Accents, accents, accents... — enhancing CM fonts with "funny" characters

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

Janusz M. Nowacki
Foto-Alfa, Grudziadz, Poland
J.Nowacki@gust.org.pl

Pracę zgłosił: Andrzej Borzyszkowski

Abstract

Accented characters play the rôle of *enfants terribles* in the world of computers. Anybody who has to communicate with another computer system in a language other than English knows that using so called "funny characters" is not fun at all.

Those pesky diacritics A giant step towards putting some order into the chaos was the Unicode standard (ISO/IEC 10646) published ten years ago. Unicode, obviously, does not remove all the problems from the font's playground, and even adds a few specific ones (e.g., the problems with the size of fonts or with the registration of non-standard characters and languages). Nevertheless, one can believe that the world will become a bit better when Unicode turns from the standard *de nomine* to the standard *de facto*.

TEX with its 8-bit (i.e., 256 characters per font) paradigm is more and more obsolescent and enhancing it by multi-byte character codes seems unavoidable. Such efforts as Ω Project [11], developed by John Plaice and Yannis Haralambous, cannot be overestimated from this point of view. But the typesetting system itself is only one side of the coin. The other is a collection of fonts it uses.

Originally, TEX was equipped with Computer Modern family of fonts (CM) which did not contain diacritical characters. Those few TEX users who would need accented letters were supposed to employ the `\accent` primitive. The immense popularity of TEX in countries that use lots of diacritical characters proved this presumption invalid. At least three reasons can be set forth: (1) accented characters do not behave like "normal" ones, e.g., they interfere with important TEX algorithms such as hyphenation and insertion of implicit kerns; (2) the CM fonts do not contain all necesary diacritics, e.g., an ogonek accent (used in Polish, Lithuanian, Navaho) is missing; (3) such diacritical elements as cedilla and ogonek, when treated as "accents," overlap with a letter, which precludes some applications, e.g., preparing texts for cutting plotters (see figure 1), even if outline fonts are used. The lesson is obvious — the CM family should be extended by a variety of diacritical letters.

In this paper we would like to present our approach to solving the problem, i.e., the open source family of fonts, *Latin Modern* (LM), in the PostScript Type 1 format [2], prepared using METATYPE1, a METAPOST-powered package [8] (see section METATYPE1). We believe that the LM family is a decent alternative to the existing extensions of the CM family — we expect it to be a handy collection of fonts for typesetting in latin-based alphabets. The fonts are also equipped with *Printer Font Metric* files (`*.pfm`) and therefore can be used as system fonts in various window systems. We plan to release the METATYPE1 sources of the LM fonts during the 24th Annual Meeting and Conference of the TeX Users Group, July 20th – 24th, 2003, Big Island, Hawaii.

A gulp of history Needless to say, the lack of diacritical letters in the CM family was recognized almost from the very beginning by TEX users who had to struggle with the languages other than English. Only in 1990, however, during the TUG meeting in Cork, Ireland, did the international TEX community decide that fonts in so called Cork Encoding (EC or, in LATEX lingo, T1) should be prepared for European TEX users [6]. The work on EC fonts started soon after the Cork meeting. Norbert Schwartz designed a prototype, so called DC fonts. The work was then continued by the team led by Jörg Knappen. The final release of EC fonts was announced in 1997.

Polska Grupa Użytkowników Systemu TEX, 2003 (http://www.GUST.org.pl)

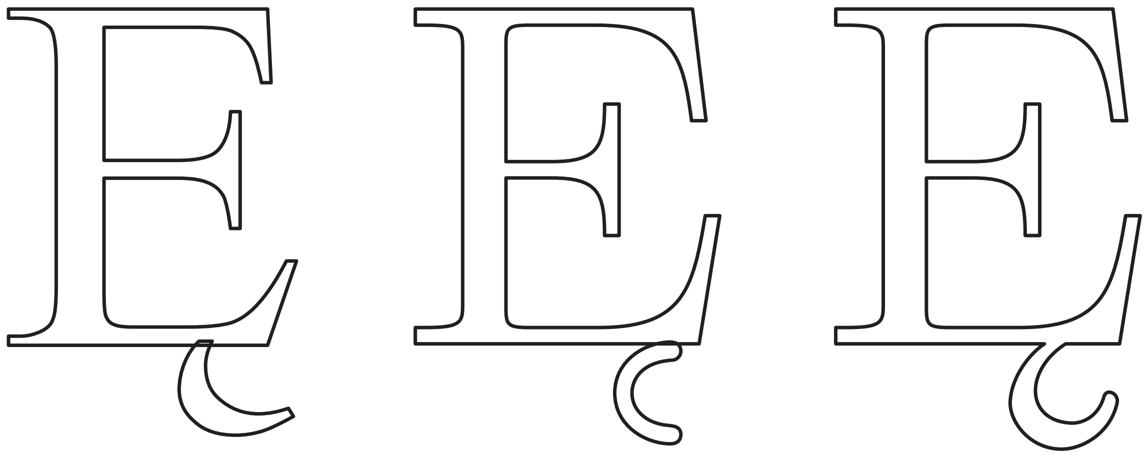

Figure1: The letter *Eogonek* from Times New Roman for Windows XP (left), from `aer10` (middle), and from `lmr10` (right); only the latter form, i.e., having a single outline, is acceptable in professional applications.

It was an important achievement. Nevertheless, the Cork Encoding conformed to TEX's 8-bit paradigm and therefore it was not able to comprise all characters occurring in European languages, not to mention other Latin alphabets, such as Vietnamese or Navaho.

For a few years, EC fonts were available only in a TEX-specific bitmap form (`pk`). Nowadays, in the advent of electronic publishing, bitmaps are hardly acceptable. Outline fonts turn out to display much better on a screen (e.g., when used in Portable Document Format, PDF).

This inspired Lars Engebretsen who prepared a set of TEX virtual fonts containing basic diacritical characters [4]. The virtual fonts could refer to the excellent outline version of the CM family which had appeared in the meantime. It had been created in 1988 by Blue Sky Research for the American Mathematical Society in PostScript Type 3 format, converted in 1990 by Y&Y into the hinted Type 1 format, and released in 1997 into the public domain by the American Mathematical Society. Engebretsen called his collection AE — "Almost EC." His virtual fonts suffer, however, from the same limitation as TEX does, i.e., the number of characters is limited to 256; moreover, as we have mentioned, superimposing a diacritical element on a character reveals undesirable features when the character is stroked rather than filled (figure 1).

