Font verification and comparison in examples

KAREL PÍŠKA
Institute of Physics, Academy of Sciences
182 21 Prague, Czech Republic
piska (at) fzu dot cz

Abstract

This contribution demonstrates several techniques for verifying and comparing fonts widely used with TEX: METAFONT fonts and outline fonts in the PostScript Type 1 and OpenType formats. The aim is to generate various proofsheet files in PDF or PostScript with node and control points, control vectors and hinting zones for the subsequent visual scanning of graphic glyph representation, calculations of differences between metric data (e.g. character widths), between contour curves for different versions or releases, etc., thus accomplishing the auditing process more quickly and efficiently. Numerous tools — METAFONT, METAPOST, (pdf)(LA)TEX, dvips, gv, FontForge, MetaType1, TEXtrace, mftrace, t1utils, awk, sed, sort and other programs — are used. Resulting differences "greater than negligible" often indicate problems with compatibility, sometimes they may signal a bug undetected even for a long time. The examples are mostly taken from the current TEX Live 2005. The results of verification of CM, EC, LM, CS and other fonts available from TEX Live or CTAN, comparison for compatibility and consistency and the information about differences and bugs will be reported.

Introduction

A short version of the article (low level font oriented and technical) is presented here. After a brief explanation of font elements important for typesetting with TEX (metrics and glyph images) we will show a limited number of illustrations. The motivation of the work was to prepare tools and intermediate results in a textual (lists, tables) and a visual form to find, detect and demonstrate differences, mistakes, cases of inconsistency and incompatibility and, in the next step, to improve past, current or future fonts.

Font types and font data

TFM character dimensions

The tfm (TEX font metric) files contain four dimensions for each character [1]:

- *charwd*, the width
- *charht*, the height above the baseline
- *chardp*, the depth below the baseline
- *charic*, the character's "italic correction"

These define the size of each character's "bounding box" which TEX needs to typeset. For formatting text TEX uses only metric information and does not need glyph shapes. The tfm files also contain information about *ligatures* and *kerning* pairs defining the space adjustment between two adjacent characters. The dvi output produced by TEX contains only references to glyphs. The real glyphs are absent in dvi and are included into the final output in PS/PDF by device drivers (e.g. dvips) or by pdfTEX. The tfm files can be converted by the tftopl program to human-oriented property list files. We can read, edit and process them in this text form more easily. afm is a metric format for Type 1 PostScript fonts. The tfm and afm formats represent and store metric data differently. The afm bounding box has a different meaning, and in afm the glyph width is defined by the WX parameter.

```
ec-lmr10.tfm/pl:
(CHARACTER C y
   (CHARWD R 0.5278)
   (CHARHT R 0.43055)
   (CHARDP R 0.194443)
   (CHARIC R 0.008)

lmr10.afm:
C 121 ; WX 527.77777 ; N y ; B 19 -205 508 431 ;
```

During our processing we check, compare and test for compatibility the metric data, especially the character widths, ligatures and kerning pairs crucial for typesetting, taking into account TEX's font limitations: A font contains at most 256 character codes, 255 different nonzero widths, at most 15 different nonzero heights, 15 different nonzero depths, and 63 different nonzero italic corrections.

Difference between TEX and outline fonts

The 'native' TEX font formats (pk/tfm) have no more than 256 characters, no character names, and any comparison using TEX is based on character sets defined by available encodings. On the other hand, the PostScript Adobe Type 1 fonts (pfb/afm) have glyph names, may contain many glyphs, but only 256 of which can be encoded, their encodings may be flexible, and all glyphs may be compared by names. Additionally, in OpenType (otf) many glyphs are encoded and available. Therefore, operations with an outline font could be independent of the TEX limitations and all glyphs present in the font can be processed.

Font tables and font specimens with TEX

To test completeness of a font's glyph set, we start with font tables and font specimens. For this task, testfont.tex (available from macros/plain/base on CTAN) from the basic TEX distribution, as well as fonttabs (texmf/tex/csplain/fonttabs.tex), and OFS [4] developed by Petr Olšák can be recommended.

