Problems of the conversion of METAFONT fonts to PostScript Type 1.

Basil K. Malyshev
INSTITUTE FOR HIGH ENERGY PHYSICS
IHEP, OMVT
Moscow Region
142284 PROTVINO
RUSSIA
Internet: `malyshev@mx.ihep.su`

Abstract

The paper describes a problems of automatic conversion of METAFONT fonts into the PostScript Type 1 font format. Several methods of conversion are observed. A short description of *Paradissa Fonts Collection* is presented which contains Computer Modern fonts (conventionally used in (LA)TEX) in the ATM compatible PostScript Type 1 format. The use of the collection and the problems related to it are discussed.

1 Introduction.

Intensive informatization of human activities invokes rapid modifications of methods of data production, processing and use. This requires the adaptation of collected data arrays to efficient modern techniques of information processing.

One of these problems is the conversion of METAFONT defined fonts into PostScript font format. The works on automatic conversion of METAFONT defined fonts to PostScript Type 3 font format have been appearing since 1987 (see *Carr*, *Henderson*, *Hobby*, *Berry*). Hewever, the obtained results are little used due to poor rasterization of PostScript Type 3 fonts at low and middle resolutions on widely used output devices. Furthermore, the fonts are incompatible with ADOBE TYPE MANAGER (ATM) which essentially narrows their application area.

The PostScript Type 1 font format published by *Adobe* in 1990, allows the possibility to perform METAFONT → PostScript Type 1 conversion that promises high quality fonts with wide application area (including ATM).

The methods of automation of such conversion are discussed in this paper.

2 METAFONT and PostScript Type 1.

Comparing METAFONT and PostScript Type 1, the following features should be noted.

METAFONT is a high level language for font description. To draw a letters, METAFONT describes the curve traveled by the center of the pen, and the shape of this pen is allowed to vary as the pen moves. The coordinates of all reference points can be defined by parameters and linear equations. The main advantage of this approach is that the same definition readily yields a family of infinitely many related fonts of type, each font being internally consistent. Thus, infinitely many typeface styles can be obtained from a single definition by changing only a few parameters. METAFONT does not dictate a designer its own font parametrization technique, but provides him with all necessary tools.

METAFONT does not satisfy up-to-date standards of rasterization, because it yields suitable results only at the resolution higher than 300*dpi*. For example, 240×216*dpi* is not good enough yet. At about 100*dpi*, which is typical for many graphic displays, the results seem to be quite unsatisfactory.

PostScript Type 1 is a low level tool for font description. It manages a character by outline description and declarative hints which indicate such important letter features as stems and alignment zones. ATM and PostScript interpreters included into printers and display servers give excellent results by using Type 1 fonts. Furthermore, the font structure of PostScript Type 1 allows a rasterizer to perform rasterization of only actually required characters, and to do it much faster than by METAFONT.

Adobe Type 1 rasterizers can represent fonts at any affine transformation, but this is not sufficient to generate several typeface styles with different optical sizes, weights, widths, etc., from a single description.

To solve this problem, Adobe Corporation has developed a *Multiple Master Extension of the Adobe Type 1 font format* which can be used to generate a wide variety of typeface styles from a single font program. This font format contains several outline typefaces called *master designs* which describe one or more design axes. Design axes represent a dynamic range of one typographic parameter, such as the weight or width.

METAFONT, however, provides font parametrization tools which are more flexible and natural for a designer. Also, new symbols can readily be added to the completed font definition in such a way that they are automatically consistent with the old ones.

Thus, the combination of the flexibility of METAFONT font description with the high quality of rasterization and simple use of PostScript Type 1 fonts seems to be desirable. Such a link can be represented by an automatic METAFONT → PostScript Type 1 convertor generating fonts of rather high quality.

3 Steps and Methods of MF → Type 1 Conversion.

The task of the conversion of METAFONT defined fonts into PostScript Type 1 font format may be divided naturally into two rather independent steps:

- **extraction of character outlines** from the METAFONT font definitions (see section 3.1);
- **generation of declarative hints** which are used by Type 1 rasterizer to fit character outlines to a discrete grid (see section 3.2).

Both steps can be solved by using different methods discussed below.

3.1 Extraction of Character Outlines. Each character in the PostScript Type 1 font is described by an outline specified by a set of lines and Bezier cubic curves. For a Bezier cubic curve four points are used: start point (P_1), endpoint (P_4), and two control points (P_2 and P_3), as shown in figure 1. The tangent vectors of endpoints are determined from the line segments P_1, P_2 and P_4, P_3. The algebraic equation for this curve is:

$$P(t) = (1-t)^3 P_1 + 3(1-t)^2 t P_2 + 3(1-t)t^2 P_3 + t^3 P_4$$

for $0 \le t \le 1$.

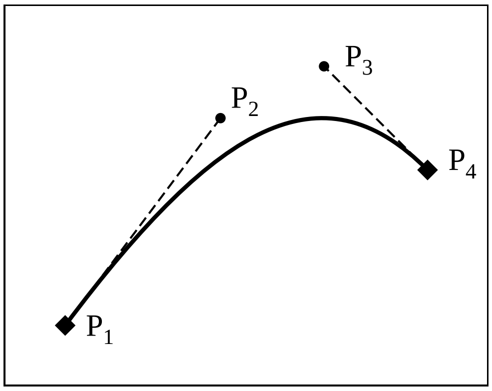

Figure 1: Bezier cubic curve defined by four points.

Note that labelling used in figure 1 for endpoints and control points will be conserved afterwards for all elements forming character outlines.

To produce high quality PostScript Type 1 fonts, character outlines should not only describe letter shapes with sufficient accuracy but also satisfy some important rules formulated in *Adobe PostScript Type 1 font format.*

A *Points at extremes.* An endpoint should be placed at most horizontal or vertical extremes. This implies that most curves should not include more than 90 degrees of arc.

