Guide to commutative diagram packages

J.S. Milne

June 18, 2006[*]

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[1]). When publishers first began using 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 commutative diagrams. Despite their wide use by mathematicians[2] and others, these packages are barely mentioned in the usual books on 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, kuvio, DCpic, xymatrix, diagxy.

In future versions I plan to include `diagmac, xygraph, ....` Among the other resources available on the web, there is a 1994 TUGboat article at
`http://tug.org/TUGboat/Articles/tb15-4/tb45vali.pdf`

Except where noted, each package is included in the standard large MikTeX installation and can be used without restriction. Except for `array`, all can produce high quality diagrams. The differences lie in their versality and their ease of use.

To avoid conflicts between the packages, I produced this document in segments which I joined using Acrobat. All segments were produced using `pdflatex` except `kuvio` which doesn't support it.

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 and suggestions.

Summary

The package `amscd` is so easy to use that it should be the first choice for the simple diagrams it can handle. For more complicated diagrams, because of possible conflicts, it is probably necessary to choose one of the remaining programs or the family based on `XY-pic`.

---

[*]Available at `http://www.jmilne.org/not/CDGuide.pdf`.

[1]A diagram is said to commute if the arrows in different paths between two objects compose to the same arrow.

[2]A glance at 20 papers on the largest archive of mathematics papers showed that 15 used at least one commutative diagram package, the most popular being `xymatrix` and `amscd`.

Those familiar with the syntax of `array` will find `diagrams` easy to use and it is among the best at making automatic adjustments, but it has some disadvantages: no curved arrows, some arrow shafts don't quite meet their heads correctly, it doesn't support `dvi` (except for drafts), and it has a quirky license.

The package `kuvio` is very similar to `diagrams`, and appears to be more versatile, but it wouldn't run completely on my system. As it hasn't been revised since 1996, it may be necessary to consider it obsolete.

The package `DCPic` is elegantly simple, but has only a small number of arrow types. It is based on `Pictex`, which has a reputation as a resource hog.

The packages `xymatrix` and `diagxy` based on `XY-pic` are the most complicated to use. For example, `\ar@{^{(}->}` is a clumsy way to produce a hook arrow, and it is necessary in `xymatrix` to adjust manually for long labels and large objects. For those diagrams that it provides templates for, `diagxy` is easier to use. In combination with `XY-pic` itself, these packages are much the most versatile, and for most people who need to go beyond `amscd`, they will probably be the most useful.[3]

Using array

It is possible to produce simple diagrams very easily as arrays, but the quality is poor:

$$
\begin{array}{ccc}
A & \xrightarrow{a} & B \\
\downarrow b & & \downarrow c \\
C & \xrightarrow{d} & D
\end{array}
\qquad\qquad
\begin{array}{ccccc}
A & \rightarrow & B & \leftarrow & C \\
& \searrow & \downarrow & \swarrow & \\
& & D & &
\end{array}
$$

The quality can be improved somewhat, for example, by using stretchable arrows, but commutative diagrams produced with `\array` are best confined to drafts.[4]

---

[3]Among publishers of mathematics, `Xypic` seems to be becoming the standard for more complicated commutative diagrams. For example, on the American Mathmatical Society's FAQ page, one finds:
How can I get diagonal arrows in commutative diagrams?
We recommend the Xypic package for more complicated diagrams, especially if diagonal connections are needed. Xypic is considered a core component of the AMS production environment, so there is no need to include it among the files you must submit with an article or book.

[4]Poor quality diagrams produced by `array` are surprisingly common in mathematics books produced even by leading mathematics publishers.

The amscd package

The American Mathematical Society's package `amscd` can produce only rectangular diagrams (no diagonal arrows) and supports only plain labeled 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:

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

```
\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:

$$A \longleftarrow B \longrightarrow C$$
$$\| \qquad \uparrow$$
$$D =\!\!=\!\!= E$$

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

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

$$A \xrightarrow[b]{a} B$$
$$l \downarrow r \qquad l \uparrow r$$
$$C \xleftarrow[b]{a} D$$

```
$\begin{CD}
A      @>a>b>   B\\
@VlVrV          @AlArA\\
C      @<a<b<   D
\end{CD}$
```

Arrows stretch to match long labels:

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

```
$\begin{CD}
A  @>>> B@>\text{very long label}>>C\\
@VVV      @VVV                      @VVV\\
D  @>>> E@>>>                       F
\end{CD}$
```
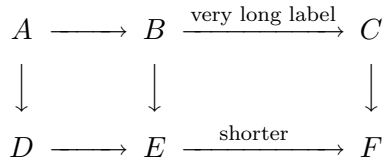
Notice that the lower arrow doesn't stretch to match the upper arrow. To fix this, add a "phantom" label:

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

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

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

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

$$
\begin{CD}
A @>>> B @>{\text{very long label}}>> C \\
@VVV @VVV @VVV \\
D @>>> E @>{\text{shorter}}>> F
\end{CD}
$$
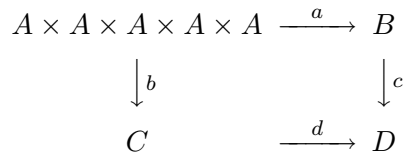
Alternatively, increase the minimum length of a horizontal arrow by replacing the code with `\[\minCDarrowwidth55pt\begin{CD}...\end{CD}\]`.

$$
\begin{CD}
A @>>> B @>{\text{very long label}}>> C \\
@VVV @VVV @VVV \\
D @>>> E @>{\text{shorter}}>> F
\end{CD}
$$
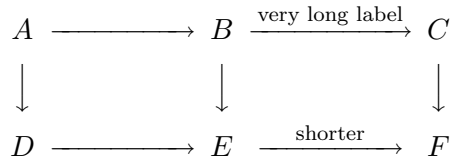
This trick also allows you to shorten arrows if necessary to fit a long diagram onto a page.

Here is how `amscd` handles large objects:

$$
\begin{CD}
A \times A \times A \times A \times A @>{a}>> B \\
@V{b}VV @V{c}VV \\
C @>{d}>> D
\end{CD}
$$

The `diagrams` package

This is not part of the standard MikTex installation, but you can get it from the author's home page `http://www.cs.man.ac.uk/~pt/diagrams/`. To install it, simply change the name of diagrams.tex to diagrams.sty, and put it somewhere TeX can find it, for example, in /localtexmf/tex (and remember to refresh the filename database from MiKTeX Options). A somewhat dated manual 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 (with pdflatex, but not from dvi files or using dvipdfm). 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=littlevee]{diagrams}
\diagramstyle[labelstyle=\scriptstyle]
```