METAFONT and bitmap fonts

METAFONT generates the metric tfm files and a bitmap representation of glyphs for a selected device ('mode'). The shapes of bitmap fonts are represented as a bitmap (a matrix of pixels) in *any* resolution. Unfortunately, probably no reference resolution exists. We run METAFONT and dvips with modes available from the widely distributed modes.mf (CTAN: fonts/modes):

```
mf '\mode='$MOD';' input font.mf
dvips -mode $MOD -D $RES
```

and with the corresponding resolutions, for example

```
MOD  300    600     1200    2400    2602    5333
RES   cx   ljfour  ljfzzz  supre   proof    crs
```

to test fonts for all designed sizes and also for various (low, middle and high) resolutions. There is no direct correlation between correctness or incorrectness of shapes in different design sizes or different resolutions (magnifications).

Outline fonts

Outline fonts, e.g. Adobe PostScript Type 1 and OpenType, are represented by their outline contour curves independent of resolution. Of course, rendering always depends on an output device for both outline and bitmap fonts. Good outline fonts that should be rendered properly elsewhere, for example, may be embedded in a PDF document and are more flexible than bitmap fonts because "rerendering" of a bitmap font for any device often causes a loss of quality. The aim of testing outlines is to verify *consistency* (font and glyph elements are unified for all fonts of a font family, preserved for all design sizes) and *compatibility* of versions, changes may be only improvements or corrections of mistakes (not producing new mistakes).

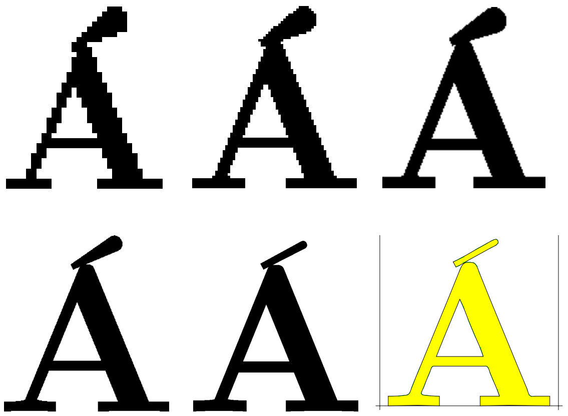

Figure 1: **csbx10 Á** at various resolutions; the shape of the accent changes.

Proofs of METAFONT fonts

csbx10: Á

Our approach is not to prove correctness of METAFONT programs but to check their products — glyphs in the pk format. Because it is impossible to choose one resolution to verify, we test the glyphs at various resolutions. Fig. 1 shows tests of **Á** from the **csbx10** font at following resolutions: 300, 600, 1200, 2602, and 5333 dpi. The last is the result of mftrace (autotracing bitmaps). We can detect a bug in CS fonts (from TEX Live 2005) — serif upper case accents depend on resolution (mode). We thus also observe a consequence of such bugs: the results of conversion to the outline font by autotracing high resolution bitmaps cannot be correct.

cmbx9/cmbx10: y

The next example illustrates behavior of **"y"** in the bold Computer Modern fonts (see Fig. 2 with 600,

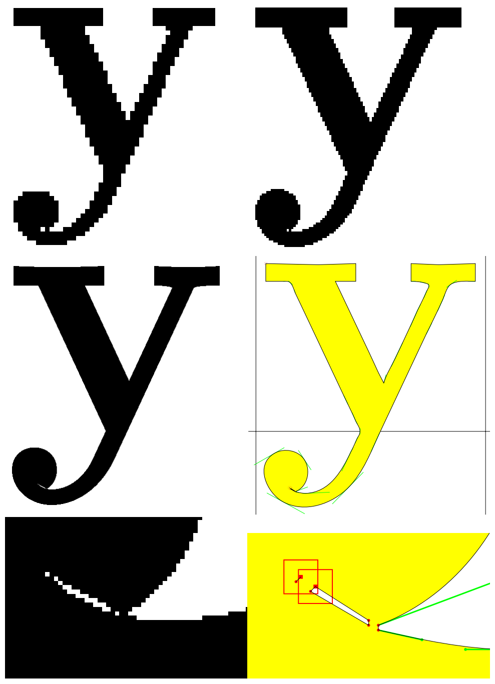

Figure 2: **cmbx10 y**: an artifact in the tail.

