Generating an *APL* Font

Aarno Hohti and Okko Kanerva
University of Helsinki

ABSTRACT. The *APL* language is well known for its peculiar symbols which have inhibited the use of this language in many programming environments. Making *APL* documents of good quality has been difficult and expensive. We describe here a simple way how to use **METAFONT** to generate an *APL* font for TEX by using existing font definitions as far as possible.

Introduction

This note describes an interesting exercise in using **METAFONT** to produce new typefaces by combining letters from standard fonts. As we know, the *APL* language [6] of Kenneth Iverson has never gained the popularity it deserves largely because of its strange symbol set. Indeed, true *APL* users require a special keyboard to support the nonstandard but powerful operator symbols. Moreover, putting *APL* into print has always been a problem, and modern low cost computerized typesetting programs do not usually support *APL* style. TEX can be used to produce high quality printouts for technical text, and it would be desirable to have a possibility to mix in *APL* code. The companion program of TEX — **METAFONT** — provides a full means for a simple generation of an *APL* font for TEX, and the purpose of this note is to inform other people about the result we have obtained at the University of Helsinki. Let us note that there are at least three other *APL* fonts available for small computer environments. Indeed, the newsletter *APL Quote Quad** is produced by using Troff, and there exists a PostScript *APL* font for the Apple LaserWriter [5] and another font for the TEXT typesetting system [2].

Consider the following usual kind of function definition in *APL*:

```
       ∇S←SUMSQ N;I
[1]    S←I←0
[2]    →(N<I←I+1)/0
[3]    S←S+I⋆2
[4]    →2
[5]    ∇
```

How to write in this code, providing that we have a suitable font? The *APL* font should represent the screen output style of *APL* code and obey the same laws of spacing. Hence, it should be a typewriter-like typeface with fixed spacing; the same approach for representing TEX input was adopted by Knuth in the TEXbook. The *verbatim* macros have often been used for importing screen or paper outputs into TEX documents; some people misuse them for an easy construction of tables etc. In *verbatim*, the typewriter mode is entered by the control sequence `\begintt` — that mode is ended by `\endtt`. In the same vein, we could enter the *APL* mode by the control sequence `\beginapl`, and to end it by `\endapl`. However, it is more convenient to augment *verbatim* with `aplstyle` so that it can be used with several different typewriter-like fonts. (The *verbatim* macros can be found in the TEXbook, p. 421.) Since `@` (the *at sign*) is used as the escape character inside *verbatim* mode, our TEX code might (and in fact does) look as follows:

```
\choosett{apl}
\begintt
       @DL~S_SUMSQ N;I
 [1]   S_I_0
 [2]   @GO(N<I_I+1)/0
 [3]   S_S+I*2
 [4]   @GO~2
 [5]   @DL
\endtt
```

The control sequences `\DL` and `\GO` are not chosen arbitrarily but follow the conventions used in Digital's VAX *APL* interpreter [1]. As terminals usually do not support the *APL* character set, an alternative representation by two-letter mnemonics is provided by the interpreter. For the most part we have adopted

* Newsletter of SIGAPL, the Special Interest Group for *APL*. Quote Quad has the same status in the world of *APL* as in TEXnical world.

these mnemonics also for our *APL* font. Hence, as an additional bonus the user should find it easy to combine his or her *APL* code with usual TEX code. Thus, it is not necessary for a VAX *APL* user to retype the definitions of *APL* functions in order to be able to use them in documents. However, the syntax of TEX code for *APL* text is somewhat different from that of Digital *APL*. Indeed, mnemonics must be followed by a non-letter (a character whose *catcode* is different from 12) and the escape character is `@`, not period as in the interpreter. Moreover, all spaces are obeyed; thus, if the user does not want a space after an *APL* symbol obtained from a control sequence, the *tilde* character (here a character with zero width) must be used as an end character. Using a macro for removing these differences would make `aplstyle` both slower and unnecessarily complicated.

The *APL* font table

APL symbols are divided into two classes: the primitive symbols and those obtained by overstriking two primitive ones. The overstrikes are traditionally obtained by typing the first symbol, by using backspace to go back one space and then typing the second symbol **over** the first one. However, in modern *APL* keyboards these double symbols are assigned to non-alphanumeric keys (for example, to keys under the `ALT` key). We decided to include only the primitive *APL* symbols in the font table; this enabled us to include also the lower-case letters, following modern conventions. (The original *APL* letters were restricted to capitals.) The comment symbol ⍝ is the only exception since it is keyed in as a double quote. Our font is a fixed size typeface with strongly slanted letters. Moreover, we have followed the style of best books in *APL*: all symbols should be drawn with a thin pen to get a touch of a typewriter. (This point is clearly witnessed, for example, in [5].) The places of some symbols are determined by the TEX font tables. For example, the hash sign `#` is used for the multiplication sign in VAX *APL*, and hence the corresponding symbol has the the same octal code (043) as the hash sign in TEX. The font table has the following form:

	0	1	2	3	4	5	6	7
'000		⎕	∇	∆	⍺	⍵	∊	¨
'010	∨	◊	≤	≥	⍴	⍳	∘	○
'020	∪	∩	⊂	⊃	_	¯	~	≠
'030	⊤	⊥	⊣	⊢	⌊	⌈	→	↓
'040		!	⍝	×	$	÷	∧	'
'050	(	)	⋆	+	,	−	.	/
'060	0	1	2	3	4	5	6	7
'070	8	9	:	;	<	=	>	?
'100		A	B	C	D	E	F	G
'110	H	I	J	K	L	M	N	O
'120	P	Q	R	S	T	U	V	W
'130	X	Y	Z	[	\	]	↑	←
'140		a	b	c	d	e	f	g
'150	h	i	j	k	l	m	n	o
'160	p	q	r	s	t	u	v	w
'170	x	y	z	{	|	}		

The necessary **METAFONT** files

The whole process started when the first author had a paper containing *APL* symbols and was disappointed with the quality of the symbols available on the typewriter. Moreover, the secretary who had typed in the text had forgotten a couple of lines in the middle of the paper, and the correction of such mistakes seemed to be very clumsy in comparison with modern typesetting. Then he decided (together with the second author) to remedy the situation by creating an *APL* font for TEX he was using for other kinds of document. Many of the *APL* symbols needed were contained in standard fonts; for example, *diamond* can be found in *cmsy10*. For alphanumeric characters one could use *cmsltt10*. The simplest try for a solution of the problem would be to write a list of definitions that pick symbols from appropriate fonts. However, this brute force method does not really work since these symbols come from very different typefaces and, moreover, do not provide a fixed typeface. Hence, we decided to find an easy way of producing an *APL* font by using **METAFONT**.

As the starting point, we took the font *cmtex10*. This is a fixed typeface for an extended typewriter-like font including some Greek characters and mathematical symbols. The **METAFONT** file for this font, *cmtex10.mf* contains (as usual) a preamble that assigns values to several global variables, and the command *generate textset;*. Now the **driver** file *textset.mf* contains the commands *mode_setup;font_setup;* (establishing the values of the variables for this font) and several input files from which the **METAFONT** descriptions of the characters are to be found. Since some of these files treat the characters by name and since some of the definitions have to be changed (and some dropped), we considered it advisable to discard the driver file and to collect the separate **METAFONT** files, together with the preamble, to form a large single file *cmapl10.mf*. The *.mf* character files needed for *cmapl10.mf* are the following:

greekl	(rho,omega,alpha)
italms	(iota)
romand	(roman digits)
punct	(punctuation symbols)
romanp	
symbol	(math symbols)
sym	
romanu	(upper case letters)
romanl	(lower case letters)

Definitions

First, put *font_identifier:="CMAPL";* and set *slant:=0;* in the preamble. Many definitions can be copied verbatim from the *.mf* files, but some of them need changes. The Greek *iota*, as given in *greekl.mf*, is strange to *APL* style; we use instead the *dotless i* from *italms.mf*. Moreover, we used the symbol *elt* (element) from *sym.mf* instead of the Greek epsilon. The symbols *del* and *delta* are taken from *symbol.mf* (where their names are *large triangle* and *large inverted triangle*). However, they are too sturdy and too short to be placed in a proper *APL* font. Further, *del* must be lifted up so that it is vertically aligned with other symbols. The modifications are very easy to do, and the modified definition is shown below.

```
% sqrt48 was changed to 6.25 since the
% Del symbol in APL has a narrower top
% than the original reversed triangle
% symbol
% rule.nib has been changed to
%     light_rule.nib
% bot y3=-d-o has been changed to y3=0
% top y1=h-d has been changed to
%     top y1=h+2o

cmchar "Del";
beginchar(oct"002",16u#,asc_height#,0);
adjust_fit(0,0); pickup light_rule.nib;
top y1=h+2o; y2=y1; bot y3=0;
.5[x1,x2]=x3=good.x .5w; w:=r:=2x3;
lft x1=hround(.5w-u*6.25);
draw z1--z2--z3--cycle; % stroke
labels(1,2,3); endchar;
```

As can be seen from this example, the pen strokes were made thinner. Actually only one symbol was directly missing — this is the *quad box*. However, it can be obtained from the above by adding one control point:

```
cmchar "Quad";
beginchar(oct"001",16u#,body_height#,0);
adjust_fit(0,0); pickup light_rule.nib;
bot y1=0; y2=y1; top y3=h+2o; y4=y3;
.5[x1,x2]=x5=good.x .5w; w:=r:=2x5;
% The quad box is slightly wider than Del
lft x1=hround(.5w-u*7);
x3=x1; x4=x2;
draw z1--z2--z4--z3--cycle; % box
labels(1,2,3,4); endchar;
```

After taking care of the special symbols, the letters can be treated by finding a suitable value for *tilt ratio* (slant). Indeed, *APL* letters are **very** slanted. We suggest the value 1/5 for this font (the font *cmsl10* uses 1/6). One should remember to give the command *font_setup;* after setting *slant* to 1/5. Finally, one has to make the *verbatim* macros suitable for *APL* style. Since *verbatim* might be used for several different fonts in one document, we decided to include a control sequence *\choosett*.

```
\def\ifundefined#1{\expandafter
      \ifx\csname#1\endcsname\relax}
\outer\def\choosett#1{\ifundefined{#1}
      \message{Undefined font(?),
               replaced with cmtt10}
      \let\tt=\tentt
    \else
      \def\tt{\expandafter
              \csname#1\endcsname}\fi}
```

(The control sequence *\ifundefined* comes from the TEXbook, p. 308.) With the help of *\choosett*, the standard *verbatim* macro can be used without changes.