B *Tangent Continuity.* Whenever one path element should make a smooth transition to the next element the endpoint joining the two elements and the Bezier control point associated with that endpoint (for curve) or the other endpoint (for line) should all be colinear.

C *Consistency.* All character features (stem width, height, spacing, shapes) that are intended to be the same should be exactly the same.

D *Conciseness.* Character outline definitions should be as concise as possible, without breaking the other rules.

There are two main approaches to the extraction of character outlines:

- generation of character raster images by METAFONT with subsequent rediscovering the outlines by **tracing the pixels**, and the approximation of the resulted outline by lines and Bezier cubic curves (see section 3.1.1);
- **direct extraction of outlines** from the METAFONT program with subsequent removal of overlapping elements and geometric optimization of resulted outline (see section 3.1.2).

3.1.1 Tracing the pixels of the bitmaps. At first, we need to generate bitmapped fonts by METAFONT. In PostScript Type 1 font format the endpoints and control points are defined on a 1000 × 1000 grid. To avoid rounding errors after outline scaling, magnification should be chosen so that font

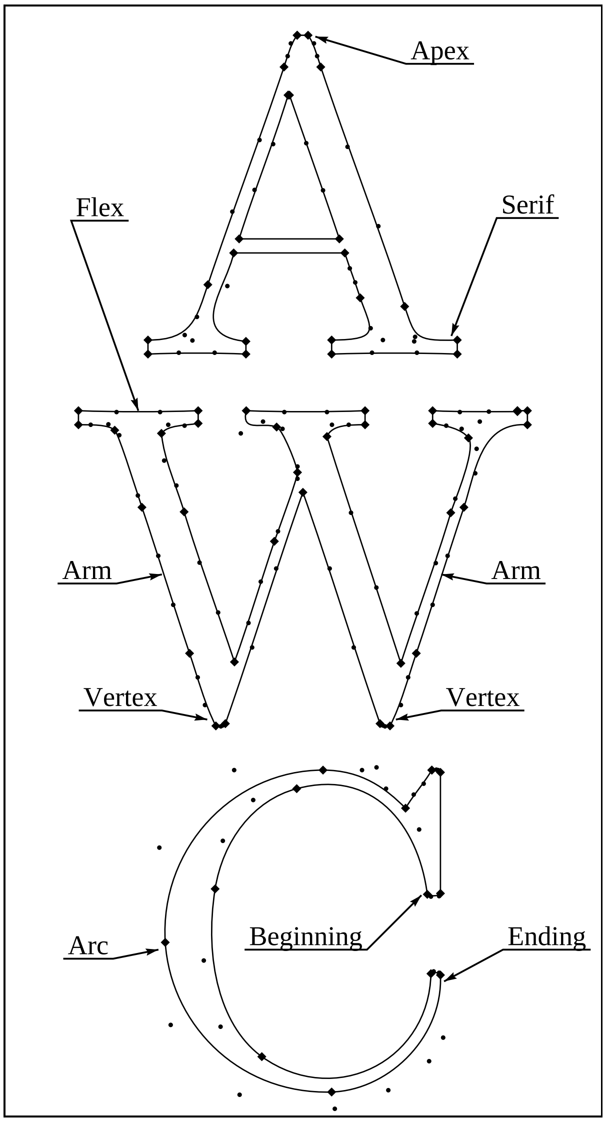

Figure 2: Neil Raine and Graham Toal CMR10 outlines

design size is rasterized into 1000 pixels. Therefore, the resolution should be chosen as

$$\text{<resolution>}\frac{dots}{inch} = \frac{72.27\frac{pt}{inch} \times 1000 dots}{\text{<design size>}pt}.$$

Thus, the best resoltion for the design size of 10 points is 7227*dpi*. This resolution does not require additional scaling of a resulted outline, and the METAFONT program performs correct coordinate rounding.

It is easy to make METAFONT itself to compute the required magnification by adding

```
pixels_per_inch:=4000 + 3227;
pixels_per_inch:=pixels_per_inch*4pt#;
tmp:=designsize/2.5;
pixels_per_inch:=pixels_per_inch/tmp;
```

to `mode_def` macro definition in the `local.mf` file.

There are many programs intended for the reconstruction of raster image outlines by tracing the pixels. However, font generation requires the outlines to satisfy the rules discussed in section 3.1. Below we consider several results focusing on whether these rules are obeyed or violated.

The software developed by Neil Raine rediscovers the outlines from the bitmaps by tracing the pixels. Graham Toal has generated Computer Modern fonts at 3000*dpi* resolution and produced PostScript Type 3 fonts by using Raine's program. These fonts are available in CTAN in `/tex-archive/fonts/cm/ps-type3` directory. The examples of these outlines are shown in figure 2. It is easy to see that these outlines are not free from some visible defects:

- The violation of rule **3.1.A** in the letter 'C'. This defect is not casual, it is common to most curved stems.
- Asymmetrical *vertices* in the letter 'W' violate rule **3.1.C**.
- Bad transitions from *arms* to *serifs* in the letters 'A' and 'W'.
- A middle *serif* in the letter 'W' is not satisfactory.

Karl Berry and Kathryn Hargreaves have developed GNU FONT UTILITIES, that had been announced in TEXhax (Volume 92, Issues 8 and 17). These utilities contain LIMN program which takes the bitmap fonts and rediscovers the outlines by tracing the pixels. The results of their operation are shown in figure 3.

The outlines presented in this figure also have some visible defects:

- The violation of rule **3.1.A** in the letter 'C'. Similarly to the above example, this defect is regularly repeated.
- Each *flex* is split into many short line segments, violating rule **3.1.D**.
- Multiple consecutive collinear line segments violate rule **3.1.D**.