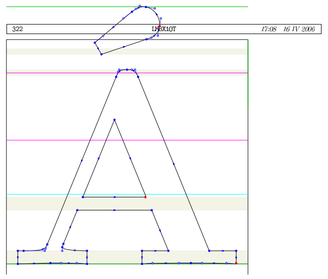

Figure 3: Standard proofsheet from MetaType1.

1200, 5333 dpi, and mftrace) where a strange scrap is generated. This bug in CM is very small, its occurrence is rare and well hidden (only in some sizes), unlike the previous easily evident bug in CS. A similar effect can be observed for cmbx9, cmbx7, cmbx8 (see also [2]). Probably, weaker correlations suppress this defect for cmbx5, cmbx6, cmbx12, and cmbx17.

The role of METAFONT today

METAFONT fonts may be simple or very complex, therefore debugging of a bitmapped glyph representation may be difficult. Although probably the users do not expect bitmap fonts in distributions today, METAFONT and METAPOST are still and will continue to be very important tools for font developers.

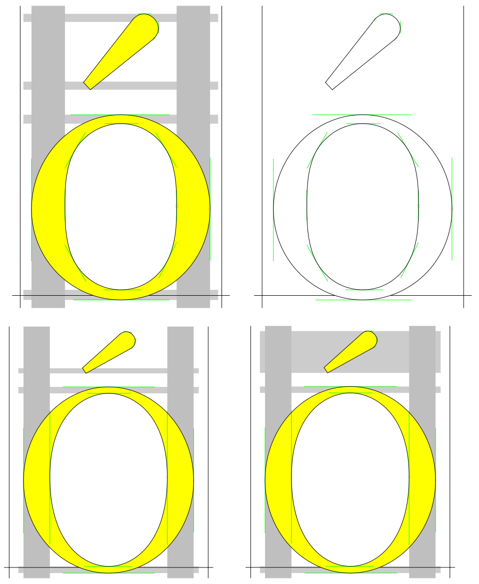

Figure 4: Outline font proofs.

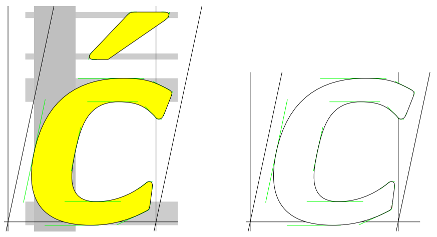

Figure 5: Extrema points and hints.

Proofs of outline fonts

Figure 3 shows the proofing page produced from the Latin Modern sources directly by programs from the MetaType1 package [3]. The following pictures in Fig. 4 demonstrate my own variants of proofsheets for screen and printer to recognize better the tinest details (using zoom) invisible in the previous figure because of mutual overlapping of such small details, e.g. the control points and vectors. Figures 4 and 5 also raise questions about an optimal approximation and a hinting strategy: To add the nodes at extrema or not, how to hint accents, etc.

FontForge [5], an open source font editor developed by George Williams, allows us to read and generate various font formats. We can also use it for checking and analyzing fonts, importing TEX font bitmaps, and exporting glyph outline curves in eps for subsequent processing.

