Latin Modern fonts: how less means more

Bogusław Jackowski
BOP s.c., Gdańsk, Poland
b.jackowski@gust.org.pl

Janusz M. Nowacki
Foto-Alfa, Grudziądz, Poland
j.nowacki@gust.org.pl

Streszczenie

Rzecz będzie o najświeższej wersji rodziny fontów Latin Modern. Aktualnie w skład rodziny wchodzi 57 fontów zawierających średnio ok. 665 znaków, głównie diakrytyków. Nowością jest udostępnienie źródeł (fonty LM zostały przygotowane za pomocą pakietu METATYPE1, bazującego na systemie METAPOST). Dlaczego mniej to więcej postaramy się wyjaśnić w trakcie referatu.

1. Introduction

The Latin Modern project was launched during the 13$^{\text{th}}$ European and 10$^{\text{th}}$ Polish TEX Conference, May 2002, Bachotek, Poland. The aim was to prepare a family of outline fonts, compatible with Computer Modern fonts ([8]), but, unlike CMs, equipped with a rich collection of diacritical characters.

At that time, two solutions to the problem existed: (1) Lars Engebretsen's AE (Almost EC, [3]) family of virtual fonts, based on Computer Modern fonts in PostScript Type 1 format released by AMS; (2) Vladimir Volovich's collection of PostScript Type 1 fonts, CM-Super ([15]). Engebretsen's approach has an important drawback — virtual fonts can be used only with TEX. From this point of view, Volovich's CM-Super fonts would be a better choice. The fonts were produced by Péter Szabó's TEXtrace ([12]) which, in turn, is based on Martin Weber's Autotrace ([16]). Volovich's achievement is really impressive, nevertheless we would cast doubt upon the quality of the outlines of glyphs. This is perhaps the intrinsic drawback of the autotracing approach. Moreover, there is a problem with the size of the CM-Super package. It contains more than four hundred fonts; the size of the PFB files is almost 60 MB. Finally, it is not easy to to repeat the process of the font generation if changes are needed, as manual tuning was involved. (A comprehensive discussion of alternative approaches can be found in [5], [6], and particularly [11].)

Finding the situation unsatisfactory, some representatives of European TEX Users groups decided to prepare yet another family of fonts, Latin Modern, being in a way a continuation of Engebretsen's approach, but going further: the aim was to comprise all existing Latin-based alphabets, not necessarily European. We were invited to lead the project, which we gladly accepted.

2. The initial stage of the LM project

Like Engebretsen, we decided to make use of the Computer Modern fonts in the PostScript Type 1 format, released by AMS. But being bent on working with human readable sources, we decided to employ our (anyway favourite) METAPOST-based program, METATYPE1 ([6], [7]). One of the modules of the METATYPE1 package is a converter from PostScript fonts to METAPOST sources. So, the first step was the conversion of PostScript Type 1 fonts to METAPOST sources that were to be manually adjusted.

The decision was not obvious at all. The main disadvantage of such an approach is the "freezing" of parameterization. As an alternative, we considered a conversion of METAFONT CM sources into METATYPE1-conforming ones. It turned out, however, that this method, although practicable, would be time-consuming. Taking into account that our main goal was the extension of the *standard TEX fonts* with diacritical characters, we abandoned eventually the idea of working with METAFONT sources, although access to the CM parameterization is provided (see section 4.1).

3. Interim stages of the LM project

The LM family of fonts has been developing evolutionary. Our main concern, as already mentioned, was the enhancement of the character set, but, as a result of pressure from users, the number of fonts also increased.

3.1. Serious matters and trifles

The number of glyphs per font grew from less than two hundred to more than six hundred. Also, the number of fonts grew. We started with 50 fonts (following Engebretsen); currently, the LM family contains 57 fonts. Among them are fonts that do not belong to the "Knuthian canon," for example, the bold companion for `cmssq8` and `cmssqi8` (prepared by Pierre A. MacKay) — see [5], p. 67, for the complete list of LM fonts. The increase of the number of glyphs resulted of course in the rapid growth of the number of kern pairs.