These outlines, when compared with Toal's ones, show some evident advantages. The apex of the letter 'A' is symmetrical, also the vertices of the letter 'W' are symmetrical and matching to each other.

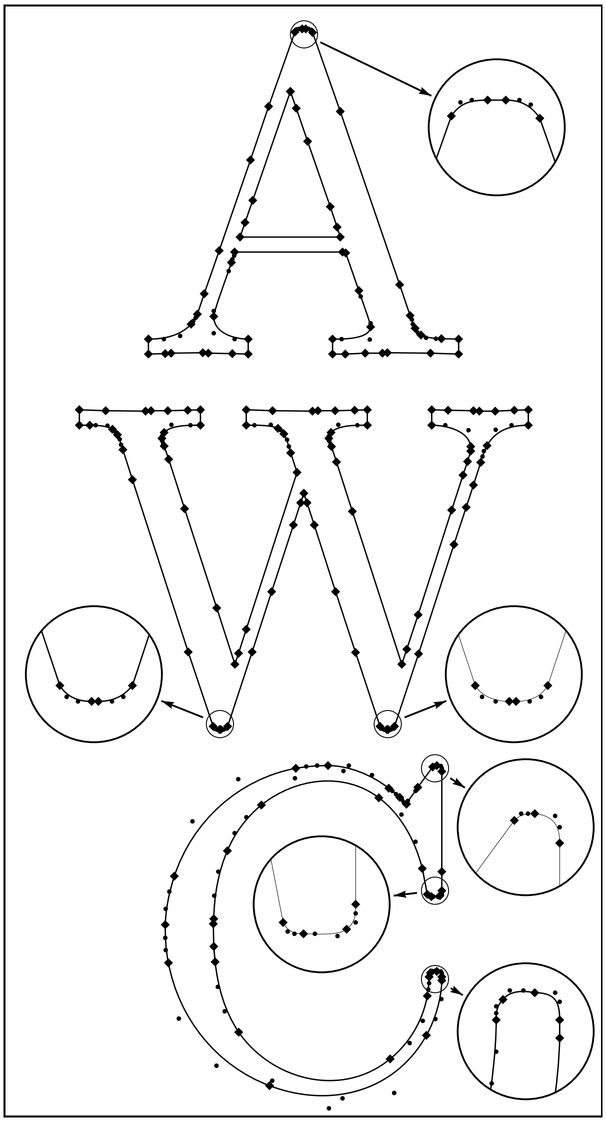

Figure 3: CMR10 outlines produced by `LIMN` from GNU FONT UTILITIES

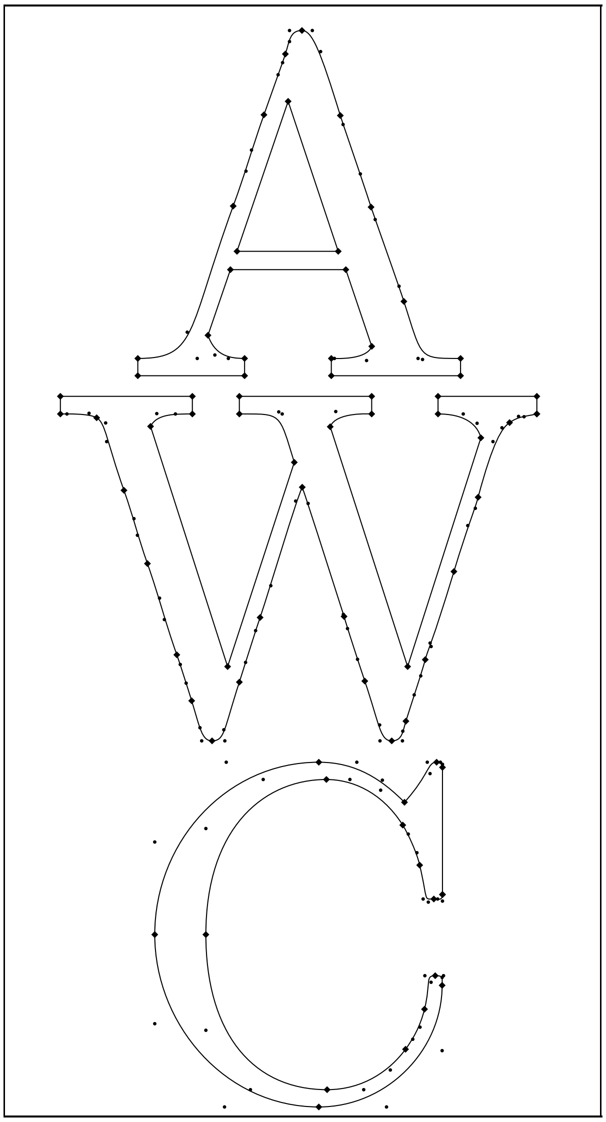

Figure 4: CMR10 outlines produced by FONTOGRAPHER

To make more comparisons, the bitmap font (at 3613*dpi* resolution) was traced by FONTOGRAPHER (version 3.5.1). The resolution $3613dpi = \frac{7227dpi}{2}$ has been chosen because the tracing procedure built in FONTOGRAPHER could have been broken at too high resolution. The result of this experiment is shown in figure 4. In spite of the fact that rule **3.1.A** is satisfied here (it has been violated in the above examples), the outlines are not free from some defects:

- The *apex* in the letter 'A' is asymmetrical (violation of rule **3.1.C**).
- The *vertices* in the letter 'W' are asymmetrical and different (violation of rule **3.1.C**).
- *Arm – serif* transitions in the letters 'A' and 'W' are not satisfactory.