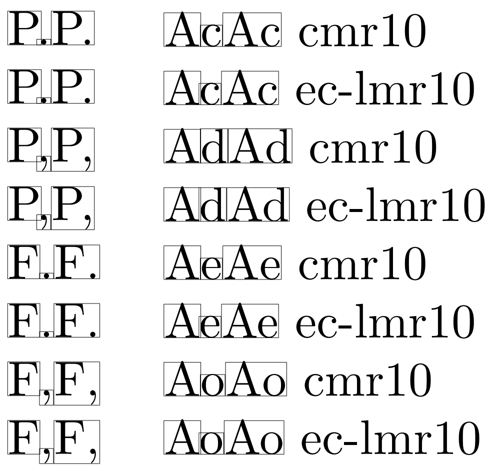

Figure 6: Kerning pairs in a visual form.

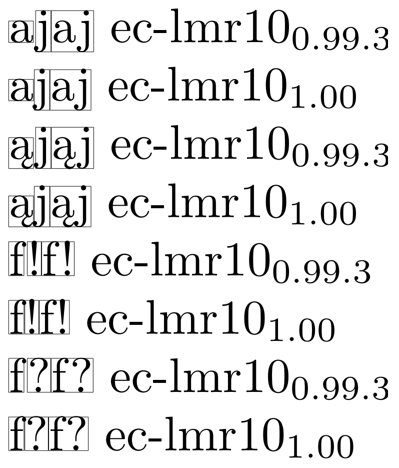

Figure 7: Kerning changes in LM.

Comparison of metric data

Analyzing the metrics we detect (automatically) cases of agreement or disagreement in dimension and kerning values. In Fig. 6 we present a small part of the comparison between the CM and LM `tfm` files in the T1 (ec-lm) encoding. `cmr10` and `ec-lmr10` are compatible in relation to the kerns `"P" : ",","."` and absence of kerns for `"F","T","V","W","Y" : ",","."`.

In `ec-lmr10` new kerning pairs have been introduced: `"A" : "c","d","e","o"`.

Fig. 7 demonstrates the changes between two versions of LM.

```
\newlength{\bbox}\newlength{\cbox}%
\def\fboxsep{0pt}\def\fboxrule{0.1pt}
\def\pair#1#2{%
  \settowidth{\bbox}{#1#2}%
  \settowidth{\cbox}{\mbox{#1}\mbox{#2}}%
  \addtolength{\bbox}{-\cbox}%
  \fbox{#1}\kern-0.2pt\kern\bbox\fbox{#2}%
  \fbox{#1#2}}
\pair{a}{j}
```

A short LaTeX macro `\pair` calculates the shift between two “kerned” boxes.

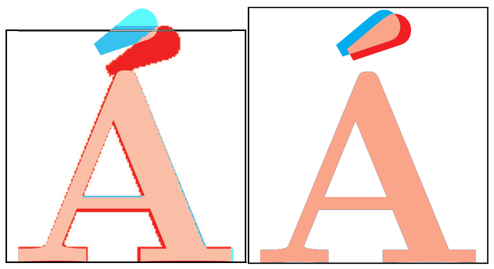

Figure 8: A color mix.

Comparison of glyph images

We will demonstrate two techniques:

- color mix with pdfTeX for visual scan
- outline comparison for outline fonts

Color mix with pdfTeX

Generating of comparative proofsheets using pdfTeX for mixing colors is possible for both bitmapped and outline fonts. Searching for differences needs subsequent human visual postprocessing. In our examples, the combination of red and cyan produces pink in the intersection. Here is the TeX code:

```
\usepackage{graphicx}
\def\Default{\pdfliteral{0 g 0 G}}
\pdfpageresources{/ExtGState
 << /Luminosity
    << /Type /ExtGState /BM /Luminosity >>
 >>}
\def\Acolor{1 0 0 rg}% red
\def\Bcolor{0 1 1 rg}% cyan

\renewcommand\C{\char#1}%
\Default  \fbox{\makebox[0pt][l]%
{\pdfliteral{/Luminosity gs \Acolor}\Afont\C}%
{\pdfliteral{\Bcolor}\Bfont\C}}%
```

Fig. 8 demonstrates data from two samples:

left outline font `lmbx10` (ver. 0.99.3) [cyan] vs. bitmap font `csbx10` [red]

right lmbx10 (ver. 0.99.3) [cyan] vs. lmbx10 (ver. 1.00) [red] (both outline Type 1)

In a grayscale printing the red color looks dark, the cyan is lighter, and the intersection is the lightest.

