

CM-Super: Automatic creation of efficient Type 1 fonts from METAFONT fonts

Vladimir Volovich

Voronezh State University

Moskovsky prosp. 109/1, kv. 75, Voronezh 304077 Russia

vvv@vsu.ru

Abstract

In this article I describe making the CM-Super fonts: Type 1 fonts converted from METAFONT sources of various Computer Modern font families. The fonts contain a large number of glyphs covering writing in dozens of languages (Latin-based, Cyrillic-based, etc.) and provide outline replacements for the original METAFONT fonts. The CM-Super fonts were produced by tracing the high resolution bitmaps generated by METAFONT with the help of $\text{\TeX}trace$, optimizing and hinting the fonts with FontLab, and applying cleanups and optimizations with Perl scripts.

1 The idea behind the CM-Super fonts

The Computer Modern (CM) fonts are the default and most commonly used text fonts with \TeX . Originally, CM fonts contained only basic Latin letters, and thus covered only the English language. There are however a number of Computer Modern look-alike METAFONT fonts developed which cover other languages and scripts. Just to name a few:

- EC and TC fonts, developed by Jörg Knappen, which are the default \LaTeX fonts for the T1 and TS1 font encodings and cover many Latin-based scripts (mainly European).
- EC and TC Concrete and Bright fonts, developed by Walter Schmidt, which are additional font families containing the same glyphs as EC and TC fonts.
- LH Cyrillic fonts, developed by Olga Lapko, which support the family of T2 font encodings: T2A, T2B, T2C, X2, and others. They support the same font families as the original EC fonts, EC Concrete and EC Bright fonts.
- TIPA (International Phonetic Alphabet) fonts, developed by Rei Fukui, which support the T3 font encoding. There exist Concrete (CIPA) and Bright (BIPA) families of the TIPA fonts too.
- FC fonts, developed by Jörg Knappen, which support the T4 font encoding for African languages.
- VNR fonts, developed by Cuong Nguyen, Werner Lemberg and Hàn Thế Thành, which support the T5 font encoding for Vietnamese.
- CBgreek fonts, developed by Claudio Beccari, which support the Greek font encoding (LGR).

There exist free Type 1 versions of the original CM fonts, provided by Blue Sky Research, Elsevier Science, IBM Corporation, the Society for Industrial and Applied Mathematics (SIAM), Springer-Verlag, Y&Y and the American Mathematical Society, but until not long ago there were no free Type 1 versions of other “CM look-alike” fonts available, which limited their usage in PDF and PostScript target document formats. The CM-Super fonts were developed to cover this gap.

Such a conversion from METAFONT to Type 1 fall into one of two general categories. First, base the conversion on analytic study of the METAFONT output (which may include “patching” the METAFONT program, analyzing the output of METAPOST, or similar approaches). Such an approach can give (potentially) the most accurate results, but I did not choose to use it, since it is much harder to develop. (There are some commercial translators of METAFONT to Type 1 which I did not evaluate, mainly due to their “closedness”, which I wanted to avoid.) The second, straightforward, approach is based on tracing the high-resolution bitmaps generated by METAFONT, and thus obtaining outline versions of the fonts.

I was considering several approaches to converting the METAFONT fonts to Type 1 format — it seems that the GNU fontutils package developed by Karl Berry is capable of this, but it had some problems with glyph positioning. Since the appearance of $\text{\TeX}trace$, it became very easy to do this. $\text{\TeX}trace$ is a free automatic converter of METAFONT fonts to Type 1 format, developed by Péter Szabó. It is based on Autotrace by Martin Weber.

And, as far as I know, some code was taken from the GNU fontutils package.