Evidently, it is difficult to obey rule **3.1.C** when directly tracing raster images. Meanwhile, rule **3.1.A** seems to be easy to satisfy, and the violation of this important rule in the first two examples could probably be attributed to lack of authors' attention to it.

3.1.2 Extraction of Outlines from METAFONT**.** Each program considered in section 3.1.1 is not free from its own defects of outline rediscovering. Besides, there are defects common to all programs, such as poor rediscovery of *flex* features or bad *serif–arm* transitions.

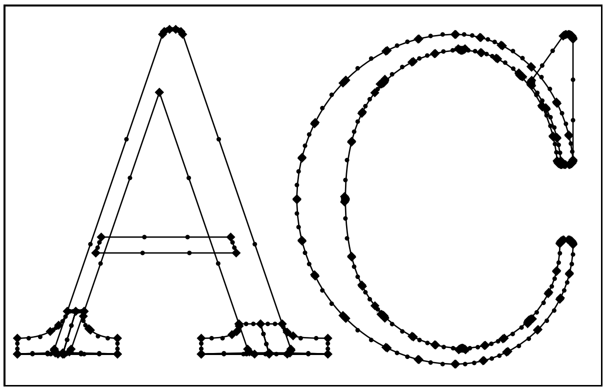

Figure 5: METAFONT internally generated envelopes

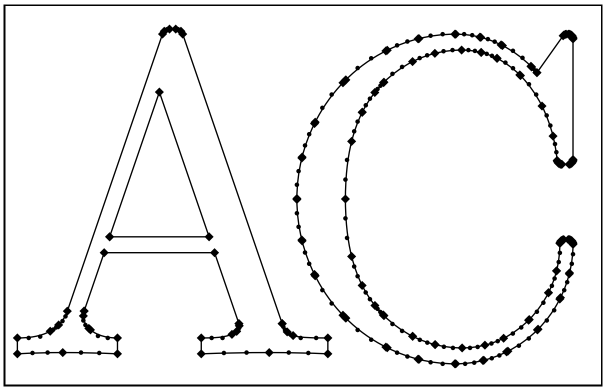

Figure 6: METAFONT envelopes after removing overlappings

The information critical for good appearence of characters is evidently lost when tracing an outline on bitmap. Therefore, the extraction of outlines from METAFONT definitions without raster representation of fonts seems to be more fruitful.

The first attempt of extracting the character envelopes from METAFONT was undertaken by Leslie Carr at 1987. Carr's programs take as input the log output METAFONT file which contains a description of all the paths that METAFONT traces out in drawing character. But using this method one should take into account the METAFONT's pen shape.

Later Shimon Yanai and Daniel M. Berry have developed a more successful program, `mf2ps`, that finds the internally generated METAFONT's envelopes, used as the boundaries of the inked region, and uses these envelopes as the PostScript outlines.

In both these attempts, PostScript Type 3 fonts have been generated. The outlines generated by the `mf2ps` program are shown in figure 5.

To present such envelopes for PostScript Type 3 font, the `mf2ps` program reorders cycled subpaths

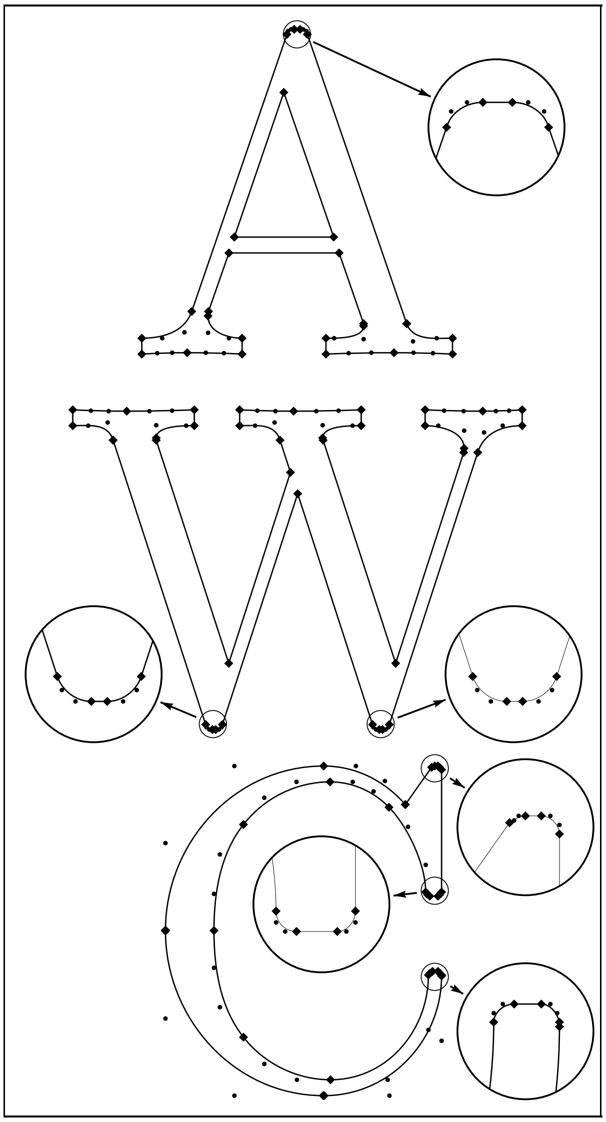

Figure 7: METAFONT envelopes after geometrical optimization

and chooses black and white filling for each of them. This method being suitable for Type 3 fonts, fails for Type 1.

To produce outlines suitable for Type 1 fonts, envelope overlappings should be removed. The result of this operation is shown in figure 6.

Note that the resulted outlines contain too many consecutive lines and curves split into many pieces. To obtain outlines free from such a defect, I have done some geometrical optimization. The result is shown in figure 7.

