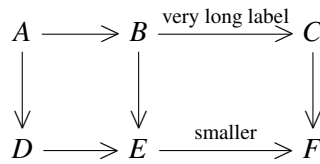
To add labels to arrows, place them as superscripts or subscripts on the arrow (between braces if necessary), as illustrated by:



A superscript places the label above (or to the left) of an arrow. Labels can also be made to break arrows by using `~` instead of `^`

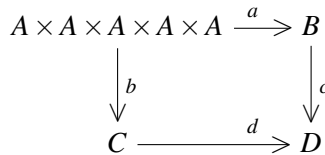


Arrows stretch to match long labels:

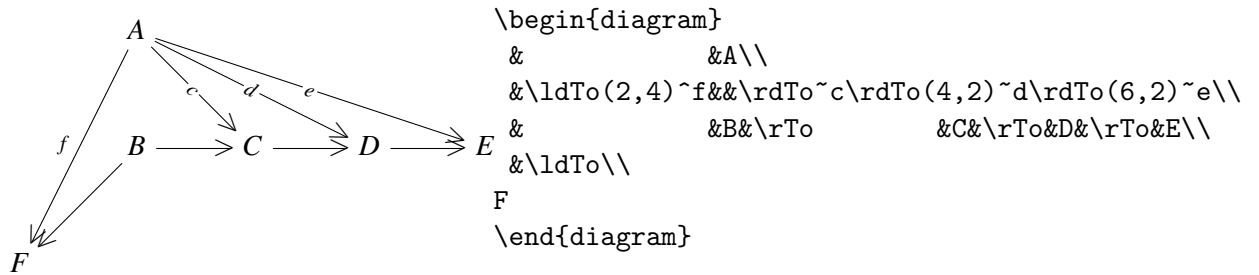


Notice that the lower arrow also stretched.

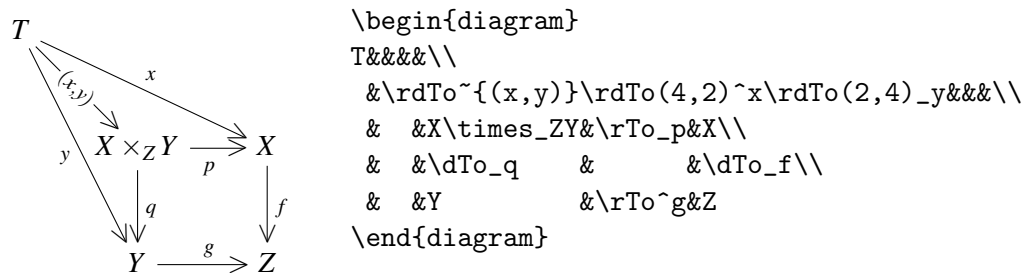
Arrows also stretch (or contract) to match large objects:



If a diagonal arrow is to go across  $x$  cells horizontally and  $y$  cells vertically instead of the usual  $2+2$ , this can be achieved by placing  $(x,y)$  after the arrow command, as illustrated by:



Another example to illustrate the above rules:



It is possible to introduce curved arrows into a diagram if you first define them using the picture environment:

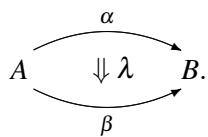
```

\newcommand{\lift}[2]{%
\setlength{\unitlength}{1pt}
\begin{picture}(0,0)(0,0)
\put(0,{#1}){\makebox(0,0)[b]{${#2}$}}
\end{picture}
}

\newcommand{\lowerarrow}[1]{%
\setlength{\unitlength}{0.03\DiagramCellWidth}
\begin{picture}(0,0)(0,0)
\qbezier(-28,-4)(0,-18)(28,-4)
\put(0,-14){\makebox(0,0)[t]{${\scriptstyle #1}$}}
\put(28.6,-3.7){\vector(2,1){0}}
\end{picture}
}

\newcommand{\upperarrow}[1]{%
\setlength{\unitlength}{0.03\DiagramCellWidth}
\begin{picture}(0,0)(0,0)
\qbezier(-28,11)(0,25)(28,11)
\put(0,21){\makebox(0,0)[b]{${\scriptstyle #1}$}}
\put(28.6,10.7){\vector(2,-1){0}}
\end{picture}
}