Only recently, automatically traced fonts in PostScript Type 1 format, based on the EC fonts, have been published: Péter Szabó's `Tt2001`, Vladimir Volovich's CM-Super (both in 2001; [14] and [15], respectively), and a newfangled Alexey Kryukov's CM-LGC (March, 2003). Note, however, that Szabó courteously "recommends the wonderful CM-Super package instead of his own `Tt2001`." Indeed, Volovich's collection contains much more font variations and covers a broader character set than Szabó's one. Kryukov's collection is, in a way, a supplement to CM-Super. The creation of these packages was possible thanks to a marvellous tool provided by Martin Weber, namely, `autotrace` [16].

Volovich's accomplishment seems to bring to an end the long-lasting endeavours to introduce diacritical characters into the TEX's realm. Do we really need one more collection of fonts?

Another viewpoint

Autotraced fonts, in spite of their many advantages, have drawbacks. Objectively, the most important one is perhaps the size of a font. Such fonts are usually larger than similar fonts having carefully designed outlines because of the greater number of nodes. Compare, for example, fairly tidy Volovich's CM-Super fonts with AMS CM and LM: the number of bytes per character is 260, 200, and 135, respectively. Twice is not too much, but in connection with lots of magnifications included (see section *Too many font sizes*) it makes a difference. Incidentally, the size of the CM-super fonts can be reduced by circa 10 percent by using a subroutine compression module PACKSUBR from METATYPE1 (actually, it is a short `awk` script).

For us, however, more important are arguments of a rather imponderable nature. We stay firmly by the conception underlying the TEX and METAFONT design: *every detail*, be it a typesetting or a typeface design, *should be controllable and replicable*.

This is not the case with autotraced fonts. You must relay, e.g., on the nodes selected by the tracing engine. Volovich admits that `FontLab` program (very good but commercial) was used for improving the fonts, namely, for hinting and reducing the number of nodes; therefore, the process cannot be easily repeated somewhere else. In other words, there are actually no sources for the CM-Super family. The consequence is that `tfm` files have to be generated from `afm` ones (using, e.g., AFM2TFM program) which adds further uncontrolled factors. For example, one cannot suppress overshoots, i.e., characters 'o' and 'x' will usually have slightly different heights, in disaccordance with the original CM design.

Speaking about AFM2TFM converter, please note that it cannot produce mathematical fonts. One has to use METAFONT or METAPOST (or edit manually property lists generated by `tftopl` or `vftovp`) in order to exploit such features as `charlist` or `extensible`. Ignoring this aspect would mean, in our opinion, the waste of TEX equipment for mathematics.

Having said this, we would like to emphasize that we esteem the job Szabó, Volovich, and Kryukov did. Our predilection to another solution may be regarded as a natural, if not advisable, difference of viewpoints.

Too many font sizes There is one more issue, related indirectly to the problem of 'bitmaps versus outlines,' namely, the number of font sizes for a given typeface, or more adequately — proportions. Donald E. Knuth's idea of having different proportions for different font sizes (5, 6, 7, 8, 9, 10, 12 and 17 points) has no precedent in typography. John Sauter attempted to go even further [13]. He prepared METAFONT programs that interpolate (and even extrapolate) Knuth's font parameters to non-integer font sizes. We can accept Sauters's approach as an interesting experiment, admissible for bitmap fonts. Nevertheless, using it for outline fonts is at least controversial.

We believe that, in general, four font proportions would suffice: heading (17 pt), normal (10 pt), script (7 pt), and second order script (5 pt, "scriptscript"). Because of the well-established tradition, we cannot refrain from using Knuth's scheme, but we would strongly discourage extending it.

For these reasons, we accept with difficulty the enormous number of different sizes/proportions both in EC and CM-Super fonts. This is apparently the inheritance of Knuth's and Sauter's ideas. We would gladly discard most of fourteen alterations of a single typeface (corresponding to font sizes 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28, 20.74, 24.88, 29.86, and 35.83 points). The series proposed by Knuth plus TEX `scaled` and `at` operations provide sufficient means to deal with font scaling in most of practical applications.

Too few typefaces If anything, completely new typefaces are needed. The number of fonts prepared with METAFONT is surprisingly small compared, e.g., to what is available on the commercial market. Well, not so surprisingly. As we have already mentioned, METAFONT generates TEX-oriented `pk` bitmap fonts which have not become popular outside the TEX world. In principle, the conversion of `pk` bitmaps into PostScript Type 1 form is possible, as Szabó and others proved. Which does not mean that looking for alternative tools is impractical.

Alternative tools In general, there are two classes of computer tools: visual (interactive) and logical (programmable). Perhaps someday both classes will converge into, say, "visual-and-logical" tools which will prevail, but at present, no doubt, interactive tools are in vogue. The majority of contemporary visual typographic programs are commercial products. Fortunately, George Williams launched (in 2000) an impressive open source project, `PfaEdit` [17]. This font editor is already a powerful tool and, being extensively developed, it promises even more, which countervails the grasping market up to a point. An interesting visual tool for generating PostScript Type 1 fonts is also Richard Kinch's `MetaFog` which enables visual tuning of METAPOST-generated PostScript files [9].

Programming tools are not so popular. Are they to go extinct some day? We hope they will not. It would be a pity, because in some applications programmability is better. Fortunately, there exist people who share our point of view. One of them is Włodek Bzyl who found a plausible application for the logical approach in typography. His amazing colour PostScript Type 3 fonts are no mean challenge for those who use visual tools [3].

Fonts are very complex structures. They are governed by the ample set of interdependent parameters, such as character dimensions, font-specific parameters (italic angle, x-height, typical stems), characteristic shapes (serifs and arcs), not speaking about such technicalities as hints or subroutines. And here an important aspect of programmability enters. By definition, programmable tools require sources in a

lmb10
lmbo10
lmbx10
lmbx12
lmbx5
lmbx6
lmbx7
lmbx8
lmbx9
lmbxo10
lmbxi10
lmcsc10
lmcsco10
lmr10
lmr12
lmr17
lmr5
lmr6
lmr7
lmr8
lmr9
lmri10
lmri12
lmri7
lmri8
lmri9
lmro10
lmro12
lmro8
lmro9
lmss10
lmss12
lmss17
lmss8
lmss9
lmssbo10
lmssbx10
lmssdc10
lmssdo10
lmsso10
lmsso12
lmsso17
lmsso8
lmsso8
lmsso9
lmssq8
lmssqbx8
lmssqbo8
lmssqo8
lmtcsc10
lmtt10
lmtt12
lmtt8
lmtt9
lmtti10
lmtto10
lmvtt10
lmvtto10
lmvtto10

Figure2: The Latin Modern collection of fonts.

human-readable text form. A plethora of text processing utilities around (awk, perl, grep, diff) can therefore be employed to crosscheck the consistency of the data describing the font. This can hardly be achieved with purely interactive programs, although it should be noted that some interactive typographic programs have implemented limited programmability.