The augmentation of the LM family of fonts was certainly the most important part of the whole enterprise. We wrote several tools (mostly `awk` scripts) that helped to control the herd of glyphs and interdependences between them — it is unimaginably difficult to fiddle with dozens of thousands of glyphs and hundreds of thousands of kern pairs by hand.

As we pointed out in [5], the lion share of our time was spent on the struggle against tiny details and exceptions. We were not expected to come up with brand-new concepts. On the contrary, we had to comply with the established practice. This, as it turned out, necessitated looking closely into lots of various aspects.

One can call most of these problems trifles, but their amount, relating of course to the size of the project, created a real problem. Such "trifles" had to be carefully analysed and even if the result of the analysis was simply "let's do nothing," it took time. Listing all details would be impracticable, but we cannot resist to mention just a few in order to demonstrate how seemingly less important things may cause more trouble.

3.2. Detail 1: asymmetry of double quotes

One of such problems turned out to be, somewhat unexpectedly, the asymmetry of double quotes (see figure 1). Observe that single quotes are positioned symmetrically, while double ones are not. This is the CM fonts heritage: the glyphs *quoteleft* (reverse apostrophe), *quoteright* (apostrophe), *quotedblleft* (opening quotes), and *quotedblright* (closing quotes) were designed by Donald E. Knuth ([8], p. 140 – 141 and p. 280 – 281), who decided to introduce asymmetry. But the glyph looking like an English opening quote is used in some languages, for example, Czech and German, as a closing one. Therefore, Czech TeX users introduced a special glyph with differently asymmetric sidebearings (*quotedblright.cs* in figure 1) in their variant of CM fonts. In consequence, the character *quotedblbase* (used as an opening quote, for example, in Czech, German, and Polish) also inherited the asymmetry[1].

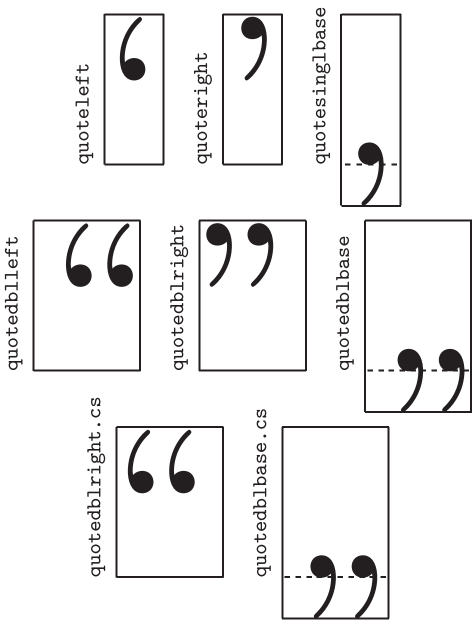

Figure 1: A seemingly innocuous asymmetry of double quotes turned out to be fairly bothersome; PostScript names of the glyphs have been used for the description.

The proliferation of glyphs caused by the asymmetry of *quotedblleft* and *quotedblright* is of course a disadvantage because fonts needlessly swell. We decided, however, to inflate them even more: symmetric quotes were provided as an alternative. We believe that only the latter ones should be used, but because of the remnants of history, the problem cannot be resolved once forever — asymmetric quotes should be retained.

3.3. Detail 2: non-uniform width of accents

Typically, accents should have the same width. This is, however, not the case with CM fonts: *cedilla*, *dotaccent* and *ring* have widths different from the remaining accents, that is, *acute*, *breve*, *caron*, *circumflex*, *dieresis* (umlaut), *grave*, *Hungarian umlaut*, *macron*, and *tilde*, all of which have the same width of $1/2$ em. We cannot say why *cedilla* and *dotaccent* are an exception. The idea behind the extraordinary width of *ring* can be easily understood if one inspects the code of the plain TeX ([9])

[1] For historical reasons, the glyphs *quotedblbase* and *quotedblbase.cs* slightly differ; the latter is placed asymmetrically also in typewriter fonts.

macro `\AA` which typesets the symbol Å (*Aring*). The glyph *ring* is designed to align with the top of the letter *A*:

```
\def\AA{\leavevmode\setbox0\hbox{!}%
  \dimen@\ht0\advance\dimen@-1ex%
  \rlap{\raise.67\dimen@\hbox{\char'27}}A}
```

It is tempting to have a unique width for all accents. But this would mean upward incompatibility, as the `\AA` macro would cease to work. On the other hand, it is not urgently needed, as *Aring* obviously belongs to the repertoire of LM glyphs. Perhaps the cure would be to introduce alternative accents and use the odd-sized ones only with TeX, and only when compatibility is needed, for example, if the LM fonts are to be used as a replacement for CMs (see section 4.2). But, as we complained already in section 3.2, a supererogatory increase of the number of glyphs would be an obvious disadvantage.

