ISSN 1364-0380

Geometry & Topology
Sample 3 (1996) 301–320
Published: 6 June 1996

Further topics for authors

Colin Rourke

Mathematics Institute
University of Warwick
Coventry, CV4 7AL, UK

Email: `cpr@maths.warwick.ac.uk`

URL: `http://www.maths.warwick.ac.uk/~cpr/`

Abstract

This file is the third sample file for *Geometry & Topology*. It contains a more thorough discussion of the topic of graphics in `tex` and `latex` files (discussed in [3]) and also some further topics. It is addressed to users of `tex` in all its various flavours. It comprises five topics:

(1) The `pictex` drawing package
(2) Including `.eps` files in a `tex` or `latex` document with the macro package `epsf.tex`
(3) Relabelling `.eps` files in a `tex` or `latex` document using the macro package `rlepsf.tex`
(4) Notes on the relabelling package `psfrag`
(5) Adding labels using the package `labelfig.tex`
(6) Inserts in plain `tex`

Topic (3) depends on topic (2), but otherwise the topics are independent and can be read in any order.

AMS Classification numbers Primary: 00-01, 00-02, 68-00, 68-01

Secondary: 51-00, 51-01, 55-00, 55-01, 57-00, 57-01

Keywords: Graphics, `tex`, `latex`, `pictex`, eps files, labelling, relabelling, inserts

Proposed: James Williams | Received: 7 June 1996
Seconded: Alan Humphries, Tracy Brakes | Revised: 9 June 1996

We are concerned here with the problem of creating and inserting graphics into a `tex` or `latex` document and particularly with the problem of making the labels in the included diagrams match the text accurately. All the macro packages mentioned in this document except for `psfrag` can be collected from the *Geometry & Topology* `\info\macros` subdirectory. (Many of these packages are standard macro packages—particularly the `pictex` and `epsf` packages—and will probably already be installed on your system.)
This is the `latex` *version of this file.*

1 The classic `pictex` drawing package

Michael Wichura's classic package `pictex` can be used to produce excellent diagrams in a `tex` or `latex` document. See for example [2] for macros which use `pictex` to draw braid and knot diagrams. The great advantage of this package is that it is pure `tex`. Thus `pictex` diagrams view correctly on all dvi viewers and no special software is needed to create, view or print them. The disadvantages are firstly that creation of `pictex` diagrams is a non-interactive process and secondly that a complicated `pictex` diagram can use a vast amount of `tex`'s memory leading to a `TeX capacity exceeded, sorry ...` message (and no output). For those of you who are lucky enough to use Eberhard Mattes' suberb program suite `emtex` running on a modern PC, this latter problem is not a real problem. Switch to the `emtex` program `htex386` ("huge" `tex`); the memory size on this version of `tex` is large enough to contain the most ambitious `pictex` diagrams![1] Users of other installations of `tex` may have to persuade their systems manager to enlarge the main memory size of `tex` if they meet this problem—not always an easy task.

The main purpose of this section is to explain how to use the X–windows drawing program `xfig` to overcome the first of these two drawbacks and create `pictex` diagrams interactively. We start by explaining how to load pictex and, briefly, how to use it.

Loading `pictex`

To load `pictex` in plain or ams `tex` you need the file `pictex.tex` (which is already installed in most `tex` installations) and then the line

[1] This file needed "huge" `tex` to process it. The two large sample `pictex` figures drawn using `xfig` (figures 3 and 4) occur on one page, and together overflow the main memory of standard "big" `tex`!

```
\input pictex
```

somewhere near the start of your file. In `latex` you need three files `pictex.tex`, `prepictex.tex` and `postpictex.tex` (again usually already installed) and the following four lines:

```
\font\fiverm=cmr5
\input prepictex
\input pictex
\input postpictex
```

For convenience we have supplied a `pictex.sty` file which loads `pictex` for you. Collect this file (and the other three files if they are not already installed) and then you can load pictex by adding `pictex` to your style instruction (old `latex` — latex2.09) or your `usepackage` command (new `latex` — latex2e) thus:

```
\documentstyle[11pt,pictex]{article}         %(old latex)
```

or

```
\documentclass[11pt]{article}                %(new latex)
\usepackage{latexsym,pictex}
```

Using `pictex`

Detailed instructions on using the package are given in [16]. Here we shall just consider those instructions which you need to understand in order to use `xfig` as a "front end" for `pictex`. Here is the code for the example shown in figure 1 (reproduced from [3]):

```
$$\beginpicture\small
\setcoordinatesystem units <.4truein,.4truein> point at 0 0
\put {$b$} [r] <-3pt,0pt> at 0 1
\put {$a$} [t] <0pt,-3pt> at 1 0
\put {$a^b$} [b] <0pt,3pt> at 1 2
\put {$b$} [l] <3pt,0pt> at 2 1
\setlinear
\plot 0 0 2 0 2 2 0 2 0 0 /
\arrow <4pt> [.2, .7] from 0 2 to 0 2.4
\arrow <4pt> [.2, .7] from 2 0 to 2.4 0
\setplotsymbol ({.})
\plot 1 0 1 0.85 /
\plot 1 1.15 1 2 /
\plot 0 1 2 1 /
\put {$\scriptstyle1$} [l] <2pt,0pt> at 2.4 0
```