METATYPE1 We prefer unlimitted programmability. Being provoked by the irksome scarcity of fonts prepared using METAFONT, we contrived another font generating package, METATYPE1 [8], based on METAPOST, which produces results in the world-wide accepted PostScript Type 1 format. The package makes use of two sets of METAPOST macros (general purpose plain_ex and task-oriented fontbase) and a few other utilities, such as awk (for processing METAPOST output), T1utils (for converting text data into a binary form), and mft (for neat proofing). Originally, METATYPE1 was developed for DOS; thanks to Włodek Bzyl, it is available also for Linux.

Those who are repelled by the sophisticated software and therefore are unwilling to experiment with METATYPE1 may find Han-Wen Nienhuys's opinion encouraging: "METATYPE1 is a very simplistic approach" [10].

One of the first results obtained with METATYPE1 was Donald E. Knuth's logo font and an electronic replica of a traditional Polish font, Antykwa Półtawskiego [7]. We also used METATYPE1 for auditing and enhancing selected fonts from the URW++ collection distributed with Ghostscript.

In 2002, during the TEX meeting in Bachotek, Poland, the representatives of European TEX users group, having discussed the matters on email, came up with a proposal of converting AE virtual fonts into a more universal PostScript Type 1 format and augmenting them with a set of necessary diacritical characters. Thus the opportunity arose to embark METATYPE1 upon a new, unconventional task. We took up the gauntlet without hesitation.

The LM family of fonts or details, details, details...

Our intention was to preserve the AE name, as we wanted to emphasize the rôle of Engebretsen's idea in this enterprise. Soon it became clear, however, that the differences would be fundamental and that the change of the name would be necessary in order to avoid a mess. Therefore, we coined the name "Latin Modern" which is to betoken further development—we would like the final version of LM to comprise as many latin-based alphabets as possible, e.g., Vietnamese which regretfully is not included yet.

The collection of AE fonts consisted of 50 fonts, reasonably selected from the abundance of Computer Modern. We decided to add a variable-width typewriter font and a few oblique derivatives, arriving finally at 57 fonts (see figure 2).

Observe two details:

1. We adopted a more regular (although unorthodox) font naming convention with respect to slant/italic variants: we preserved the 8-character limit but we have used the letter 'o' as a suffix for oblique (slanted) fonts and the letter 'i'—as a suffix for truly italic fonts.

2. The LM family contains font lmssqbx8 (i.e., the bold version of lmssq8); a corresponding font occurs neither in CM nor in EC. Actually, the respective AE fonts (aessq8, aessqi8, and aessqb8) refer to the fonts lcmss8 lcmssi8, and lcmssb8. These fonts, added by Pierre A. MacKay, were meant for using with SliTEX. Their regular variants are nearly identical with Knuth's cmssq8 and cmssqi8. The only difference is the capital 'I' (see figure 3).

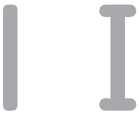

Figure3: The letter *I* from Knuth's cmssq8 (left) and MacKay's lcmss8 (right).

The issue of font names was triggered by the slanted fonts that we decided to add: what the name should we assign to the oblique variant of lmvtt10? The name lmvttsl10 did not conform to the Knuthian 8-character canon, the name lmvtti10 did not tell the truth. Having thought the problem over, we could not find the reason why oblique fonts, i.e., the mechanically skewed ones, received the designator 'i' in some cases (e.g., cmssi10) and the designator 'sl' in other (e.g., cmbxsl10), and why the designator appeared either at the end of the kernel of the name, as in the mentioned examples, or — in some cases — immediately after the prefix 'cm' (cmsltt10, cmitt10). We could either uphold traditional Knuth's terminology (but how then should we call oblique lmvtt10?) or take an opportunity and introduce some regularity in font naming at the risk of commencing an incompatibility mess. We have chosen the latter solution…

The issue of an alternative letter 'I' necessitated, besides undertaking a decision whether to introduce it or not (we decided to introduce it as a variant letter), some extra work due to the addition of variant accented characters and a variant ligature *IJ*. The lmssq* fonts became thus somewhat exceptional. This is usually undesirable but sometimes cannot be avoided.

The reader may wonder why to dwell on such trifles? The answer is simple: it was the bulk of details of this kind that made the work on the LM family laborious, although individual tasks were relatively simple. In other words, the problem with details is that each of them, even the tiniest one, has to be handled *somehow* — if the amount of details grows, the job becomes complex.

Enumerating all dilemmas, technicalities, subtleties or even puzzles with which we had to struggle is obviously pointless. On the other hand, our work consisted nearly exclusively of details — how to describe such a work? Perhaps the best method is to let the reader perceive the scent of the battleground by showing representative examples. Two such examples we have already indicated. The rest of the paper presents a few more of them.

From PostScript to METATYPE1 sources The process of conversion of fonts from PostScript Type 1 form into METATYPE1 sources is moderately relevant since the potential users of the LM fonts are not expected to repeat this operation any more. The METATYPE1 sources are legible and can easily be modified, if necessary.

We used a stand-alone utility PF2MT1 (belonging to the METATYPE1 package) for the translation of pfb+afm couples from CM fonts into METATYPE1 code. The virtual AE fonts provided the necessary information for merging appropriately the results of conversion. awk is a very convenient tool for such operations. Thanks to it, the framework of the LM sources was ready after a few hours; amending the LM sources took a few months.

Tuning and augmenting the METATYPE1 sources of the LM fonts The main part of the job, although the simplest one, was adding accents. METATYPE1 provides a use_accent operation, similar to the TEX \accent primitive, that can conveniently be used for this purpose. By default, use_accent aligns the centre of an accent with the centre of its accentee and raises the accent by $x - h$, where x is the value of the x-height parameter, and h is the height of the character. This is the procedure that is used by TEX for accenting. Such an algorithm is not always eligible. Occasionally, the position of accent may have to be adjusted. The command use_accent enables an arbitrary shift of both accent and accentee. Moreover, a supplementary parameter can optionally be specified for each character — a glyph axis (see figure 4).

All in all, adding accented letters was a child's play. Somewhat more difficult was adding extra characters.