The plain TeX macro `\AA` is not the only one that heavily exploits the metric properties of CMs. The macros `\l` and `\L` (which define glyphs *lslash* and *Lslash*, respectively) also are defined in a CM-dependent manner. Both macros rely on the assumption that there is a special glyph in slot 32 (*suppress*; of course, the width of this "accent" glyph is different from a typical one) and that there are specific, unusually large kerns between this glyph and the letters *l* and *L*:

```
\def\l{\char32l}
\def\L{\leavevmode\setbox0\hbox{L}%
  \hbox to\wd0{\hss\char32L}}
```

For example, the respective kern amounts in `cmr10` are -2.78 pt and -3.19 pt, while other kerns are generally in the order of a fraction of a point.

There are two intrinsic problems with *suppress*: (1) it does appear in worldwide standards, such as *Unicode* ([14]) or *Adobe Glyph List* ([2]), although *lslash* and *Lslash* appear there (the *Unicode* standard uses the name *stroke* instead of *slash*); (2) according to ASCII, slot 32 should be occupied by the *space* glyph. The result is easy to predict: most non-TeX fonts will not work properly with plain TeX and most non-TeX software will not work properly with standard TeX fonts...

3.4. Detail 3: bogus x-height in small caps

One might expect that *x-height* (that is, the *ex* unit in TeX) is approximately the height of the lowercase letter *x*. True, but it depends on the accuracy of approximation. The difference between the *x-height* and the height of the lowercase *x* is queerly large in CM caps and small caps (see figure 2). For example, in `cmcsc10` it reaches 0.83 pt. The answer

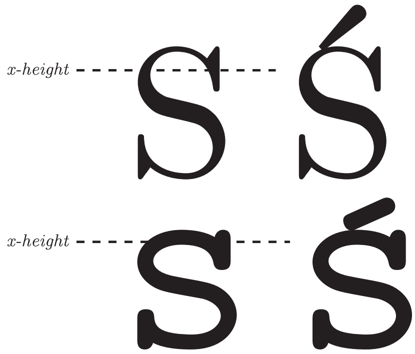

Figure 2: The lowercase letters in CM caps and small caps fonts, that is, `cmcsc10` (top) and `cmtcsc10` (bottom), are higher than the nominal *x-height*; this may befool some typesetting programs which may assume that lowercase letters should be accented without an additional vertical shift of the accent — the potential disastrous results are shown to the right; TeX rises an accent by the difference between the *ex* unit and the actual height of an accentee.

to this riddle is simple: all roman fonts of the CM family have *exactly the same* value of *x-height*; in other words, `cmcsc10`'s *x-height* is the height of the `cmr10`'s lowercase *x*. This discrepancy is harmless, if not advantageous, for TeX, but if the fonts are to be used outside the TeX world, then one may expect weird results if a given system is capable of making composed characters. Nevertheless, we adopted the CM convention for the LM fonts in the hope that if a font is used in a different environment, all necessary characters that could potentially be composed will be already in it.

4. The present stage of the LM project

The project seems to be approaching a development plateau: in comparison with the state of the art reported in [5], the number of fonts was not changed, although the repertoire of characters has been augmented by approximately one hundred glyphs per font — now each font contains circa 650 glyphs. In particular, diacritical characters for Vietnamese and Navajo alphabets have been added — many thanks to Hàn Thế Thành, Karl Berry and Hans Hagen for their warm-hearted help.