It would be possible to simply generate Type 1 variants of all needed METAFONT fonts (each containing no more than 256 glyphs), but this will have disadvantages: the total number of files will be truly vast (several thousands), and also the total size of these fonts will be much bigger than necessary, because of glyph duplication: Latin letters (and some other characters and glyphs) will be present in more than one font, e.g., the fonts `ecrm1000` (T1 encoding), `larm1000` (T2A), `lbrm1000` (T2B), `lcrm1000` (T2C), `fcr10` (T4), and `vnrms1000` (T5) are all of the same family and shape (Computer Modern Roman) and design size (10pt), and will thus duplicate at least the basic Latin alphabet. Given the total number of font files (several thousands), such duplication will give significant overhead.

Thus, it seems natural to try to combine all fonts which have the same font family, shape and design size, and differ only by font encoding (set of supported glyphs) into *one* SuperFont (the name comes from Karl Berry’s Fontname package), which will contain all the *unique* glyphs from these fonts just once. Then it is possible to create map files for `dvips`, `pdftex` or other programs which will re-encode the big Super-fonts into specific 256-character font encodings, selecting the glyphs needed for a particular font encoding.

Such an approach is undertaken in the CM-Super font package: each Type 1 SuperFont includes all the glyphs from several METAFONT fonts which have the same font family, font shape and design size. Currently, only text font encodings are supported. The name “super” does not imply that this font collection contains Type 1 variants of all existing CM look-alike fonts (e.g., there are no glyphs from math fonts in CM-Super), but that each font in this collection is a SuperFont supporting many glyphs, languages, and scripts. These SuperFonts can be used with other applications as well as \TeX .

2 Font encodings, families, shapes, and names

The New Font Selection Scheme (NFSS) in \LaTeX serves as a very good classifying mechanism in the chaos of various fonts: each font is classified by its encoding, family, shape and design size. The font encoding defines the set (and order) of glyphs which are contained in a font. Currently, the CM-Super font collection supports glyphs from the following \LaTeX font encodings: T1, TS1, T2A, T2B, T2C, X2. In the near future I’ll try to add support for these encodings as well: T3, T4, T5, LGR, and others (e.g.,

additional Cyrillic and Latin glyphs which are not already present in supported encodings).

The original CM fonts used file names such as `cmr10`, `cmbx12`, etc., while their EC analogues use font names consisting of four letters, the first two of which are always “ec” while the second two denote font family and shape: `ecrm1000`, `ecbx1200`, etc. Design sizes are specified differently in the two families: EC fonts use a 4-digit scheme, while CM fonts use two digits. I’ve chosen to use font names of SuperFonts which follow the scheme used in EC fonts (with “sf” instead of “ec”, “tc”, etc.), since the majority of METAFONT fonts included into the CM-Super package follow this naming scheme (EC, TC, LH).

Families and shapes currently supported (the README file in the CM-Super distribution contains a more detailed list):

1. 29 font shapes supported by EC/TC fonts. Each font shape comes in 14 font design sizes ranging from 5pt to 35.83pt (or 11 design sizes for type-writer font shapes ranging from 8pt to 35.83pt), giving $23 \cdot 14 + 6 \cdot 11 = 388$ font files;
2. 13 font shapes for `Sl \TeX` (each comes in one design size);
3. 14 fonts from Computer Modern Concrete family (font file names correspond to the scheme used in EC Concrete fonts, again with “sf” instead of “ec”);
4. 19 fonts from Computer Modern Bright family (font file names correspond to the scheme used in European Computer Modern Bright fonts).

The total number of Type 1 font files included in the CM-Super package is 434.

Each Type 1 font contains glyphs from several METAFONT fonts with the same font shape and design size. For example, `sfrm1000.pfb` combines unique glyphs from the following METAFONT fonts: `ecrm1000.mf`, `tcrm1000.mf`, `larm1000.mf`, `lbrm1000.mf`, `lcrm1000.mf`, `rxrm1000.mf` (for encodings T1, TS1, T2A, T2B, T2C, X2, respectively). In a future version, this font will also include glyphs from `tipa10.mf`, `fcr10.mf`, `vnrms1000.mf`, `grms1000` (for font encodings T3, T4, T5, LGR).