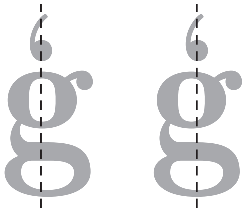

Figure4: The optical axis of a glyph does not necessarily coincide with the center of the glyph. Compare the corrected placement of the accent in *gcommaaccent* (left) with the default one (right).

In the AE family, the characters were brought together from a few CM sources. Consider, e.g., `aer10`: *arrow left hook* (i.e., faked *ogonek*), *less*, and *greater* characters were taken from `cmmi10`; *bar*, *backslash*, *braceleft*, *braceright*, and *section*—from `cmmi10`; *sterling*—from `cmu10`. Some characters were drawn using rules, e.g., *visiblespace* (missing characters were marked by a rule having width and height equal to ½ em), and some were assembled from a few components (*Aogonek*, *aogonek*, *Eogonek*, *eogonek*). We went even further: we "borrowed" characters *asciicircum* and *asciitilde* from `cmex10`; *mu*—from `cmmi10`; *dagger*, *daggerdbl*, and *paragraph*—from `cmsy10`.

It is debatable whether borrowing characters is acceptable. The *section* sign from `cmsy10` is certainly an alien in a sans serif font. Therefore, characters that seemed to us more important (*section*, *sterling*) were programmed from scratch. We used, of course, appropriate parameters from the CM driver files, but we did not follow Knuth's recipe rigorously. This might have been done (see the comments on symbol *Euro* below). We prefered, however, our shapes of glyphs. This may evoke some compatibility-related issues but, anyway, the full compatibility with CM, EC, and AE fonts cannot be achieved (see section *Compatibility issues*).

Actually, some characters were borrowed not from CM fonts but from their PL counterparts (i.e., CM fonts equipped with Polish diacritical letters; note that the relevant code from the PL fonts was incorporated into the EC fonts). The acute and grave accents over capital and small letters in PL fonts diminutively differ, namely, accents over capital letters are flattened. The same approach we applied in LM fonts (see figure 5) which is consistent with EC and inconsistent with CM fonts.

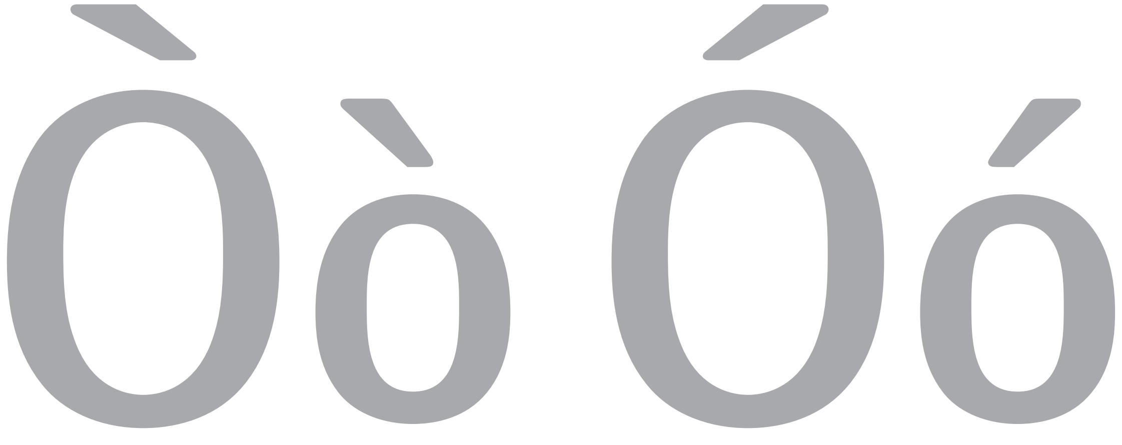

Figure5: There are actually two acute accents in LM fonts: a flattened variant is used for capital letters. This idea was implemented in PL fonts and next in EC ones. Note that, in general, the flattening is neither a slanting nor a rotation.

Besides the accented, borrowed and newly programmed characters, a few glyphs had to be programmed as consistently as possible with the CM character set. A notable example is an *Euro* currency symbol. It looks as though it became so important recently that even Adobe assigned it a name beginning with a capital letter (cf. *dollar*, *yen*, *sterling*, etc., in *Adobe Glyph List For New Fonts* [1]). We attempted to exploit the METAFONT code for the letter *C* and—it worked (see figure 6).

