GRAY FONTS FOR METAFONT PROOFS

Gray fonts for METAFONT proofs. (Preliminary draft: May 2, 1984)

The GFtoDVI program converts a GF file into a DVI file that, when printed, gives a hardcopy proof of the characters. The proof diagrams can be regarded as an array of rectangles, where each rectangle is either blank or filled with a special symbol that we shall call x. A blank rectangle represents a white pixel, while x represents a black pixel. Additional labels and reference lines are often superimposed on this array of rectangles; hence it is usually best to choose a symbol x that has a somewhat gray appearance, although any symbol can actually be used.

In order to construct such proofs, GFtoDVI needs to work with a special type of font known as a "gray font"; it's possible to obtain a wide variety of different sorts of proofs by using different sorts of gray fonts. The purpose of this memo is to explain exactly what gray fonts are supposed to contain.

The simplest gray font contains only two characters, namely x and a another symbol that is used for dots that identify key points. If proofs with relatively large pixels are desired, a two-character gray font is all that's needed. However, if the pixel size is to be relatively small, practical considerations make a two-character font too inefficient, since it requires the typesetting of tens of thousands of tiny little characters; printing device drivers rarely work very well when they are presented with data that is so different from ordinary text. Therefore a gray font with small pixels usually has a number of characters that replicate x in such a way that comparatively few characters actually need to be typeset.

Since many printing devices are not able to cope with arbitrarily large or complex characters, it is not possible for a single gray font to work well on all machines. In fact, x must have a width that is an even multiple of the printing device's unit of horizontal position, since rounding the positions of grey characters would otherwise produce unsightly streaks on proof output. Thus, there is no way to make the gray font as device independent as the rest of the system, in the sense that we would expect approximately identical output on machines with different resolution. Fortunately, proof sheets are rarely considered to be final documents; hence GFtoDVI is set up to provide results that adapt suitably to local conditions.

This understood, we can now take a look at what GFtoDVI expects to see in a gray font. The character x always appears in position 1. It must have positive height h and positive width w; its depth and italic correction are ignored.

Positions 2–120 of a gray font are reserved for special combinations of x's and blanks, stacked on top of each other. None of these character codes need be present in the font; but if they are, the slots should be occupied by characters of width w that have certain configurations of x's and blanks, prescribed for each character position. For example, position 3 of the font should either contain no character at all, or it should contain a character consisting of two x's one above the other; one of these x's should appear immediately above the baseline, and the other should appear immediately below.

It will be convenient to use a horizontal notation like 'XOXXO' to stand for a vertical stack of x's and blanks. The convention will be that the stack is built from bottom to top, and the topmost rectangle should sit on the baseline. Thus, 'XOXXO' stands actually for a character of depth $4h$ that looks like this:

blank ⟵ baseline
x
x
blank
x

We use a horizontal notation instead of a vertical one because column vectors take too much space, and because the horizontal notation corresponds to binary numbers in a convenient way.

Positions 1–63 of a gray font are reserved for the patterns X, XO, XX, XOO, XOX, ..., XXXXXX, just as in the normal binary notation of the numbers 1–63. Positions 64–70 are reserved for the special patterns XOOOOOO, XXOOOOO, ..., XXXXXXO, XXXXXXX of length seven; positions 71–78 are, similarly, reserved for the length-eight patterns XOOOOOOO through XXXXXXXX. The length-nine patterns XOOOOOOOO through XXXXXXXXX are assigned to positions 79–87, the length-ten patterns to positions 88–97, the length-eleven patterns to positions 98–108, and the length-twelve patterns to positions 109–120.

Position 0 of a gray font is reserved for the "dot" character, which should have positive height h' and positive width w'. When GFtoDVI wants to put a dot at some place (x, y) on the figure, it positions the dot character so that its reference point is at (x, y). The dot will be considered to occupy a rectangle

$(x+\delta, y+\epsilon)$ for $-w' \le \delta \le w'$ and $-h' \le \epsilon \le h'$; the rectangular box for a label will butt up against the rectangle enclosing the dot.

All other character positions of a gray font (namely, positions 121–255) are unreserved, in the sense that they have no predefined meaning. But GFtoDVI may access them via the "character list" feature of TFM files, starting with any of the characters in positions 1–120. In such a case each succeeding character in a list should be equivalent to two of its predecessors, horizontally adjacent to each other. For example, in a character list like

53, 121, 122, 123

character 121 will stand for two 53's, character 122 for two 121's (i.e., four 53's), and character 123 for two 122's (i.e., eight 53's). Since position 53 contains the pattern XXOXOX, character 123 in this example would have height h, depth $5h$, and width $8w$, and it would stand for the pattern

xxxxxxxx

xxxxxxxx

xxxxxxxx
xxxxxxxx

Such a pattern is, of course, rather unlikely to occur in a GF file, but GFtoDVI would be able to use it if it were present. Designers of gray fonts should provide characters only for patterns that they think will occur often enough to make the doubling worthwhile. For example, the character in position 120 (XXXXXXXXXXXX), or whatever is the tallest stack of x's present in the font, is a natural candidate for repeated doubling.

Here's how GFtoDVI decides what characters of the gray font will be used, given a configuration of black and white pixels: If there are no black pixels, stop. Otherwise look at the top row that contains at least one black pixel, and the eleven rows that follow. For each such column, find the largest k such that $1 \le k \le 120$ and the gray font contains character k and the pattern assigned to position k appears in the given column. Typeset character k (unless no such character exists) and erase the corresponding black pixels; use doubled characters, if they are present in the gray font, if two or more consecutive equal characters need to be typeset. Repeat the same process on the remaining configuration, until all the black pixels have been erased.

If all characters in positions 1–120 are present, this process is guaranteed to take care of at least six rows each time; and it usually takes care of twelve, since all patterns that contain at most one "run" of x's are present.

Fonts have optional parameters, as described in Appendix F of *The TEXbook*, and some of these are important in gray fonts. The slant parameter s, if nonzero, will cause GFtoDVI to skew its output; in this case the character x will presumably be a parallelogram with a corresponding slant, rather than the usual rectangle. METAFONT's coordinate (x, y) will appear in physical position $(xw+yhs, yh)$ on the proofsheets.

Parameter number 8 of a gray font specifies the thickness of rules that go on the proofs. If this parameter is zero, TEX's default rule thickness (0.4 pt) will be used.

The other parameters of a gray font are ignored by GFtoDVI, but it is conventional to set the space parameter to w and the xheight parameter to h.

For best results the designer of a gray font should choose h and w so that the user's DVI-to-hardcopy software will not make any rounding errors. Furthermore, the dot should be an even number $2m$ of pixels in diameter, and the rule thickness should work out to an even number $2n$ of pixels; then the dots and rules will be centered on the correct positions, in case of integer coordinates. Gray fonts are almost always intended for particular output devices, even though 'DVI' stands for 'device independent'; we use DVI files for METAFONT proofs chiefly because software to print DVI files is already in place.

Editor's note: The following article by Georgia Tobin was set using TEX82.9999 on Apollo microcomputers, and printed on an Imprint-10 laser printer at 240 dots per inch. Ms. Tobin hopes that it is clear to TUGboat readers that this is a "font-in-progress". Her article "Computer Calligraphy", which appeared in vol. 4, no. 1, described the design and construction of a Copperplate script font.