```

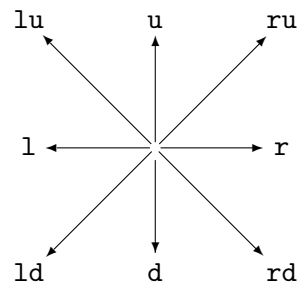
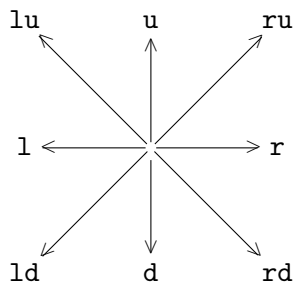


```

\begin{diagram}[w=3em]
A & \upperarrow{\alpha} & \\
\lift{-2}{\ \ \ \Downarrow{\lambda}} & & \\
\lowerarrow{\beta} & & B.
\end{diagram}

```

The following two diagrams were produced with the “heads=littlevee” and “heads=LaTeX” options.



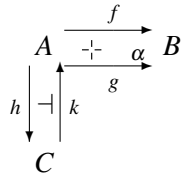
To my eyes, the “vee” option gives heads that are too large, the “littlevee” option gives heads that don’t fit correctly on the shaft, and the “LaTeX” option gives arrows that don’t match the inline arrows.

Finally, an example from the manual:

```

\begin{diagram}[heads=LaTeX]
A          & \pile{\rTo^f\ \ \puncture\quad\scriptstyle{\alpha}\ \ \rTo_g} & B \\
\downarrow h & \dashv\ \ \uTo_k & \\
C          & & 
\end{diagram}

```



According to the manual: “Permission is now granted for [the software’s] use for the production of academic research and textbooks, journals and conference proceedings, subject to the conditions that

- acknowledgement be given as above,
- an up-to-date version of the package be used for the final production,
- and one copy of the book be sent to me on publication in lieu of royalty, at the above address.

Use by commercial organisations is considered (for this purpose) to be academic if the results are intended for publication in an academic forum, concern pure research and do not relate to any particular commercial product.

The software may not be used for any military purpose under any circumstances.”

## The xymatrix package

The `xymatrix` package is included in the drawing package `Xy-pic`. It is possible to use `Xy-pic` directly for drawing commutative diagrams,<sup>1</sup> and some recommend its component `xygraph` for complicated diagrams, but in this version of the guide I will only discuss `xymatrix`.

The documentation for `xymatrix` is embedded in that for `Xy-pic`. There are a user's guide and reference manual available at <http://www.maths.mq.edu.au/~ross/Xy-pic.html>.

The package supports a great variety of arrows, including curved arrows. Arrows do not automatically stretch to match their labels, but this can be done manually. To load it for this document, I used: `\usepackage[all,cmtip]{xy}`.

The syntax is illustrated by

$$\begin{array}{ccc}
 A & \xrightarrow{a} & B \\
 \downarrow b & & \downarrow c \\
 C & \xrightarrow{d} & D
 \end{array}
 \quad
 \begin{array}{l}
 \text{\code{xymatrix}\{ \\
 A \text{\code{\ar[d]~b} \ar[r]^a \&B\ar[d]^c\ \\
 C \text{\code{\ar[r]^d} \quad \quad \quad \&D}\}
 \end{array}$$

Note that only objects are separated by `&` and that all arrows are attached to their source. In contrast to `array`, the diagram when set with `xymatrix` has only two lines of code.