The *APL* symbols not in the font table are obtained — as usual — by overstriking two table symbols.

```
\newskip\charwidth
\def\overstrike#1#2{\setbox0=\hbox{#1}%
  \charwidth=\wd0 #1\hskip-\charwidth#2}
```

For example, the *grade up* and *grade down* symbols ⍋ and ⍒ are obtained by striking the *stile* symbol | over ∇ and ∆, respectively.

```
\def\GU{\overstrike{\DL}{\AB}}  % grade up
\def\GD{\overstrike{\LD}{\AB}}  % grade down
```

Now let us take another example of *APL*. Figure 1 shows some input and the resulting output.

User extension

A modern user of a computerized typesetting facility will probably ask if it is possible to extend or modify fonts coming with the system. As with [5], where the font has an *analytic* and a *bitmapped* variant, we can distinguish between the need of modifying *cmapl10* via **METAFONT** and modification of the pixel files. *APL* symbols of various "blackness" or "thickness" may be desirable. Indeed, in [3] the user input is written with boldface *APL* symbols and the answers from the interpreter with thin ones. Furthermore, certain screen previewers use specific small size pixel files, and do not support the *APL* font.

The modification of the font by changing the values of some global variables in the preamble of the **METAFONT** file is easy and can be done by following how it is done in standard fonts such as *cmbx10* etc. This is the recommended way, too. However, if **METAFONT** is not available, then one must attack the pixel files. For direct hand editing, we use a program that converts a pixel file into a (bitmapped) text file acceptable to any standard screen editor, and another program reading the edited file back into a TEX pixel file. Scaling fonts down to a desired size can be done in a similar (but automatic) manner; this facility is needed by a previewer (written by the first author) not using runtime scaling.

References

[1] Digital Equipment Corporation: *VAX-11 APL Reference Manual*, 1983.
[2] Feldberg, Ian: *TEXT: Publication-Quality Characters Come To APL Graphics*, Proceedings of the 1986 *APL* Conference, SIGAPL, pp. 306 – 313.
[3] Gilman, Leonard, and Allen J. Rose: *APL, An Interactive Approach*, John Wiley & Sons, Inc., 1984.
[4] Grenander, Ulf: *Mathematical Experiments on the Computer*, Academic Press, 1982.
[5] Howland, John E.: *Typesetting APL using a Macintosh*, Proceedings of the 1986 *APL* Conference, SIGAPL, pp. 301 – 305.
[6] Iverson, Kenneth: *A Programming Language*, Wiley, New York, 1962.

The address

University of Helsinki
Department of Mathematics
Hallituskatu 15
SF–00100 HELSINKI
FINLAND

Figure 1. An *APL* example

The input

```
\choosett{apl}
\setbox0=\vbox{\hsize=5.5truein
\begintt
    [0]   Z_A1 PROD A2;A;I;V
    [1]   "RETURNS THE PRODUCT OF THE POLYNOMIALS A1 AND A2
    [2]   "THE ARGUMENTS ARE GIVEN AS COEFFICIENT ARRAYS
    [3]   I_@RO~A1
    [4]   Z_A1@SO.#A2
    [5]   LOOP:V_@RO~Z @DM V[(@RO~I)+1]-1
    [6]   V_@RO~Z_Z,[(@RO~I)+1]V@RO0
    [7]   A_(1+-@IO~V[1])@SO.#((@NT(@IO@RO~V)@EP(1,(@RO~I)+1))/V)@RO1
    [8]   Z_+/[1]A@RV[(@RO~I)+1]Z
    [9]   ((@RO@RO~Z)>@RO~I)/LOOP
\endtt
               }
$$\boxit{\boxit{\box0}}$$
\centerline{\sevenrm An APL function for polynomial multiplication}
```

gives the output

```
    [0]   Z←A1 PROD A2;A;I;V
    [1]   ⍝RETURNS THE PRODUCT OF THE POLYNOMIALS A1 AND A2
    [2]   ⍝THE ARGUMENTS ARE GIVEN AS COEFFICIENT ARRAYS
    [3]   I←⍴A1
    [4]   Z←A1∘.×A2
    [5]   LOOP:V←⍴Z ◊ V[(⍴I)+1]-1
    [6]   V←⍴Z←Z,[(⍴I)+1]V⍴0
    [7]   A←(1+-⍳V[1])∘.×((~(⍳⍴V)∊(1,(⍴I)+1))/V)⍴1
    [8]   Z←+/[1]A⌽[(⍴I)+1]Z
    [9]   ((⍴⍴Z)>⍴I)/LOOP
```

An APL function for polynomial multiplication