Commutative Diagrams in TeX (version 4)

Paul Taylor
Department of Computing,
Imperial College of Science, Technology and Medicine,
180 Queen's Gate, London SW7 2BZ
+44 71 594 8263
`<pt@doc.ic.ac.uk>`
FTP: `theory.doc.ic.ac.uk` (146.169.2.27)

12 July 1994

Abstract

TeX and LaTeX have become standard as a way of writing papers in Computer Science and Category Theory. Even in source form they are easier to compose and read than attempts to write mathematics in ASCII. In Category Theory "commutative diagrams" are essential for a clear visual understanding of the paper, but the graphics capabilities of TeX are so limited that it is very difficult to draw them nicely, if at all. This manual describes a new but reverse-compatible version of a package to draw such diagrams, expressed in a language in which many users have already found it very easy to express themselves.

1 Introduction

In papers in mathematics and computer science which employ Category Theory, there is much benefit in clarity if "commutative diagrams" are used as much as possible to illustrate definitions, equations and universal properties. Here is a typical such diagram: it is one of the Mac Lane-Kelly equations.

$$\begin{tikzcd}
A*(B*(C*D)) \arrow[r, "\mathrm{assl}"] \arrow[d, "\mathrm{id}*\mathrm{assl}"'] & (A*B)*(C*D) \arrow[r, "\mathrm{assl}"] \arrow[d, phantom, "="] & ((A*B)*C)*D \\
A*((B*C)*D) \arrow[rr, "\mathrm{assl}"] & & (A*(B*C))*D \arrow[u, "\mathrm{assl}*\mathrm{id}"']
\end{tikzcd}$$

This manual describes version 4 of the author's package for drawing diagrams line this in (plain) TeX or LaTeX. Version 3 is already very widely used in the Category Theory and Theoretical Computer Science communities. Most of the underlying code has been rewritten, with a great improvement to the appearance of the diagrams, but it remains compatible with the previously developed and very popular straightforward language.

2 Design Criteria

Drawing such a diagram using the LaTeX `picture` environment takes about sixty lines of code, though some saving is possible (with the positioning of labels on arrows) by means of some simple macro programming. To get the arrows to match up neatly with the objects takes quite a lot of experimentation, and the whole job has to be repeated with each new diagram or each modification. Again, the use of macros for commonly occurring diagrams such as squares and triangles can save effort, but this does not postpone the difficulty very far, because as soon as we want to draw a slightly more complicated diagram we're back to square one.

The now widespread use of workstations with big screens and "personal" computers has lead to a kind of religious fervour that a mouse with at most three buttons is always easier to use than a keyboard with maybe over a hundred. This might be so if the tracking software were accurate and could use telepathy to ascertain what the user wanted, but in reality the attempts I have seen to "draw it on the screen" and include the result in a TEX document have looked as professional as what children bring home from their first day at school. My view is, if you want *wysiwyg,* use pen and paper!

Besides the awful results, mouse-driven methods take longer and are less portable. If you want to write a joint paper with a colleague on the other side of the world, it is a great deal simpler to send a single ascii file by electronic mail than to package twenty of them (including one file per diagram as well as the main text) encoded in some weird commercial binary format.

On the basis of these remarks, the design criteria of this package are as follows:

1. The entire diagram must form part of the source of the document itself. In other words, there must be no preprocessing (*cf.* `eqn` in UNIX) or inclusion of files (such as POSTSCRIPT pictures).

2. Simple diagrams must be able to be drawn "on the fly" and not need to be drawn on paper first. Obviously, complex diagrams will already have been worked out on paper anyway.

3. The layout of the source code must resemble the intended diagram as far as syntactically possible.

4. There must be no measuring of labels to calculate co-ordinates or lengths of arrows.

5. There must be a variety of arrow styles, with facilities for defining new ones. Diagonal arrows (which, through lack of appropriate primitives, TEX makes very difficult to draw) should be provided at various slopes, albeit with limited choice and features.

6. The package must be compatible both with `plain` TEX and with LATEX, and not rely on non-standard fonts[1] or language features.

7. Future versions which improve the appearance of the diagrams must, as far as possible, be compatible with past papers written using the package — but you hack at your own risk!

In addition there are æsthetic criteria, some of which may be a matter of opinion:

- Arrows should stretch to meet the objects which are intended to be their endpoints.

[1]Diagonal lines can *only* be drawn using LATEX's `line10` and `linew10` fonts, and for exotic arrow styles the AMS maths symbols fonts may be useful: we regard these as standard. Alan Jeffery's St. Mary Road symbols font is also useful.

- Arrows should be aligned (both horizontally and vertically) with the centres of the objects.
- Labels on arrows should not affect the spacing of the diagram except to avoid overlapping.
- Stretching of arrows should not affect the centering of their labels.

3 Typing the diagram

The diagram above is produced in LaTeX as follows:

```
\input diagrams
\def\Assl{{\rm assl}}\def\Id{{\rm id}}
\begin{diagram}
 A*(B*(C*D))   & \rTo^\Assl & (A*B)*(C*D) & \rTo^\Assl &   ((A*B)*C)*D   \\
\dTo^{\Id*\Assl} &          &      =      &            & \uTo_{\Assl*\Id} \\
 A*((B*C)*D)   &            & \rTo^\Assl  &            &   (A*(B*C))*D   \\
\end{diagram}
```

In `plain` TeX you do the same thing, writing `\diagram` and `\enddiagram` wherever we have `\begin{diagram}` and `\end{diagram}`.

In LaTeX 2_ε you can put

```
\usepackage{diagrams}
```

in place of the `\input` command, but you have to rename or alias the file to `diagrams.sty`.

The basic rule is to divide the diagram into cells, just like the cells of a

$$\begin{tikzcd}
A*(B*(C*D)) \arrow[r, "\mathrm{assl}"] \arrow[d, "\mathrm{id}*\mathrm{assl}"'] & (A*B)*(C*D) \arrow[r, "\mathrm{assl}"] & ((A*B)*C)*D \\
 & = & \\
A*((B*C)*D) \arrow[rr, "\mathrm{assl}"'] & & (A*(B*C))*D \arrow[uu, "\mathrm{assl}*\mathrm{id}"']
\end{tikzcd}$$

matrix, and then type the contents of the cells with columns delimited by "`&`" and rows by "`\\`". The bottom arrow extends through the empty cells either side of it. Notice that although the matrix imposes a kind of co-ordinate system, the widths of the columns and the heights of the rows are variable and chosen automatically — by TeX, in the same way as it does for matrices, tables, *etc.*

At first you will probably need to draw the diagram carefully on paper and divide it into cells in this way before typing it in. The main difficulty is working out how many `&`s to insert; for this it is useful to observe that *in the simplest cases*[2]

- objects and verticals go in odd-numbered columns, and
- horizontals and diagonals go in even-numbered columns.

[2]The bottom arrow disobeys this rule, since it goes across three columns.

Then, of course, you need an even number of `&`s between columns of the same parity and an odd number between different ones. After a little practice you'll learn other rules of thumb, but even if you make a mistake, the DVI previewer will make it clear how to correct it. Error messages about *clashing* or *unterminated* arrows indicate that something is wrong without previewing.

Each cell should contain *either* an object (an ordinary mathematical expression, set in maths mode) *or* a morphism (an arrow such as `\rTo^f`). Horizontal and vertical arrows cannot be mixed in one cell; moreover only one horizontal arrow per cell is allowed, but see the section 6 on parallel maps below.

The horizontal and vertical arrows extend through the empty cells either side until they meet a non-empty cell, just like the rook (castle) in chess. For this purpose anything other than white space (space, tab and newline), comments (`%`) and `\empty` in the source make a cell non-empty For example `\null`, `\␣` or `{}`may be used to terminate arrows: it's not necessary that anything be printed. If you don't terminate an arrow, it will extend to the edge of the diagram, but just exactly where the "edge" *is*, particularly the right one, will be determined somewhat arbitrarily.

The chess rule does not apply to diagonals, whose endpoints are specified differently: see section 10.

Do not enclose the arrow commands in boxes or braces, because this prevents the automatic stretching from working.

The *horizontal* arrow commands (including `\pile`)may also be used in text, $A \xrightarrow{f} B$. Since there is no enclosing matrix, this is written as `$A \rTo^f B$` *without* `&`. You still need the dollars because it's still a mathematical expression. Arrows participate in the horizontal stretching and shrinking of spaces between words in a paragraph, but of course their labels also force the lines of the paragraph apart.

The arrow commands are *fragile* in the LaTeX sense: if you want to use them in section headings you must `\protect` them.

4 Labels

Each arrow carries up to three labels, whose position is specified analogously to superscripts[3] by

`^` above, `_` below, `<` left, `>` right and `~` middle.

For reverse compatibility, above=left and below=right for vertical arrows. Very old versions of the package used positional arguments; these are also still supported, but *must be enclosed in braces, e.g.* `\rTo{f}{g}` but not `\rTo f g`.

Explicitly, the labels are placed as follows:

- for **horizontal** arrows, `\rTo^f_g` and `\rTo{f}{g}` give $\xrightarrow[g]{f}$;
- for **vertical** arrows, `\dTo <f >g`, `\dTo^f_g` and `\dTo{f}{g}` give $f\big\downarrow g$;
- for **positive** gradient **diagonal** arrows, `\ldTo^f_g`, `\ldTo <f >g` and `\ldTo{f}{g}` all give ${}^f\!\swarrow_g$ (similarly `\ruTo`).