Arrows are specified in the form `\ar@{shape}[dir]` where `shape` describes the shape of the arrow and `dir` is a sequence of single letters — l for left, r for right, u for up, d for down — describing the direction of the arrow. For example, `\ar[rd]` is an arrow from its source to a target one right and one down. Here are some of the possible arrows:

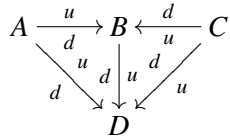
$$\begin{array}{ll}
 A \longrightarrow B & \text{\code{\ar[r]}} \\
 A \longmapsto B & \text{\code{\ar@{|->}[r]}} \\
 A \longdashrightarrow B & \text{\code{\ar@{-}[r]}} \\
 A \lhookrightarrow B & \text{\code{\ar@{^{\{ }->}[r]}} \\
 A \twoheadrightarrow B & \text{\code{\ar@{->>}[r]}} \\
 A \dashrightarrow B & \text{\code{\ar@{.>}[r]}} \\
 A \dashrightarrow B & \text{\code{\ar@{->}[r]}} \\
 A \Longrightarrow B & \text{\code{\ar@{=>}[r]}} \\
 A \rightsquigarrow B & \text{\code{\ar@{~>}[r]}}
 \end{array}$$

For example:

$$\begin{array}{ccc}
 A & \xhookrightarrow{\quad} & B & \xleftarrow{\quad} & C \\
 & \searrow & \vdots & \swarrow & \\
 & & D & & 
 \end{array}
 \quad
 \begin{array}{l}
 \text{\code{xymatrix}\{ \\
 A \text{\code{\ar@{->>}[rd] \ar@{^{\{ }->}[r]}} \\
 \&B \text{\code{\ar@{.>}[d]}} \\
 \&C \text{\code{\ar@_{-}{\{ }->}[1] \ar@{->>}[1d]\ \\
 \&D}\}
 \end{array}$$

To get labels on arrows, place them as superscripts or subscripts on the arrow (between braces if necessary), as illustrated by:

<sup>1</sup>See the articles by Paul Blaga in *The PracTeX Journal*, Issue 4, 2006, and Issue 1, 2007



```

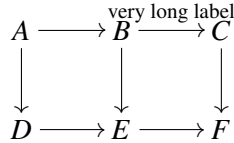
\xymatrix{
A \ar[r]^u_d \ar[rd]^u_d &
B \ar[d]^u_d &
C \ar[l]^u_d \ar[ld]^u_d \\
& & & & D}

```

Note that the position of the label rotates with the arrow. To have a label break an arrow, use the syntax: `\ar[r] | f`

$$A \xrightarrow{| f} B.$$

Arrows don't automatically stretch to match long labels,

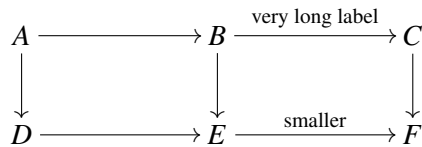


but you can fix this manually by stretching all the horizontal arrows (increasing the column separation):

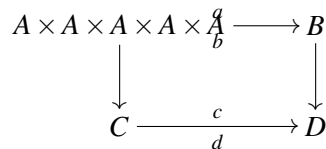
```

\[\xymatrixcolsep{5pc}\xymatrix{...}

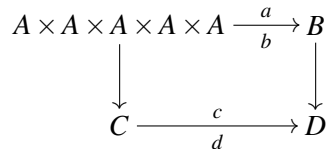
```



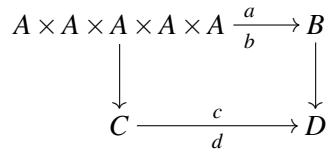
Arrows do stretch (or contract) to match large objects, but then the placement of the label may be wrong (it is placed halfway between the centres of the source and target objects)



To centre the label on the arrow, replace `\ar[r]^a_b` with `\ar[r]^~a_b`



It is also possible to move the labels manually by using, for example, `\ar[r]^<<<a_<<<b \ar[d]`



Objects are positioned using the "centre" of the object, which may not be what you want:

$$\hat{A} \longrightarrow \prod_{n \in \mathbb{Z}} A_n \longrightarrow \prod_{n \in \mathbb{Z}} A_n.$$

Replacing `\xymatrix` with `\xymatrix@1` helps a little:

$$\hat{A} \longrightarrow \prod_{n \in \mathbb{Z}} A_n \longrightarrow \prod_{n \in \mathbb{Z}} A_n.$$