As was mentioned, the LM family of fonts developed evolutionary. Everybody knows that evolution is capable to bring forth really bizzare creatures. So were the METATYPE1 sources of the LM fonts after two years of evolution. Two years more — and we would be lost in them. Hence the decision to repeat the initial step: the METATYPE1 sources were once more generated from the now current LM fonts

Polska Grupa Użytkowników Systemu TeX, 2005 (http://www.GUST.org.pl)

in PostScript Type 1 format. Of course, manual tuning was again necessary, as the structure of automatically generated sources is not always fit for a particular purpose. The newly generated sources turned out to be satisfactorily legible, so we decided to release them publicly. Thus, one of the project's main goals was reached.

4.1. Structure of the LM sources — an overview

The LM family of fonts consists of a few general purpose files and files containing specific data for every font (see the listing of a sample driver file below). The data for each font is split into five files that contain:

- metric data,
- PostScript-oriented data,
- encoding data,
- the definition of shapes of basic glyphs,
- the information about ligatures and kerns.

All these files are governed by a single driver that inputs them — see lines 5, 6, 7, 10, and 13 in the listing below:

```
1  % A driver file for lmb10 Latin Modern font
2  input fontbase;
3  vardef cm_pal = "cmb10" enddef;
4  input comm_mac;   % common defs, CM params
5  input lmb10.mpm;  % metric
6  input lmb10.mph;  % PS-oriented header
7  input lmb10.mpe;  % encoding
8  input comm_mph;   % common header
9  beginfont
10 input lmb10.mpg;  % ‘‘frozen’’ glyphs
11 input comm_mpg;   % common glyphs (diacritics)
12 if known generating:
13  input lmb10.mpl; % ligatures and kerns
14 fi
15 endfont
```

There is no parameterization in these files. All entities are defined using bare numbers. The files are assumed to be “frozen” and are not expected to be altered in the future, unless new basic characters are added or severe bugs are spotted. The exception is, of course, the encoding file (line 7) that can be modified as need be.

Each LM font is associated with its CM pal (line 3). The respective CM driver file is being read and its parameters are stored for further use; they are exploited, for example, in the file `comm_mpg.mp` (line 11) by the programs defining the characters depicted in figure 3.

The `comm_mpg.mp` file is actually a “pivot” of the LM fonts. Its main purpose is to define accented glyphs, that is, diacritical characters that can be defined as composites. But not only. The file begins with three peculiar inputs:

```
input gly_euro.mp;
input gly_guil.mp;
input gly_vspa.mp;
```

The files being input are exceptional as they do not define accented characters. They contain a parametric METAFONT-based code for the following glyphs: *euro* (`gly_euro.mp`), *guillemotleft*, *guillemotright*, *guilsinglleft*, *guilsinglright* (`gly_guil.mp`), and *visible space* (`gly_vspa.mp`). The selection of glyphs is more or less arbitrary. The glyphs could be “frozen” as well; however, we decided to leave them in order to demonstrate what the METAFONT code would look like after a manual conversion to the METATYPE1 jargon.

Next, the definitions of letters i and j come. If one is surprised, one shouldn't. After all, the letters i and j are simply *dotlessi* and *dotlessj* accented with *dotaccent*.

Then, the main part of the `comm_mpg.mp` file ensues. It reads as follows:

```
%% \vb\- Aacute:\- \PICT{Aacute}\-
acc_glyph(_A)(_Acute)(_Aacute);

%% \vb\- aacute:\- \PICT{aacute}\-
acc_glyph(_a)(_acute)(_aacute);

%% \vb\- Abreve:\- \PICT{Abreve}\-
acc_glyph(_A)(_Breve)(_Abreve);

%% \vb\- abreve:\- \PICT{abreve}\-
acc_glyph(_a)(_breve)(_abreve);

%% \vb\- Abreveacute:\- \PICT{Abreveacute}\-
acc_glyph(_A)(_breveacute)(_Abreveacute);

...
```

The details of the code are unimportant — the reader is expected to understand what is going on here without arcane knowledge of the METAFONT language. We only mention that the persistently appearing macro `acc_glyph` automatically generates an accented character and that lines beginning with a double percent are meant for the preparing of proofs of a font.

The code looks a bit boring. Indeed, the majority of diacritical characters are composed using the macro `acc_glyph` which roughly corresponds to the TeX `\accent` primitive. In particular, the Vietnamese diacritics are defined in this way (see [4] for the details concerning the Vietnamese alphabet).