[3]However they are *not* recognised syntactically in the same way, and so for instance `\nolimits` will not work. The sub- and superscript characters are recognised by their `\catcode`s, so `\sp` and `\sb` will work, but the others are compared using `\ifx`, so have to be the same characters, with the same `\catcode`s as when `diagrams.tex` was read in. The `Sb` and `Sp` environments in AMS-TeX and AMS-LaTeX will not work, and the text of the label itself must be either a single token or be explicitly enclosed in `{}` or `\bgroup\egroup`.

- but for **negative** gradient arrows, above=right and below=left, so `\rdTo^f_g`, `\rdTo <g >f` and `\rdTo{f}{g}` give ${}_g\searrow^f$ (similarly `\luTo`);
- Using tilde, the label may instead **break** the arrow: `\rTo~f` gives A —f→ B. If the arrow had a middle (*e.g.* + in `\rCrossedInto`, page 8), the label would replace it. This is sometimes useful to preserve the symmetry of a diagram with three verticals.

Although the arrows extend into adjacent cells, the labels on horizontal and vertical maps remain centred *in the cell in which the arrow was declared.*

Any ordinary (mathematical) text you put in the cell with an arrow will appear in the usual way:

- For horizontal arrows it will therefore be attached to the arrowhead or tail.

  This offers a simple way of annotating the head of an arrow, for example

  `$ A \rTo^f \relax ^*  B  {}_s \lTo_g C$` gives
  $A \xrightarrow{f} {}^*B_s \xleftarrow[g]{} C.$

  where `\relax` is needed to stop `\rTo` from reading `^*` as another label and `{}` serves as an object to which s is subscripted. The spacing in this example is not entirely satisfactory, because `\rTo` and `\lTo` generate the `shortfall` spacing as part of themselves and regard everything else as their targets. You can use `\!` and `\;` commands to compensate for this, remembering that should you subsequently change the `shortfall` amount then you must also adjust the compensation.

  An alternative way of doing this is to define a new arrow command: section 7 explains how.

  Don't make a habit of putting `\relax` after arrow commands: it can lead to gaps in short double-shafted arrows, for reasons which are too complicated to explain here.

- For vertical and diagonal arrows additional text will lie to the side of (or between) the arrows. In the case of verticals this is at the same height as the middle and labels of the arrows; see section 6.

By default the labels on the arrows are set in `\textstyle` (the same size as in text) when they are part of a matrix, and in `\scriptstyle` (like subscripts) in text. However this may be changed using the `labelstyle` option. This re-defines the command `\labelstyle`, for instance

```
\begin{displaymath}
\renewcommand\labelstyle{\scriptscriptstyle}
\begin{diagram}
A &\rTo^{\sum^n_1 f_i}_{\rm scriptscript} & B
\end{diagram}
\qquad
...
\end{displaymath}
```

gives

$$A \xrightarrow[\scriptscriptstyle\text{scriptscript}]{\scriptscriptstyle\sum_1^n f_i} B \qquad A \xrightarrow[\scriptstyle\text{script}]{\scriptstyle\sum_1^n f_i} B \qquad A \xrightarrow[\textstyle\text{text}]{\textstyle\sum_1^n f_i} B \qquad A \xrightarrow[\displaystyle\text{display}]{\displaystyle\sum_1^n f_i} B$$

Similarly, objects within a diagram are set in `\displaystyle` unless the `\objectstyle` command is redefined. (In text, *e.g.* `$A \rTo B$`, the setting of the objects is out of the control of the diagrams package. Unless you have done some special programming of your own, they will be set in `\textstyle`.)

5 The Cube

The cube offers a good example of how horizontal and vertical arrows stretch. Notice how the cells have been sub-divided further to give the "3D" effect, and that the positions of the labels on the maps betray the cells in which they were declared. Because of the sub-division, the parity rule for the column in which the horizontal arrows go has broken down. You should now formulate for yourself a new rule of thumb for the cases when you need one, two, three or four `&`s between two commands.

```
\begin{diagram}
  A'     &          & \rTo^{f'} &           &            & B'          &           &        & \\
         & \rdTo_a  &           &           &            & \vLine^{h'} & \rdTo_b   &        & \\
\dTo^{g'}&          &     A     &  \rTo^f   &            &   \HonV     &           &   B    & \\
         &          & \dTo^g    &           &            &   \dTo      &           &        & \\
  C'     & \hLine   &  \VonH    & \rTo^{k'} &            &    D'       &           & \dTo_h & \\
         & \rdTo_c  &           &           &            &             & \rdTo_d   &        & \\
         &          &     C     &           &            &  \rTo^k     &           &   D    & \\
\end{diagram}
```

The broken lines (h' and k') are simply two arrows. The commands `\HonV` ("horizontal on vertical") and `\VonH` ("vertical on horizontal") allow arrows in one direction to pass through, whilst obstructing arrows in the other direction as if an object were present. They each have an optional argument: `\HonV[=]` and `\VonH[\|]` should be used to allow extra space for double lines. An alternative is to declare the "front" line in the intersection cell, where it will have the effect of obstructing the other arrow, but then its label (if any) would clash with the other arrow.

There is also a command `\HmeetV` which causes single horizontal and vertical lines to meet, forming a corner.

$$\begin{tikzcd}
A' \arrow[rr, "f'"] \arrow[dr, "a"'] \arrow[dd, "g'"'] & & B' \arrow[dd, "h'"'] \arrow[dr, "b"] & \\
& A \arrow[rr, "f"] \arrow[dd, "g"'] & & B \arrow[dd, "h"] \\
C' \arrow[rr, "k'"] \arrow[dr, "c"'] & & D' \arrow[dr, "d"'] & \\
& C \arrow[rr, "k"] & & D
\end{tikzcd}$$

You should soon be able to read and compose the source of these diagrams as easily as the printed version, although one would not normally go to the trouble of aligning the `&`s in the source code![4]

6 Parallel arrows

You can draw two arrows between the same two vertices, including extra things like 2-cells ($\Downarrow$, `\Downarrow`), the adjoint symbol ($\dashv$, `\dashv`) in between. Peter Freyd's `\puncture` symbol is also defined in the diagrams package.

```
\begin{diagram}
        A          & \pile{\rTo^f\\ \puncture\quad\alpha \\ \rTo_g} & B \\
\dTo^h\dashv\uTo_k &                                                 &   \\
        C          &                                                 &   \\
\end{diagram}
```

$$\begin{tikzcd}
A \arrow[r, shift left=2, "f"] \arrow[r, shift right=2, "g"'] \arrow[r, phantom, "\dotplus\quad\alpha"] \arrow[d, shift right=2, "h"'] \arrow[d, phantom, "\dashv"] & B \\
C \arrow[u, shift right=2, "k"']
\end{tikzcd}$$

Horizontal arrows may be `\pile`d on top of each other. Effectively, a one-column diagram is created, which may have *either* a horizontal arrow *or* an object in each row. This works both in diagrams and in text. The spacing (`\baselineskip`) between the rows is *half* of that specified by the `pilespacing` option (page 13), on the assumption that you will put something between parallel arrows; a blank line (`\\ \\`) will suffice, then the spacing is the same as that for vertical arrows.

Vertical arrows may simply be put together in the same cell, with any text between them. The spacing is always that given by `\pilespacing`. (`\pile` must not be used.)

[4] To do so neatly in this manual required special programming: the `verbatim` environment produced unsatisfactory results.

Diagonal lines are currently implemented as fixed (non-stretching) boxes and can therefore be juxtaposed, separated with explicit space and moved with `\raise` or `\raisebox` commands.

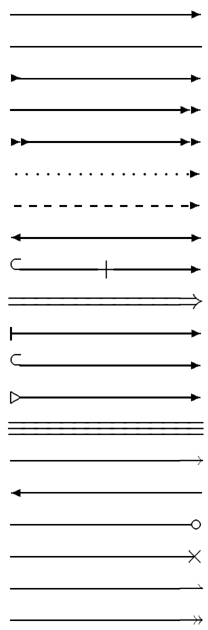
```
\raise 5pt \hbox{$\ldTo^R$}\rightthreetimes\raise-5pt\hbox{$\ruTo^L$}
```

However this will be incompatible with future versions; there is an error message ("you must not put arrows in braces") to warn you of this but you can ignore it for the time being.

7 Defining arrow styles

It is easy to define other arrows besides the basic line with arrowhead given by `\rTo` and friends. For example,

`\newarrow{To}`	`---->`
`\newarrow{Line}`	`-----`
`\newarrow{Embed}`	`>--->`
`\newarrow{Onto}`	`----{>>}`
`\newarrow{EEmbedd}`	`{>>}---{>>}`
`\newarrow{Dotsto}`	`....>`
`\newarrow{Dashto}`	`{}{dash}{}{dash}>`
`\newarrow{Corresponds}`	`<--->`
`\newarrow{CrossedInto}`	`{boldhook}-+->`
`\newarrow{Implies}`	`===={=>}`
`\newarrow{Mapsto}`	`|--->`
`\newarrow{Into}`	`C--->`
`\newarrow{Openinto}`	`{triangle}--->`
`\newarrow{Congruent}`	`33333`
`\newarrow{TeXto}`	`----{->}`
`\newarrow{Backwards}`	`<----`
`\newarrow{Multi}`	`----o`
`\newarrow{Crossto}`	`----X`
`\newarrow{Partial}`	`----{harpoon}`
`\newarrow{TeXonto}`	`----{->>}`



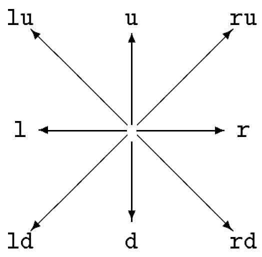
Note that the `{->>}`, `{harpoon}`, `{->}` and `{=>}` heads use TeX's `\rightharpoondown` ($\rightharpoondown$), `\rightarrow` ($\rightarrow$) and `\Rightarrow` ($\Rightarrow$) symbols, consisting of heads *with shafts.* Also, the `boldbook` tail uses LaTeX's `\boldmath`: please see the note about `heads=littlevee` on page 11. The `hook` and `C` tails are the same.