Fortunately, Alexander Perlis<sup>2</sup> has found a fix for the problem, namely, add the line

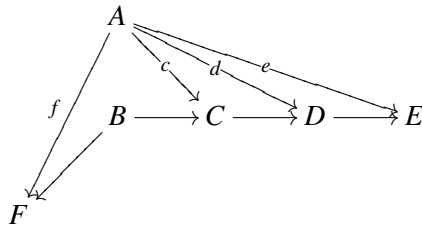
`\entrymodifiers={+!!<0pt,\fontdimen22\textfont2>}`

prior to each `\xymatrix`, or simply add it once and for all in the document's preamble:

$$\hat{A} \longrightarrow \prod_{n \in \mathbb{Z}} A_n \longrightarrow \prod_{n \in \mathbb{Z}} A_n.$$

An arrow that goes, for example, 3 cells right and 1 down, is invoked by `\ar[rrrd]`, as illustrated by:

```
\[
\xymatrix{
&A \ar[lld]_f \ar[r] | -{c} \ar[rrd] | -{d} \ar[rrrd] | -{e} \\
&B \ar[l] \ar[r] &C \ar[r] &D \ar[r] &E \\
F & & & &
}
```



To curve an arrow, insert `/~/` or `/_/_`, as in

$$A \overset{\curvearrowright}{\longrightarrow} B \quad \text{\xymatrix{A \ar@{~}/[r] \ar@{/_}/[r] & B}}$$

Add a dimension to increase the curvature:

$$A \overset{\curvearrowright}{\longrightarrow} B \quad \text{\xymatrix{A \ar@{~}/[r] \ar@{~}1pc/[r] \ar@{~}2pc/[r] & B}}$$

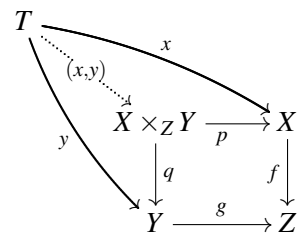
Change the arrow as follows:

$$A \overset{\curvearrowright}{\dashrightarrow} B \quad \text{\xymatrix{A \ar@{->>}@{~}/[r] & B}}$$

The next example illustrates most of these capabilities:

```
\[
\xymatrix{
T \ar@/_/[ddr]_y \ar@/^/[drr]^x \ar@{.>}[dr] | -{(x,y)} \\
&X \times_{\mathbb{Z}} Y \ar[d]^q \ar[r]_p & X \ar[d]_f \\
&Y \ar[r]^g & Z
}
```

<sup>2</sup><http://math.arizona.edu/~aprl/publications/axisalignment/>



## The diagxy package

This is a front end to Xy-pic that contains templates for diagrams. <sup>3</sup> To load it for this document, I used

```
\usepackage[all,cmtip]{xy}
\usepackage{diagxy}
```

The simplest template is

```
\morphism(x,y)|p|/{sh}/<dx,dy>[N'N;L]
```

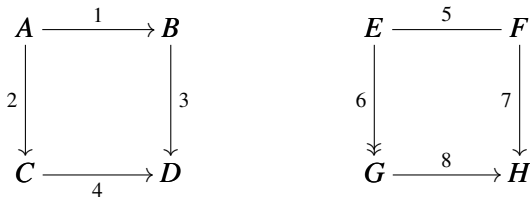
where  $(x,y)$  gives the position of the start of the arrow in units of .01em,  $|p|$  gives the position of the label (**above**, **below**, **left**, **right**, or **mid**),  $\{sh\}$  gives the shape of the arrow (the part in parentheses in the table on p10),  $\langle dx,dy \rangle$  gives the coordinates of the end of the arrow relative to the start,  $N$  is an object, and  $L$  is a label, as in:

$A \xrightarrow{1} B$		$\backslash\backslashbfig$
$A \xrightarrow{2} B$		$\backslashmorphism(0,800)[A'B;1]$
$A \xrightarrow{3} B$		$\backslashmorphism(0,600)/{-}/[A'B;2]$
$A \xrightarrow{4} \gg B$		$\backslashmorphism(0,400) b [A'B;3]$
$A \xleftarrow{5} B$		$\backslashmorphism(0,200) m /{->>}/[A'B;4]$
$A \xrightarrow{6} B$		$\backslashmorphism/{<-}/[A'B;5]$
$A \xrightarrow{7} B$		$\backslashmorphism(800,500) r <0,-500>[A'B;6]$
$A \xrightarrow{8} B$		$\backslashefig\backslash$

It is not possible to have labels both above and below an arrow.