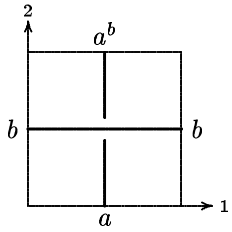


Figure 1: from [2]

```
\put {$\scriptstyle2$} [b] <0pt,2pt> at 0 2.4
\endpicture$$
```

The enclosing `$$ ... $$` are there to place the picture in the centre of the page (like a dispayed equation). The guts of the code is the enclosing `\beginpicture ... \endpicture` which creates a box inside which the various drawing commands are given. We shall need to consider two particular commands in detail, the `\setcoordinatesystem` command and the `\put` command, but notice the command `\small` which sets the labels in the figure in 10pt type. The `\beginpicture ... \endpicture` forms a `tex` group, so the `\small` command stays local to this group and does not affect the subsequent text.

By editing the `\setcoordinatesystem` command you can scale the figure. For example if we change the units to `<.3truein, .5truein>` the result is illustrated in figure 2.

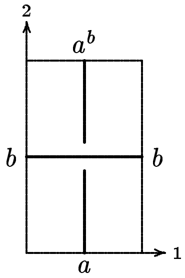


Figure 2: the result of scaling figure 1

Thus you can very easily adjust figures for size after they are drawn.

Now look at the syntax of the `\put` command. In its simplest form it is:

`\put` {TeX material} `at x y`

which places the `tex` material inside the curly brackets into the `pictex` picture centred at the point with coordinates (x, y). The other items in the `\put` command are optional. Firstly there is an orientation command in square brackets eg [lB] (which defines the left end of the baseline as the point in the tex material to be placed at the given coordinates); the default orientation is the centre of the material. Secondly there is a very useful adjustment command in diamond brackets:

```
\put {TeX material} <dx,dy> at x y
```

will make a fine adjustment to the position of the tex material by (dx, dy) which are any tex dimensions. Thus in figure 1 the label a^b has been placed exactly 3pt above the point $(2,1)$ using the adjustment <0pt, 3pt> and notice that this adjustment was unaffected by the scaling in figure 2.

Using xfig as a "front end" for pictex

We are now ready to discuss the main topic of this section namely using xfig as a "front end" for pictex.

If you have access to a computer running X–windows with xfig available, you can draw your figure interactively in xfig and then export the figure as pictex macros. The resulting file can be pasted into your main tex document. You will need to edit the labels, since xfig cannot handle mathematical symbols. So what you do is to use "dummy" labels in the xfig picture which are roughly the correct size and suitably mnemonic. For example the label ab could be used where you want to place a^b.

After exporting your figure, look for the text \put commands (which the export programme conveniently places near the end of the file), and edit them to the required tex formulæ. The exporter makes quite a song-and-dance out of the text \put's so you will have to discard quite a lot of the code. Here is an example of a text \put command in pictex macros output from xfig:

```
\put{\SetFigFont{12}{14.4}{rm}ab} [lB] at  7.855 22.172
```

which you need to edit to:

```
\put{$a^b$} [lB] at  7.855 22.172
```

The control sequence \SetFigFont is defined in a preamble which the exporter places before the main figure. Since you are discarding all the uses of \SetFigFont you can discard the definition as well, which will tidy up your file. In fact all the preamble before the line:

```
\mbox{\beginpicture
```

can be discarded except for the line:

```
\font\thinlinefont=cmr5
```

which is essential (though only needs to be placed once early in your file). Type the size command `\small` somewhere in the file before the text `\put`'s (for example just after `\beginpicture`) to set the new labels in the correct size. Now paste the file containing the figure into your main file.

If you are using `latex` then that is all you have to do. If you are using plain or ams `tex` then there is a little more work to do because the `xfig` exporter writes the macros in `latex` and the author seems to have deliberately chosen `latex` commands none of which are understood by plain `tex`, instead of the corresponding plain `tex` commands which are understood by both! (If the person responsible for the next version of this exporter reads this, then please take note!)