Caps and small caps fonts do not include glyphs from TC fonts (TS1 font encoding), since there are no small caps TC fonts (but it may make sense to include glyphs from the TS1 encoded fonts into these fonts, for completeness, by duplicating glyphs from the corresponding non-smallcaps font shapes).

Our approach provides big savings: if we were making separate Type 1 fonts for each of the above mentioned METAFONT fonts, we would have 256 ·

$5 + 128 = 1408$ glyphs from the five METAFONT sources mentioned; but `sfrm1000.pfb` contains only 585 unique glyphs, which includes some glyphs which were added for the sake of completeness.

The CM-Super fonts come with AFM and INF files and are thus usable with non- \TeX -related applications as multilingual fonts.

3 Details on making the CM-Super fonts

As mentioned above, CM-Super fonts were made by tracing the high resolution bitmaps created by METAFONT. Below is a more detailed description of this process.

First, I created map files which contain each font shape per line corresponding to each METAFONT font which is to be traced. Each map file contains fonts for some particular font encoding (T1, TS1, T2A, T2B, T2C, X2). Also there is a map file for additional glyphs (ellipsis and alternative variant of the sharp “s”).

Also, for each supported font encoding, I created the encoding vector with standard glyph names following the Adobe Glyph List (AGL) conventions (a few glyphs absent from the AGL and even from Unicode were named arbitrarily).

Now we can run the script `traceall.sh` from \TeX trace to make Type 1 fonts from each METAFONT font, using the map files and encoding vectors made in the previous step.

Now we need to combine these small Type 1 font files into Super-fonts. First, we “disassemble” them using `t1disasm` from the `t1utils` package, to convert them into plain text format, convenient for processing. Then we can run a script which combines unique glyphs from the fonts with the same font shape and design size into one (disassembled) Type 1 font. The script not only check for unique names, but also checks that the glyphs with the same names from different fonts (e.g., the Latin letters) are represented identically.

Since we combined several Type 1 fonts into one big font, the `FontBBox` parameter needs to be fixed. This is done using the `pf2afm` PostScript program from the Ghostscript distribution.

We also fix the `isFixedPitch`, `ItalicAngle`, and `Weight` values using a script, since \TeX trace doesn’t set them right. The value of `ItalicAngle` is extracted from the TFM file using the `Font::TFM` perl module.

To make the resulting fonts smaller, we make some cleanups, like removing redundant “`0 hmoveto`” from glyph charstrings dictionary, setting the default font encoding to `StandardEncoding`, setting the value of `lenIV` parameter in the private dictionary to 0, etc.

At this point, we have “raw” Type 1 fonts which should be optimized (which is done later).

Now we gather information contained in the TFM files (which are generated by METAFONT), and apply it to PFB files, and also create AFM font metric files. This in itself involves several steps:

- First, we extract kern values from each of the TFM files, using the script based on the `Font::TFM` perl module, and generate the textual representation of kerns which is used in AFM file format.

Then we combine the kern values from individual TFM files which correspond to the same font shape and design size (but differ by font encoding) into one big kern table. While doing such combining, we always check that there no inconsistent kerns (for the same glyph pairs) in different fonts. A few such inconsistencies were indeed found.

- Now we’d like to make the glyph widths in the PFB files use precise (non-integer) values which better match the values in the TFM files. These widths are generated using the best approximation (based on continued fractions) with the denominator not exceeding 107 to fit in 1 byte in CharStrings (giving space economy), and are stored using the `div` operator in CharStrings. Apparently, such a subtle technique was used first in the BSR/Y&Y CM fonts.

Again, we combine the exact glyph widths obtained from different METAFONT fonts into glyph widths for SuperFonts. And finally, we fix the `hsbw` operators in the PFB files to use the calculated precise glyph widths.