For more complicated templates, such as that for a square, the syntax is similar:

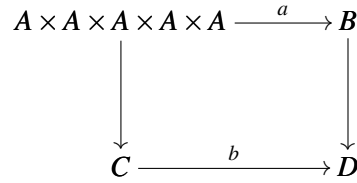
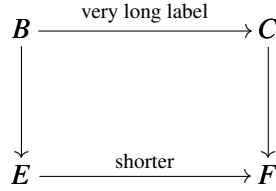
```
\backslashbfig
\square(0,0)[A'B'C'D;1'2'3'4]
\square(1200,0)|aaaa|/{-}'{>>}'>'>/[E'F'G'H;5'6'7'8]
\efig\backslash
```



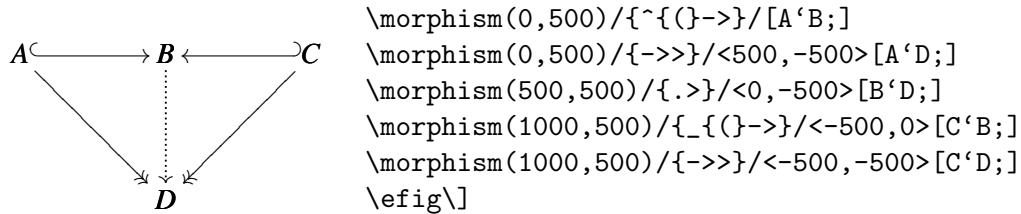
square doesn't adjust for long labels or large objects but its variant Square does:

```
\backslashbfig
\Square|aaaa|[B'C'E'F;\text{very long label}'{}'{}'\text{shorter}]
\Square(2000,0)|aaaa|[A\times A\times A\times A\times A'B'C'D;a'{}'{}'b]
\efig\backslash
```

<sup>3</sup>If your TeX system doesn't have it, you can get `diagxy.tex` from the author's home page <ftp://ftp.math.mcgill.ca/pub/barr/>, rename it to `diagxy.sty` and place it somewhere your TeX system can find it. There is a comparison of `diagxy` with `xymatrix` at <http://www.emis.de/journals/TAC/style/diagxy-xymatrix.pdf>.



It is possible to combine templates to get more complicated diagrams, as in:

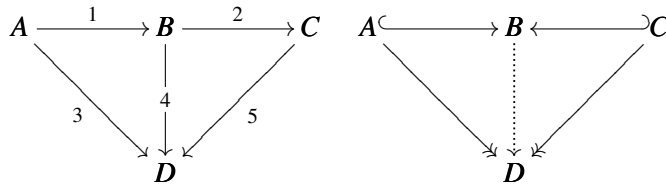


Fortunately, there is a template `Vtrianglepair` that makes this much easier:

```

\[\bfig
\Vtrianglepair[A'B'C'D;1'2'3'4'5]
\Vtrianglepair(1200,0)/{\{(->)}{\{<-~{}}}\{->>}\{.>}\{->>}/[A'B'C'D;{}'{}'{}'{}'{}]
\efig\

```

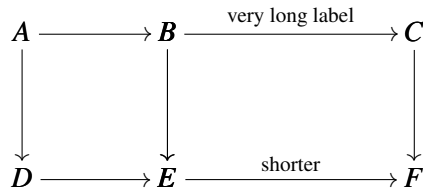


By combining two Squares, one can build more complicated diagrams:

```

\[\bfig
\Square[A'B'D'E;{}'{}'{}'{}]
\Square(500,0)|aaaa|[B'C'E'F;\text{very long label}'{}'{}'\{\text{shorter}}]
\efig\