Now the outlines look significantly better than in figures 2,3 and 4. All the rules from section 3.1 have been satisfied. The excessive fragmentation of

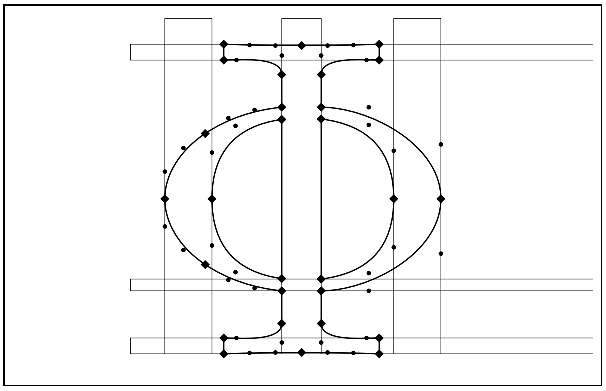

Figure 8: Asymmetrical coding of the like symmetrical form in letter Φ

the inner side of arc in the letter 'C' occurs due to high (probably too high) accuracy of approximation of the original shape.

There are some delicate peculiarities related to different encoding of too close shapes. In the letter 'Φ', for example (figure 8), outline representation is somewhat asymmetrical. The outline have slight asymmetry, but in the process of rasterization its asymmetry of encoding may be exaggerated.

Nevertheless, this method is a step forward in improvement character outlines obtained from METAFONT font definitions.

3.2 Generation of Declarative Hints. One of the main problems arising in font rasterization on a discrete grid is the conservation of important geometrical properties of outlines. Identical parts of the letters differently located on the grid may take different shapes in discrete representation.

In the PostScript Type 1 font format this problem is solved by using *declarative hints* which indicate horisontal or vertical stem occuring in certain coordinates. Those part of the outlines which appear inside of the so–called stem hints, will be rendered by special technique.

The declarative hints for outlines obtained at the previous stage can be generated by font editors like FONTMONGER or FONTOGRAPHER as follows:

- styling the outlines in the form of ATM compatible font format;
- loading the outlines to a font editor;
- saving the completed font on a disk.

As a result, the saved font will be provided with the declarative hints.

The hints generated by FONTMONGER (version 1.0.4) are shown in figure 9. Comparing these outlines with the original ones from figure 3, it is easy to see that FONTMONGER has changed the outlines so that rule **3.1.A** is satisfied. However, the stem hints for the serifs in the letters 'A' and 'W' are missing.

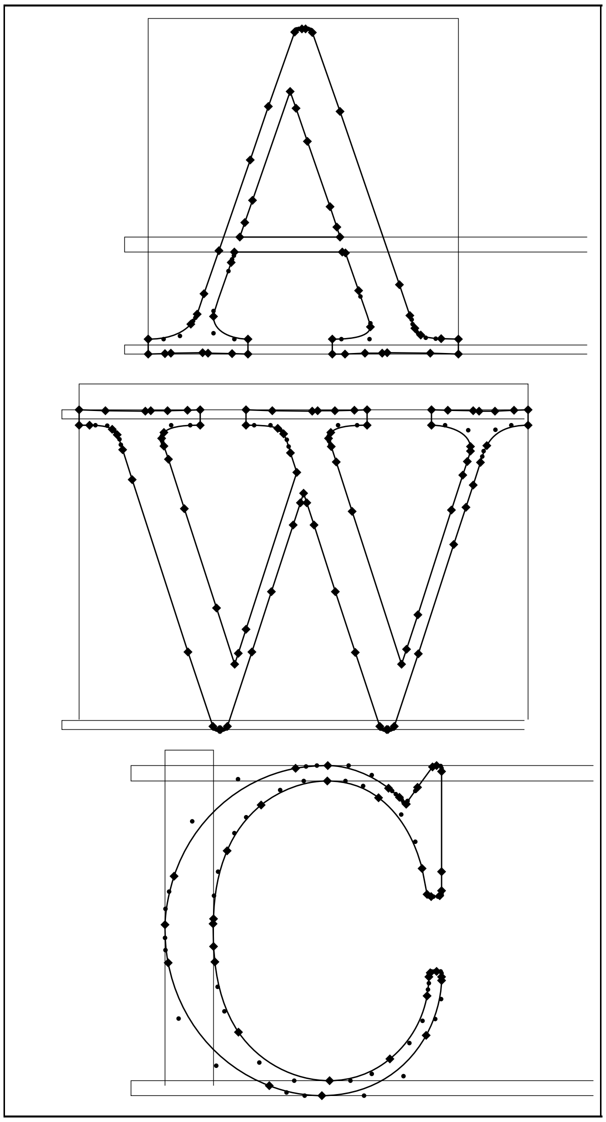

Figure 9: `LIMN` generated outlines hinted by FONTMONGER

The declarative hints generated by FONTOGRAPHER for the same outlines are shown in figure 10. In this figure the outlines are unchanged, while the hints for the curved stems in the letter 'C' are missing. That is, FONTOGRAPHER does not even try to correct the violation of rule **3.1.A**, and this violation has a pernicious effect on its operation. However, unlike the latter case, all hints for the serifs in the letters 'A' and 'W' are found correctly now.