Anyway, you have to translate five `latex` commands into the corresponding plain commands by pasting the following lines somewhere in your file before the figure:

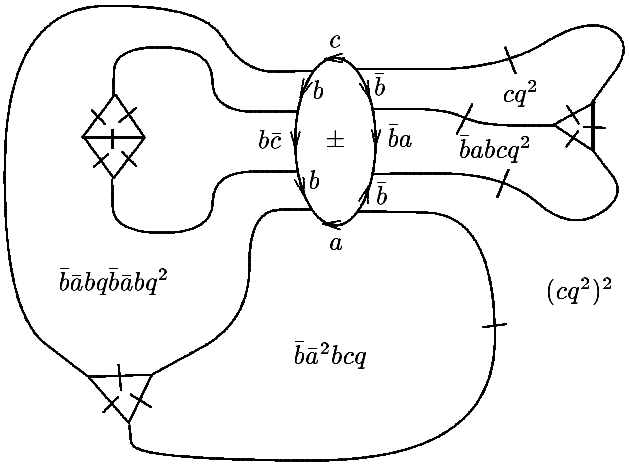
```
\def\mbox#1{\hbox{#1}}
\def\makebox(0,0)[l]#1{\hbox to 0pt{#1\hss}}
\def\symbol#1{\char#1}
\font\tencirc=circle10
\newdimen\unitlength
```

The result after all this editing is a `tex` box containing your figure, which you can place in a `\centerline{ ...}` or a `$$ ... $$` for display or include in a `tex` insert (see section 5) or combine with other `tex` boxes to form a larger display.

Finally preview the figure and adjust the size if necessary by editing the `\setcoordinatesystem` command as explained above and adjust the positions of the labels (if necessary) by inserting adjustments of the form `<dx,dy>` before the `at` in the appropriate `\put` commands.

This all sounds very complicated, but the results are well worth the effort and, moreover, the editing described above is much easier to do than describe. Two examples of figures which I drew recently using the above procedure are given in figures 3 and 4. Figure 3 is from [13] and figure 4 from [12]; both were drawn because I decided to put these papers on my home page and had to replace the existing hand-drawn digrams by electronic diagrams.

Here is a summary of the steps needed to create a `pictex` diagram using `xfig` as a "front end":



where $q = \bar{b}\bar{a}bc$ and unlabelled regions carry q^3, a^2 or c^3

Figure 3: from [10]

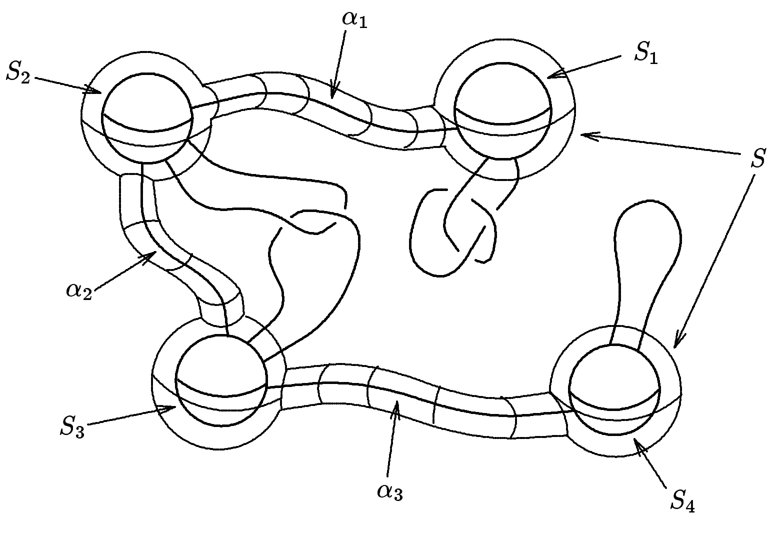


Figure 4: from [9]

Summary of procedure

1. Copy and paste the following lines somewhere near the start of your file (you'll find a copy near the start of the `.tex` source file of this file):

```
\def\mbox#1{\hbox{#1}}
\newdimen\unitlength
\def\makebox(0,0)[l]#1{\hbox to 0pt{#1\hss}}
\def\symbol#1{\char#1}
\font\tencirc=circle10
\font\thinlinefont=cmr5
```

Note: `latex` *users should only paste in the last line.*

2. Export your `xfig` picture as `pictex macros` and edit the output by discarding the preamble before the line:
`\mbox{\beginpicture`

3. Insert the command `\small` somewhere before the text objects (eg just after `\beginpicture`).

3. Edit the text of the labels to the correct `tex` text, by discarding the `\SetFigFont{12}{14.4}{rm}` (or similar material) and changing your dummy labels into the required `tex` labels.

4. Paste the picture into your main file and preview it. Make adjustments to the overall size and the label positions if necessary.

(If you inspect the `tex` source file to this file, you can see the results of this editing for the two sample pictures in figures 3 and 4.)

Health warning

The export program from `xfig` to `pictex macros` is not stable. The output changed considerably a couple of versions ago. The details of this section are based on the current exporter (`xfig` version 3.1). If the exporter changes its output again then the instructions in this section will need changing to suit; however the basic idea of the editing will remain constant.