The "small" reduces the distances between objects, "nohug" stops the labels rotating with their arrows (mostly), and "heads=littlevee" determines the size and shape of the heads on the arrows (alternatives LaTeX, vee, triangle, ...). The second row makes the labels print in scriptstyle. Another option that is useful is "noPostScript". This produces lower quality diagrams, but allows MikTeX (for example) to produce dvi files, and is therefore useful for previewing.

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

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

```
\begin{diagram}
A           &\rTo^{a}    &B\\
\dTo_{b}    &            &\dTo_{c}\\
C           &\rTo^{d}    &D
\end{diagram}
```
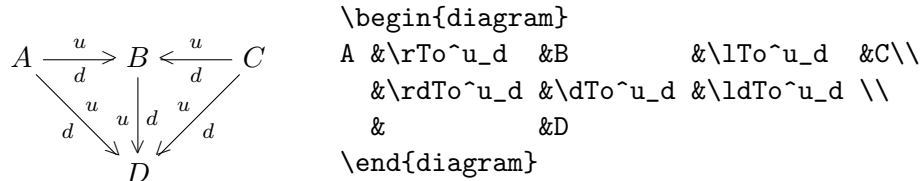
Arrows are specified by a one- or two-letter prefix describing the direction, and a suffix describing the body of the arrow. For example:

lu	u	ru	$\longrightarrow$	To	
			$\longmapsto$	Mapsto	
			$\relbar\relbar$	Line	
l		r	$\hookrightarrow$	Into	
			$\twoheadrightarrow$	Onto	
			$\dashrightarrow$	Dotsto	
ld	d	rd	$\dashrightarrow$	Dashto	
			$\Longrightarrow$	Implies	

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

$$
\begin{array}{ccc}
A & \hookrightarrow B \hookleftarrow & C \\
 & D &
\end{array}
$$

```
\begin{diagram}
A &\rInto  &B          &\lInto &C\\
  &\rdOnto &\dDotsto   &\ldOnto\\
  &        &D
\end{diagram}
```
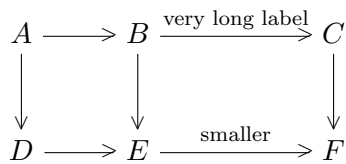
To add labels to arrows, place them as superscripts or subscripts on the arrow (between braces if necessary), as illustrated by:

$$A \xrightarrow[d]{u} B \xleftarrow[d]{u} C$$

(diagram with diagonal arrows labeled u, d to D)

```
\begin{diagram}
A &\rTo^u_d  &B          &\lTo^u_d  &C\\
  &\rdTo^u_d &\dTo^u_d &\ldTo^u_d \\
  &          &D
\end{diagram}
```

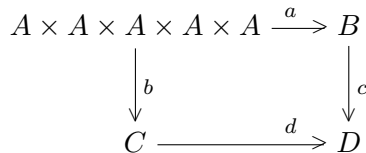A superscript places the label above (or to the left) of an arrow. Labels can also be made to break arrows by using ˜ instead of ^

$$A \overset{a}{-\!\!\!-\!\!\!\rightarrow} B \qquad \texttt{A \&\rTo\~{}\{a\} \&B}$$

Arrows stretch to match long labels:

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

Notice that the lower arrow also stretched.

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

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

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:

(diagram with A, B, C, D, E, F and diagonal arrows labeled c, d, e, f)

```
\begin{diagram}
&                 &A\\
&\ldTo(2,4)^f&&\rdTo\~c\rdTo(4,2)\~d\rdTo(6,2)\~e\\
&                 &B&\rTo          &C&\rTo&D&\rTo&E\\
&\ldTo\\
F
\end{diagram}
```

Another example to illustrate the above rules:

(diagram with T, $X \times_Z Y$, X, Y, Z and arrows labeled (x,y), x, y, p, q, f, g)

```
\begin{diagram}
T&&&&\\
 &\rdTo\~{(x,y)}\rdTo(4,2)^x\rdTo(2,4)_y&&&\\
 &   &X\times_ZY&\rTo_p&X\\
 &   &\dTo_q    &      &\dTo_f\\
 &   &Y         &\rTo^g&Z
\end{diagram}
```
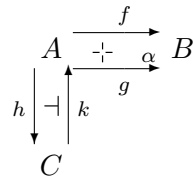
6
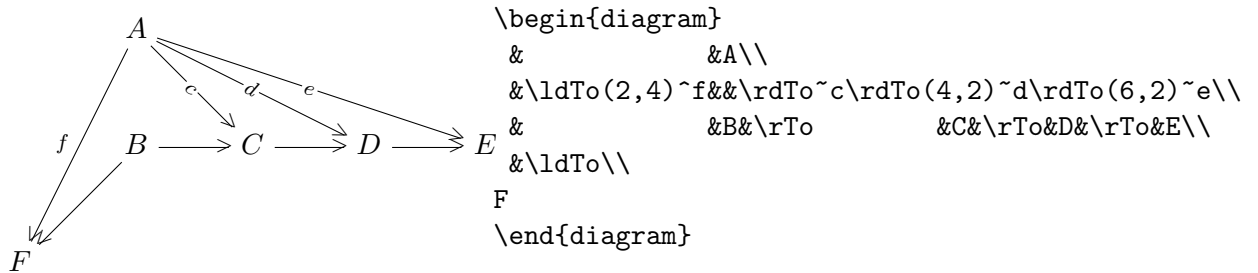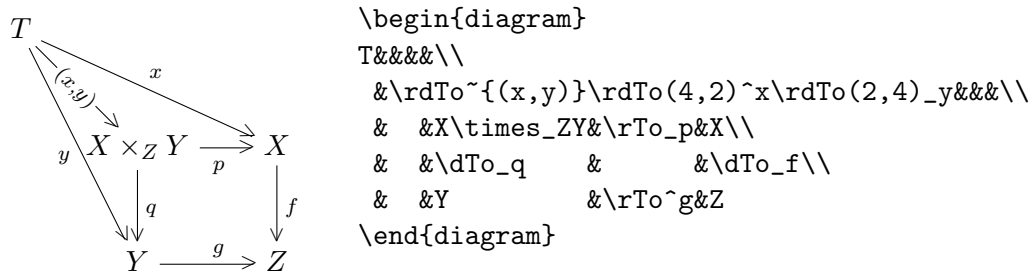
It is possible to introduce curved arrows into a diagram if you first define them using the picture environment:

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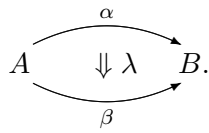
```
\begin{diagram}[w=3em]
A  & \upperarrow{\alpha}
\lift{-2}{\ \ \Downarrow\lambda}}
\lowerarrow{\beta} & B.
\end{diagram}
```

Finally, an example from the manual:

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

$$
\begin{array}{c}
A \underset{g}{\overset{f}{\rightleftarrows}} \; B \\
h \downarrow \dashv \uparrow k \\
C
\end{array}
$$

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 `kuvio` package

MikTeX had not installed this package on my computer, but did so from the internet when I ran it on a file requiring `kuvio`. The files for kuvio can also be found at: `http://ftp.agh.edu.pl/pub/tex/systems/generic/diagrams/kuvio/`. There is a manual "Typesetting diagrams with kuvio.tex" available on the web.

The package uses specials that are recognized only by dvips, not pdflatex, and so this section of the Guide was produced tex → dvi → ps → pdf. According to the manual, the program should be loaded with

    \usepackage[arrsy]{kuvio}.

The option loads special fonts, but on my system it produced garbage, and so I ran it without the option. This limited the package. Perhaps with tweaking, it can be made to run correctly. If not, since it hasn't been updated since 1996, it may need to be considered obsolete. As recommended in the manual p24, I loaded it with the option `forcekdg`.

Except that it lacks curved arrows and doesn't automatically stretch arrows to match labels, it is a very capable package.

The syntax is similar to that of `array` (and `diagrams`), as illustrated by:

```
A ———a——→ B   \Diagram
                A           &\rTo^{a}    &B\\
b|          c|  \dTo_{b}  &            &\dTo_{c}\\
                C           &\rTo^{d}    &D\\
C ———d——→ D   \endDiagram
```

Note that it is necessary to end the last line with \\.

As with `diagrams`, arrows are specified by a one- or two-letter prefix describing the direction, and a suffix describing the body of the arrow. For example:
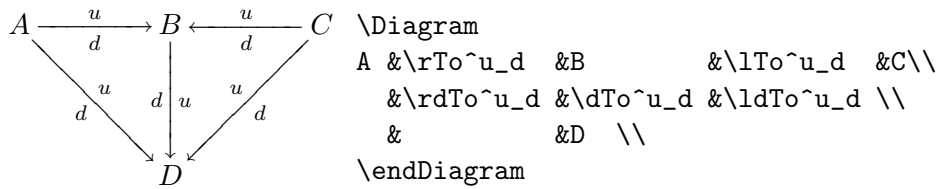
lu	u	ru		→	To		Nul
					Mapsto	⇒	Two
l ←		→ r		↪	Into	=	Eq
				↠	Epi		Dots
ld	d	rd		↔	Bij	—	Line

Notice that `\rMapsto` and `\rDots` produce nothing on my system. To invoke an arrow, combine the two, as illustrated by:

```
A ↪————→ B ←————↩ C   \Diagram
                          A &\rInto  &B        &\lInto &C\\
                            &\rdEpi  &\dDots   &\ldEpi\\
                            &        &D \\
          D               \endDiagram
```
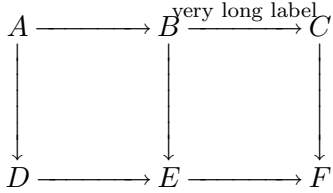
To add labels to arrows, place them as superscripts or subscripts on the arrow (between braces if necessary), as illustrated by:

$$A \xrightarrow[d]{u} B \xleftarrow[d]{u} C$$

```
\Diagram
A &\rTo^u_d  &B          &\lTo^u_d  &C\\
  &\rdTo^u_d &\dTo^u_d &\ldTo^u_d \\
  &            &D  \\
\endDiagram
```
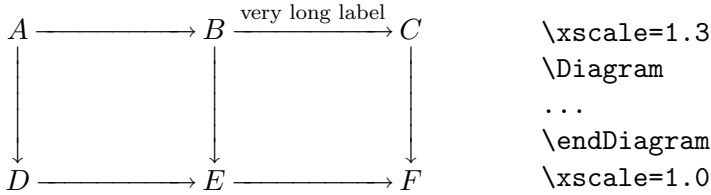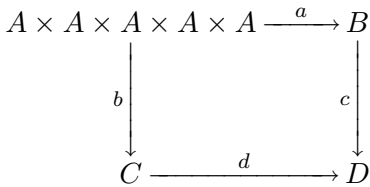
A superscript places the label above (or to the right) of an arrow.
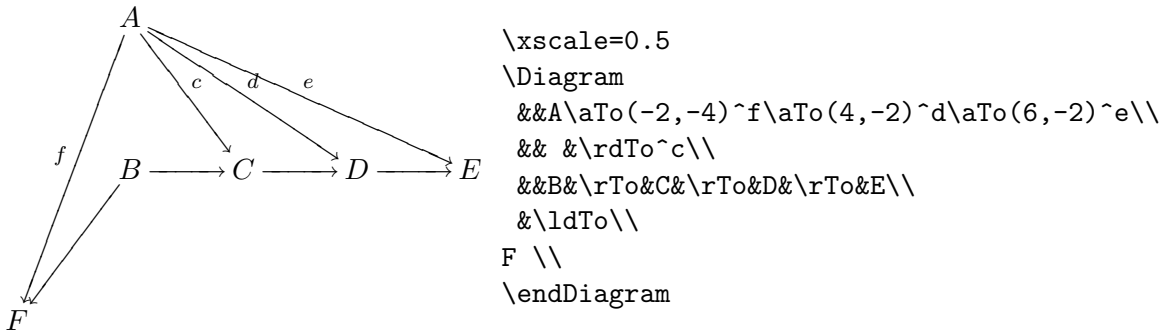
Arrows don't stretch to match long labels:


To fix this, scale the diagram in the x-direction:


```
\xscale=1.3
\Diagram
...
\endDiagram
\xscale=1.0
```

Arrows stretch (or contract) to match large objects but, as in the following diagram, it may be necessary to scale the diagram in the x-direction.

An arrow that points to an object x columns to the right and y rows above is invoked by `\aTo(x,y)`, as illustrated by:


```
\xscale=0.5
\Diagram
&&A\aTo(-2,-4)^f\aTo(4,-2)^d\aTo(6,-2)^e\\
&& &\rdTo^c\\
&&B&\rTo&C&\rTo&D&\rTo&E\\
&\ldTo\\
F \\
\endDiagram
```

Another example to illustrate the above rules:

```
\Diagram
T\aTo(4,-2)^x\aTo(2,-4)_y&&&&\\
&\rdTo^{(x,y)}&&&\\
&&X\times_ZY&\rTo_p&X\\
&&\dTo_q&&\dTo_f\\
&&Y&\rTo^g&Z\\
\endDiagram
```
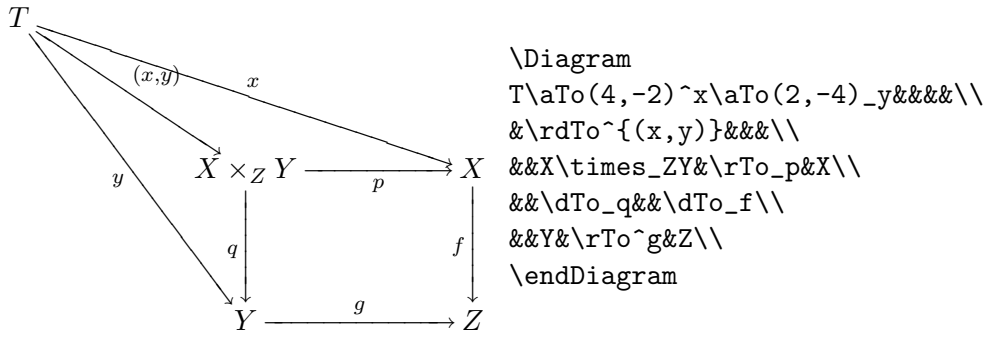
Finally, an example from the manual.

```
\dotted\grid=7mm\yscale=2\Diagrampad=0pt
\Diagram
&&&&&&&&&xz&&&&xyz&&&&&xy^2z&&&&&xy^3z&\\
\\
z&&&&zy&&&&zy^2&&&&zy^3\\
\dy{-.2}
&&&&&&&&x&&&&xy&&&&xy^2&&&&xy^3\\
\\
&&&&y&&&&y^2&&&&y^3\\
\Modify
\Line (0,0) (4,0)\dt{1pt}
\Line (4,0) (8,0)
\Line (8,0) (12,0)
\Line (12,0) (22.5,0)
\Line (0,0) (0,3)\dt{1pt}
\Line (0,3) (0,5.2)
\To (9,2) (12,0)
\To (9,5) (12,3)
\endDiagram
```
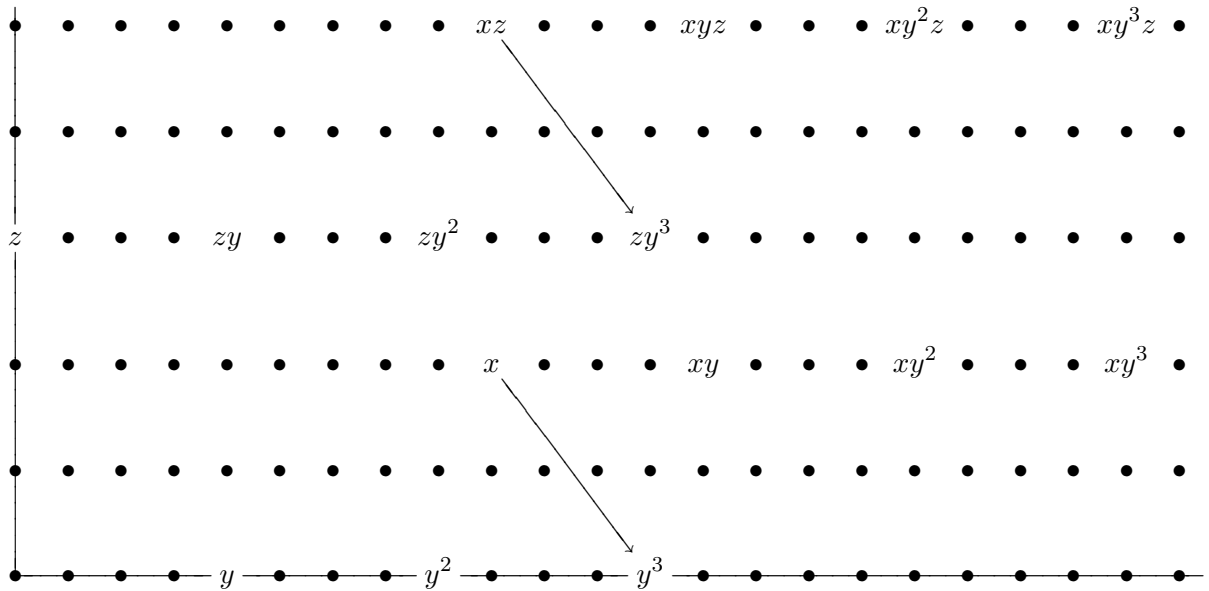
The DCpic package

This package makes use of Pictex (rather, its more modern variant pictexwd) to produce commutative diagrams. The file dcpic.sty and a manual for DCpic can be found on CTAN or at http://hilbert.mat.uc.pt/~pedro/DCpic/man/.