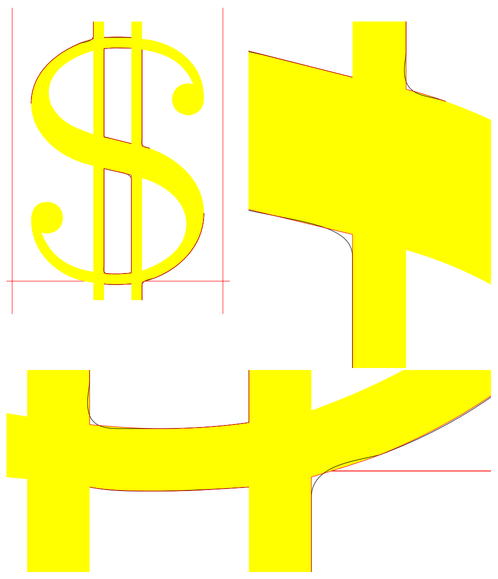
Figure 9: Improvement of outline approximation.

Comparison of outline fonts

Some elements of two outline fonts allow an automatic comparison. We can compare two fonts with a common glyph repertoire, e.g. to test two releases, alternatives, extensions or subsets of any font, or to test two similar fonts for differences. We compare all the glyphs available in both fonts by their names automatically and detect the following differences:

- presence and absence of a glyph with a given name (This may mean that a new glyph has been added or an old glyph has been removed, a glyph has been renamed, or sometimes, a glyph name may be invalid)
- different glyph shapes (at least one segment is different)
- different glyph widths (even after rounding to integer in the glyph coordinate space)

In the examples, the contour curves of the first (older) font are black. The contour curves of the second (newer) font are red and the filled glyph area is yellow. The glyph box frames in the older font are blue. The black and dark lines denote the older version in printing without colors.

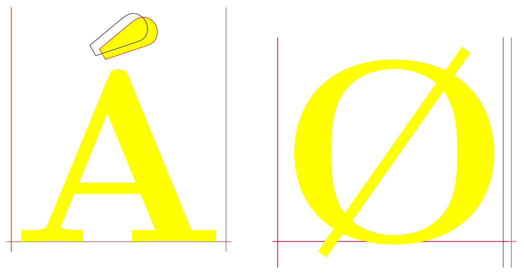
Figure 10: Modification and correction.

Figures 9 and 10 show differences between LM 0.99.3 and LM 1.00. In Fig. 9 the `dollaroldstyle` from `lmr10` has been improved (its conversion to outlines is better). Fig. 10 gives information about the modification of the acute accent and correction of glyph width in the `lmbx10` font.

Conclusion

Techniques extending and complementary to existing testing tools have been presented in various examples, to help font authors and maintainers in their time comsuming and expensive work. The results of tests were or may be applied as warnings, bug reports or suggestions. They were or are used for verification of the Type 1 version of public Indic fonts [6] available from CTAN and for tests of the LM fonts.

References

[1] Donald Knuth. *The METAFONTbook*.

[2] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk. “Programming PostScript Type 1 Fonts Using MetaType1: Auditing, Enhancing, Creating. *TUGboat* 24:3, pp. 575–581, *Proceedings of the XIV EuroTEX 2003 conference*, Brest, France, 24–27 June 2003.

[3] MetaType1 distribution. `ftp://bop.eps.gda.pl/pub/metatype1`.

[4] Petr Olšák. The OFS font management system. `ftp://matf.feld.cvut.cz/pub/olsak/ofs`

[5] George Williams. FontForge: an outline font editor. `http://fontforge.sourceforge.net`

[6] `CTAN:fonts/ps-type1/indic`