

## Font Forum

### Multiple Master math extension fonts

D. Men'shikov, A. Kostin, and M. Vulis

#### Abstract

From its inception,  $\TeX$  has relied on the concept of extensible (composite) characters to implement large glyphs that occur in mathematical typesetting. While thousands of papers have been successfully written using this technology, it nevertheless has obvious shortcomings. This article demonstrates them, as well as an alternative solution, free from the problems.

### 1 Problems with the traditional approach

The extensible glyphs in  $\TeX$  are composed out of several smaller characters, and, in some cases, rules. The construction itself is carried either by code inside the  $\TeX$  compiler itself, using the special information in the `.tfm` files, or by macros, implemented as part of the format. We will consider each in turn.

#### 1.1 Extensions done by $\TeX$

An example of extensible glyphs constructed within  $\TeX$  is the construction of large delimiters, such as brackets. In this case, a vertical segment is inserted to stretch the delimiters vertically:

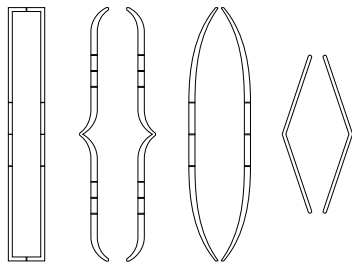
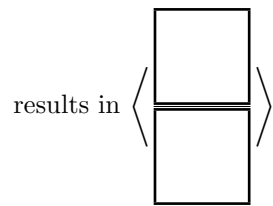


Figure 1: Vertical delimiters

One of the shortcomings of the extensible characters is that some character shapes do not lend themselves to the insertion of extension pieces; for example, angle brackets are not extensible. Thus, when used to encompass a large formula, they would not scale to the required size. For example, the input:<sup>1</sup>

<sup>1</sup> `\emptyfbox` is a macro which makes a frame box of the specified width and height, used to avoid printing large black squares in this journal.

```
 $\left< \emptyfbox{1cm}{1cm} \over
\emptyfbox{1cm}{1cm} \right>$ 
```



A variant of the situation is  $\TeX$ 's treatment of the radical symbol: while the vertical extension is driven by the `.tfm` information, the horizontal bar is supplied by the  $\TeX$  program itself—and in this case, the bar is a rule, rather than an extension character.

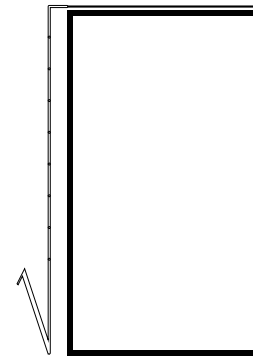


Figure 2: Radical delimiters

In the above picture, the rule component is shown filled.

The problem with the use of rule components in building a composite symbol is that rules are subjected to different roundoff rules than characters. With DVI files and bitmap fonts, these roundoff errors could in principle be handled by careful calculations within the  $\TeX$  program and the DVI driver; but in the modern world of scalable fonts and PS/PDF targeting this becomes an impossibility.

For instance, the picture below shows a snapshot of a square root constructed by Pdf $\TeX$ :

The underlying  $\TeX$  code is simply  
 $\$ \sqrt{\emptyfbox{1cm}{2cm}} \$$

Depending on factors such as the magnification used in the PDF previewer and the resolution of the output device, the rule may be thinner, or thicker, or above, or below the part of the `sqrt` glyph it is supposed to connect to smoothly. Sometimes it might even fit correctly.

The root of the problem is that glyph rounding is subject to font hinting, while the thickness and positioning of rules is not precisely controllable by the PS/PDF rendering.

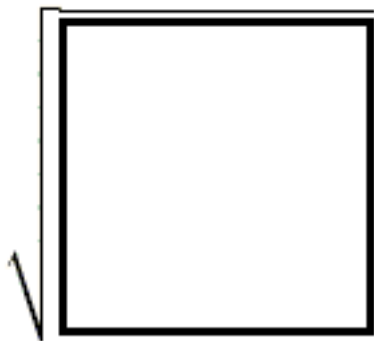


Figure 3: Incorrect radical

### 1.2 Extensions done by macros

The second type of extension mechanism is via  $\TeX$  macros. This is typically used to construct long horizontal symbols, as done by commands similar to `\underbrace`:

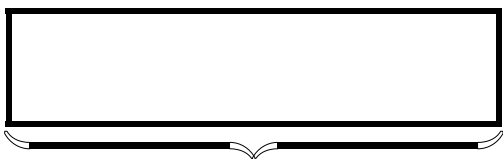


Figure 4: Underbrace extension

Once again, we get mismatches in the width and positioning of the connecting rules comparing to that of the symbols used in the middle and at the ends of the long brace. A snapshot:



Figure 5: Incorrect extensible braces

### 1.3 Shading characters

A third problem with extensible characters is that sometimes (usually, in slides) it is desirable to outline, or shade characters; this cannot be done correctly with composite characters. In fact, the diagrams above that show the structure of extensible characters, are *unintended output* of an attempt to stroke extensible symbols, making them suitable for this article, but not for a use within a slide.