The package supports a limited number of arrows. Its greatest strength is its ability to produce curved arrows. Arrows don't stretch to match their labels, and there is no switch to change the size of all labels. The diagrams are of high quality. Its syntax is very different from the other diagrams. For this file, I loaded it using
\usepackage{pictexwd,dcpic}

Its syntax is illustrated by:
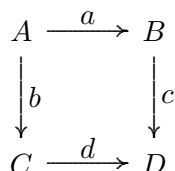
```
\begindc{\commdiag}[50]
\obj(0,1){$A$}
\obj(1,1){$B$}
\obj(0,0){$C$}
\obj(1,0){$D$}
\mor{$A$}{$B$}{$a$}
\mor{$A$}{$C$}{$b$}
\mor{$B$}{$D$}{$c$}
\mor{$C$}{$D$}{$d$}
\enddc
```

Note that the position of each object in the diagram is specified by a pair of integers, and not by its position in the code. The "50" specifies the size of the arrows. The same diagram is produced by the following code:
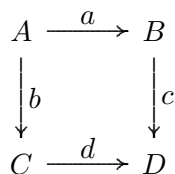
```
\begindc{\commdiag}[5]
\obj(0,10){$A$}
\obj(10,10){$B$}
...
\mor{$C$}{$D$}{$d$}
\enddc
```

There are the following types of arrows:

0	$A \longrightarrow B$	\solidarrow
1	$A \dashrightarrow B$	\dasharrow
2	$A \longrightarrow B$	\solidline
3	$A \hookrightarrow B$	\injectionarrow
4	$A \longmapsto B$	\aplicationarrow (sic)

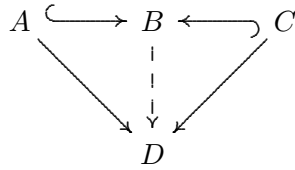An arrow is invoked by

\mor{source}{target}{label}[labelposition,type]

where labelposition can be +1 (=\atright) or -1 (=\atleft) and type is 0,1,2,3,4 or the corresponding command in the table above. Alternatively, the arrow can be invoked by

\mor(w,x)(y,z){label}[labelposition,type]

where (w,x) and (y,z) are the integer coordinates of the source and target. For example, both samples of code below produce the diagram:
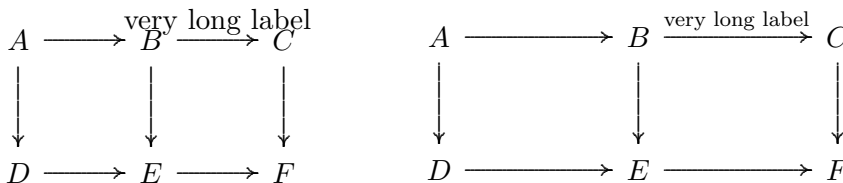
```
\[\begindc{0}[50]              \[\begindc{0}[50]
\obj(0,1){$A$}                 \obj(0,1){$A$}
\obj(1,1){$B$}                 \obj(1,1){$B$}
\obj(2,1){$C$}                 \obj(2,1){$C$}
\obj(1,0){$D$}                 \obj(1,0){$D$}
\mor(0,1)(1,1){}[+1,3]         \mor{$A$}{$B$}{}[\atright,\injectionarrow]
\mor(2,1)(1,1){}[+1,3]         \mor{$C$}{$B$}{}[\atright,\injectionarrow]
\mor(0,1)(1,0){}               \mor{$A$}{$D$}{}
\mor(1,1)(1,0){}[+1,1]         \mor{$B$}{$D$}{}[\atright,\dasharrow]
\mor(2,1)(1,0){}               \mor{$C$}{$D$}{}
\enddc\]                       \enddc\]
```

In the diagram at right below, all arrows have the (default) `\atright` option; in the diagram at left, they have the `\atleft` option. (It is not possible to have labels on both sides of an arrow.)
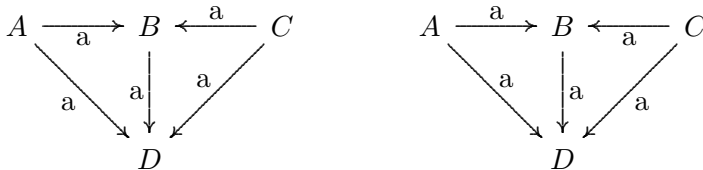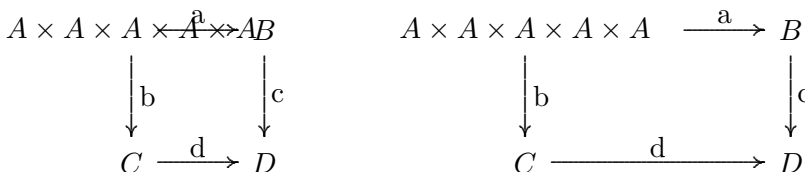

Arrows do not stretch to match long labels:

To fix this, replace `\begindc{0}[50]` with `\begindc{0}[5])`, put $A, B, C \ldots$ at (0,10),(15,10),(30,10) rather than (0,1),(1,1),(2,1) and shrink the label (`\scriptsize{very long label}`), as in the diagram at right.