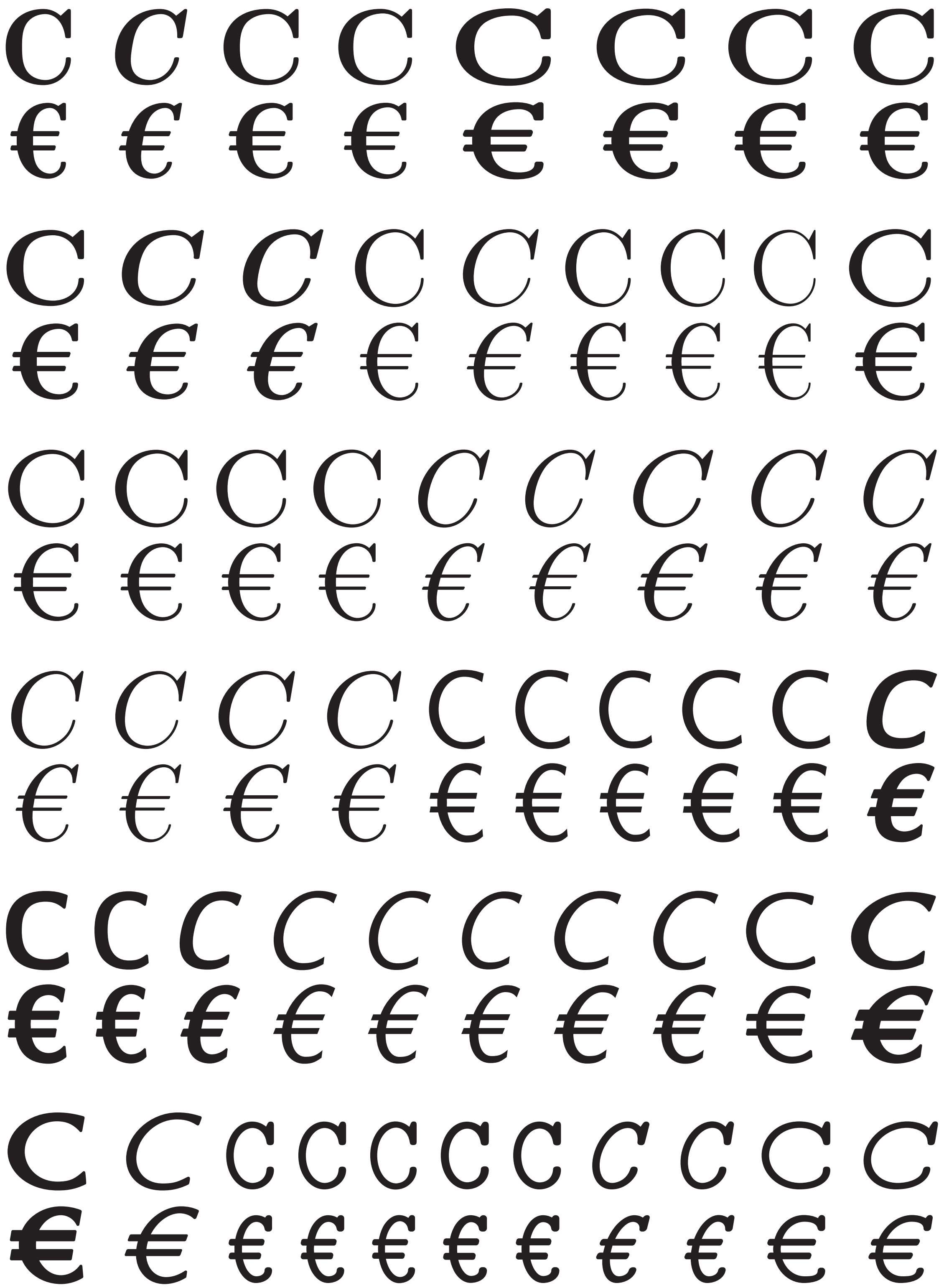

Figure6: *Euro* symbols from the LM fonts; observe that an *Euro* symbol is narrower than the corresponding letter *C* (above), but that the stem sizes are preserved. Note, however, that there is no slot for an *Euro* symbol in the *Cork Encoding*.

The LM fonts contain also a few idiosyncratic symbols. We wanted, e.g., to have a ligature *f_k* in the repertoire of characters (see figure 7) because there are several words in Polish containing the digram 'fk.' They are less numerous than words with digrams 'fi' and 'fl' but more than words with trigrams 'ffi' and 'ffl' (which occur exclusively in words of foreign origin). The electronic *Collins English Dictionary* retrieved only three words containing the digram 'fk': Aufklärung (sic), Kafka, Kafkaesque. There are more candidates for nonstandard ligatures, e.g., 'fb,' 'fh,' 'ffb,' and 'ffh.' These groups of letters occur sporadically in English and German (they are absent from Polish). We are not in a position, however, to decide whether introducing the respective ligatures would make sense.

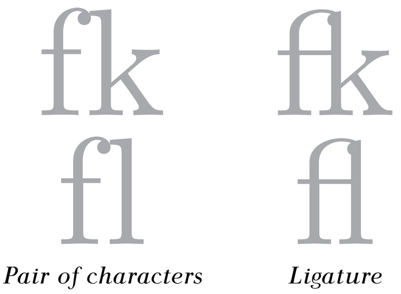

Figure7: There are several words in the Polish language that contain the digram 'fk'; therefore, we included the ligature *f_k* (top-right) in the LM character set for the sake of consistency with native CM ligatures, such as *fl* (bottom-right).

Compatibility issues The answer to the question whether the LM fonts can serve as a replacement for CM or EC ones is obviously 'no.' First of all, the collection of fonts is different — LM is a subset of CM (except `lmssqb8` and a few oblique derivatives), not speaking about EC. Therefore, not every text typeset with CM or EC fonts can be re-typeset using LM ones.

On the other hand, it should be noted that LM fonts are based on the data taken from CM driver files. Therefore, all relevant dimensions are (or at least should be) the same in LM and CM fonts within the accuracy of rounding errors. It is thus possible to use existing LM fonts as a replacement for CM in `dvips` driver `psfonts.map` file — it suffices to prepare appropriate encoding (`*.enc`) files.

In order to reach this level of compatibility, we had to add two more characters, namely *arrowup* and *arrowdown* which, somewhat surprisingly, are present in `cmr5`, but not in other fonts in the `cmr*` series. At the moment, we resisted the temptation to include consequently a full quiver of other arrows. The main reason was that arrows are absent from the basic *Cork Encoding* (they appear only in the *Text Companion Encoding* — see, e.g., the file `dcdoc.tex` distributed with the EC sources); moreover, PostScript is anyway involved and therefore various transformations can easily be applied, if necessary. In future, however, we may change our opinion.

The METATYPE1 programs for the arrows are based on METAFONT sources excerpted from `sym.mf`. While adapting the code, we encountered a quandary which is a good example of a seemingly trivial yet embarrassing detail. It turned out, that the arrow programs produce questionable results for certain driver files, namely, sidebars disappear. The arrow programs were perhaps never tested against all driver files. One can live with this, nevertheless, we decided to preserve minimal space at both sides — the result is certainly more palatable (see figure 8).

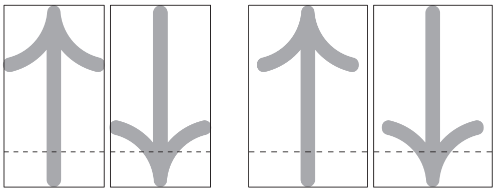

Figure8: The METAFONT program for arrows (from `sym.mf`) would produce glyphs stripped of sidebars for parameters from `cmssdc10` (left); arrows in LM fonts always have sidebars (right).

Another quandary of a similar kind is related to accents. For inexplicable reason, the caron accent in CM fonts is lowered in comparison to the remaining accents (see figure 9). We considered it a fault and decided to raise all carons appropriately. We relinquished thus the full compatibility between CM and LM families.

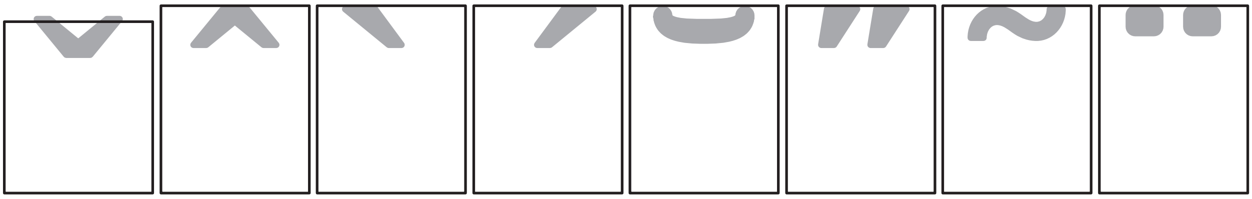

Figure9: The caron alias hachek accent (the leftmost box) is slightly lowered in the CM fonts; in the LM fonts, all accents are aligned horizontally.