```

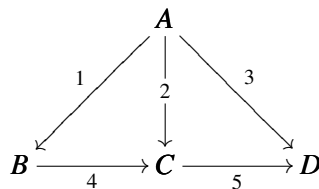


Here are some other templates.

```

\[\bfig
\Atrianglepair[A'B'C'D;1'2'3'4'5]
\efig
\]

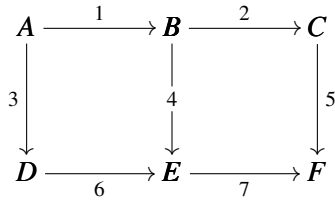
```



```

\[\bfig
\hSquares[A'B'C'D'E'F;1'2'3'4'5'6'7]
\efig
\]

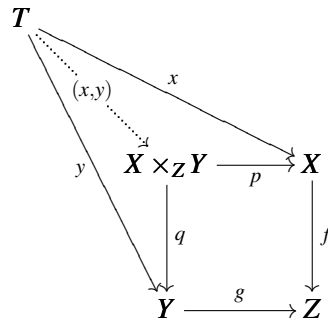
```



```

\[\bfig
\pullback|brra|[X\times_ZY'X'Y'Z;p'q'f'g]%
/>'{.>}'>/[T;x'(x,y)'y]
\efig
\]

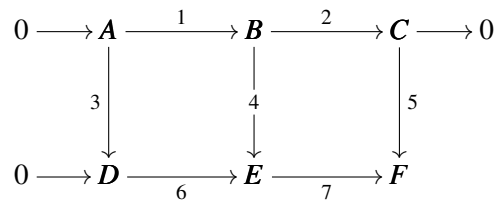
```



```

\[\bfig
\iiixii {7}<300>[A'B'C'D'E'F;1'2'3'4'5'6'7]
\efig

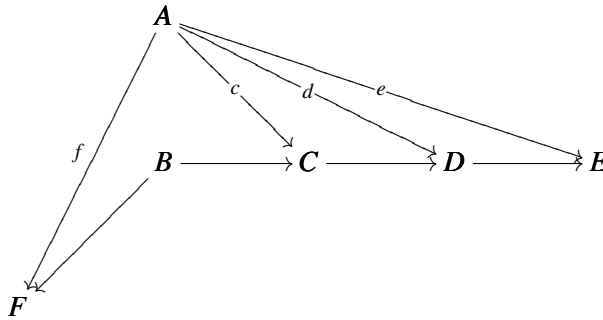
```



Which 0s appear is determined by the first number in braces, which must be between 0 and 15 (it is 7 in the above example), and depends on the binary expansion of the number, as illustrated by the examples at right:

	1	2	4	8	
5	1		1		0 0
7	1	1	1		0 0 0
14		1	1	1	0 0 0

The diagram

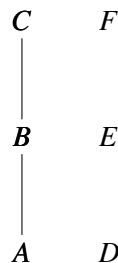


doesn't fit any template, but `\diagxy` offers an alternative method of building diagrams:

```
\[\bfig
\node a(500,1000) [A]
\node b(500,500) [B]
\node c(1000,500) [C]
\node d(1500,500) [D]
\node e(2000,500) [E]
\node f(0,0) [F]
\arrow[a'f;f]
\arrow|m|[a'c;c]
\arrow|m|[a'd;d]
\arrow|m|[a'e;e]
\arrow[b'f;{}]
\arrow[b'c;{}]
\arrow[c'd;{}]
\arrow[d'e;{}]
\efig\]
```

The line `\node a(500,1000) [A]` places the object *A* at (500,1000) and labels it with a (for internal purposes). The line `\arrow[a'f;f]` runs an arrow from the node “a” to the node “f” and labels it with *f*.