Each declaration

`\newarrow{`*Name*`}{`*tail*`}{`*filler*`}{`*middle*`}{`*filler*`}{`*head*`}`

defines eight arrow commands

`\r`*Name*	right	or	east
`\l`*Name*	left	or	west
`\d`*Name*	down	or	south
`\u`*Name*	up	or	north
`\ru`*Name*	right & up	or	northeast
`\rd`*Name*	right & down	or	southeast
`\lu`*Name*	left & up	or	northwest
`\ld`*Name*	left & down	or	southwest



from the five named components.

Some but not all of the names given in the examples above are defined in the source of the package as distributed. However this is only really intended for reverse compatibility, because just as it is better to define a macro `\isomorphic` for the symbol $\cong$ (if that is what you use it to mean) than to write `\cong` in your documents, so it is advisable to define the arrow command names in the file where you keep your private collection of symbols.

For example, if in your subject there are special kinds of functions known standardly as immersions, inclusions and internalisations and written as $\hookrightarrow$, $\rightarrowtail$ and $\triangleright\!\!\longrightarrow$, then you should define `\rImmerse`, `\rInclude` and `\rInternalise` instead of using the names `\rInto`, `\rEmbed` and `\rOpeninto`. The motto is *what you see is what you mean*! This avoids remembering or getting confused about the so-called standard macro names, and also enables you to change your mind about the notation if it is not standard but subject to experimentation.

The components as given in the `\newarrow` command are mnemonics, which are themselves defined using the commands

`\newarrowhead`, `\newarrowtail`, `\newarrowfiller` and `\newarrowmiddle`.

Most of the characters in the standard TEX Computer Modern fonts which are appropriate for these components have already been used in the package and are illustrated above. If you wish to define other components, the examples at the end of the source show how to do this. If you are doing this with publicly available fonts, please contribute them for the benefit of other users by sending the definitions to me by electronic mail. It is in your interests to do so — and to confine use of these four component-declaration commands to a single macro file — because a more elaborate syntax may be used in future to facilitate adjustment of positioning. The `\newarrow` command itself, however, will remain as it is.

The components `>`, `{>>}`, `<`, `{<<}`, `x`, `+`, `O` and `o` can be used both as heads (fifth position) and tails (first). The declaration `<----` defines `\rBackwards` to be a left-ward pointing arrow and `\lBackwards` to point right; this is not recommended, as you're certain to get confused. We shall see in the next section how to change the style of the four arrowhead components in a systematic way, thereby avoiding the ugly mixture of vee and LATEX arrowheads obtained using earlier versions of this package.

In some of the examples above (with the dot and rule fillers) the filler was repeated as the middle and one or both of the tips, whereas with others (dash) we used empty middle and tips instead. This was just to make the definitions easier to type: `\newarrow` treats middles and tips which agree with the fillers as empty ones. It is possible to use two different fillers, but this is not recommended.

The commands `\HorizontalMap` and `\VerticalMap` are still supported, but should be replaced by `\newarrow`. These ought only to have been used in macro files — not in the text of documents — to define four arrow commands together (right, left, up and down). The twenty arguments defining the five components of each of these four arrows can (if they're not already provided) be re-arranged as the four arguments of each of five `\newarrowhead`, *etc.,* declarations with the same effect. The mnemonics of the five components are then given to `\newarrow`. If you have any difficulty doing this conversion, please contact me.

8 Options

The `\diagram` or `\begin{diagram}` command and the arrow commands may be followed by a list of options in square brackets, for example

```
\begin{diagram}[size=2em,textflow]     \rTo[abut]
```

which apply to the diagram or arrow to which they are attached. Options may be given for an entire document, or within TEX's normal scoping rules, by a stand-alone command such as

```
\diagramstyle[centredisplay,PostScript=dvips]
```

You may use any number of `\diagramstyle` commands. In LATEX 2$_\varepsilon$ these options can be given when loading:

```
\usepackage[centredisplay,PostScript=dvips]{diagrams}
```

though you should avoid using macros in the option values.

Note: if the text of the first cell in a diagram begins with a square bracket, the program will attempt to read it as an option, just as square brackets within LATEX `array`s can cause trouble. This usually leads to a catastrophic error, which can be avoided using `\diagramstyle[nooptions]`. This problem doesn't arise in LATEX if you use `\begin{diagram}` followed by a (space or) new line before the text of the diagram.

Some options take a value, using an equals sign; those enclosed in square brackets below are optional, the others mandatory. Most of the options are appropriate for diagrams rather than arrows.

`abut` — Leave no gap between maps and objects; useful for ("Hasse") diagrams displaying order structures (page 17).

`alignlabels` — The labels on horizontal arrows remain centred in the cells where they are defined, despite different size endpoints; *cf.* `midshaft`. This is the default, and it is recommended for diagrams of a regular character. If there is insufficient space to position the labels in this way, they are centred (automatically and silently) instead. Whichever option is used, the middle of the arrow, if present, is always positioned with the labels.

`amstex` — Enable recognition of $\mathcal{AMS}$-TEX's commutative diagram commands; see section 14.

`balance` — The whole diagram is centred horizontally without regard to the left- and rightmost labels, so that its weight lies in the middle of the page. This is not the default, because if you were not aware of it the gaps would come as a surprise, but it is recommended.

`bottom` — When the diagram is placed alongside some simple text, the baseline of that text is aligned with that of the bottom row of the diagram.

`centre` *or* `center` — The diagram is centred[5] naïvely, using `nobalance` and `vcentre`; *cf.* `middle`.

`centredisplay` *or* `centerdisplay` — Disables `flushleft`, enables `balance` and gives a warning when used within `$$...$$` (see below).

[5]Notice that I have provided the "-er" alternative as a courtesy to American users which Donald Knuth and Leslie Lamport did not afford to British ones.

`dpi=`*resolution* — Anticipate the output resolution in pixels per inch (default 300). A fudge factor is applied to horizontal and vertical ruled lines, without which pixel rounding at 300dpi would cause the *bottom* rather than the middle of the horizontal rule to align with the cusp of the vee arrowheads: compare $\longrightarrow$ and $\longrightarrow$. Other similar adjustments may also be made to character positioning. Most office laser printers are 300dpi, but if you intend to use a printer of a different resolution for your final output, set `dpi=1270` or whatever — otherwise the shafts will be too low instead. The same applies if you intend to magnify or reduce; for example `dpi=212` should be used for two-up or A4-on-A5 final printing. In `plain` TeX you may set `\magnification` at the beginning of the file, but *do not* change `dpi` as this compensates automatically. The `nopixel` option disables the fudge factor.

`dpm=`*resolution* — As `dpi` but in dots per metre; *e.g.* 1270dpi=50000dpm.

`eqno=`*label* — Place *label* (in maths mode) beside the diagram, as an equation number.

`fixed` — Force diagonal arrows to be set on the first pass and not stretched.

`flushleft[=`*width*`]` — Display the diagram on the left of the page instead of the centre. If *width* is given, the leftmost vertical of the diagram is, if possible, aligned at that distance from the left edge of the page; *cf.* `fleqn.sty`. This is considered good book design by certain publishers. (This option is experimental: in future versions it may do something slightly different if the first column contains parallel verticals or none at all.)

`h=`*distance* — Same as `height`.

`hcentre` *or* `hcenter` — Same as `nobalance`.

`heads=`*name* — Set the `>` and `{>>}` arrow heads and tails to those defined by `\newarrowhead{`*name*`}` and `\newarrowtail{`*name*`}`.The following styles are currently available:

`LaTeX`			(default; uses `line10`)
`vee`			
`littlevee`			
`boldlittlevee`			(uses `\boldmath`)
`triangle`			
`o`			
`O`			
`X`			
`+`			
`curlyvee`			(uses AMS symbols)
`blacktriangle`			(uses AMS symbols)
`littleblack`			(uses AMS symbols)

The vertical `curlyvee` heads come from the AMS symbols, and the horizontals, from `cmsy` in the Computer Modern fonts, seem appear to be slightly mis-aligned with the `vee` heads from `cmmi`. If you use them but forget to `\usepackage{amssymb}`, you'll get an "`undefined control sequence`" error in the middle of a lot of garbage (deeply nested): hit return several times and carry on.

The `boldlittlevee` heads rely on LaTeX's `\boldmath` command, and default to `littlevee` in `plain` TeX. They may not work correctly if you use Leslie Lamport's old LaTeX2.09 font selection. Even if you use Frank Mittelbach and Rainer Schöpf's new one (which is part of LaTeX 2_ε) some PK files which are not in the standard distribution may be needed. It is only intended for final copy in the event that the `littlevee` appears too feint. The same applies to the `boldhook` tails.