The game of names Among many technicalities related to the representation of PostScript fonts, we would like to comment upon only one — the particularly upsetting problem of character names.

```
Gcommaaccent; LATIN CAPITAL LETTER G WITH CEDILLA
Kcommaaccent; LATIN CAPITAL LETTER K WITH CEDILLA
Lcommaaccent; LATIN CAPITAL LETTER L WITH CEDILLA
Ncommaaccent; LATIN CAPITAL LETTER N WITH CEDILLA
Rcommaaccent; LATIN CAPITAL LETTER R WITH CEDILLA
Scommaaccent; LATIN CAPITAL LETTER S WITH COMMA BELOW
gcommaaccent; LATIN SMALL LETTER G WITH CEDILLA
kcommaaccent; LATIN SMALL LETTER K WITH CEDILLA
lcommaaccent; LATIN SMALL LETTER L WITH CEDILLA
ncommaaccent; LATIN SMALL LETTER N WITH CEDILLA
rcommaaccent; LATIN SMALL LETTER R WITH CEDILLA
scommaaccent; LATIN SMALL LETTER S WITH COMMA BELOW
```

Figure10: An excerpt from the up-to-date *Adobe Glyph List For New Fonts* [1]. How sweet...

There exists a standard of glyph naming worked out by Adobe [1]: *Adobe Glyph List 2.0* (of 20th September 2002) and *Adobe Glyph List For New Fonts 1.0* (of 31st January 2003). Regretfully, the standard contains numerous entries that are at best dubious. We have already jeered at the name of an *Euro* symbol that singularly begins with a capital letter. But this is nothing. The excerpt from the *Adobe Glyph List For New Fonts* concerning characters with *commaaccent* is really astounding (see figure 10). Even more astounding is a part of this story pertaining to *Tcedilla* and *tcedilla*:

- The version 1.1 of *Adobe Glyph List* mentioned the characters described as ‘T with cedilla’ and ‘t with cedilla’ and assigned them names *Tcommaaccent* and *tcommaaccent*, respectively; characters that could be described as ‘T with comma below’ or ‘t with comma below’ were just ignored.
- In the version 1.2 of the *Adobe Glyph List*, the names *Tcommaaccent* and *tcommaaccent* were assigned both to characters described as ‘T or t with cedilla’ and ‘T or t with comma below’.
- The up-to-date *Adobe Glyph List For New Fonts* says that one of the most recent changes was “renaming tcommaaccent to tcedilla and Tcommaaccent to Tcedilla.”

To untangle the "commaaccent story" a little bit, we would like to quote a more reliable opinion from Michael Everson's web site devoted to European alphabets [5]:

- Concerning Latvian: "The [accented] characters g, k, l, n, r, G, K, L, N, and R must always be drawn with a *comma below*, although these characters are identified in ISO standards as *letters with cedilla.* Note particularly the reverse comma accent used with the *latin small letter g with cedilla.*" (Cf. figure 4.)
- Concerning Romanian: "Note that Romanian uses the characters *s with comma below* and *t with comma below.* In inferior Romanian typography, the glyphs for these characters are sometimes drawn with *cedillas*, but it is strongly recommended to avoid this practice."

There were more pitfalls of this kind, not as ridiculous as the case of the *commaaccent*, but sufficiently confusing to make this part of the job arduous.

Given such a state of the art, we decided to copy some glyphs under different names—just in case. We repeated, e.g., the glyphs *scommaaccent*, *tcommaaccent*, *Scommaaccent*, and *Tcommaaccent* under the names *scedilla*, *tcedilla*, *Scedilla*, and *Tcedilla*, respectively. Altogether, there are approximately 10 duplicated characters per 400-character font.

Note that the duplication of glyphs does *not* lead to an enormous inflation of the size of font files because of a very efficient subroutine packing mechanism (cf. section *Another viewpoint*, p. 4). Actually, a duplicated character increases the size of a font by 30–40 bytes. This means that 10 duplicated characters would increase a font size by less than 1 percent, as the average size of an LM font (pfb) is 60 kb.

Beware of your friends The basic tools we used (METAPOST, tftopl, vftovp, awk, T1utils) worked nearly infallibly. Only once we met a truly intricate problem. It was a bug persistently offered by our friend, METAPOST.

One of the important operations in the process of font generation is determining the orientation of a path: anticlockwise-oriented paths are used for filling and clockwise-oriented—for unfilling. There is a function turningnumber in METAFONT and METAPOST that returns +1 and −1 for anticlockwise-oriented and clockwise-oriented paths, respectively. In METAFONT it works correctly, in METAPOST—unfortunately it does not. The bug manifests its presence even in such trivial cases as the following code:

```
path p;
p=(0,10)..controls (5,10) and (10,5)
  ..(10,0)..controls (10,-5) and (5,-10)
  ..(0,-10)..controls (-5,-10) and (-10,-5)
  ..(-10,0)..controls (-10,5) and (-5,10)
  ..cycle;
```

This nearly circular 4-node path is evidently clockwise-oriented. Nevertheless, METAPOST maintains that turningnumber p = 0.

We did not analyse the METAPOST source code as we were not going to fix the bug, but circumventing it was crucial. The only method that proved to work was the "straightening" of a path prior to the application of the turningnumber function; in other words, each Bézier segment of a path was changed to a straight line and then the turningnumber function was applied to a modified path. It works so far, although the method is not general (see figure 11) and, moreover, frequently used straightening slows down the process of the generation of fonts.

Encodings In olden days, there was a one-to-one correspondence between a *font name* and the name of a *font metric file* (tfm). This is not possible any longer. If there are more characters in a font than 256, like in CM-Super or LM, one has to select a subset of characters and to assign codes to every character. Even not knowing the precise results of combinatorial analysis, one may fancy how many such encoding may coexist. It seems that there is no choice—*metric files must not use the same name as the basic font*, otherwise a mess is bound to ensue.

One may think of a distinguished (main) encoding that would inherit the basic name, but we would rather equate all encodings. We supplied two encodings in the official distribution of the LM fonts: the *Cork Encoding* and the *QX Encoding*.