Some further hints

(1) You can avoid most of the editing described above if you redefine `\SetFigFont` as follows:

```
\def\SetFigFont#1#2#3#4{{\small$#4$}}
```

The effect of this is that the text `\put`'s will consider only the last part after the `{rm}` and will set the first character `\small` and in mathematics mode. Thus if you have only one character in the label which is wanted in mathematics mode (eg a single letter label) then there is no editing needed at all. If you have a mathematical formula, then edit it and add `{,}`'s eg:

```
\put{\SetFigFont{12}{14.4}{rm}{a^b}} [lB] at  7.855 22.172
```

and if you have a text label then precede it with an empty formula `{}` eg:

```
\put{\SetFigFont{12}{14.4}{rm}{}Some text} [lB] at  6.832 21.004
```

(2) If you are using plain or ams `tex` then you can exploit the fact that the exporter puts the figure into a `latex` `\mbox` to display the figure by your favourite method. For example if you always display in `$$ ... $$` then you could define:

```
\def\mbox#1{$$\hbox{#1}$$}
```

to do this automatically.

(3) It is possible to make substantial "fine tuning" to the finished picture by directly editing the `pictex` instructions. For example, the export program always uses `{\thinlinefont .}` as the plot symbol for thin lines. By judiciously changing the definition of `\thinlinefont` to `cmr6` or `cmr7` these thin lines can be made just a little bit more solid.

2 Including `.eps` files in a `tex` document

This section is a brief introduction to the macro package `epsf.tex` [11]. Complete detailed instructions are given at the start of [11], my purpose here is merely to publicise this package and particularly to show how easy it is to use.[2]

Loading `epsf.tex`

You need the file `epsf.tex` and the line

[2]The `epsf.tex` package can only be used with the Rokicki–Knuth `dvips` converter normally installed with Unix or IbmPC `tex` instalations. For other installations you have to use another package, such as Larry Siebenmann's `BoxedEPS.tex` [15] which works with all `tex` installations. You have to "switch on" the appropriate `\special` but, after that, the package is equally easy to use. If you are using the current version (latex2e) of `latex` then you can use the `latex` graphics package [10] as an alternative to `epsf.tex`. Like `BoxedEPS.tex` this works with all `tex` installations. Instructions for collecting and installing this package are given in section 4.

```
\input epsf
```

somewhere in your file near the beginning.

In `latex` you can load `epsf` by adding it to your `\documentstyle` or `\usepackage` command as described for pictex above (you need the file `epsf.sty` as well to do this). Both these files are usually already installed in `tex` installations. Note that this is *instead* of inserting the line `\input epsf`.

Using `epsf.tex`

If your figure is the `.eps` file called say `spheres.eps` then to insert the figure in your file you merely type

```
\epsfbox{spheres.eps}
```

which will paste the figure into your file at the current point. To display it you would type for example:

```
\centerline{\epsfbox{spheres.eps}}
```

These instructions will insert the figure at its natural size. But you can also rescale it to any desired size very easily. The line

```
\centerline{\epsfxsize 4.4truein\epsfbox{spheres.eps}}
```

will display the figure but with width scaled to 4.4 inches (and height scaled appropriately to preserve the shape of the figure). A similar command `\epsfysize` will scale using height as the measure. Figure 5 was inserted in this file by using this command. This is the same figure as figure 4 except that it was output from `xfig` as `encapsulated postscript` and the "dummy" labels in the original `xfig` diagram were first changed to appropriate postscript labels.

Notice that I took care to select postscript fonts to make the labels match the computer modern fonts of the main text as far as possible. I chose Times-Italic for the letter "S", Times-Roman (in a smaller font size) for the numerical subscripts and Symbol (Greek) for the α's. This is a very painstaking process, especially placing the numbers to simulate correct mathematical spacing. It would be quite impractical for a diagram such as figure 3, which has complicated mathematical formulæ as labels. For this diagram the methods described in sections 1, 3 or 4 are far more practical (as well as producing a better finished product).