Note that there are different accents for uppercase and lowercase letters, for example, *Acute* and *acute*. Note also that double accents in the LM fonts, such as *breveacute*, are not defined using

Figure 3: A group of diacritical characters in the LM fonts for which accents were positioned by hand. Note that although the suffix *caron* misleadingly appears in the glyph names, it is not an element of the respective glyphs.

the macro `acc_glyph`, that is, they are supposed to belong to the basic set of glyphs. They might have been defined as composed objects, but this would increase the complexity of the fonts (for example, *abreveacute* would depend on *a* and *breveacute* and the latter, in turn, would depend on *breve* and *acute*), which we wanted to avoid. Moreover, in some cases subtle adjustments were needed. Therefore, we decided to "freeze" the double accents, once they had been created.

There are, however, several glyphs that cannot be obtained in a simple manner (see figure 3). In the LM fonts, the following glyphs are specially programmed: *dcaron*, *gcommaaccent*, *Lcaron*, *lcaron*, *tcaron*, and *ydotbelow*. A punctilious reader may wish to examine the source code for the details of the implementation.

The final part of `comm_mpg.mp` defines duplicated glyphs, that is, glyphs of the same shape, but different names. For example, we decided to keep the glyphs named *Tcedilla* and *tcedilla* for historical reasons, although their proper names are *Tcommaaccent* and *tcommaaccent* (see [5], p. 70 – 71). Such duplication increases of course the size of a font, but not excessively. As already mentioned in [5] (p. 71), the duplication of a character adds only 30 – 40 bytes to a font. This is done by a METATYPE1 module which compresses PostScript Type 1 fonts. The module defines multiple occurrences of the same PostScript code as subroutines. In particular, whole characters can be defined as subroutines. This means that only the code that invokes these subroutines is to be added. Thus, the duplication of glyphs is moderately harmful which does not mean that it is always reasonable. In future, some of the duplicated glyphs might be deleted.

4.2. Using LM fonts with other tfm files

Obviously, as shown in [5], full LM and CM font compatibility can not be expected, that is, LM metric files cannot be used instead of CM ones. Still, it is possible to use LM fonts as a replacement for a subset of CMs: one should use CM metric files, a few special encoding files and a special font map file for the `dvips` driver. A typical line (broken here into two lines for technical reasons) from the relevant font map file for CM fonts looks as follows:

```
cmb10 LMRomanDemi10-Regular
    "enccmrm ReEncodeFont" <cmrm.enc <lmb10.pfb
```

This line says that TEX should use the metric file `cmb10.tfm` for typesetting, while the `dvips` driver should embed files `lmb10.pfb` and `cmrm.enc` instead of respective files for the CM fonts. Because the dimensions of the glyphs occuring both in the LM and CM fonts are the same (within the accuracy of rounding errors) and glyph shapes are very similar to each other, a user should not notice any difference, unless there is a bug in the LM fonts.

We hope that this solution will prove sufficient in most of practical cases. Similar files are provided for the PL and CS fonts, that is, for the Polish and Czech variants of the CM fonts. At present, work is being done on the support for VNS, that is, the Vietnamese variant of the CM fonts.

4.3. LM fonts in the *OpenType* format

The PostScript Type 1 format is claimed to be obsolete since many years. Actually, all PostScript engines support Type 1 fonts and are expected to support them also in the future. Recently, however, the *OpenType* format becomes a worldwide-accepted standard (see, for example, [10]). We believe that the TEX world should acquiesce to this. Therefore, we also prepared the collection of the LM fonts in the *OpenType* format.

The current release of the *OpenType* LM fonts should be considered experimental, although we gathered some experience during the preparation of the *OpenType* fonts for the *Antykwa Toruńska* family. We employed the *Adobe Font Development Kit for OpenType* (free but not open; see [1]) for the conversion from the PostScript Type 1 to *OpenType* format. An alternative could be *FontForge*, a marvellous openware font program by George Williams (see [17]). Currently, AFDKO better suits our purpose, but as *FontForge* is being constantly developed we hope to switch to it before long.