Summarizing, we have identified three separate problems with the  $\TeX$  approach to the extensible symbols:

- Some delimiters (angle brackets, for example) cannot be correctly scaled.
- Rules and characters together cannot be correctly aligned.
- Composite symbols cannot be shaded or correctly stroked.

All these problems can be solved, however, if we switch to a different mechanism of constructing large symbols: Multiple Master fonts.

## 2 Multiple Master fonts

For those unfamiliar with MM fonts, they represent a PostScript world answer to METAFONT. Just as METAFONT allows creation of different designs from the same font program, so does the MM technology. While MM fonts are less flexible than METAFONT, they are easy to use. The output of MM font instancing is a Type 1 outline font, whereas a generic METAFONT emits bitmapped fonts. (See [3,6] for additional information on MM fonts.)

MM technology was introduced by Adobe in 1991; the first description appeared in PC Week [4]. The first MM font was Myriad (1992), with two axes: weight and width.

Since 1992 Adobe has designed at least 36 MM font families with about 100 fonts. Perhaps the most successful is the Minion family.

While Adobe originally intended to include MM in the OpenType specifications, this effort has been abandoned, and Adobe has stopped making new MM fonts. The last Adobe MM font, VerveMM, was designed in 1998; Adobe had announced that it was giving up on MM technology at the 1999 ATypI Congress.

Following the Adobe announcement, an article in *MacObserver* lamented [5]:

*Was it necessary for Multiple Masters to die? Probably not. Several factors contributed: inexpert and uninteresting designs; a purportedly “open” technology that was in fact proprietary; and inadequate interface support early on to Aldus, Quark and Macromedia. None of this had to be. Pity, pity, pity!*

Despite its abandonment by Adobe, MM fonts still represent a very convenient technology for use in typesetting applications like  $\TeX$ . While MM functionality is not supported within PDF documents, instances of MM fonts are, and the entire MM font can be made available to the  $\TeX$  document.

VT<sub>E</sub>X has supported MM fonts since 2001, and the use of MM fonts in math extension fonts builds on the existing MM support; we will start by outlining the current support.

VT<sub>E</sub>X recognizes MM fonts by the name of the font; a font name that includes the bracket character is taken as an instance of a Multiple Master font. This does not represent much of a restriction on the font name selection, since use of brackets in file names is not common and is in fact invalid on some operating systems.

The left bracket in the font name is followed by the instancing parameters. For example:

```
\font\sm=xo_____
\font\mm=xo_____ [300,10]
```

The `\sm` font declaration defines the default shape of the Multiple Master CrononMM font; this is the “normal” way to use the typeface. The second declaration, however, defines an instance of this MM font to be constructed dynamically.

In the above example, the first definition makes VT<sub>E</sub>X to load the usual TFM metrics file, namely `xo_____.tfm`; the second declaration causes VT<sub>E</sub>X to load the multiple font metrics file `xo_____.mfm` and generate the instance metrics on the fly. Because of the use of `.mfm` files, providing metrics for each instance becomes unnecessary.

In some MM fonts, all the instances of the font have the same metrics; if so, it is possible to use the ordinary `.tfm` file. In most MM fonts, however, the metrics of each instance are different, and this made the development of the new metrics format necessary.

Upon seeing a MM font used in the document, VT<sub>E</sub>X automatically uses its built-in PostScript interpreter G<sub>E</sub>X [1] to instance it. Since the instancing is done within PostScript, the instance font is always built correctly. The T<sub>E</sub>X compiler then automatically packs it into the output PDF file.

Additional details on the MM support can be found in the `mmsupp.pdf` document in VT<sub>E</sub>X distributions.

### 3 Math extension Multiple Master fonts

Supporting math extension MM fonts requires several additional steps.

First, of course such fonts need to be developed. At this writing, two such fonts exist: `cmex10mm` and `paex10mm`. The first is intended for use with Computer Modern, the second for use with the alternative PaMATH fonts, available from MicroPress.






Math extension MM fonts include symbols for vertical delimiters, long horizontal symbols (like the

ones constructed by the `\underbrace` macro), and the radical symbol. Since the radical requires two MM axes, the entire font is a two-axes MM font, even though the majority of symbols are actually “one-dimensional”.