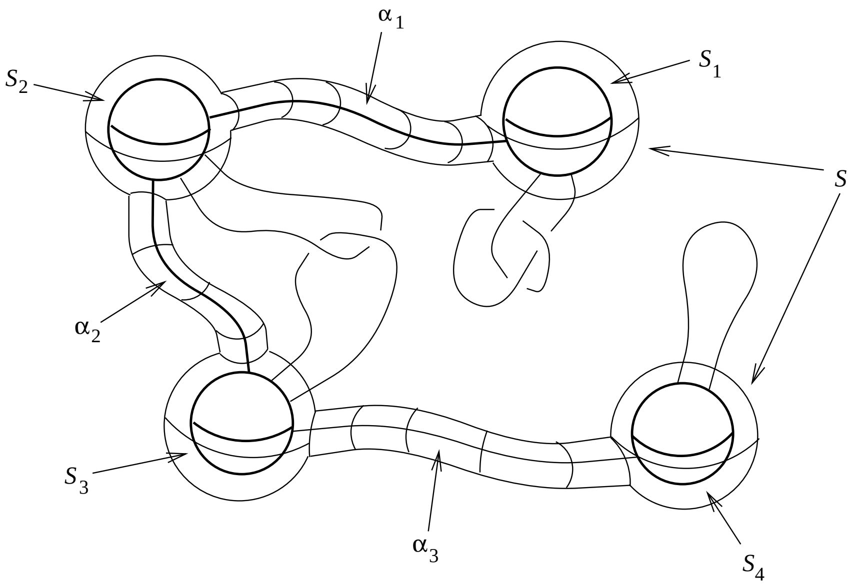


Figure 5: the `.eps` version of figure 4

3 Relabelling `.eps` files using `rlepsf.tex`

The macro package **`rlespsf.tex`** is an extension of **`epsf.tex`** which allows the postscript labels in the **`.eps`** file to be replaced by **`tex`** labels. This means that you can draw your figure in any drawing program which is capable of **`eps`** output (or even draw the figure "by hand") using dummy postscript labels, which are then replaced by the correct **`tex`** labels.

Loading `rlepsf.tex`

You need both the files **`epsf.tex`** and **`rlepsf.tex`** and the line

`\input rlepsf`

somewhere in your file near the beginning. Do *not* also type **`\input epsf`** (if you already have this line then edit it to **`\input rlepsf`**). This is because **`rlepsf.tex`** loads **`epsf.tex`** automatically so it should not be loaded separately.

In **`latex`** you can load **`rlepsf`** by adding it to your **`\documentstyle`** or **`\usepackage`** command as described for **`pictex`** above (you need the three files **`rlepsf.sty`**, **`epsf.tex`** and **`rlepsf.tex`** for this). Note that this is *instead* of inserting the line **`\input rlepsf`**. Also note that the package **`rlepsf`** includes the package **`epsf`**, which should not be loaded separately.

Using `rlepsf.tex`

Draw your figure in any drawing program and use “dummy” labels, ie choose labels which are simple strings but which remind you of the correct label; for example you could type `babcq2` to stand for the label $\bar babcq^2$. Now export your figure as an encapsulated postscript (`eps`) file. Place your `eps` file inside a `\relabelbox ... \endrelabelbox`. Insert the `eps` file exactly as usual with `epsf.tex` as explained in the last section. Type your relabelling instructions after the `\epsfbox{..}` command. A typical relabelling instruction (for the dummy label suggested above) is:

```
\relabel {babcq2}{$\bar babcq^2$}
```

There are three (re)labelling macros:

```
\relabel {pslabel}{TeX label}
\adjustrelabel <dx,dy> {pslabel}{TeX label}
\extralabel <x,y> {extra label}
```

`\relabel` uses the position of the dummy ps label for the new `tex` label (which can be any `tex` output). The new label is positioned with same reference point (left-hand end of baseline) as the dummy ps label. Usually this means that the new label position is perfect, but if it needs adjusting then `\adjustrelabel` can be used to adjust the position by (dx, dy). `\extralabel` allows extra `tex` labels to be inserted. The positions x, y are measured from the bottom right-hand corner of the diagram (so x is typically negative)! The re/extra-labelling instructions can be given in any order and any `tex` dimensions can be used for x, y, dx, dy.

Preview the diagram (with ghostview or some similar ps previewer) after inserting the `\relabel` instructions but before inserting the `\extralabel` instructions. Decide if the figure needs scaling and if so choose the scaled width or height. Decide which labels need adjusting (if any) and measure the positions of any extra labels which are needed from the bottom right. Add the appropriate scaling command to the `\epsfbox` command as explained in the last section. Edit the `\relabel` instructions which need adjusting into `\adjustrelabel` instructions inserting appropriate <dx,dy> and add the `\extralabel` instructions.

Preview again and make any final adjustments to the various variables. This all sounds complicated, but it is in fact very straightforward and the amount of adjustment is usually minimal.

The result is a box with the same dimensions as the original epsfbox but with new `tex` labels. This can then be inserted in `$$ ... $$` or `\centerline{ ... }` for display, or combined with other boxes to form a larger display.

Here is a specific example. This is the code for relabelling figure 6, which is the same `xfig` picture as figure 3, but exported from `xfig` as an `eps` file.