Polska Grupa Użytkowników Systemu TEX, 2005 (http://www.GUST.org.pl)

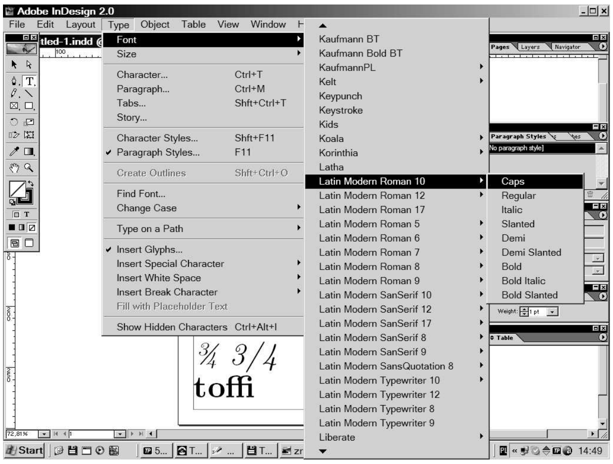

Figure 4: The *OpenType* LM family of fonts as seen by the *Adobe InDesign* program. Note that *Adobe Type Manager* used to accept subfamilies that contained at most four variants of a font, that is, normal, italic, bold, and bold italic. This is no longer the case with *OpenType* fonts — see the list displayed in the right part of the screen shot.

One of the most important innovations introduced in the *OpenType* format are so called *features*. These are tags that provide additional information about how to use the glyphs in a font. So far, five features have been built into the *OpenType* LM fonts:

- `cpsp` (*Capital Spacing*).
- `dlig` (*Discretionary Ligatures*),
- `frac` (*Fractions*),
- `liga` (*Standard Ligatures*),
- `onum` (*Old Style Numerals*).

The availability of these features depends upon application support, for example, the *Adobe InDesign* program under the control of the *Microsoft Windows 98* operating system offers all of them (see figure 4), while *Microsoft Word 2002* in the same system ignores *OpenType* features.

Perhaps the toughest problem is the grouping of LM fonts into subfamilies. The idea of a series of point sizes, as implemented by Knuth in the CM fonts, seems to be athwart the nowadays praxis. Nevertheless, we followed the Knuthian tradition — see figure 4. Feel warned, however, that the adopted grouping may likely change after consultations with experienced *OpenType* users.

5. Conclusions

As one can infer from the title of the paper, our aim was to obtain a product handy in use at the price of abandoning features that — as far as we perceive it — are only moderately usable.

Just two examples:

- There are some fonts in the CM family that we never happened to use: `cmff10` and `cmfi10`. We decided not to include such fonts into the LM family.
- We did not follow the idea underlaying the EC fonts to provide a complete series of font sizes (our arguments are set forth in [5], p. 66), unlike Volovich with his CM-Super fonts.

It is for the users to judge whether the goal was achieved.

Actually, the LM family of fonts in many respects offers simultaneously less and more, not always unequivocally, for example:

- The number of fonts is less than in the CM family, but the repertoire of characters in each font is much larger.
- The LM parameterization is limited in comparison with the CM one, but we expect the potential modifications and augmentations of LMs to be easier, although the LM sources are much larger (6 MB after a compression) than the CM ones. Note that LM fonts take up much less space than the CM-Super ones (but reckoning with the uncompressed LM sources, the sizes become comparable).
- If the LM family would become a basic set of fonts for TEX (which we hope for), then the national variants of the CM fonts (PL, CS, VNS) could be dismissed, which would introduce more order into the TEX font distribution.
- The area of possible applications for the LM fonts is broader in comparison with the TEX fonts available so far, because of the furnishing of the LM distribution with the *OpenType* format.
- Although the LM glyph repertoire is already fairly rich, it can and should be extended further: the next step will be perhaps the addition of glyphs specific for African Latin-based alphabets (cf. [13]). It is not within the scope of the project, however, to include Cyrillic and Greek alphabets.

There is, however, at least one case where more means a not wanted more: the significantly large repertoire of glyphs per font means that the one-to-one correspondence between an LM font and its metric no longer exists and that a multitude of font metric files can be generated for a given font. This abundance is not necessarily what we want. But as long as TEX accepts 1-byte fonts only, the situation cannot be improved. This, however, is quite a different story.

