4.1 Digression: Applying Shading to Regions in Metafont

I'd like to briefly digress to indicate the method I used to shade Metafont regions. The crucial idea behind this was suggested to me by Yannis Haralambous.

Suppose we want to apply a striped vertical pattern to a simple rectangle, as schematically illustrated in figure 9. It's helpful to remember that regions created by Metafont are patterns of pixels. In this figure, the absence of a pixel is indicated by a period and the presence of a single pixel by a 1. Metafont, though, can doubly, triply, or multiply write on a pixel so that pixel values may possess other whole number values. Pixels may also be multiply erased, and possess negative whole number values as well. These observations are crucial in what follows.

```
. . . . .
. . . . .
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
. . . . .
. . . . .
 picture
```

```
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
 pattern
```

```
1 . 1 . 1
1 . 1 . 1
2 1 2 1 2
2 1 2 1 2
2 1 2 1 2
2 1 2 1 2
1 . 1 . 1
1 . 1 . 1
  adding
```

```
. . . . .
. . . . .
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
1 . 1 . 1
. . . . .
. . . . .
 culling
```

Figure 9: *Applying a pattern to a region in* Metafont.

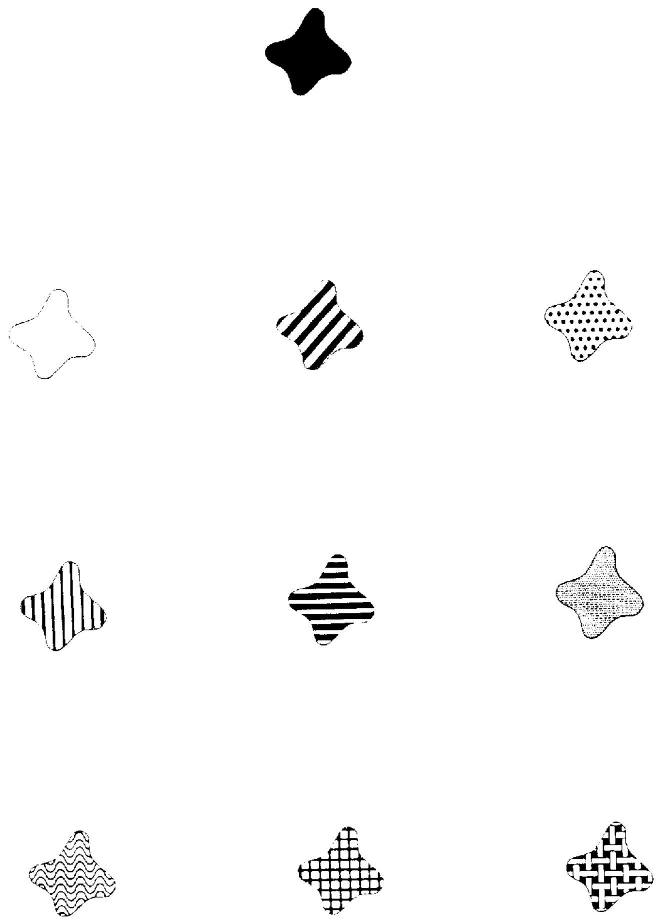

Figure 10: *A region and some background patterns.*

What happens when we add the pattern to the region? To add two pictures together, simply add the corresponding pixel values. Consequently, figure 9 shows the new picture. We see that some pixels now have values of 2, and *it is precisely these pixels that we wish to preserve in the final picture*, as these pixels trace out the striped pattern only in the region of the original shape (in this case, a rectangle). But now our work is done, for we can instruct Metafont to strip away all pixels except those with a value of two. Once this culling is done, we are left with the striped rectangle that we were after, as shown finally in figure 9.

A similar set of operations allows us to outline any region, but as this is treated in *The* Metafont*book*, I will say no more about it here.

There is nothing special about stripes and rectangles; this reasoning applies to arbitrary shapes and arbitrary patterns. I can't resist showing off some patterns. In figure 10, we see a general amorphous shape on which I have drawn an outline and patterns using these operations. Of course, we can use letterforms of a font on which to superimpose these patterns, and you see an example of "Computer Modern Candystripe" in figure 11, which is based on `cminch` (or `cm-half-inch`, to be precise). Figure 14 shows some more fonts. The author of the Computer Modern fonts has provided so many hooks into the programs generating the fonts that very little modification of the existing Metafont programs was needed to generate these fonts.

4.2 Other Work with Labelled Figures

Returning now to my main discussion, I'd like to complete the discussion by referring to other work done in the field of labelling figures.

First is the `diagramf` package developed by Alan Jeffrey. He uses a variant method which is superior in many respects to the one I have just outlined.

Metafont is able to produce only three kinds of files—`gf` pixel files, `tfm` font metric files, and the ubiquitous `log` files. I used the `tfm` to communicate with TEX, but he uses the `log` file in the following manner. The Metafont `message` command displays its argument on screen, but also records it in the log file. If we create a macro that automatically appends a flag to any message, then a simple filter can process the `log` file, collecting all this special `message` data. In a Unix environment, `grep` does this straightforwardly, but even on other platforms, it's simple enough to create a utility to do that for you.