`height=`*distance* — The distance between the baselines of successive rows in the diagram is as specified. Note that as objects and arrows commonly alternate, this is usually half the distance between one horizontal arrow and the next.

`hmiddle` — Same as `balance`.

`htriangleheight=`*distance* — Set `height`=*distance* and then `width` in such a way that the minimal 3×5-grid will make an equilateral $\triangleright$ triangle, and a 5×7-grid makes a regular hexagon (page 19). If the *distance* is not specified, the existing `height` is used and the `width` adjusted accordingly.

`htrianglewidth=`*distance* — Set `width`=*distance* and then `height` to make these figures.

`inline` — Use this option on individual diagrams which are being displayed alongside one another, for example in `$$`...`$$` or LaTeX's `center` or `displaymath` environments, when the global display option is `flushleft` or `centredisplay`.

`l>=`*distance* — Forces arrows (particularly horizontals and rotated diagonals) to have at least the specified length (default 2em), to avoid getting birds' feet instead of arrows: $A \succ B$. Sometimes this makes the arrow over-print an object or appear displaced; in this case you'll see an "over-full `\hbox`" or "increase cell width" error message.

`labelstyle=`*command* — Inserts this command in every label text; `\scriptstyle` is the commonest choice apart from the default.

`landscape` — If `PostScript` is enabled, rotate the entire diagram by 90° anticlockwise. Options which refer to the extreme rows and columns are modified accordingly. Useful for big diagrams with long objects or labels.

`large` — Same as `size=5em`.

`lefteqno` — Place the equation number, as given by `eqno`, on the left of the diagram.

`leftshortfall=`*distance* — The gap between the arrow (to which it is applied individually) and the object on its left is as specified.

`lowershortfall=`*distance* — Similarly below.

`loose` — The rows and columns of the diagram have *at least* the `height` and `width` specified, but may stretch, in the same way as those of an array or table. This is the default, because the results are more of less right in most circumstances, but this can result in gaps in the diagram, so it is recommended that `tight` be used in the final version of a document, with appropriate manual adjustment of the size of the grid.

`middle` — The diagram is centred both horizontally using `balance` and vertically using `vmiddle`, *q.v.*

`midhshaft` *or* `midshaft` — Labels are centred in the shafts of horizontal arrows; *cf.* `alignlabels`. This happens automatically for in-text or one-line diagrams, or if the arrow is too short to align the labels.

`midvshaft` — The same for vertical arrows: not recommended.

`moreoptions` — Allow another list of options in square brackets. This is intended for macros of the form `\def\funnydiagram{\diagram[`*options*`,moreoptions]}` to allow `\funnydiagram[`*options*`]`. The arrow commands always allow any number of lists of options.

`nobalance` — Horizontal centring of diagrams is done with respect to the extremities.

`nohcheck` — Disables certain error checking.

`nooptions` — Used in `\diagramstyle`, this disables the parsing of options. This is for reverse compatibility in the case where several diagrams begin with a square bracket.

`noorigin` — Disables `origin` and `balance`.

`nopixel` — Disable the `dpi` fudge factor, *q.v.*, by setting a very high resolution.

`noPostScript` — Disables the use of embedded PostScript *and* the `PS` and `noPS` options. The value, if any, is ignored, so you need only edit in/out the prefix `no` whilst leaving the choice of PostScript translator intact.

`noPS` — Disable PostScript on individual maps and diagrams.

`notextflow` — Disables `textflow`. This is needed on some individual diagrams (where they appear as nouns in a sentence, for instance) when you use `\diagramstyle[textflow]`.

`noTPIC` — Disable the use of TPIC `\special`s.

`objectstyle=`*command* — Inserts this command in every object text; `\scriptstyle` is the commonest choice apart from the default.

`origin` — Makes the width, height and depth of the whole diagram zero, locating it at the baseline of the bottom row, in the centre of the leftmost column. This allows it to be positioned by the user; useful for mixing diagrams with LaTeX pictures and other graphics.

`p=`*distance* *or* `pilespacing=`*distance* — Set the distance between parallel verticals (in the same cell); that between the rows of a `\pile` is half of this to allow things to be put between parallel horizontals.

`pixelsize=`*distance* — Anticipate this output resolution; `pixelsize=.02true mm` is the same as `dpi=1270` or `dpm=50000`.

`portrait` — Disable the `landscape` option, *q.v.*

`PostScript=`*author* — Enable the use of embedded Adobe PostScript `\special` commands in a form supported by *author*'s translator. The following are currently recognised by the *author*s' surnames, their programs or their companies. The surname is the preferable form, since the programs have very similar names. The master FTP source is also given.

- (Tomas) `Rokicki`, `dvips`, `RadicalEye`: labrea.stanford.edu /pub/dvips9999.tar.Z

- (Stephan) `Bechtolsheim`, `dvitps`, `IntegratedComputerSystems`: arthur.cs.purdue.edu /pub/TeXPS-9.99.tar.Z
- (James) `Clark`, `dvitops`

These are currently used to implement diagonals by rotating horizontals. It is not possible to use Andrew Trevorrow's OzTEX.

`PS` — Suppose you want to use PostScript for some but not all maps and diagrams in the final version. Then use `PS` for each of them but put `\diagramstyle{noPostScript}` in the preamble during drafting.

`righteqno` — Put the equation number (`eqno`), if any, on the right.

`rightshortfall=`*distance* — Analogous to `leftshortfall`.

`s=`*distance* — Same as `size`.

`scriptlabels` — Labels on maps are set in `\scriptstyle`; useful if they're rather long.

`shortfall=`*distance* — Set the gap between arrows and the objects to which they point.

`size=`*distance* — Set both `height=`*distance* and `width=`*distance*.

Most geometrical problems with diagrams can be solved by enlarging the cells. Please try this and the `tight` option before contacting me.

`small` — Same as `size=2em`.

`textflow` — The text which follows the diagram in the TEX source is brought up to fill out the line preceding it; useful to avoid the "such that the diagram ♣ commutes" cliché. This has been used for several diagrams in this manual.

`thick[=`*breadth*`]` — As LATEX `\thicklines`, optionally setting the width of orthogonal ruled lines. The LATEX command itself has no effect within diagrams. The default rule breadth is twice that for the `thin` option.

`thin[=`*breadth*`]` — This is similarly analogous to `\thinlines`.The default rule breadth is the same as that used by TEX, namely `\fontdimen8\textfont3`.

`tight` — Force all of the cells in the grid to have exactly the size you specify. This is recommended for the final version of a document, but is not the default because it may cause over-printing, which requires the intervention of the user to cure (by setting `height` and `width`); *cf.* `loose`.

`top` — Analogous to `bottom`, except that the top row is used.

`TPIC` — Use TPIC `\special` commands instead of LATEX line segments to draw diagonal lines. These are supported by Textures, Tomas Rokicki's `dvips` and Paul Vojta's `xdvi`.

`uppershortfall=`*distance* — Analogous to `leftshortfall`, except above.

`vcentre` *or* `vcenter` — Vertical positioning is the same as with TEX's `\vcenter` command or LATEX's `[c]` option. However `vmiddle` is what is usually wanted.

`vtriangleheight=`*distance* — Set `height=`*distance* and then `width` in such a way that the minimal 5×3-grid will make an equilateral $\triangle$ triangle and a 9×5-grid makes a regular hexagon (page 19).

`vtrianglewidth=`*distance* — Set `width=`*distance* and then `height` to make these figures.

`vmiddle` — If a diagram with an odd number of rows of constant height (the usual case) is placed alongside some simple text, the baseline of the text will be aligned with that of the middle row, irrespectively of the height and depth of the labels on the top and bottom rows. In general, the baseline of the diagram is half-way between those of the top and bottom rows.

`w=`*distance or* `width=`*distance* — Set the distance between the centre of one column and the next to the right.

9 Application to adjunctions

Here is a side application of commutative diagrams to displaying "adjoint correspondences." It also illustrates the way arrows stretch to meet their endpoints but keep their labels centred in the column of arrows. The options are chosen to avoid getting lots of extra space around the cells, which in this case we don't want. Notice also the invisible "objects" terminating the `\hLine` "morphism" command.

```
\begin{diagram} [loose,height=.8em,width=0pt]
            & Z\times X &       \rTo^f          &  Y  \\
         \  &           &       \hLine          &    & \  \\
            &     Z     & \rTo^{{\bf curry}(f)} & Y^X \\
\end{diagram}
```

$$\frac{Z \times X \xrightarrow{\;f\;} Y}{Z \xrightarrow{\mathbf{curry}(f)} Y^X}$$

If you want the Zs lined up, it's no good `\hfill`ing the second one on the right, because the arrow wont stretch to meet it[6]. The best solution is probably to split the object into two cells:

```
\begin{diagram} [loose,height=.8em,width=0pt,l>=3em,midshaft]
          & Z & {}\times X &       \rTo^f          &  Y  \\
        \ &   &            &       \hLine          &    & \  \\
          & Z &            & \rTo^{{\bf curry}(f)} & Y^X \\
\end{diagram}
```

$$\frac{Z \times X \xrightarrow{\;f\;} Y}{Z \xrightarrow{\mathbf{curry}(f)} Y^X}$$

The `{}` makes sure `\times` still gets the spacing appropriate to a binary operator.

[6] There is no way of detecting *where* glue is placed inside a box, so the stretching algorithm *assumes* the object is centred and extends the arrow by half the amount of glue in the object box.