- We extract font parameters (`fontdimen` values) from individual METAFONT fonts. They are converted to the corresponding values like `Ascender`, `Descender`, `XHeight`, `CapHeight` which are stored in the AFM files.
- The ligatures contained in the TFM files are also extracted and put into the AFM files.
- Finally, glyph bounding boxes are extracted from the PFB files using `pf2afm`, and all the pieces obtained in previous steps are combined into final AFM files for the CM-Super fonts.

Now it’s time to optimize the PFB files, since they contain a lot of “junk” control points and do not follow the rules which should be obeyed in good Type 1 fonts. This is the only step which was performed using commercial software: FontLab, but now it is also possible to use PFAedit¹ which may give comparable

¹ This program has since been renamed to FontForge and is available from <http://fontforge.sourceforge.net/>. Ed.

results. The optimization consists of adding nodes at extremes, removing overlaps in contours, optimizing contours (removing unnecessary control points, simplifying the contours), and also autohinting. We intentionally used only automatic optimization (in packet mode, without human interaction). The aim was to use *totally* automatic conversion of METAFONT fonts to Type 1 format, automatic optimization and hinting, with the best achievable quality of final Type 1 fonts, to be able to re-generate the fonts if necessary (e.g., when a new version of original METAFONT fonts is released). Undoubtedly, there is room for improvement of this approach, which we will attempt in future versions of the fonts. After passing the fonts through FontLab, we perform some cleanup again by disassembling, processing and re-assembling back the fonts.

Finally, we create the INF files using a simple script.

The above description may seem a bit complicated at the first glance, but all steps are performed by running a few simple Perl scripts. Some of them may appear to have wider application, so I'll put them into the distribution at some point.

4 Related works

Some packages have been developed which may be useful in conjunction with the CM-Super font package. First, the `type1ec` package which is analogous to the existing `type1cm` package and makes the EC-based fonts available at any size (as opposed to the set of the standard font sizes defined in the default \LaTeX font definition files). This will work efficiently since the CM-Super fonts are vector fonts and will be preloaded only at the few design sizes scaled appropriately.

The second package, recently developed, is the `cmap` package, to be used with `pdf \LaTeX` , which “hooks” into the low-level \LaTeX font pickup command to preload the CMap resources for the fonts which are used in the document. This adds searching and copying capabilities to the PDF files, by defining the “meaning” (Unicode values) for the

glyphs used in the document. At the moment this works only for Type 1 fonts, since `pdf \TeX` had been ignoring the `\pdffontattr` command for the Type 3 fonts, but this was fixed recently and thus it will soon be possible to make the files “searchable” even if the document uses some bitmap fonts.

Both these packages are available on CTAN.

5 Future work

There are many possibilities for improving and extending the CM-Super package. Unfortunately, I haven't had much time recently to work on it, but hopefully I'll move forward soon. Some of the ideas are written in the `TODO` file in the distribution. Most important are:

- cover more fonts—support some other \LaTeX font encodings: T3 (TIPA), T4 (African writings, FC fonts), T5 (Vietnamese fonts), LGR Greek font encoding (CB-Greek fonts), . . . ;
- make the fonts even more efficient (smaller) by using techniques similar to the ones described in Thanh's paper (putting glyphs and accents into subroutines, and constructing accented glyphs from them instead of putting the whole definition of accented glyph into the font);
- make an ultimate step forward to improve the quality of glyph shapes by using analytic transformation rather than tracing.

6 Acknowledgements

I am grateful to the following people who made this work possible: Hàn Thế Thành who inspired me to make a joint talk at TUG 2003 and who gave me many ideas for improving the package and helped to understand some mechanisms (in particular, how to add the CMap entries into font dictionaries into PDF files); Peter Szabo and Martin Weber and Karl Berry who made it possible to easily generate Type 1 fonts from bitmaps; Yuri Yarmola from FontLab for providing a copy of FontLab; William Adams, Wendy McKay, Robin Laakso and Patricia Monohon for the help with corrections of this article and friendly help before and during this conference.