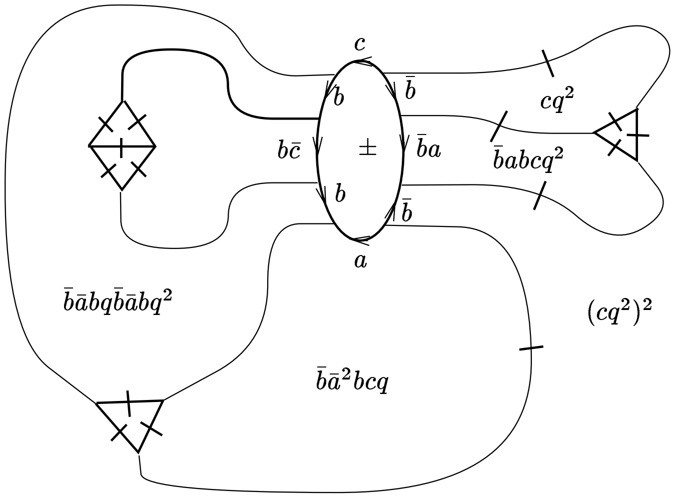
```
\centerline{\relabelbox\small
\epsfxsize 3.5truein \epsfbox{KDiag.eps}
\relabel {c}{$c$}
\adjustrelabel <-2pt, 0pt> {a}{$a$}
\relabel {b}{$b$}
\adjustrelabel <-1pt, 0pt> {b1}{$b$}
\adjustrelabel <-1pt, -1pt> {bb}{$\bar b$}
\relabel {bb1}{$\bar b$}
\relabel {bc}{$b\bar c$}
\relabel {ba}{$\bar ba$}
\relabel {cq2}{$cq^2$}
\relabel {babcq2}{$\bar babcq^2$}
\relabel{(cq2)2}{$(cq^2)^2$}
\relabel {ba2bcq}{$\bar b\bar a^2bcq$}
\relabel {babqbabq2}{$\bar b\bar abq\bar b\bar abq^2$}
\relabel {where q=babc etc}{where $q=\bar b\bar abc$ and
unlabelled regions carry $q^3$, $a^2$ or $c^3$}
\extralabel <-1.7truein, 2.15truein> {$\pm$}
\endrelabelbox}
```

Remarks (1) The `\relabelbox ... \endrelabelbox` forms a `tex` group; thus the instruction `\small` which sets the new labels in 10pt type remains local to this group and does not affect the subsequent text.

(2) If the `eps` file has two or more labels the same, then only one will be replaced. Therefore it is necessary to use different dummy labels even if you wish the final labels to be the same. In the above example the dummy labels `b` and `b1` were both replaced by b and the dummies `bb` and `bb1` were both replaced by $\bar b$.

(3) The macros are sensitive to spaces. Be careful to leave a space after the $<dx\,, dy>$ adjustment but not between the {pslabel} and the {TeX label}.

(4) All the existing ps labels are deleted from the `eps` digram, whether or not instructions for relabelling are given. Thus you can use the macros to relabel an existing diagram by choosing the appropriate subset of labels to replace. However if you want to replace two or more ps labels, which are the same, then you have to use `\extralabel` to replace the one(s) which do not get relabelled



where $q = \bar{b}\bar{a}bc$ and unlabelled regions carry q^3, a^2 or c^3

Figure 6: the relabelled `.eps` version of figure 3

at the first attempt. (Alternatively if you know how to find the labels in the `eps` file—look for strings enclosed in round brackets—then you can edit the `eps` file to make the repeated labels different.)

(5) The output will not view correctly on a dvi viewer which will typically show the new labels superimposed at the bottom right-hand corner. However those produced by `\extralabel` will be shown correctly positioned.

Health warning The package `rlepsf.tex` is still under development. The present version [7] is an "beta test" version. It has been thoroughly tested with `xfig` and Adobe Illusrator output and also with 'hand-drawn' figures. However before committing any time to using the package, test it with a simple test figure to check that it will work properly with your chosen drawing program.

4 Notes on the relabelling package `psfrag`

There is a macro package `psfrag` [6], written by Michael Grant et al, which does a similar job to `rlepsf.tex`. This package can be collected from the CTAN `tex` archive and contains its own instructions. It needs a fairly recent version of `latex` (not more than a year or so out-of-date) in order to work. It will *not* work with old latex (latex2.09). It also needs the latex graphics package [10] installed. To install either of these packages, proceed as follows. Connect by anonymous ftp (or a Web browser) to your nearest CTAN tex archive site (in the UK this has address ftp.tex.ac.uk). Change to the appropriate subdirectory (see the bibliography to this paper). Collect the contents of this subdirectory.[3] Once you have the files on your computer (unpacked into a suitable directory) you have to "install" the package by running latex on the .ins file; for example to install psfrag you type latex psfrag.ins.

There is an interface written by David Carlisle [1], which adapts psfrag (and the latex graphics package) for use with plain tex. You can collect and install this in the same way as the other packages mentioned above. However note that this package, although it is designed for use with plain tex, needs a recent version of latex installed (including the graphics and psfrag packages) in order to work.

In use psfrag is similar to rlepsf.tex; for details see the documentation in the package [6]. Unlike rlepsf.tex, it is designed to work with all tex installations; however a similar health warning should be issued: Before committing any time to using the package, test it with a simple test figure to check that it will work properly both with your chosen drawing program and with your dvi to ps converter.