Nor do arrows stretch (or contract) to match large objects:

To fix this, move B from (1,1) to (2,1) and set the arrow to run from (1,1) to (2,1), as in the diagram at right.

While the syntax of DCpic seems a little clumsy for simply diagrams, it makes it easy to construct complicated diagrams — once the objects are correctly placed, it is easy to add the arrows, as illustrated by,
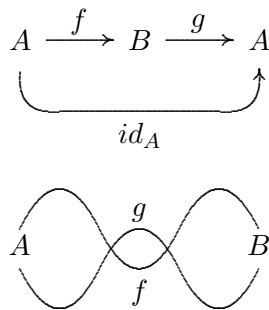
12
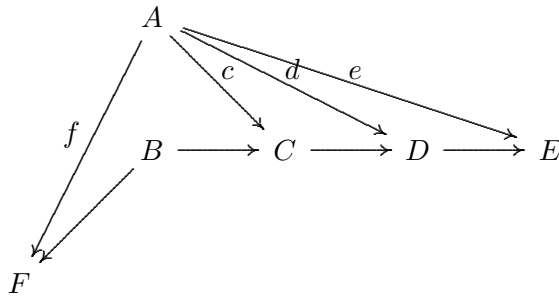
```
\begindc{0}[50]
\obj(1,2){$A$}
\obj(1,1){$B$}
\obj(2,1){$C$}
\obj(3,1){$D$}
\obj(4,1){$E$}
\obj(0,0){$F$}
\mor{$A$}{$F$}{$f$}[-1,0]
\mor{$A$}{$C$}{$c$}
\mor{$A$}{$D$}{$d$}
\mor{$A$}{$E$}{$e$}
\mor{$B$}{$F$}{}
\mor{$B$}{$C$}{}
\mor{$C$}{$D$}{}
\mor{$D$}{$E$}{}
\enddc
```

One of DCpic's strengths is its ability to draw curved arrows. Here are two examples from the "examples" files with the package.

To use DCpic to draw "direct graphs" or "unidirect graph" instead of commutative diagrams, invoke it with \begindc{1} or \begindc{2}. There is an optional argument `placement of the object` for objects, which (apparently) has no effect for commmutative diagrams, and an option argument `correction factors` which, when invoked with numbers other than [10,10], will move the starting or ending point of the arrow.

According to the documentation, with v4.1 "arrows now automatically adjust their size to the object's box size", but I was only able to find v4.0 on the web.

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, 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://evarose.net:800/~krisrose/Xy-pic.html`,
and a tutorial available at
`http://www.dpmms.cam.ac.uk/~al366/xyintroduction/`.

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

$$\begin{array}{ccc} A & \overset{a}{\longrightarrow} & B \\ \downarrow{\scriptstyle b} & & \downarrow{\scriptstyle c} \\ C & \underset{d}{\longrightarrow} & D \end{array}$$

```
\xymatrix{
A \ar[d]^b \ar[r]^a &B\ar[d]^c\\
C \ar[r]^d        &D}
```

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 `hop` 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:

$A \longrightarrow B$ `\ar[r]`

$A \longmapsto B$ `\ar@{|->}[r]`

$A \longrightarrow B$ `\ar@{-}[r]}`

$A \hookrightarrow B$ `\ar@{^{(}->}[r]`

$A \longrightarrow\!\!\!\rightarrow B$ `\ar@{->>}[r]`

$A \dashrightarrow B$ `\ar@{.>}[r]`

$A \dashrightarrow B$ `\ar@{-->}[r]`

$A \Longrightarrow B$ `\ar@{=>}[r]`

$A \rightsquigarrow B$ `\ar@{~>}[r]`

For example:

$$\xymatrix{ A \ar@{->>}[rd] \ar@{^{(}->}[r] & B \ar@{.>}[d] & C \ar@{_{(}->}[l] \ar@{->>}[ld] \\ & D & }$$

```
\xymatrix{
A \ar@{->>}[rd] \ar@{^{(}->}[r]
&B \ar@{.>}[d]
&C \ar@{_{(}->}[l]\ar@{->>}[ld]\\
&D}
```

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

```
\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}
```

$$A \xrightarrow[d]{u} B \xleftarrow[u]{d} C$$

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

$$A \;\text{---}f\text{---}\!\!\!\rightarrow B.$$

Arrows don't automatically stretch to match long labels,

$$A \longrightarrow B \xrightarrow{\text{very long label}} C$$

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

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

$$A \longrightarrow B \xrightarrow{\text{very long label}} C$$

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)

$$A \times A \times A \times A \times A \xrightarrow[b]{a} B$$

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

$$A \times A \times A \times A \times A \xrightarrow[b]{a} B$$

It is also possible to move the labels manually by using, for example,

`\ar[r]^<<<a_<<<b \ar[d]`

$$A \times A \times A \times A \times A \xrightarrow[b]{a} B$$

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

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

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

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

Fortunately, Alexander Perlis (`http://math.arizona.edu/~aprl/publications/axisalignment/`) has found a fix for the problem, namely, add the line
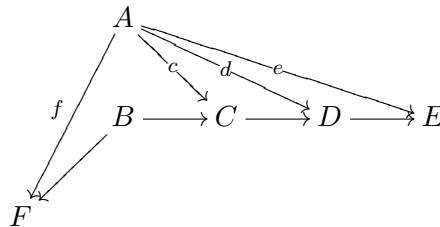
```
\entrymodifiers={+!!<0pt,\fontdimen22\textfont2>}
```

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

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

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

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


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

    `\xymatrix{A \ar@/^/[r]\ar@/_/[r] & B}`

Add a dimension to increase the curvature:

    `A \ar@/^/[r] \ar@/^1pc/[r] \ar@/^2pc/[r]& B`

The next example illustrates most of these capabilities:

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

$$
\begin{array}{ccc}
T & & \\
 & \xrightarrow{\;\;x\;\;} & \\
\ \ \downarrow{\scriptstyle(x,y)} & & \\
 & X \times_Z Y \xrightarrow{\;p\;} X & \\
{\scriptstyle y}\ \ & \downarrow{\scriptstyle q} \qquad \downarrow{\scriptstyle f} & \\
 & Y \xrightarrow{\;\;g\;\;} Z &
\end{array}
$$
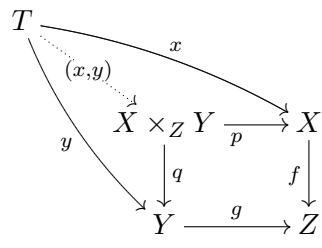
The `diagxy` package

To make sure you have the latest version, get `diagxy.zip` from the author's home page `ftp://ftp.math.mcgill.ca/pub/barr/`, rename diagxy.tex to diagxy.sty and place it somewhere that MikTeX can find it, for example, in /localtexmf/tex, and refresh the filename database from MiKTeX options.

This is an add-on to `XY-pic` that contains templates for diagrams. To load it for this document, I used

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

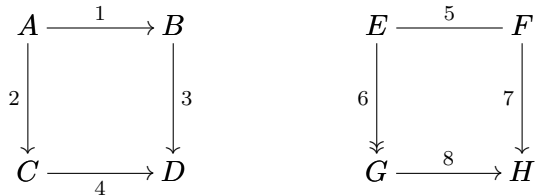The simplest template is

`\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 (**a**bove, **b**elow, **l**eft, **r**ight, or **m**id), `{sh}` gives the shape of the arrow (the part in parentheses in the table on p14), `<dx,dy>` gives the coordinates of the end of the arrow relative to the start, `N` is an object, and `L` is a label, as in:

$A \xrightarrow{\ \ 1\ \ } B$

$A \xrightarrow{\quad 2 \quad} B$

$A \xrightarrow[3]{\quad\quad} B$

$A \xrightarrow{4} B$

$A \xleftarrow{\ \ 5 \ \ } B$

A
$\downarrow{6}$
B

```
\[\bfig
\morphism(0,800)[A'B;1]
\morphism(0,600)/{-}/[A'B;2]
\morphism(0,400)|b|[A'B;3]
\morphism(0,200)|m|/{->>}/[A'B;4]
\morphism/{<-}/[A'B;5]
\morphism(800,500)|r|<0,-500>[A'B;6]
\efig\]
```

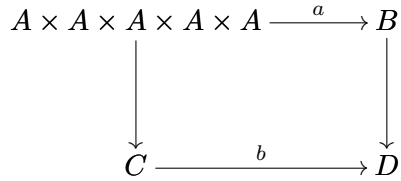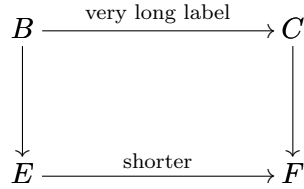It is not possible to have add labels both above and below an arrow.

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

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

$\begin{array}{ccc} A & \xrightarrow{\ 1\ } & B \\ {\scriptstyle 2}\downarrow & & \downarrow{\scriptstyle 3} \\ C & \xrightarrow[4]{} & D \end{array} \qquad \begin{array}{ccc} E & \xrightarrow{\ 5\ } & F \\ {\scriptstyle 6}\downarrow & & \downarrow{\scriptstyle 7} \\ G & \xrightarrow[8]{} & H \end{array}$
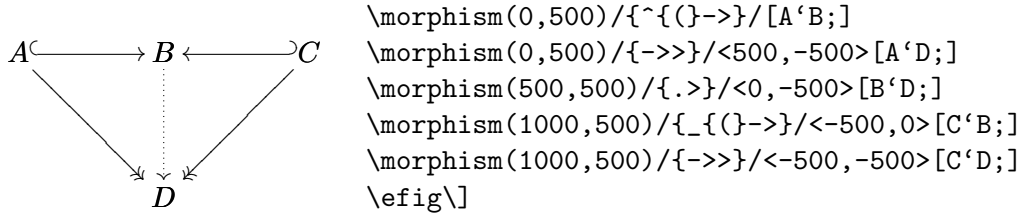
`square` doesn't adjust for long labels or large objects but its variant `Square` does:

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

$$B \xrightarrow{\text{very long label}} C$$

Diagram: B → C (very long label), B → E (down), C → F (down), E → F (shorter)

$$A \times A \times A \times A \times A \xrightarrow{a} B$$

Diagram with $A \times A \times A \times A \times A \xrightarrow{a} B$ top, $C \xrightarrow{b} D$ bottom, vertical arrows.

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

```
\morphism(0,500)/{^{(}->}/[A`B;]
\morphism(0,500)/{->>}/<500,-500>[A`D;]
\morphism(500,500)/{.>}/<0,-500>[B`D;]
\morphism(1000,500)/{_{(}->}/<-500,0>[C`B;]
\morphism(1000,500)/{->>}/<-500,-500>[C`D;]
\efig\]
```
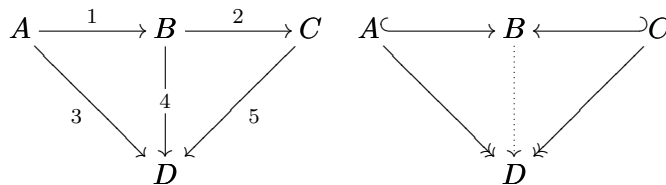
Diagram: A ↪ B ↩ C, with arrows down to D.

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

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

Diagram: A →¹ B →² C with arrows 3, 4, 5 down to D; and A ↪ B ↩ C with arrows down to D.
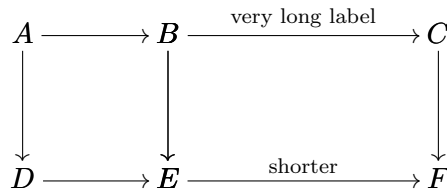
By combining two `Square`s, one can build more complicated diagrams:

```
\[\bfig
\Square[A`B`D`E;{}`{}`{}`{}]
\Square(500,0)|aaaa|[B`C`E`F;\text{very long label}`{}`{}`{\text{shorter}}]
\efig\]
```
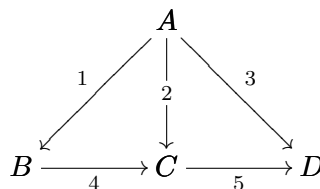
Diagram: A → B → C (very long label), D → E → F (shorter), with vertical arrows.