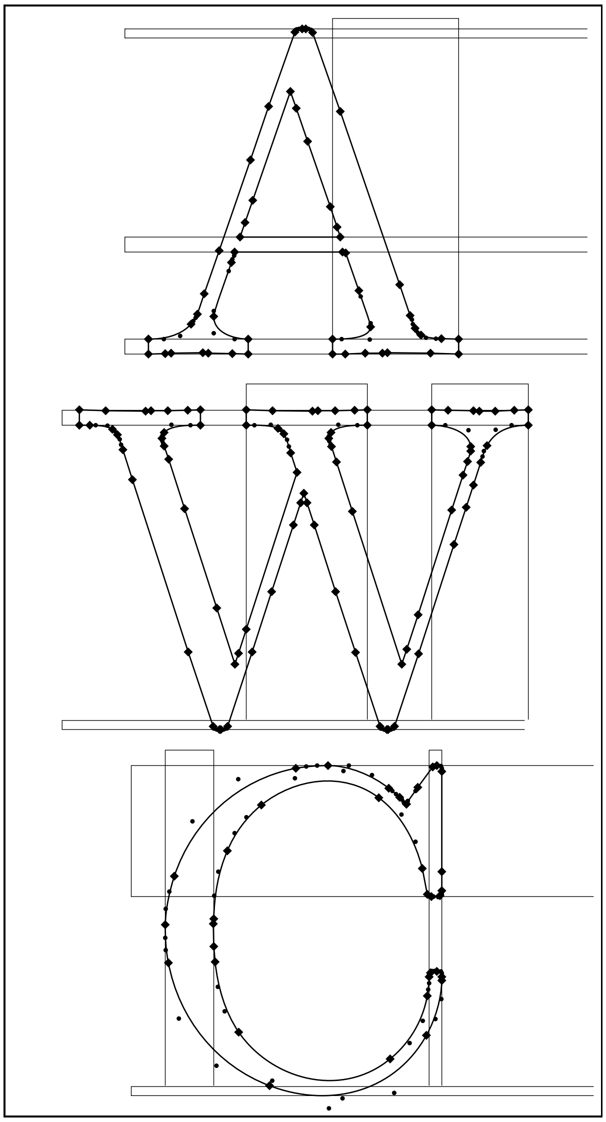

Figure 10: `LIMN` generated outlines hinted by FONTOGRAPHER

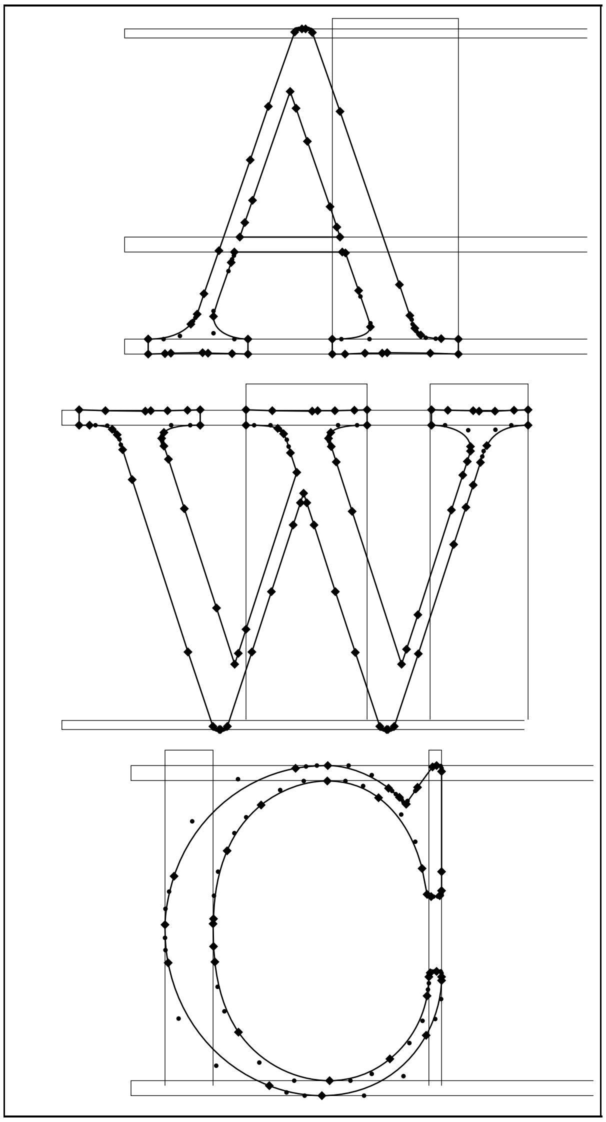

Figure 11: `LIMN` generated outlines hinted first by FONTMONGER and then by FONTOGRAPHER

If the outlines are not too accurate (such as those generated by `LIMN` program), then the operation of both FONTMONGER and FONTOGRAPHER is not good enough. However, if a font processed by FONTMONGER is subsequently processed by FONTOGRAPHER, the obtained results can be significantly better (figure 11).

In the case of more accurate outlines directly extracted from METAFONT (figure 7), this approach is quite suitable, but FONTMONGER can hardly find the serifs (figure 12), while the results obtained by FONTOGRAPHER are quite acceptable (figure 13).

Thus, one can see that FONTOGRAPHER performs hinting somewhat better than FONTMONGER. Still FONTMONGER also has an advantage useful for mass conversion of fonts: it has conversion batch.

Paradissa Fonts Collection has been created by using homegrown algorithm for generating character hints. This algorithm has been developed specially for processing the outlines generated by the `LIMN` program. The result of its operation is shown in figure 14. It can be seen that almost all hints have been found, but the outlines have not been corrected as in the case of running FONTMONGER (figure 9).

In the case of outlines directly extracted from METAFONT (figure 7), the homegrown algorithm gives the results (figure 15) competitive with those obtained by FONTOGRAPHER (figure 13).