10 Diagonal lines

Whereas horizontal and vertical arrows such as `\rTo` and `\dTo` stretch to meet their endpoints without your telling them where those are, the diagonals (`\ruTo`) need to be told explicitly if you want to use them for anything other than to go across a 3×3 square.

The approximate direction of the arrow stays the same — right/left and up/down — as given by its name. The length (and exact direction) are given by *positive* coordinates. As with LaTeX's `\lines` and `\vectors`, these are given pairs of (small) positive integers in round brackets, except that there is no restriction on the values.

However these numbers count the *cells,* horizontally and vertically, through which the arrow passes, rather than the absolute distance. This means[7] that if the `height` of the rows and the `width` of the columns are different, `\ruTo(2,2)` is no longer a 45° slope.

The names of the arrow commands specify the horizontal direction first, as a reminder that this is also the convention for coordinates.

Another difference from the orthogonal arrows is the rule for where to place the command. Somewhat arbitrarily, this is

> in the row *below the upper endpoint* (irrespectively of the direction of the arrowheads, if any), and in the adjoining column

In the commonest case (the diagonal of a small square) this means the middle of the square; more generally for positive gradients (`\ruTo` and `\ldTo`) the arrow command goes below and to the left of the upper endpoint, and for negative gradients (`\rdTo` and `\luTo`) below and to the right.

For example, here is the diagram which defines a pullback. The corner symbol is a macro (`\SEpbk`) placed in the same cell as the object, but which prints way outside it.

```
   \begin{diagram}
U &                                      &                 &         &        \\
  & \rdTo~{(x,y)}\rdTo(4,2)^x\rdTo(2,4)_y &                 &         &        \\
  &                                      & X\times_Z\SEpbk Y & \rTo_p & X      \\
  &                                      &     \dTo^q      &         & \dTo_f \\
  &                                      &       Y         & \rTo^g  & Z      \\
   \end{diagram}
```

$$\begin{tikzcd}
U \arrow[dr, "(x,y)"] \arrow[drr, "x"] \arrow[ddr, "y"'] & & \\
& X \times_Z Y \arrow[r, "p"'] \arrow[d, "q"'] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d, "f"] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}$$

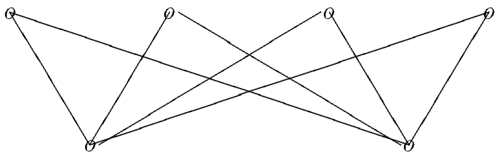
For reverse compatibility, some commands with "compass" names are provided, such as `\NW`, `\WNW`, `\NNNW`, together with dotted and other variants.

[7] This represents a change from version 3, in which they were required to be the same for the diagonals to work. The coordinates are now converted internally into distances, and then back into a rational number, and the appropriate LaTeX arrow characters chosen. The macros `\laf`, `\lah`, *etc.* and their octal arguments are now obsolete.

However *the auxiliary file* `extra-diagonals.tex` *is obsolete and must not be used.*

The main use of diagonals other than the simplest ones is for illustrating order structures ("Hasse diagrams"). Posets are trivial categories, but in practice they're the most difficult diagrams to draw!

```
\begin{diagram}[height=2.5em,width=1.5em,abut]
o&&&&o&&&&o&&&&o\\
&\luLine\luLine(10,2)&&\ruLine&&\relax\luLine(6,2)&&
\ruLine(6,2)&&\luLine&&\ruLine\ruLine(10,2)&\\
&&o&&&&&&&&o&
\end{diagram}
```



The obsolete command `\DiagonalMap` is also still supported, but should be replaced by the new commands. If you have used this with fonts other than `line10` (for example Michael Spivak's fonts `lams1` to `lams5`), please contact me to assist with conversion.

By default, the diagonal arrows are set on the first pass using LaTeX line segments. By contrast, the horizontals and verticals use ruled lines wherever possible and are adjusted on a second pass of the diagram (within the `\enddiagram` command) to meet their endpoints.

When the diagonals are set during the first pass, everything happens within the cell where they are declared, where the algorithm has no access to information about the size (and exact whereabouts) of the objects at which the arrows are meant to point. Therefore the lengths of the diagonals are chosen somewhat arbitrarily, though the `abut` option will make them touch a `\cdot`.

To get better results we have to use more sophisticated methods, and go outside what is strictly "standard" TeX.

11 Alternative TEXnology

In order to get the best results, you should be aware of the way in which diagonals are constructed. TeX is ultimately only capable of positioning characters from various fonts and drawing black rectangles with horizontal and vertical sides: there is no primitive for diagonal lines or for rotation. This means that we have

- either to juxtapose characters with a limited choice of angles, as LaTeX's `\line` and `\vector` commands do, with the result that (because of pixel rounding errors) they may not line up correctly,
- or to use the loophole in the DVI language provided by the `\special` command to tell some post-processing software (such as `dvips`) to do the work for us, thereby surrendering portability,
- or to over-print a large number of dots, making the DVI file very large since at least 12 bytes are needed for each dot. This is what PICTeX does.

In this package options are provided to employ the first two of these methods.

- By default, characters from LaTeX's[8] `line10` font are used. Arrowheads from this font are used irrespectively of those chosen by the `heads` option. The nearest[9] available LaTeX slope is chosen, with the effect that arrows may sometimes fall short or overshoot: `height` and `width` should be set with this in mind.

- If the `PostScript` option is enabled, horizontal arrows are used, and `\special` commands[10] are embedded in the DVI file which have the effect of rotating them. If the DVI file is post-processed by a program which does not understand Adobe PostScript an error message will probably be generated and the arrows will appear horizontal.

  Currently the labels are rotated with the arrows; later there will be an option either to do this or to set them horizontally.

  Literal PostScript commands are included in the DVI file using TeX's `\special` command. The various DVI to PostScript translators (page 13) make use of this in different ways, so it is necessary to specify which one you are using. At Imperial College we use Tomas Rokicki's `dvips`; if the program you use is not listed, please consult its manual for the appropriate way of including literal PostScript commands and tell me the result so that I can add this as an option. The rotation code should be independent of the design of the translator.

  It does not and cannot work with the OzTeX PostScript translator because this sorts the characters on each page by font and cannot deal with `\special` commands.

- If the `TPIC`[11] option is enabled, a simpler set of `\special` commands is used to draw diagonal lines, but rotation is not available. `dvips` and `xdvi` understand these, but other post-processors may give error messages and print nothing.

No additional macro packages or special knowledge are needed to make use of these methods: all of the necessary code is built in to the diagrams package.

Future versions may exploit these extensions to provide other features, such as curved lines; indeed this is likely to be the main direction of further development. Other methods may also be provided. However

- if no extension option is selected a standard DVI file (using the `line10` font) will be always generated, giving an approximation to the features requested.

- development of new features will use `PostScript` mode, without which a "second class service" will be provided.

---

[8] The code is similar to that underlying the LaTeX commands, but is not the same. In particular, the line segment characters are equally spaced, instead of a single overlap between the last two.

[9] The approximation uses continued fractions, *alias* Euclid's algorithm, with a correction. There is also code for computing Pythagorean sums (for the lengths of rotated diagonals) and trigonometric functions (currently unused).

[10] The code is adapted from Tomas Rokicki's `rotate.sty`.

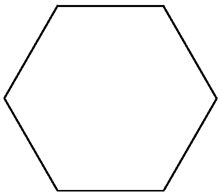
[11] TPIC is a graphics preprocessor for TeX by Tim Morgan which uses this set of `\special` commands. The same ones were also adopted byDirk Grunwald for `texpictex` and by Conrad Kwok for `eepic`, which are LaTeX macro packages for graphics.

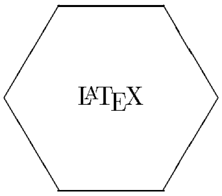
12 Regular figures

The difference between these options can be seen by trying to draw a regular hexagon.

```
\begin{diagram} [vtrianglewidth=1em,abut,tight]
           &         & {} &  & \rLine &  & {} \\
           & \ruLine &    &  &        &  &    & \luLine \\
        {} &         &    &  &        &  &    &         & {} \\
           & \luLine &    &  &        &  &    & \ruLine \\
           &         & {} &  & \rLine &  & {} \\
\end{diagram}
```

The PostScript and TPIC options will make it meet up exactly, whereas the default LaTeX method does not — though it is surprisingly accurate!





The vtrianglewidth option and its variants set the width and height of the cells, but the array retains the symmetry of a rectangle. Tesselations of triangles and hexagons may be drawn in this way.

Drawing a regular pentagon requires rows and columns of different sizes. It is envisaged that only a few special ways of doing this will be used, so this feature is provided by a *declaration*[12] of the form

```
\newdiagramgrid{pentagon}%
   {0.618034,0.618034,1,1,1,1,0.618034,0.618034}%
   {1.17557,1.17557,1.902113,1.902113}
```

(In fact this is already made within the source of the package.) The numbers in the first list specify the distances between the centres of the first and second columns, the second and third, and so on, and similarly those in the second list the distances between the baselines of the rows. The units are the specified cell height and width.