Here are some other templates.

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

Diagram: A at top with arrows 1, 2, 3 down to B, C, D; B →⁴ C →⁵ D.

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

$$A \xrightarrow{1} B \xrightarrow{2} C$$

with vertical arrows 3, 4, 5 down to

$$D \xrightarrow{6} E \xrightarrow{7} F$$

```
\[\bfig
\pullback|brra|[X\times_ZY`X`Y`Z;p`q`f`g]%
/>`{.>}`>/[T;x`(x,y)`y]
\efig
\]
```

$$T$$

with arrows x, (x,y), y to

$$X \times_Z Y \xrightarrow{p} X$$
$$\downarrow{q} \qquad \downarrow{f}$$
$$Y \xrightarrow{g} Z$$
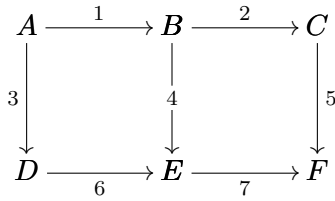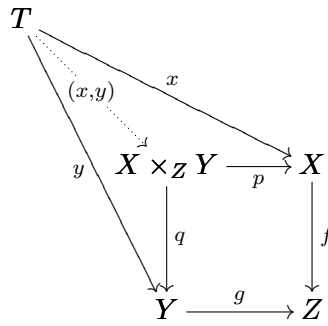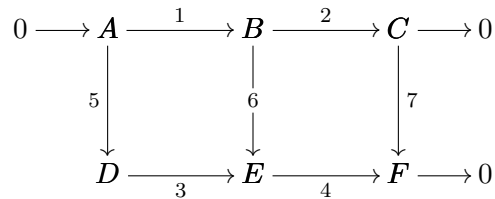
```
\[\bfig
\iiixii {5}<300>[A`B`C`D`E`F;1`2`3`4`5`6`7]
\efig
```

$$0 \longrightarrow A \xrightarrow{1} B \xrightarrow{2} C \longrightarrow 0$$

with vertical arrows 5, 6, 7 down to

$$D \xrightarrow{3} E \xrightarrow{4} F \longrightarrow 0$$

Which 0s appear is determined by the first number in braces, which must be between 0 and 15 (it is 5 in the above example). To discover the correspondence between zeros and numbers, you are asked to solve a riddle, but as the sixteen numbers give only eight of the sixteen possible positions, the riddle has no solution.

The diagram

doesn't fit any template, but \diagxy offers an alternative method of building diagrams similar to that of DCpic:

```
\[\bfig
\node a(500,1000)[A]
\node b(500,500)[B]
\node c(1000,500)[C]
\node d(1500,500)[D]
\node e(2000,500)[E]
\node f(0,0)[F]
\arrow[a`f;f]
\arrow|m|[a`c;c]
\arrow|m|[a`d;d]
\arrow|m|[a`e;e]
\arrow[b`f;{}]
\arrow[b`c;{}]
\arrow[c`d;{}]
\arrow[d`e;{}]
\efig\]
```
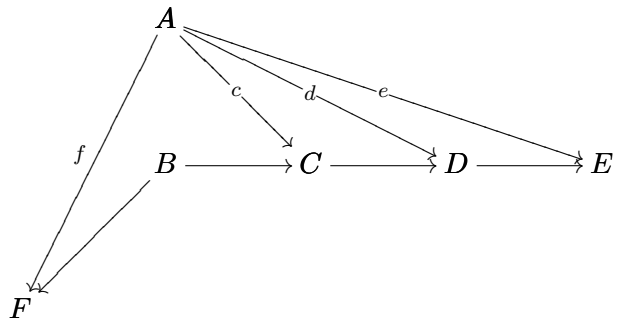
The line \node a(500,1000)[A] places the object A at $(500, 1000)$ and labels it with a (for internal purposes). The line \arrow[a`f;f] runs an arrow from the node "a" to the node "f" and labels it with f.

For users of Scientific Word/Workplace (only!)

The only package SW fully supports is `array`, and so the best strategy is to insert your diagrams as arrays (matrices in SW's language) when writing your document, and then rewrite them using a different package before printing the final version. For `amscd` and `diagrams` this is especially easy as their syntax is similar to that of array.

Only `amscd` comes installed with SW (version 4.1). To install `diagrams` for SW, simply copy diagrams.sty to somewhere in your SW directory where TrueTeX can find it (where the other *.sty files are). According to a post on `http://forum.mackichan.com:81/~mackichan` (October 9, 2004; search for xy-pic), it is probably impossible to install `xy-pic` (hence `xymatrix`) in the TrueTex that comes with SW because of problems with fonts, but it is probably possible to install it with the pdflatex that comes with v5.0 of SW. Alternatively, you can replace TrueTeX with MikTeX, which is freely available at `http://www.miktex.org/`.

When SW reads a file, it translates the TeX into its own code, and then translates it back when it writes it out. This can cause problems with code it doesn't understand. I've had no problems with the code for `amscd` and `diagrams` in recent versions of SW. The `xymatrix` code beginning with `\xymatrix@C=xpc` caused problems (a `\\` was changed to `\newline`), but this can be avoided by using the syntax `\xymatrixcolsep{xpc}`. In general, you can avoid problems of this type by "encapsulating" the foreign code, or by inputting it from a separate file.

When using packages not directly supported by SW, it is advisable to save your documents as "portable latex" (Save As, Type Portable LaTeX) to avoid conflicts between the package and the file tcilatex.tex which is otherwise included.