5 Relabelling .eps files using labelfig.tex

The macro package labelfig.tex [14] allows tex labels to be superimposed onto any picture, or in fact onto any box. Essentially this is a sophisticated version of the \extralabel facility in rlepsf.tex and indeed you can combine the two methods—replace the convenient ps labels by tex labels using rlepsf.tex—then add extra labels using labelfig.tex, see the example given below. The sophistication comes from the fact that the new labels attached by labelfig.tex are given using "scalable" coordinates, ie coordinates which are fixed to the picture and scale with it; moreover you can superimpose a temporary grid over the picture in order to read off the coordinates of the new label positions.

[3] If you are using anonymous ftp you can take advantage of the archive's "zipping on the fly" capability: for instance to collect psfrag change to the subdirectory tex-archive/macros/latex/contrib/supported/, change file-type to binary, and type get psfrag.zip. Note that you must be in the *parent* directory of the directory you wish to collect.

There is excellent documentation in the file `labelfig.tex` both as comments near the beginning and as an appended document; for full details on using the package, see this documentation. Here we shall just give an overview together with an example of use.

Loading `labelfig.tex`

You need the file `labelfig.tex` and the line

```
\input labelfig
```

somewhere in your file near the beginning.

In `latex` you can load `labelfig` as usual by adding it to your `\documentstyle` or `\usepackage` command as described for `pictex` in section 1. You need the two files `labelfig.sty` and `labelfig.tex` for this.

Using `labelfig.tex`

Place your figure in an `\AffixLabels{ ... }` command preceeded by a `\ShowGrid` command, eg:

```
\ShowGrid
\AffixLabels{\epsfbox{figure.eps}}
```

Now preview the figure. The figure will be shown with a 10×10 grid superimposed over it. Read off the coordinates of the positions where you want the labels and then construct a label list of the form:

```
\SetLabels
(x1*y1) first label \\
(x2*y2) second label \\
...
\endSetLabels
```

which should come before the `\Showgrid`. You can prefix the coordinates with extra orientation commands (such as `\L`) which cause `labelfig.tex` to attach the new labels to the given coordinate by particular points. For example `\L` means: attach by left-hand end of baseline. (The default is centre of baseline.) For a complete list of these prefix commands see [14]; see also the worked example below. Preview again. When you are happy with the new labels, comment out the `\ShowGrid` command to remove the grid.

Here is a worked example. I used both `rlepsf.tex` and `labelfig.tex` to relabel an existing diagram (from [9]) with `tex` labels. The diagram with grid attached is shown in figure 7. The commands for displaying this figure were:

```
\centerline{\ShowGrid
\AffixLabels{\epsfbox{Closure.eps}}}
```

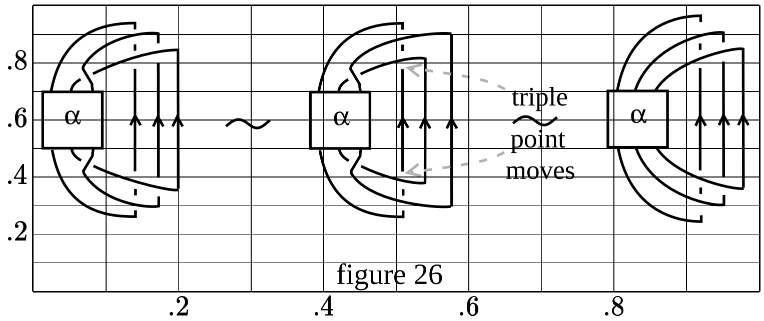


Figure 7: from [7] (drawn with Adobe Illustrator)

After previewing figure 7, I decided to replace the labels “triple”, “point”, “moves” and “figure 26” using `rlepsf.tex`, which is needed in any case in order to kill the postscript labels, and use `labelfig.tex` to replace the three α's. I read off the coordinates for the centre of the three α's as $(.05, .6)$, $(.42, .6)$, $(.83, .6)$ and typed the `\SetLabel` instructions. I previewed the figure a second time, made small adjustments to two of the `\relabels` and commented out the `\Showgrid`. The result is shown in figure 8, and the full set of instructions are as follows:

```
\centerline{%
\small
\SetLabels
\E(.05*.6) $\alpha$\\
\E(.42*.6) $\alpha$\\
\E(.83*.6) $\alpha$\\
\endSetLabels
%\ShowGrid
\AffixLabels{\relabelbox
\epsfbox{Closure.eps}
\relabel{triple}{triple}
\relabel{point}{point}
\adjustrelabel <2pt, 0pt> {moves}{moves}
\adjustrelabel <-0.5in,0pt> {figure 26}{Figure 8:
figure 7 after relabelling}
\endrelabelbox}}
```

Notes

(1) The `\E` commands prefixed to the coordinates for the α's are orientation

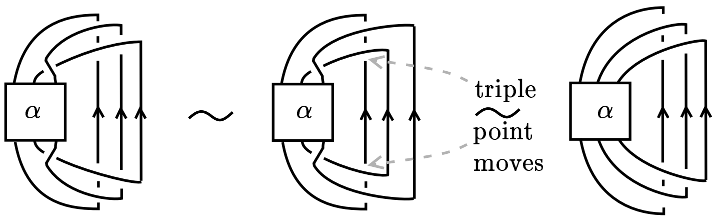


Figure 8: figure 7 after relabelling

commands which cause `labelfig.tex` to attach these labels by their centres to the point with given coordinates ("E" is for 'equator').