Once declared, the grid is chosen by name using the obvious option. Note that this forces the tight option.

```
\begin{verbdiag}[grid=pentagon,PostScript]
  &&&&(A*B)*(C*D)& \\
  &&&\ruTo(4,2)^\Assl&&\rdTo(4,2)^\Assl&\\
  A*(B*(C*D))&&&&&&&&((A*B)*C)*D&\\
  &\rdTo_{\Id*\Assl}&&&&&&\ruTo_{\Assl*\Id}&\\
  &&A*((B*C)*D)&&\rTo^\Assl&&(A*(B*C))*D&
\end{verbdiag}
```

[12] I cannot think of a natural way of specifying the distances without coding general Gaussian elimination. I'm quite happy to accept angles expressed in degrees.

$$
\begin{tikzcd}
 & & (A*B)*(C*D) \arrow[drr, "\mathrm{assl}"] & & \\
A*(B*(C*D)) \arrow[urr, "\mathrm{assl}"] \arrow[dr, "\mathrm{id}*\mathrm{assl}"'] & & & & ((A*B)*C)*D \\
 & A*((B*C)*D) \arrow[rr, "\mathrm{assl}"] & & (A*(B*C))*D \arrow[ur, "\mathrm{assl}*\mathrm{id}"'] &
\end{tikzcd}
$$

A similar grid for a pentagon with a vertical side is obtained by the `gridx` option.

The grid `perspective` has this effect on the cube on page 6:

$$
\begin{tikzcd}
A' \arrow[rr, "f'"] \arrow[dr, "a"] \arrow[dd, "g'"'] & & B' \arrow[dr, "b"] \arrow[dd, "h'"'] & \\
 & A \arrow[rr, "f"] \arrow[dd, "g"'] & & B \arrow[dd, "h"] \\
C' \arrow[rr, "k'"] \arrow[dr, "c"'] & & D' \arrow[dr, "d"] & \\
 & C \arrow[rr, "k"] & & D
\end{tikzcd}
$$

Without the `PostScript` option the diagonal arrows will not be set correctly.

13 Emulation of TEXercise 18.46

This and the next section describe how to convert diagrams in your existing documents which were drawn using other commutative diagram macro packages to use this one, wherever possible changing only the preamble or macro file and not the text itself.

Exercise 18.46 of *The TEXbook* provided some ideas for commutative diagrams, from which the present package was originally developed. The following instructions are based on what is given there, but if you have added other arrow macros you will have to work out how to re-define them by following the examples given.

If you have used the *TEXbook* macros as they stand — with `\matrix` enclosing the diagram — you will first have to distinguish between the diagrams and real matrices. (If you give arrays of numbers to the commutative diagrams package, they will come out rather widely spaced.) This process is unnecessary if you already have a macro for diagrams rather than matrices: just re-define that in a similar way.

In your macro file you probably have something like what follows the `\iffalse` command below, but with additional macros written in a similar way (this is the reason for setting it all out in gory detail in the manual rather than providing an extra input file as in the case of some of the "packages" described in the next section). If you make the following additions you will be able to revert to the original (in the unlikely event that you are not satisfied) by changing this to `\iftrue`.

This is not particularly difficult: when, as a TEX novice, I wrote my Ph.D. thesis in August 1986, I had sixty diagrams drawn using the *TEXbook* macros, together with several others that had to be drawn by hand. Recovering that from an archive tape, I recently found that all but five of the sixty could be converted without any change at all to the text, whilst the hand-drawn ones can now be drawn with the up-to-date package.

```
\iffalse
  % your macros, copied or adapted from TeXercise 18.46 (page 325)
  \def\mapright#1{\smash{\mathop{\longrightarrow}\limits^{#1}}}
 \def\mapdown#1{\Big\downarrow\rlap{$\vcenter{\hbox{$\scriptstyle#1$}}$}}
  %
  \let\cdmatrix\matrix % revert to using \matrix for diagrams
\else
  % Replacement for the above using ...
  \input diagrams % Paul Taylor's diagrams.
%
  \def\mapright#1{\global\matrixwascdtrue\rTo^{#1}\relax}
  \def\mapleft #1{\global\matrixwascdtrue\lTo^{#1}\relax}
  \def\mapup   #1{\global\matrixwascdtrue\uTo>{#1}\relax}
  \def\mapdown #1{\global\matrixwascdtrue\dTo>{#1}\relax}
%
  % Maybe you have some variants like this:
  \def\mapupbefore#1{\global\matrixwascdtrue\uTo<{#1}\relax}
%
  % The following examples may also be useful:
  \let\into\rInto
  \def\horizadjoint#1#2{\pile{\lTo^{#1}\\\bot\\\rTo_{#2}}}
  \def\vertadjoint #1#2{\dTo<{#1}\dashv\uTo>{#2}}
%
  % Here is a replacement for \matrix which gives a commutative
  % diagram, including some extra macros for use inside them.
  \def\cdmatrix{\bgroup
    \edef\matrixlineno{\the\inputlineno}\global\matrixwascdfalse
    %
    \let\matrix\pile   % inner \matrix is probably parallel horizontals
    %
    % re-define the diagonal arrows
   \let\nwarrow\luTo\let\nearrow\ruTo\let\swarrow\ldTo\let\searrow\rdTo
    %
    \diagram[]% begin the diagram (without options)
    \getthematrix% delete this line if you use amstex rather than plain
    }
  \def\getthematrix#1{#1\endcdmatrix}
  \newif\ifmatrixwascd
  \def\endcdmatrix{\enddiagram\egroup}
%
  % The following tells you whether it was actually a diagram or
  % a matrix. Remove this when you've changed those \matrix
  % commands in your text which should be diagrams to \cdmatrix.
  \def\endcdmatrix{\enddiagram\egroup
    \expandafter\message{^^JThe \string\matrix\space at lines
      \matrixlineno--\the\inputlineno\space was really a
      \ifmatrixwascd diagram\else matrix\fi.^^J}}
  \let\matrix\cdmatrix \let\endmatrix\endcdmatrix
%
```

```
\fi
```

The code as shown above assumes you have TEX version 3 (to provide `\inputlineno`); it prints a `\message` to tell you whether each `\matrix` has used any arrow commands. With a clever choice of this message you can even get your editor to make the necessary changes to the text for you in batch mode! Delete the extra code after you have done this.

Beware that AMS-TEX uses `\matrix#1\endmatrix` instead of `\matrix#1`. In this case, delete `\getthematrix` as indicated. Then (where appropriate) change `\endmatrix` as well as `\matrix`. The same applies if you have used the `array` enviroment in LATEX.

14 Emulation of other macro packages

There are several other TEX macro packages in circulation for drawing commutative diagrams, of varying degrees of sophistication. This section describes how to adapt the preamble of an existing document which was written to use such macros so that it prints diagrams like those in this manual instead. When you publish documents prepared in this fashion, please remember to acknowledge the authors of both packages, making it clear which you used to type the original source and which produced the finished product.

American Mathematical Society (AMS-TEX, and AMS-LATEX's `amscd.sty`): see Michael Spivak below.

Michael Barr's `catmac` macros were based on a principle of overlapping squares. Whilst this is perhaps closer conceptually to the categorical ideas which are being expressed, it is not possible to emulate the language using the matrix syntax. The simple shape macros can, however, be replaced by

```
\def\square[#1`#2`#3`#4;#5`#6`#7`#8]{%
   \diagram[]
      {#1} & \rTo^{#5} & {#2} \\
      \dTo<{#6} & & \dTo>{#7} \\
      {#3} & \rTo^{#8} & {#4}
   \enddiagram}
```

and similarly `\atriangle`, `\btriangle`, `\dtriangle`, `\ptriangle`, `\qtriangle`, `\Atriangle`, `\Ctriangle`, `\Driangle`, `\Vriangle`, `\Arianglepair`, `\Vrianglepair` and `\recurse`, which are easy exercises.

Karl Berry's `eplain`: see Steven Smith below.

Francis Borceux's `diagram` package uses LATEX's `picture` environment instead of TEX arrays, but since it is designed in an array fashion it can be interpreted. Instead of that file, use

```
\input diagrams
\input Borceux-to-Taylor
```

This is available from the same place as (my) `diagrams.tex` itself. Currently the curved, free and multiple arrows are not implemented, and all size parameters are ignored.

Eitan Gurari's `dratex`: I haven't looked in to this package yet.

Donald Knuth: see section 13 of this manual.

Frank Mittelbach's $\mathcal{A}_\mathcal{M}\mathcal{S}$-LaTeX `amscd.sty`: see Michael Spivak below.

John Reynolds: I haven't looked in to this package yet.

Kris Rose's XY-pic uses a different convention for where to declare arrows, namely in the cells with their sources. This convention will be supported in the next release. The syntax of Rose's arrow commands is also more complicated. I do not intend to support his fonts or turning commands.

Rainer Schöpf's $\mathcal{A}_\mathcal{M}\mathcal{S}$-LaTeX `amscd.sty`: see Michael Spivak below.

Steven Smith: I haven't looked in to this package yet.

Michael Spivak's LAMS-TeX:similar comments apply here as to Kris Rose's package.

Michael Spivak's $\mathcal{A}_\mathcal{M}\mathcal{S}$-TeX (as described in *The Joy of TeX*) had some very primitive commutative diagrams,enclosed in `\CD...\endCD`.

You can obtain a dramatic improvement in these diagrams without changing the text of your document by using my package with the `amstex` option. Given a (complete, correct) $\mathcal{A}_\mathcal{M}\mathcal{S}$-TeX document, change the header to read

```
\input amstex
\input diagrams
\diagramstyle[amstex]
```

If $\mathcal{A}_\mathcal{M}\mathcal{S}$-TeX or $\mathcal{A}_\mathcal{M}\mathcal{S}$-LaTeX (`amscd.sty`) had been loaded, this will replace the definitions.