Specific sizes of delimiters can be constructed by loading the font with different instancing parameters. For example, for the `\overbrace` symbol, we can use font commands such as

```
\font\f=cmex10mm[30,100] \f \char"7A
\font\f=cmex10mm[30,300] \f \char"7A
\font\f=cmex10mm[30,500] \f \char"7A
\font\f=cmex10mm[30,700] \f \char"7A
\font\f=cmex10mm[30,900] \f \char"7A
```

obtaining the shapes

100:	
300:	
500:	
700:	
900:	

(Only the second instancing parameter is used in long horizontal symbols.)

Since the symbols are built from single glyphs, they do not suffer from the problems listed at the beginning of this paper.

Since T<sub>E</sub>X supports `\overbrace` and other long horizontal symbols entirely through macros, switching T<sub>E</sub>X to supporting MM instances requires only changes to the macros. One of the tasks of the `mathexmm` style is to therefore redefine these macros.

Supporting vertical delimiters and radicals requires more, well, radical changes: the `.tfm` mechanism of extensible characters needs to be replaced by an MM alternative. We accomplish this as follows:

First, we prepare an alternative metrics file, `cmex10m`, which is mostly the same as `cmex10`, except that

- The TFM `CODINGScheme` is `MMEXTENSION`; this is the signal to the T<sub>E</sub>X compiler that what normally would be interpreted as `exten` instructions in the `.tfm` instead should be seen as pointers to MM glyphs. (It is unfortunate that the `.tfm` syntax does not allow for new flags; this is what forces us to use the `CODINGScheme` field.) In a `.PL` file, this appears as:

```
(CODINGScheme MMEXTENSION)
```

- Each `exten` specification in the `.tfm` is replaced by the glyph number of the character to use in the `cmex10mm` MM font. For example, the original `cmex10.pl` listing contains

```
(CHARACTER C 0
  (CHARWD R 0.875003)
  (CHARHT R 0.039999)
  (CHARDP R 1.760019)
  (VARIABLE
    (TOP C 0)
    (BOT 0 100)
    (REP C B)
  )
)
```

specifying that the character 0 is to be built from glyphs 0, ‘100 and B. In `cmex10m.pl`, however, we have

```
(CHARACTER C 0
  (CHARWD R 0.875003)
  (CHARHT R 0.039999)
  (CHARDP R 1.760019)
  (VARIABLE
    (REP 0 303)
  )
)
```

which means that the character 0 is to be built as an instance of the glyph ‘0303 in the corresponding MM font (`cmex10mm`).

- `cmex10` and `cmex10m` are otherwise identical.

The `mathexmm` L<sup>A</sup>T<sub>E</sub>X style, when used, loads the font `cmex10m` instead of `cmex10`. Unless extensible characters are involved, it functions in exactly the same way as `cmex10`; but when T<sub>E</sub>X is about to build an extensible character, it instead builds an appropriately-sized instance from the `cmex10mm` MM font.

From the user’s point of view, this is all invisible, and no action is required except for adding `\usepackage{mathexmm}` in the document preamble for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, or `\input mathexmm` for plain T<sub>E</sub>X and  $\mathcal{M}$ S-T<sub>E</sub>X.

#### 4 Acknowledgement

Related ideas previously appeared in [2]; however, they did not add up a practically usable implementation, which has been our contribution.

#### 5 Bibliography

- [1] A. Kostin & M. Vulis, “Mixing T<sub>E</sub>X and PostScript: The G<sub>E</sub>X Model”, in *TUGboat*, Vol. 23(3), 251–264 (Sep 2000, proceedings of the TUG 2000 conference).
- [2] J. André & I. Vatton, “Dynamic optical scaling and variable-sized characters”, in *Electronic Publishing*, Vol. 7(4), 231–250 (Dec 1994).
- [3] Adobe Systems Inc., “*Type 1 Font Format Supplement*”. Adobe Technical Specification 5015.
- [4] PC Week, Vol. 8(10), pg. 1 (11 Mar 1991).
- [5] Bill Troop, “A giant step backwards for Adobe?”, in *MacObserver*, Oct 6, 1999. <http://www.macobserver.com/columns/troop/99/october/991006.shtml>
- [6] John D. Berry, “*dot-font: Avant Garde, Then and Now*”. <http://www.creativepro.com/story/feature/19432.html>

◇ D. Men’shikov, A. Kostin, and M. Vulis  
 MicroPress, Inc.  
 68-30 Harrow Street  
 Forest Hills, New York 11375  
 USA  
 phone: +1 (718) 575 1818  
 fax: +1 (718) 575 8038  
[support@micropress-inc.com](mailto:support@micropress-inc.com)  
<http://www.micropress-inc.com>