The LM project is not finished yet. As all of us were taught by Donald E. Knuth, the debugging of software is a never-ending task and therefore software projects never end. But apart from fixing bugs, when the LM project reaches the stage of stability of metric data, we will consider the project essentially finished. Having legible sources, we are fairly optimistic — as far as we can assess, the LM project is rather more than less accomplished.

The current version, 0.98, of the LM fonts distribution is available either from CTAN or from `ftp://bop.eps.gda.pl/pub/lm`.

6. Acknowledgements

The Latin Modern project is supported by European TEX Users Groups, in particular by the German-speaking TEX Users Group DANTE e.V., the French-speaking TEX Users Group GUTenberg, the Polish TEX Users Group GUST, the Dutch-speaking TEX Users Group NTG, and, last but not least, by TUG — very many thanks. We would like to express our gratitude also to Harald Harders for preparing and maintaining the web page "Wishes for Latin Modern" (`http://www.harald-harders.de/latex/lmodern.html`) and to Jurek Ludwichowski for his willingly offered help, not only during the preparation of this paper.

References

[1] *Adobe Font Development Kit for OpenType*, `http://partners.adobe.com/public/developer/opentype/afdko/topic.html`

[2] *Adobe Glyph List, ver. 2.0, September 20, 2002*, `http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt` and *Adobe Glyph List For New Fonts, ver. 1.1, April 17, 2003*, `http://partners.adobe.com/public/developer/en/opentype/aglfn13.txt`

[3] Lars Engebretsen, *AE fonts*, `http://ctan.org/tex-archive/fonts/ae/`

[4] Hàn Thế Thành, *Making Type 1 fonts for Vietnamese*, *TUGboat* 24(1), Proc. of the 24th Annual Meeting and Conference of the TEX Users Group, p. 69 – 84

[5] Bogusław Jackowski, Janusz M. Nowacki, *Enhancing Computer Modern with accents, accents, accents*, *TUGboat* 24(1), Proc. of the 24th Annual Meeting and Conference of the TEX Users Group, p. 64 – 74

[6] Bogusław Jackowski, Janusz M. Nowacki, *Programming PostScript Type 1 Fonts Using METATYPE1: Auditing, Enhancing, Creating*, Proc. of 14th EuroTEX, June 24th – 27th 2003, Brest, France, p. 151 – 157

[7] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *METATYPE1: A METAPOST-based Engine for Generating Type 1 Fonts*, Proc. of EuroTEX 2001, 27th – 27th September, 2001, Kerkrade, the Netherlands, p. 111 – 119; the current version of METATYPE1 is available from `ftp://bop.eps.gda.pl/pub/metatype1`; METATYPE1 for Linux prepared by Wlodek Bzyl can be downloaded from `ftp://ftp.ctan.org/tex-archive/systems/unix/mtype13/`

[8] Donald E. Knuth, *Computer Modern Typefaces*, Computers & Typesetting / E, Addison Wesley, 1986

[9] Donald E. Knuth, *plain.tex*, `http://www-cs-faculty.stanford.edu/~knuth/plain.tex.gz`

[10] *OpenType specification version 1.4*, `http://www.microsoft.com/typography/otspec/`

[11] Karel Píška, *Creating Type 1 fonts from METAFONT sources: Comparison of tools, techniques and results*, Preprints for the 25th Annual TUG Meeting, August 30th – September 3rd 2004, Xanthi, Greece, p. 54 – 64

[12] Péter Szabó, *TEXtrace*, `http://www.inf.bme.hu/~pts/textrace/`

[13] Conrad Taylor, *Typesetting African languages*, `http://www.ideography.co.uk/library/afrolingua.html`

[14] *The Unicode Standard 4.0. Final Unicode 4.0 names list*, `http://www.unicode.org/Public/UNIDATA/NamesList.txt`

[15] Vladimir Volovich, *CM-Super Font Package*, `ftp://ftp.vsu.ru/pub/tex/font-packs/cm-super/`

[16] Martin Weber, *Autotrace*, `http://autotrace.sourceforge.net/`

[17] George Williams, *FontForge — An outline font editor*, `http://fontforge.sourceforge.net/`