You can also include $\mathcal{A}_\mathcal{M}\mathcal{S}$-TeX diagrams extracted from old papers in new `plain` TeX or LaTeX documents. In this case, do not load $\mathcal{A}_\mathcal{M}\mathcal{S}$-TeX (unless you want it for some other reason) but instead use

```
\input diagrams
\def\CD{\diagram[amstex]}
```

to confine the changes to the meaning of the `@` character to the diagrams themselves. The in-line horizontal arrows will then not be interpreted.

Timothy van Zandt's `pstricks`: I haven't looked in to this package yet.

15 Frequently asked questions

Wouldn't it be better to draw it with a mouse? No.

Is it compatible with ...? From 1986 until after version 3 of this package was announced in July 1990 I used `plain` TeX, whilst giving assistance to colleagues who used LaTeX. In December 1991 the local TeX system was converted to use Frank Mittelbach and Rainer Schöpf's font selection (NFSS1) for LaTeX. Now we use LaTeX 2_ε, which includes NFSS2.

Consequently there is a great deal of collective experience in using the diagrams package in all of these environments. I do not have experience of AMS-TeX, AMS-LaTeX, `eplain` or commercial TeX packages, but do not know of any reason why it should not work with them: please tell me if you find any difficulties.

It is, regrattably, not uncommon for publishers to copy parts of LaTeX and article.sty into their own journal and conference styles, and not keep these up to date with respect to bug-fixes and other changes. Many of the bug reports which I receive in connection with the diagrams package are in fact traceable to errors of this kind in publishers' styles. My policy is that I will no longer fix bugs in software from commercial organisations for free, but suggest some consultancy arrangement. If, however, it is possible to make the package more robust and circumvent such bugs in other software then I will do this.

Richard Stallman's `texinfo` is designed for documenting other software, in which TeX's special characters often have important meanings. For this reason many of the `\catcode`s have been changed, and in particular `@` is used where TeX uses `\`. The usual TeX meanings are restored within `@tex..\Etex`. It is possible to load this package without this, but you must do

```
@catcode`@\=0 \catcode`\%=14 \input diagrams \catcode`\%=12 \catcode`\\=13
```

You must also do ``@catcode`@&=4`` before using `@diagram`. Braces `{}` stay the same.

Does it use any special fonts? No. One of the design criteria of the package is that all of the components come from the standard Computer Modern fonts that come with TeX, except that the diagonal arrowhead characters come from LaTeX's `line10` font. As one user said,

> "I agree with you in being against the use of additional fonts. It takes some time and experience to port, say, LAMS-TeX fonts to [my 'personal' computer]. I would much prefer embedded PostScript commands. Custom DVI drivers such as `xdvi`, on the other hand, are not widely available, at least not for [my computer]."

There are *optional* arrowheads from the AMS symbols fonts, which were used when this copy of the manual was printed. They may be obtained by anonymous FTP from `e-math.ams.com`.

It is sometimes claimed that it is advantageous to have specially designed fonts in order to ensure that the components match up correctly. However the reason why they frequently do not is pixel-rounding, *even when the DVI-driver does this correctly according to the rules specified by Donald Knuth.*[13] The `dpi` option has been included to correct for this.

If you have other arrowhead fonts available, such as those provided by Kris Rose for XY-pic and Michael Spivak for LAMS-TeX, you may use them if you write your own `\newarrowhead` commands.

[13] If you don't believe me, calculate for yourself which pixels are inked when a 0.4pt-wide rule is centered on the maths axis of `cmmi10` (2.5pt above the baseline) at 300dpi (1 point = 4.15 pixels), and compare this with characters such as $>$ in the output of `gftype -i cmmi10.300gf`. The character $\succ$ in `cmsy` is one pixel lower than this.

How do I get it? It is easiest to get this package by anonymous FTP. This stands for "file transfer protocol"; it is a way in which you can log on interactively to my computer and fetch (some of) my files. A very large amount of software is now freely available by this method, so it is well worth putting pressure on your system administrator to get your machines connected.

The "`archie`" utility is extremely useful as a way of locating software and other information on the Internet. You just give it (part of) the name of the file you want (*e.g.* "diagram") and it will tell you where files of that name are located in numerous public FTP archives around the world. Beware, however, that many of them are just "magpies' nests" — they collect things from other places without verifying their authenticity or keeping them up to date.

Corruption by electronic mail. If you cannot use FTP and the route to you by electronic mail passes through non-ASCII machines (such as through BITNET), the file may suffer corruption. This doesn't matter very much with the manual, but the macro package itself has a list of characters at the top, and use of most of those which experience has shown to be vulnerable has been confined to the first section, wherever possible. Search for the word ASCII if in doubt.

TEX capacity exceeded. TEX was designed in the early 1980s, when RAM was measured in kilobytes, and does not have dynamic memory allocation. Although `tex.web` says loudly "Don't Touch!", the compile-time parameters listed on page 300 of *The TEXbook are* meant to be changeable, and Donald Knuth provided a mechanism (`tex.ch`) for doing so. Karl Berry's widely used `web-to-C` UNIX port of TEX increases them substantially — in accordance with the RAM available in 1990s hardware and that needed by 1990s software. There is nothing UNIX-specific about these changes: they can just as well be made in any other compilation environment. Currently it seems OzTEX has not made the changes — please complain about this to its author, Andrew Trevorrow, not me.

The increase in size over version 3 is largely due to the diagnostics, *i.e.* helping you to use the package!

If you get "no room for another `\dimen`" it probably means you're using PICTEX, which uses 110 out of the available 256; I use 20.

The process of stretching the arrows in a completed diagram is quite slow, but at least in this package TEX spends the time doing something useful rather than parsing (LATEX `picture`) `\put` commands inside macros. The emulation of Francis Borceux's macros, for instance, is 50% faster than the original. Kris Rose's XY-pic takes about six times as long to draw the same diagram.

Default arrowheads. Even when you define your own arrows with special arrowheads, you still get get LATEX heads on diagonals. This is because LATEX heads are available in a variety of directions, but your special ones are only defined for up, down, left and right. To get special heads on diagonals, it is necessary to rotate them, and to do that you need the `PostScript` option.

Display options. If, as was advised in earlier versions of this manual, you enclose the diagram in `$$...$$` or a , the new options in section 8 for positioning it cannot work. Except in those cases where you want two or more diagrams side by side, or a small diagram in-line in the text, it is better to remove such enclosings; then you can experiment with the options using `\diagramstyle`. LATEX's `center` environment is, for this purpose, *not* a display but a paragraph of text, with some strange results if you use the `textflow` option; in this case you should use the `inline` option on the individual diagrams to make them appear side by side.

Large gaps between diagonals and their endpoints. This happens if you try to use the default `LaTeX` line segments to draw very steep or very shallow diagonals. Try using the `PostScript` option.

By default, the (rows and) columns of the matrix forming the diagram can stretch to accommodate long pieces of text as objects and arrow labels. This is appropriate for rectangular diagrams and requires no user intervention. If, however, you have diagonals, this stretching causes them to fall short of the objects to which they are meant to point, because currently they are (unlike the horizontals and verticals) rigid.

$$\begin{tikzcd} A \arrow[r, "< 1_A , 0_A >"] \arrow[dr, "f"'] & A \times N \arrow[d, dotted, "h"] \\ & B \end{tikzcd}$$

An example provided by a user is shown. The solution is to use options like

```
\begin{diagram}[tight,width=4em,height=3em]
```

See section 8 for details. Sometimes there isn't room on the page to increase the width sufficiently. Try using the `scriptlabels` option, or, failing that, put the whole diagram in the scope of a LaTeX `\small` or even `\tiny` declaration.

This real-life example illustrates another common problem: the symbols $<$ and $>$ are *relations* $\langle$not brackets$\rangle$ — and it's not unusual to see line-breaks the wrong side of them, even in published, supposedly proof-read, books. If you don't often use the strict arithmetical relations and find it a bore to type `\langle..\rangle`, put the following in your macro file:

```
\mathcode`\<="4268     % < = \langle
\mathcode`\>="5269     % > = \rangle
\mathchardef\gt="313E  % arithmetic
\mathchardef\lt="313C  % strict order
```

As another piece of general advice, many people use `\mbox` when it is completely unnecessary. Amongst other things, it inhibits the reduction of the contents when used as a sub- or superscript. In LaTeX 2_ε, try using the `\text` and `\ensuremath` commands.

"Badly drawn diagonals" error message. This warning is given if (a) you use diagonals which are set on the first pass (*i.e.* LaTeX, TPIC or `fixed`) *and* (b) some of the columns are significantly wider than was specified by the `width` or `size` option. It indicates that the problem above, with big gaps at the ends of diagonals, may have occurred. Check this, and if necessary set the `tight` option and specify the necessary (increased) `width` yourself. Alternatively, use the `PostScript` option to get the diagonals set on the second pass; then they will meet their endpoints.

Labels on arrows over-print objects or arrowheads become detached. You're trying to squeeze too much into the column: increase `width` (as it tells you to do). If the object at one end is much longer than that at the other, the `midshaft` option may be appropriate.

Mixed or missing arrowheads. The idea of providing the `heads` option is that you should specify at the beginning of your document which style you want. The default is `heads=LaTeX` since this provides consistency between orthogonal and diagonal arrowheads when rotation is not available.

Undefined symbols. To use the curly and black arrowheads you need the AMS symbols fonts; if one of these (`\curlyvee`, `\blacktriangleleft`) is undefined, hit return several times to complete the run of TeX on your document, then go back and insert `\usepackage{amssymb}` or change the heads declaration.