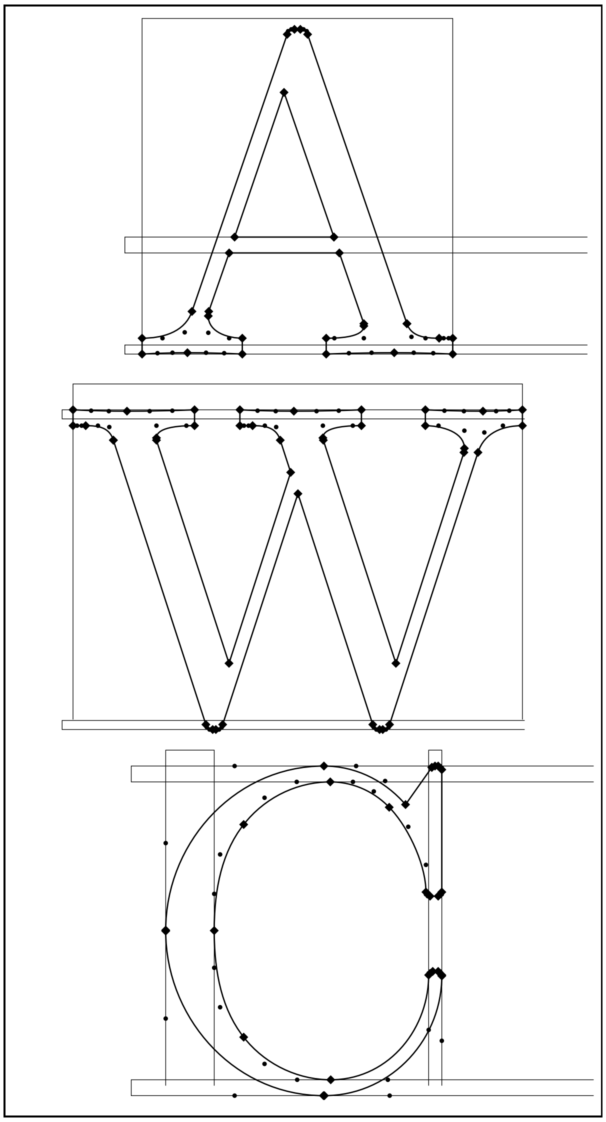

Figure 12: BKM outlines hinted by FontMonger.

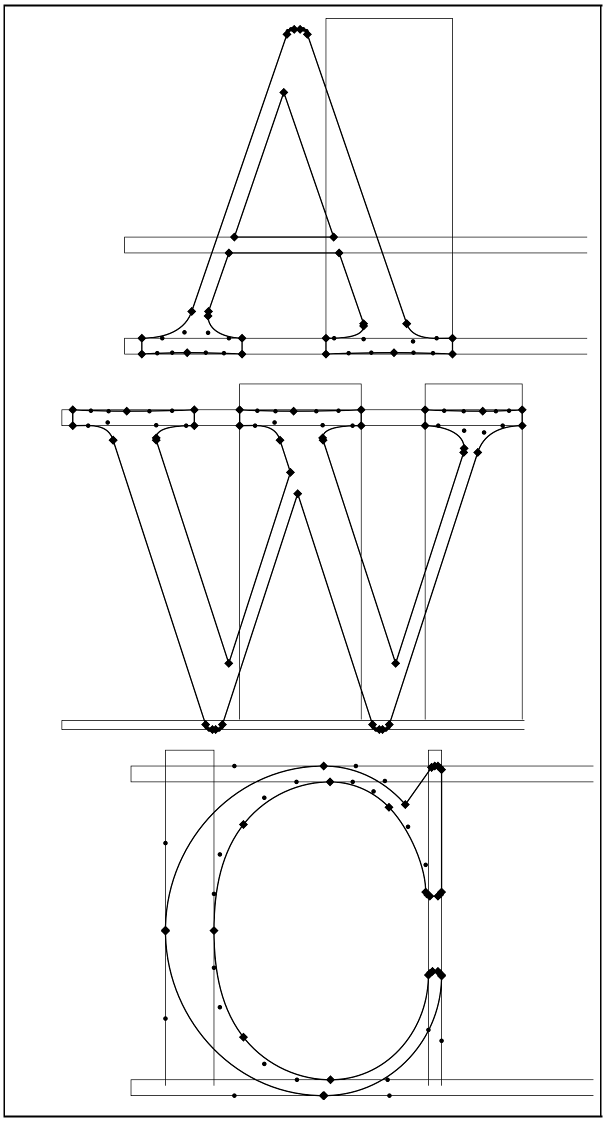

Figure 13: BKM outlines hinted by Fontographer.

4 Paradissa Fonts Collection.

Paradissa Fonts Collection has been developed using the outlines generated by `LIMN` and a specially developed outline filters and hinting algorithm. It is available in CTAN in

`/tex-archive/fonts/postscript/cm`

directory. The examples of hinted outlines contained in this collection have been already presented in figure 14.

Paradissa Fonts Collection contains:

- Computer Modern, designed by D. Knuth;
- Euler by H. Zapf;
- CM Cyrillic by N. Glonty & A. Samarin;
- Special LaTeX fonts.

Totally, it contains 165 fonts in ATM compatible PostScript Type 1 format with `AFM` and `PFM` files. This set of fonts may be used for printing most of (LA)TEX documents. It is used in Russian–English LATEX version developed and supported by ProTEX group at IHEP.