The former does not need explanation. The latter is actually a "double" encoding, i.e., there is a fixed collection of characters and two numberings—one to be used with TEX and one to be used with window systems [12]. The *QX Encoding* was worked out a few years ago by the members of the Polish TEX Users

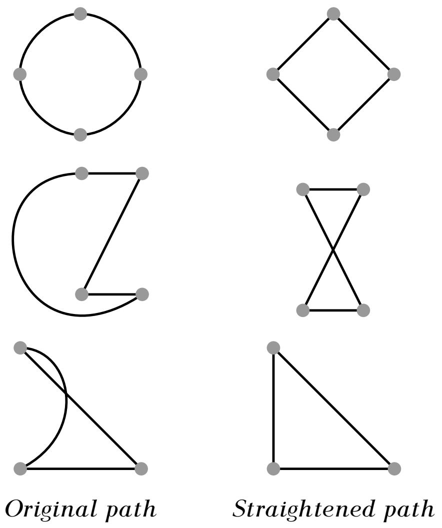

Figure11: The operation of straightening of a path typically does not change the orientation of a path (top); in general, however, this may happen — middle and bottom pictures show how a non-zero turning number can be changed to zero and vice versa. The latter situations, fortunately, are rather unlikely in fonts.

Group GUST as a difficult compromise between needs and abilities. In a nutshell: the *QX Encoding* for TEX is a variant of the *Cork Encoding* with a few characters exchanged (e.g., *gbreve*, *Gbreve*, *uring*, and *Uring* are replaced by Lithuanian *iogonek*, *Iogonek*, *uogonek*, and *Uogonek*, respectively); the *QX Encoding* for window systems is a variant of the Code Page 1250 (and also includes Lithuanian characters with ogonek).

Recall that the complete list of the LM font names is shown in figure 2. The respective `tfm` file names are derived by adding the suffix `_ec` for the *Cork Encoding* and the suffix `_qx` for the *QX Encoding*, e.g., `lmr10` with the *Cork Encoding* has the name `lmr10_ec` and with the *QX Encoding* — the name `lmr10_qx`.

This protocol is admittedly immature. Nevertheless, we do insist on recommending either this naming scheme or a similar one as a guideline for TEX users as long as TEX is not capable of handling multi-byte character codes, or even longer.

Availability The LM fonts are freely available at `ftp://bop.eps.gda.pl/pub/lm/`.

Concluding remarks

We would like to emphasize once again that our aim was not only to provide a new family of fonts, but to provide it with METATYPE1 sources that can be maintained — adjusted, augmented, improved, etc. While it is rather difficult to write a font program from scratch, it is pretty simple to modify sources, e.g., as we have mentioned, adding accented letters is straightforward.

As concerns our plans regarding the LM family, we would like to enhance fonts: to improve kerning, hinting, and shapes of certain glyphs, to add more accented characters, and, last but not least, to provide OpenType versions of the LM fonts for XP trailblazers. We consider, moreover, converting a few more CM programs from METAFONT to METATYPE1, as we would like to dismiss eventually the borrowed characters (see section *Tuning and augmenting the METATYPE1 sources...*, p. 8).

Before bringing the curtain down, we would like to draw the reader's attention to a weak point of our approach: the CM parameterization is lost. The METATYPE1 sources can be enhanced, but they cannot be used for producing, say, light or condensed versions of sanserif fonts. An experiment with the programming of the *Euro* symbol and the arrows has showed that converting METAFONT sources to METATYPE1 ones without loosing the parameterization is, in general, possible but rather time-consuming. It is an open question whether such a venture, being extremely attractive, is reasonable.

Acknowledgements

The project was 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, and the Dutch-speaking TeX Users Group NTG — very many thanks. We are also grateful to Volker Schaa and Stefan Sokołowski for their valuable comments concerning the draft version of the paper.

References

[1] *Adobe Solutions Network: Unicode and Glyph Names*, `http://partners.adobe.com/asn/developer/type/unicodegn.html`

[2] *Adobe Type 1 Font Format.* Addison-Wesley, 1990, `http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF`

[3] Włodzimierz Bzyl, *The Tao of Fonts.* Proc. of TUG 2002, 4th – 7th September, 2002, Trivandrum, India (to appear).

[4] Lars Engebretsen, *AE fonts*, `http://www.tug.org/tex-archive/fonts/ae/`

[5] Michael Everson, *The Alphabets of Europe* (ver. 3.0) `http://www.evertype.com/alphabets/`

[6] Michael Ferguson, *Report on multilingual activities*, TUGboat 11 (4), November 1990, p. 514.

[7] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *Antykwa Półtawskiego: A Parameterized Outline Font.* Proc. of EuroTeX 1999, 20th – 24th September, 1999, Heidelberg, Germany, pp. 109 – 141.

[8] 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, pp. 111 – 119, `http://www.ntg.nl/eurotex/metatyp1.pdf` and `http://www.ntg.nl/eurotex/JackowskiMT.pdf`

[9] Richard J. Kinch, MetaFog*: Converting METAFONT Shapes to Contours.* TUGboat **16** (3), pp. 233 – 243, 1995.

[10] Han-Wen Nienhuys, *MFTrace — Scalable Fonts for METAFONT*, `http://www.cs.uu.nl/~hanwen/mftrace/`

[11] John Plaice and Yannis Haralambous, *Omega System*, `http://sourceforge.net/projects/omega-system/`

[12] *QX encoding tables for TeX and for window systems*, `http://www.gust.org.pl/fonty/qx-table1.html`, `http://www.gust.org.pl/fonty/qx-table2.html`

[13] John Sauter, *Building Computer Modern Fonts*, TUGboat 7 (3), October 1986, p. 151.

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

[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, *PfaEdit — a PostScript Font Editor*, `http://pfaedit.sourceforge.net/`

Appendix: The contents of the Latin Modern family of fonts, ver. 0.82

For meticulous readers, we enclose below the complete list of LM glyph names in the alphabetic order. Note that some characters do not occur in all fonts, e.g, there are no *f*-ligatures in the typewriter fonts. Actually, there are five classes of charsets:

1. The basic class (408 glyphs); this class consists of `lmb10`, `lmbo10`, `lmbx10`, `lmbx12`, `lmbx5`, `lmbx6`, `lmbx7`, `lmbx8`, `lmbx9`, `lmbxi10`, `lmbxo10`, `lmr10`, `lmr12`, `lmr17`, `lmr5`, `lmr6`, `lmr7`, `lmr8`, `lmr9`, `lmri10`, `lmri12`, `lmri7`, `lmri8`, `lmri9`, `lmro10`, `lmro12`, `lmro8`, `lmro9`, `lmss10`, `lmss12`, `lmss17`, `lmss8`, `lmss9`, `lmssbo10`, `lmssbx10`, `lmssdc10`, `lmssdo10`, `lmsso10`, `lmsso12`, `lmsso17`, `lmsso8`, `lmsso9`, `lmvtt10`, and `lmvtto10`.
2. The class 'ssq' (419 glyphs); besides the characters present in the basic class, it contains *varI*, *varIacute*, *varIcircumflex*, *varIdieresis*, *varIdotaccent*, *varIgrave*, *varIJ*, *varImacron*, *varIogonek*, *varItilde*, and *varIvardieresis*. The following fonts belong to this family: `lmssq8`, `lmssqbo8`, `lmssqbx8`, and `lmssqo8` (cf. also figure 3 and the relevant comments in section *The LM family of fonts...*).
3. The class 'typewriter' (395 glyphs); the following glyphs are missing in comparison with the basic class: *f_k*, *ff*, *ffi*, *ffl*, *fi*, *fl*, *Germandbls*, *IJ*, *ij*, *suppress*, *trademark*, *varcopyright*, and *varregistered*. The class consists of `lmtt10`, `lmtt12`, `lmtt8`, `lmtt9`, `lmtti10`, and `lmtto10`.
4. The class containing only `lmcsc10` and `lmcsco10` (400 glyphs); the following glyphs are missing in comparison with the basic class: *dquoteright*, *f_k*, *ff*, *ffi*, *ffl*, *fi*, *fl*, and *tquoteright*.
5. The class containing only `lmtcsc10` (393 glyphs); the set of missing character is as in class 3 plus *dquoteright* and *tquoteright*.

The alphabetic list of glyphs in the Latin Modern family

`A, a, Aacute, aacute, Abreve, abreve, Acircumflex, acircumflex, Acute, acute, Adieresis, adieresis, AE, ae, Agrave, agrave, Amacron, amacron, ampersand, anglearc, Aogonek, aogonek, Aring, aring, arrowdown, arrowup, asciicircum, asciitilde, asterisk, at, Atilde, atilde, Avardieresis, avardieresis, B,` b, backslash, bar, braceleft, braceright, bracketleft, bracketright, breve, bullet, C, c, Cacute, cacute, caron, Ccaron, ccaron, Ccedilla, ccedilla, Ccircumflex, ccircumflex, Cdotaccent, cdotaccent, cedilla, cent, circumflex, colon, comma, commaaccent, compoundwordmark, copyright, currency, D, d, dagger, daggerdbl, dbar, Dcaron, dcaron, Dcroat, dcroat, degree, Delta, diameter, dieresis, divide, dmacron, dollar, dotaccent, dotlessi, dotlessj, dquoteright, E, e, Eacute, eacute, Ebreve, ebreve, Ecaron, ecaron, Ecircumflex, ecircumflex, Edieresis, edieresis, Edotaccent, edotaccent, Egrave, egrave, eight, ellipsis, Emacron, emacron, emdash, endash, Eng, eng, Eogonek, eogonek, equal, Eth, eth, Euro, euro, Evardieresis, evardieresis, exclam, exclamdown, F, f, f_k, ff, ffi, ffl, fi, five, fl, four, G, g, Gacute, gacute, Gamma, Gbreve, gbreve, Gcaron, gcaron, Gcedilla, Gcircumflex, gcircumflex, Gcommaaccent, gcommaaccent, Gdotaccent, gdotaccent, Germandbls, germandbls, Grave, grave, greater, guillemotleft, guillemotright, guilsinglleft, guilsinglright, H, h, Hbar, hbar, Hcircumflex, hcircumflex, hungarumlaut, hyphen, I, i, Iacute, iacute, Icircumflex, icircumflex, Idieresis, idieresis, Idotaccent, Igrave, igrave, IJ, ij, Imacron, imacron, Iogonek, iogonek, Itilde, itilde, Ivardieresis, ivardieresis, J, j, Jcircumflex, jcircumflex, K, k, Kcedilla, kcedilla, Kcommaaccent, kcommaaccent, L, l, Lacute, lacute, Lambda, Lcaron, lcaron, Lcedilla, lcedilla, Lcommaaccent, lcommaaccent, Ldotaccent, ldotaccent, less, Lquoteright, lquoteright, Lslash, lslash, M, m, macron, minus, mu, multiply, N, n, Nacute, nacute, nbspace, Ncaron, ncaron, Ncedilla, ncedilla, Ncommaaccent, ncommaaccent, nine, Ntilde, ntilde, numbersign, O, o, Oacute, oacute, Obreve, obreve, Ocircumflex, ocircumflex, Odieresis, odieresis, OE, oe, ogonek, Ograve, ograve, Ohungarumlaut, ohungarumlaut, Omacron, omacron, Omega, one, Oogonek, oogonek, Oslash, oslash, Otilde, otilde, Ovardieresis, ovardieresis, P, p, paragraph, parenleft, parenright, percent, period, periodcentered, perthousand, Phi, Pi, plus, plusminus, Psi, Q, q, question, questiondown, quotedbl, quotedblbase, quotedblleft, quotedblright, quoteleft, quoteright, quotesinglbase, quotesingle, R, r, Racute, racute, Rcaron, rcaron, Rcedilla, rcedilla, Rcommaaccent, rcommaaccent, registered, ring, S, s, Sacute, sacute, Scaron, scaron, Scedilla, scedilla, Scircumflex, scircumflex, Scommaaccent, scommaaccent, section, semicolon, seven, Sigma, six, slash, space, sterling, suppress, T, t, Tcaron, tcaron, Tcedilla, tcedilla, Tcommaaccent, tcommaaccent, Theta, Thorn, thorn, three, tilde, tquoteright, trademark, two, U, u, Uacute, uacute, Ubreve, ubreve, Ucircumflex, ucircumflex, Udieresis, udieresis, Ugrave, ugrave, Uhungarumlaut, uhungarumlaut, Umacron, umacron, underscore, Uogonek, uogonek, Upsilon, Uring, uring, Utilde, utilde, Uvardieresis, uvardieresis, V, v, varcopyright, vardieresis, vardotaccent, varI, varIacute, varIcircumflex, varIdieresis, varIdotaccent, varIgrave, varIJ, varImacron, varIogonek, varItilde, varIvardieresis, varregistered, visiblespace, W, w, Wacute, wacute, Wcircumflex, wcircumflex, Wdieresis, wdieresis, Wgrave, wgrave, Wvardieresis, wvardieresis, X, x, Xi, Y, y, Yacute, yacute, Ycircumflex, ycircumflex, Ydieresis, ydieresis, Yen, Ygrave, ygrave, Yvardieresis, yvardieresis, Z, z, Zacute, zacute, Zcaron, zcaron, Zdotaccent, zdotaccent, zero.