If you find a symbol with a meaningless name like `\CD@gF` or `\cD@hA` is undefined, it means that you have used internal macros from a previous version of the diagrams package. Please remove them: the names are deliberately meaningless to persuade you not to use them.

Can you put diagrams within diagrams? Yes, but it's not often that such things are needed. Remember that `\pile` is used for parallel horizontals. Sometimes you may want an array of diagrams.

During development I found problems when the first cell of the inner diagram was empty, and version 3.22 resulted in TeX's elusive "interwoven alignment preambles are not allowed" error (*The TeXbook*, page 299). Although I believe these bugs have now been fixed, this is a delicate area and it is possible that others may arise.

`Missing \endcsname inserted.` This can happen if a macro occurs within an option [in square brackets]. In general you must ensure that such macros expand to text characters only. For example in LaTeX 2_ε:

```
\usepackage[flushleft=\mainindent]{diagrams}
```

where `\mainindent` is a `\dimen`; in this case `\the\mainindent` will work, because `\the` expands the `\dimen` to its value (*TeXbook*, page 214). Values for options to individual diagrams and arrows, or parsed by `\diagramstyle` instead of `\usepackage`, may safely contain macros where appropriate.

Horizontal arrows overprint objects. You must not use `\hfill`, `\hss`, `\hspace`, `\hidewidth` or other similar commands to try to alter the effective size of the object.

What if it still doesn't work? If you have a problem which is not answered by this manual, please compile a *short* file containing your problem diagram and any macros (such as `\Assl` in section 3) it contains. Run it through `tex` or `latex` to check that no definitions are missing, and include a note of the date and version number which you are using. Then send it by *electronic* mail: please do not use the telephone or postal mail.

I am keen to know about any adverse interactions with other software, anything which is not well explained in this manual, or any cases of mis-typing in which `\scrollmode` does not get to the end of the document because of a diagrams error.

16 Conditions of use

You may freely copy and pass on this package and include it in collections of free software, but may not alter it or charge a fee for it.

Please ensure that you are registered with me as a user, so that you can be informed of new versions. Any electronic mail message containing the words "commutative" or "diagram" automatically registers you, as does quoting your electronic mail address when fetching it by FTP from `theory.doc.ic.ac.uk`.

If you consider this package good enough to use, then it is good enough to acknowledge. After all, it is academic protocol to credit prior or simultaneous discovery of techniques related to your own, even if you were unaware of them or did not rely on them when you made your own discovery. Such acknowledgement is a condition of use of this package. However this condition is waived if use amounts to no more than five diagrams, each of which is either a square or a triangle. This acknowledgement must, of course, be removed if the document is re-typeset by methods which do not use this package.

No permission was in the past given to use this package for commercial purposes. This includes a document whose copyright is seded by the author (for valuable consideration or not) to another person or body which subsequently intends to collect royalties for its reproduction. This applies to certain journals and conference proceedings. Permission is now granted for its 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.

No warranty is given with this software. It is supplied "as is", and neither the source nor this manual nor anything else shall be taken as a representation that it will perform any particular function, is suitable for any particular purpose or is of merchantable quality. In executing the software, the user implicitly accepts the above conditions and indemnifies the author, Imperial College and any person through whom the software was obtained, against liability for direct or consequential damages arising from the use of this software.

Whenever you use computers you *must* keep at least two back-up copies of all of your files, with one of them well away from the machine, in case of fire or major failure. Before sending any documents for publication or to an expendive printer make a thorough visual check using a previewer such as `xdvi` and a low resolution printer.

17 Reverse compatibility

Compatibility with past and future versions (the numbers of some of which have been assigned *post facto*) is as follows. Note the difference between "should" and "must".

Version 0: See section 13.

Version 1: Horizontal arrows made to stretch to edge of cell; 1987–9.

- `\rTo{f}{g}` works, but `\rTo f g` doesn't: it must be changed to `\rTo^f_g`.
- the `\mkern-20mu\rTo{f}{g}\mkern-20mu` idiom for manual stretching of horizontal arrows to meet objects must be removed.
- `\VerticalMapHeight` and `\VerticalMapDepth` commands are obsolete and are ignored: they should be removed.VerticalMapHeight
- Three- or four-argument uses of `\HorizontalMap`, `\VerticalMap` and `\DiagonalMap` must be changed to five-argument uses or, preferably, to `\newarrow`.
- Nested `\commdiag` commands for parallel maps must be changed to `\pile`.
- Parallel maps constructed by putting them in the rows or columns before and after must be moved to the correct cell and (in the case of horizontals) put in a `\pile`.

Version 2: Horizontal arrows made to stretch to meet objects and "superscript" labels introduced; versions with banners dated September to December 1989.

- `&\rTo\across3&` now works more accurately than in version 3, but should be changed to `&&\rTo&&`.
- The parameters `\HorizontalMapLength`, `\VerticalMapHeight`, `\VerticalMapDepth`, `\VerticalMapExtraHeight`, `\VerticalMapExtraDepth` and `\DiagonalLineSegments` are obsolete and are ignored: they should be removed.
- The labels on southeast and northwest diagonal arrows have changed their position. `^` and `_` previously meant left and right respectively for diagonals; they now mean — more logically — above and below; `<` and `>` are used for left and right, respectively. Unqualified labels are positioned as they originally were: the first above and the second below.
- Diagonal lines are constructed differently; user-defined diagonal arrows should be replaced with `\ruTo(2,4)` *etc.,* and explicit movement of them removed.
- Bent lines, with `\dlBent` and `\ruBentto`, currently do not work. Instead `\HmeetV` must be placed at the corner. Bent or half arrows will be reintroduced later but with a different naming convention.

Version 3: Vertical maps also made to stretch to meet objects; widely circulated with banners dated July 1990 to April 1992.

- `\HorizontalMap`, `\VerticalMap` and `\DiagonalMap` should be replaced by `\newarrow` declarations.
- Negative spacing around wide objects (used to avoid the stretching of the diagram which they caused) should be removed, and the `tight` option used instead. This is because objects are now allowed to extend into the neighbouring columns, with a check that there is enough space.

- Enclosing `$$...$$` or display environments should be removed, as they prevent the new display positioning options from working.
- Diagonal arrows with "compass" names, particularly the ones from `extra-diagonals.tex` (which must not be used any more), should be changed to the new geometrical names.
- The command names `\lt` and `\gt` (for $<$ and $>$) have been removed, at the request of a user who considered these to be unreasonable names for internal commands.
- The arrow commands with names like `\rArr` and `\rTo` now all use the default arrowheads; the original behaviour (`\rArr` used `LaTeX` and `\rTo` used `vee`) may be restored by changing a switch `\iffalse` in the final section of the source.

Version 4: Advertised in September 1992.

- The default arrowhead has been changed from `vee` to `LaTeX` to ensure consistency if `PostScript` is not used.
- Explicit movement of diagonal arrows vertically using `\raise`, `\lower` or `\raisebox` or horizontally by spacing commands or otherwise will not work for the diagonals which are adjusted to meet their endpoints. For the time being use either of the options `noPS` or `fixed` to suppress adjustment. A new option `crab` will be introduced shortly to allow sideways movement of arrows.

Future: The following are high on the agenda:

- Placing of labels at the head and tail of arrows.
- Curved arrows (`PostScript` only).
- Sideways movement of arrows.
- An alternative way of specifying the positions of arrows, namely at the tail with signed relative co-ordinates for the head, to allow emulation of Kris Rose's XY-pic and Michael Spivak's LAMS-TEX.
- Grids for regular figures such as the pentagon.
- Shifting objects, by specifying the left or right width.
- Option to rotate labels on diagonals or leave them horizontal.
- Application to electronic circuit diagrams.
- Overprints: pullback symbols and loops.

Suggestions for improvements and further applications (*e.g.* proof nets, Petri nets and circuit diagrams) are welcome.

18 Availability

The diagram package is currently available as a single 105kb[14] printable TEX source file which is compatible with both LATEX and `plain` TEX. It loads the LATEX `line10` font as `\tenln` (its LATEX name) and also defines some (exotic) arrow commands in terms the AMS symbols, but if these are to be used then the package `amssymb` must be loaded separately.

In Imperial College, the package is available on `/home/leonardo` and will be picked up automatically by the `tex` and `latex` commands. Users should not make their own copies unless they intend to take them away.

[14] Without comments; the development version is about 285kb.

The easiest way to obtain it is by **anonymous FTP** as

> `/tex/contrib/Taylor/diagrams/diagrams.tex` from `theory.doc.ic.ac.uk` (146.169.2.27)

Please do not ask for it by electronic mail (or less convenient media) unless you are unable to use FTP. I shall assume that users in Europe (apart from the formerly Communist countries), North America, Australia, New Zealand, Japan and Israel are able to use FTP unless they make the contrary clear.

If you are unable to use FTP it is possible to get the package (and indeed anything you can get by ftp) by electronic mail. I do not recommend this method, because it is susceptible to corruption, and you have to concatenate the files and remove the mail headers. Send the following message to `<ftpmail@doc.ic.ac.uk>`:

```
open theory.doc.ic.ac.uk
cd tex/contrib/Taylor/tex
get diagrams-V4-news
get diagrams.tex
get diagrams-manual.tex
quit
```

You will get an acknowledgement message first, followed by the files, and finally a job log file. The files may be split up into several messages.

Index