If there is no arrow between nodes, then the nodes don't print, but you can add empty arrows:

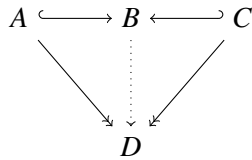


```
\[\bfig
\node a(0,0) [A]
\node b(0,400) [B]
\node c(0,800) [C]
\node d(300,0) [D]
\node e(300,400) [E]
\node f(300,800) [F]
\arrow/{-}/[a'b;{}]
\arrow/{-}/[b'c;{}]
\arrow/{}/[a'd;{}]
\arrow/{}/[b'e;{}]
\arrow/{}/[c'f;{}]
\efig\]
```

Personally, I find this to be the most convenient way to enter complicated diagrams.

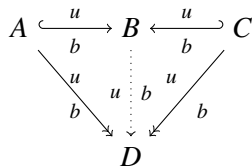


Another example



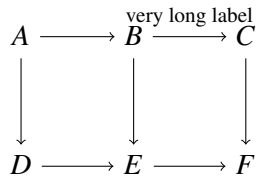
```
\begin{tikzpicture}
\matrix(a)[matrix of math nodes,
row sep=3em, column sep=2.5em,
text height=1.5ex, text depth=0.25ex]
{A&B&C\\
&D\\};
\path[right hook->](a-1-1) edge (a-1-2);
\path[->>](a-1-1) edge (a-2-2);
\path[dotted,->](a-1-2) edge (a-2-2);
\path[left hook->](a-1-3) edge (a-1-2);
\path[->>](a-1-3) edge (a-2-2);
\end{tikzpicture}
```

An example with labels on the arrows:



```
\begin{tikzpicture}
\matrix(a)[matrix of math nodes, row sep=3em, column sep=2.5em,
text height=1.5ex, text depth=0.25ex]
{A&B&C\\
&D\\};
\path[right hook->,font=\scriptsize] (a-1-1)
edge node[above]{$u$} node[below]{$b$} (a-1-2);
\path[->>,font=\scriptsize] (a-1-1)
edge node[above]{$u$} node[below]{$b$} (a-2-2);
\path[dotted,->,font=\scriptsize] (a-1-2)
edge node[left]{$u$} node[right]{$b$} (a-2-2);
\path[left hook->,font=\scriptsize] (a-1-3)
edge node[above]{$u$} node[below]{$b$} (a-1-2);
\path[->>,font=\scriptsize] (a-1-3)
edge node[above left]{$u$} node[below right]{$b$} (a-2-2);
\end{tikzpicture}
```

Long labels may cause a problem:

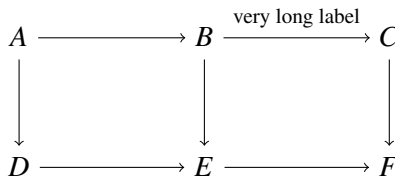


```

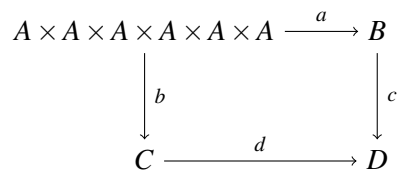
\begin{tikzpicture}
\matrix(m)[matrix of math nodes,
row sep=3em, column sep=2.5em,
text height=1.5ex, text depth=0.25ex]
{A&B&C\\
D&E&F\\};
\path[->,font=\scriptsize]
(m-1-1) edge (m-1-2)
edge (m-2-1)
(m-1-2) edge node[auto] {very long label} (m-1-3)
edge (m-2-2)
(m-1-3) edge (m-2-3)
(m-2-1) edge (m-2-2)
(m-2-2) edge (m-2-3);
\end{tikzpicture}

```

However, this can be fixed by setting `column sep=5.0em`.



TikZ has no problem with large objects



Nor does it have a problem with objects of different heights.

$$\hat{A} \longrightarrow \prod_{n \in \mathbb{Z}} A_n \longrightarrow \prod_{n \in \mathbb{Z}} A_n.$$

But that is because of the options `text height=1.5ex`, `text depth=0.25ex`. When you omit them, you get:

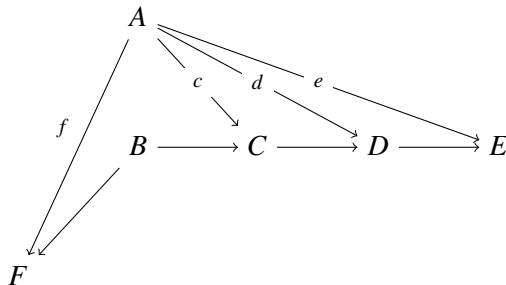
$$\hat{A} \longrightarrow \prod_{n \in \mathbb{Z}} A_n \longrightarrow \prod_{n \in \mathbb{Z}} A_n.$$

Curving arrows is easy.



```
\begin{tikzpicture}
\matrix(m)[matrix of math nodes,
row sep=3em, column sep=2.8em,
text height=1.5ex, text depth=0.25ex]
{A&B\\};
\path[->]
(m-1-1) edge [bend left] (m-1-2)
edge [bend left=40] (m-1-2)
edge [bend left=60] (m-1-2)
edge [bend left=80] (m-1-2)
edge [bend right] (m-1-2);
\end{tikzpicture}
```

Two more examples

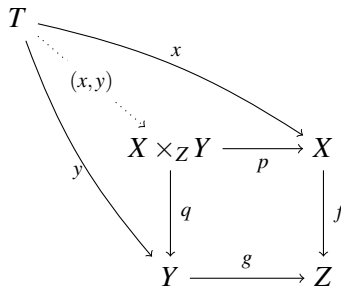


```
\[
\begin{tikzpicture}[descr/.style={fill=white}]
\matrix(m)[matrix of math nodes, row sep=3em, column sep=2.8em,
text height=1.5ex, text depth=0.25ex]
{&A\\&B&C&D&E\\F\\};
\path[->,font=\scriptsize]
(m-1-2) edge node[above left] {$f$} (m-3-1)
edge node[descr] {$c$} (m-2-3)
edge node[descr] {$d$} (m-2-4)
edge node[descr] {$e$} (m-2-5);
\path[->]
(m-2-2) edge (m-3-1)
edge (m-2-3);
\path[->]
(m-2-3) edge (m-2-4);
\path[->]
(m-2-4) edge (m-2-5);
\end{tikzpicture}
\]
```

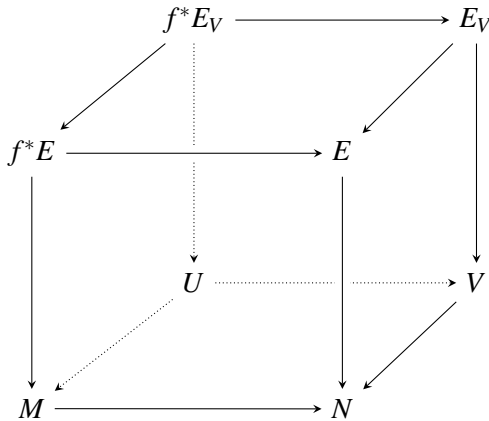
```

\l
\begin{tikzpicture}[descr/.style={fill=white}]
\matrix(m)[matrix of math nodes, row sep=3em, column sep=2.8em,
text height=1.5ex, text depth=0.25ex]
{T\&X\times_Z Y&X\&Y&Z\&};
\path[->,font=\scriptsize]
(m-1-1) edge [bend left=10] node[above] {$x$} (m-2-3)
(m-1-1) edge [bend right=10] node[below] {$y$} (m-3-2);
\path[->,dotted,font=\scriptsize]
(m-1-1) edge node[descr] {$$(x,y)$`} (m-2-2);
\path[->,font=\scriptsize]
(m-2-2) edge node[below] {$p$} (m-2-3)
(m-2-2) edge node[right] {$q$} (m-3-2);
\path[->,font=\scriptsize]
(m-2-3) edge node[right] {$f$} (m-3-3);
\path[->,font=\scriptsize]
(m-3-2) edge node[above] {$g$} (m-3-3);
\end{tikzpicture}
\l

```



Finally an examples from Stefan Kottwitz's webpage. <sup>2</sup>



Notice that he has managed to get the arrows to cross correctly.

To my eyes, the arrow heads are too small, and it seems to be difficult to change them.

<sup>2</sup><http://texblog.net/latex/graphics/pgf-tikz/>