## Labelled diagrams in METAFONT

Alan Jeffrey

### 1 Diagrams in METAFONT

In *TUGboat* 11(4), Alan Hoenig described a method of producing diagrams in METAFONT with labels provided by TEX. His method relied on passing information around via font dimensions. This is a standard method of passing information from METAFONT to TEX, but it has some drawbacks:

- There are only a limited number of font dimensions available, and each label uses up two of them.
- As METAFONT can only communicate with TEX via font dimensions, each label has to be assigned a font dimension, and it is difficult for the correspondence between font dimensions and labels to be kept automatically.
- Since TEX is providing the labels, and METAFONT is providing the diagrams, the diagrams have to be kept in a different file from the labels.
- There is no communication between TEX and METAFONT, so METAFONT cannot change the diagram depending on the size and shape of the labels. This is rather inconvenient for diagrams such as


  where the shape of the ovals depends on the size of the contents.

Fired with enthusiasm by Alan's talk at the European TEX Users Group meeting, I stole the best of his ideas, and slightly modified them to produce a simple METAFONT–TEX interface. This allows TEX code to be embedded within a METAFONT program, for example

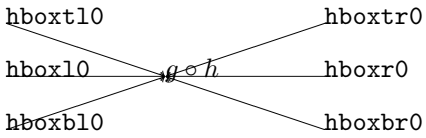
```
begindiagram(2,30pt#,7pt#,2pt#);
    hboxes(0);
    pickup pencircle scaled 0.4pt;
    .5[hboxl0,hboxr0] = (.5w,0);
    draw hboxbl0..hboxtl0
             ---hboxtr0..hboxbr0
             ---cycle;
    setbox0 "$g \circ h$";
enddiagram;
```

produces the diagram $g \circ h$ The new facilities used are:

`begindiagram(2,30pt#,7pt#,2pt#)` starts off diagram 2, which is 30pt wide, 7pt tall and 2pt deep.

`hboxes(0)` says that the only label we'll be using is number 0. This has a similar syntax to `labels`, so you can say `hboxes(1,2,7)` or `hboxes(3 upto 9)`.

`hboxl0` is the left point of label number 0, at the baseline. Similarly, `hboxbl0` is the bottom left, `hboxtr0` is top right, and so on. In this example, these points are




You can also use the numeric variables `hboxwd0`, `hboxht0` and `hboxdp0` which are the width, height and depth of label 0, and `hboxwd#0`, `hboxht#0` and `hboxdp#0` which are their sharp equivalents.

`setbox0 "$g \circ h$"` sets label number 0 to be $g \circ h$.

`enddiagram` finishes it all off.

The rest of the diagram is standard METAFONT. Within a TEX document you can use

`\diagramfile{dmfexmpl}` to load in the diagrams kept in `dmfexmpl.mf`,

`\diagramf{2}` to get the second diagram, and

`\everylabel` which is a token register added to every label, in the same fashion as `\everymath`. It should be set *before* saying `\diagramfile`.

These commands behave well inside groups, so if you say

```
\diagramfile{foo}
{\diagramfile{baz}\diagramf{1}}
\diagramf{2}
```

you get the first diagram from `baz` and the second diagram from `foo`.

### 2 How it all works

In the `diagramf` package, TEX and METAFONT communicate by auxiliary files, in a similar fashion to the MG TEX-PostScript interface ('Problems on the TEX/PostScript/graphics interface', *TUGboat* 11(3)).

When you run METAFONT on `dmfexmpl.mf` it reads in `dmfexmpl.dim`, which specifies the dimensions of all the boxes. In our example, part of `dmfexmpl.dim` is

```
wd#[2][0] := 20.3344pt#;
ht#[2][0] := 6.94444pt#;
dp#[2][0] := 1.94444pt#;
```

So, in diagram 2, label 0 has width 20.3344pt, height 6.94444pt and depth 1.94444pt. From this, METAFONT calculates where to put each label, and outputs a `.dia` file, containing TEX code. For example `dmfexmpl.dia` contains[1]:

```
\newdiagram{2}
\diagramlabel{0}{4.88908pt}{0pt}
$g \circ h$
\enddiagramlabel
\diagramchar{2}
\endnewdiagram
```

This tells TEX that diagram number 2 contains label 0 at coordinates (4.88908pt, 0pt) consisting of `$g \circ h$`. The diagram is character number 2 in the `dmfexmpl` font.

Similarly, when TEX encounters the instruction `\diagramfile{dmfexmpl}` it loads in `dmfexmpl.dia` and produces `dmfexmpl.dim`. And so we can have our METAFONT cake and eat it in TEX.

Well, almost. Unfortunately for all these grand ideas, METAFONT has *no* file-handling capabilities at all! The only files METAFONT generates are the `.tfm`, `.gf` and `.log` files.

This is rather annoying, but fortunately we can steal an idea from Section 7 of the Dirty Tricks appendix in *The* METAFONT*book*. There, Knuth uses the `.log` file as a means of communicating between METAFONT jobs. Similarly, we use the `.log` file as a way of sending messages to TEX. Our `texoutput` macro is defined

```
def texoutput text t =
    for s = t:
        message s & "% diagramf";
    endfor
    message ""
enddef;
```

So `texoutput "Fred", "Ethel"` produces the output

```
Fred% diagramf
Ethel% diagramf
```

You can then use your favourite file-handling utility to filter the `.log` file, keeping only the lines containing `% diagramf`. On my UNIX set-up, for example, I have an alias `diagramf foo` which expands out to

```
touch foo.dim
mf foo
grep "% diagramf" foo.log > foo.dia
echo Labels written on foo.dia.
```

The crucial line in this is the `grep`, which takes all the lines from `foo.log` containing `% diagramf` and puts them in `foo.dia`.

And so we've achieved labelled diagrams in METAFONT. The `diagramf` package is free software, and is available from the Aston archive.

## 3 Acknowledgements

The inspiration, and many of the original ideas, for this article came from Alan Hoenig's talk on the same subject at Cork. I'd also like to thank Jeremy Gibbons and Damian Cugley for comments, advice and allowing me to bounce ideas off them.

◇ Alan Jeffrey
School of Cognitive and
Computing Sciences
University of Sussex
Brighton BN1 9QH
UK
alanje@cogs.sussex.ac.uk



[1] Actually, each line ends with `% diagramf`.