This collection can be used for

- printing documents on a PostScript printer by using, for example, Rokiki's DVIPS. Note, that the typesetting of even a simple LATEX document may require a lot of printer memory to download fonts. To solve this problem a commercial DVIPSONE program may be used that

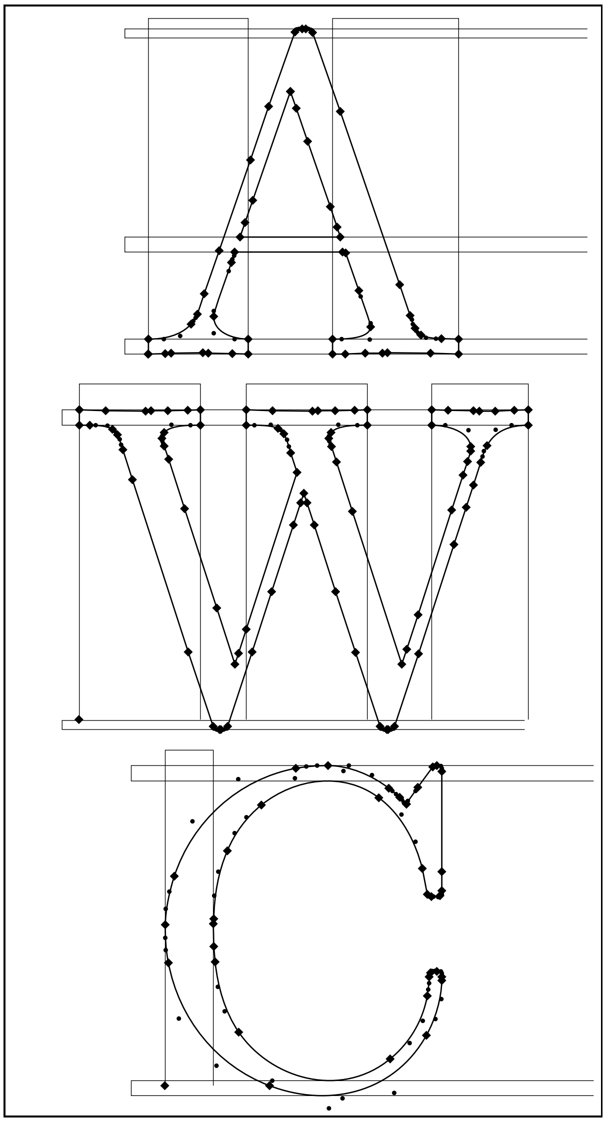

Figure 14: LIMN generated outlines hinted by homegrown hinter.

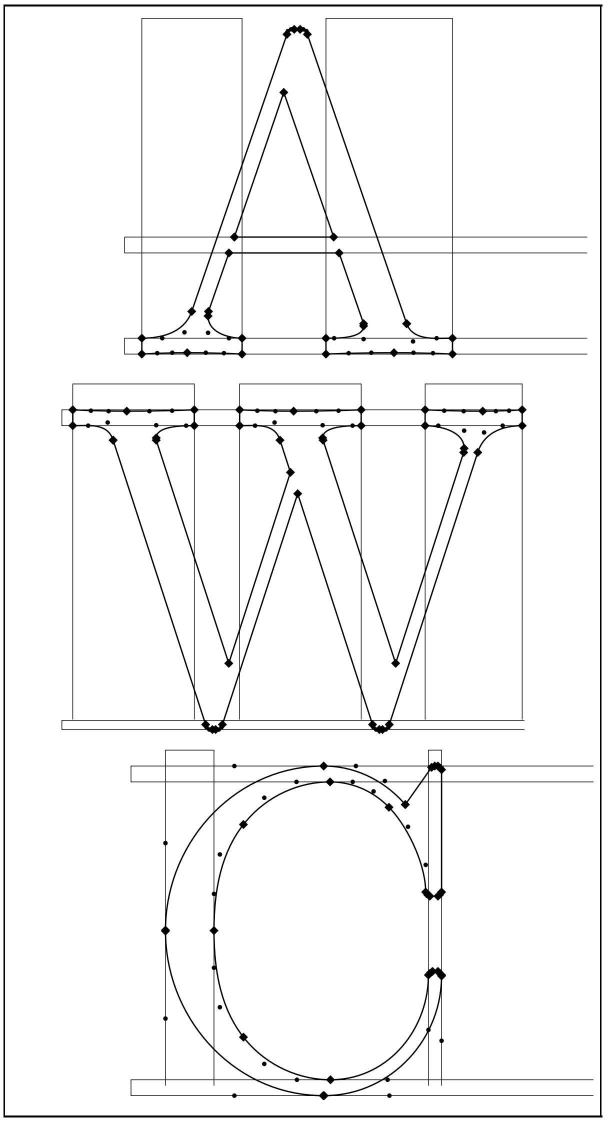

Figure 15: BKM outlines hinted by homegrown hinter.

provides a special technique with *partial font downloading* to conserve printer memory.

- printing documents on a wide set of matrix printers by using DVIPS and GhostScript.
- drawing slides on vector plotters by using PostScript `plot.ps` program which is supplied with the collection. For drawing documents on HPGL plotters, the PS2HPGL utility can be used. It is available in `ftp.mathworks.com` host in the `/pub/contrib/tools` directory.
- displaing documents under MS Windows with ATM installed by using commercial DVIWINDO program. It can be expected that the capability of using Type 1 fonts will be added to DVIWIN by Hippocrates Sendoukas.

5 Bibliography.

Adobe Systems Inc. *PostScript Language Reference Manual*. Addison-Wesley, 1985.

Adobe Systems Inc. *Adobe Type 1 Font Format*. Addison-Wesley, August 1990, Version 1.1.

Adobe Systems Inc. *Adobe Type 1 Font Format: Multiple Master Extensions*. Adobe Developer Support, 14 February 1992.

Berry, Daniel, and Shimon Yanai. "Environment for Translating METAFONT to PostScript." slTUGboat 11(4), p. 525-541, 1990.

Carr, Leslie. "Of Metafont and PostScript." *TEXniques* 5, p. 141-152, August, 1987.

Henderson, Doug. "Outline fonts with METAFONT." slTUGboat 10(1), p. 36-38, 1989.

Hobby, John D. "A METAFONT-like System with PostScript Output." slTUGboat 10(4), p. 505-512, 1989.

Knuth, Donald E. *The* METAFONT*book*. Reading, Mass.: Addison-Wesley, 1986.

Knuth, Donald E. METAFONT *The Program*. Reading, Mass.: Addison-Wesley, 1987.