(2) It is not recommended to attach the caption to the figure as part of the figure. This was merely done in the example because the caption was part of the original figure and, if nothing had been substituted, there would have been an unsightly blank line at the bottom of the figure.

(3) The `\SetLabels .... \endSetLabels` pair forms a `tex` group and I could have typed `\small` after `\SetLabels` to set the new labels for `labelfig.tex` in the correct point size without affecting any other part of the text. However, by placing the `\small` command just after the `\centerline{`, both sets of new labels were set in the correct point size.

6 Inserts in plain `tex`

This section is intended for users of plain or ams `tex` *only.*

If you examine the `tex` source file for this file (the plain `tex` version) you will see that the figure in the example given at the end of the last section, was displayed inside a `\midinsert ... \endinsert` pair. This is a facility provided by plain `tex` to allow the figure to "float" to a position where it will fit well on the page without causing badly filled pages. It is the recommended way to display figures if you use plain or ams `tex`. (In `latex` the `begin{figure} .... end{figure}` environment does a similar job of floating figures to a suitable place for display and this was used for the `latex` version of this file.)

There are macros for handling figures given in our macro packages [5] which call `\midinsert` automatically and you will find instructions for using these macros in [4].

However the plain `tex` routines for handling `\midinsert` have some bugs in them—the spacing of an insert at the top of a page can often be too high and the spacing betwen successive inserts is sometimes strange. These bugs have been corrected by a short macro package `\newinsert.tex` written by Walter Neumann, and we strongly recommend that you use this package if you have figures to display. To use this package you simply type

```
\input newinsert
```

somewhere near the start of your file and let the package work for you. There are two adjustable lengths defined by the package that need to be set: `\insertskipamount` and `\inserthardskipamount`. For best compatibility with the recommended *Geometry & Topology* style these should be set as follows:

```
\insertskipamount 12pt plus 2pt
\inserthardskipamount 4pt
```

If you use the copy of `\newinsert.tex` which is included in the *Geometry & Topology* macro directory, then these lengths are correctly set. If you have the package already or obtain it from another source then either edit the package to set these lengths (the relevant lines are clearly marked near the start of the macros proper) or type these two lines in your main file just after `\input newinsert`.

References

[1] **David Carlisle**, **The `psfrag` package for plain `tex`**, CTAN `tex` archive, `tex-archive/macros/plain/graphics/`

[2] **Roger Fenn**, *PicTeX macros for drawing braids and links*, available from Roger Fenn's home page:
`http://www.maths.sussex.ac.uk/Staff/RAF.html`

[3] **Geometry and Topology**, *Instructions for authors*, Geometry and Topology, `/info/samples/author.tex`

[4] **Geometry and Topology**, *The Geometry & Topology macro packages*, Geometry and Topology `/info/macros/gtmacins.tex`

[5] **Geometry and Topology**, *Macro files:* `gtmacs.tex` *and* `gtmacros.tex`, Geometry and Topology `/info/macros` subdirectory

[6] **Michael Grant**, **David Carlisle**, **Craig Barratt**, *The* `psfrag` *package*, CTAN `tex` archive,
`tex-archive/macros/latex/contrib/supported/psfrag/`

[7] **Michael Greene**, **Colin Rourke**, *The macro package* `rlepsf.tex`, Geometry and Topology, `/info/macros` subdirectory

[8] **Donald Knuth**, *The TeXbook*, Addison–Wesley (1984)

[9] **Sofia Lambropoulou**, **Colin Rourke**, *Markov's theorem in 3–manifolds*, Topology and its Applications (to appear) (available from my home page)

[10] **Latex3 project team**, *The* `graphics` *package*, CTAN `tex` archive, `tex-archive/macros/latex/packages/graphics/`

[11] **Tomas Rokicki (and Donald Knuth)**, *The macro package* `epsf.tex`, Geometry and Topology, `/info/macros` subdirectory

[12] **Colin Rourke**, *Algorithms to disprove the Poincaré conjecture*, Talk given at the Gökova Topology Conference, May 1994 (available from my home page)

[13] **Colin Rourke**, *Dunce hats and the Kervaire conjecture*, Papers presented to EC Zeeman, Warwick (1988) 221–230 (available from my home page)

[14] **Ray Séroul**, **Larry Siebenmann**, *The macro package* `labelfig.tex`, Geometry and Topology, `/info/macros` subdirectory

[15] **Larry Siebenmann**, *The macro package* `BoxedEPS.tex`, CTAN `tex` archive, `tex-archive/macros/generic/boxedeps/`

[16] **Michael Wichura**, *The PicTeX manual*, TeXniques series, 6, TUG (1986)