Which math font related issues can be addressed in future extensions to TEX?

Ulrik Vieth

Abstract

In the course of discussions on the math font mailing list, a number of problematic issues have been brought to attention regarding the rather obscure and idiosyncratic ways in which certain math typesetting features are presently implemented in TEX, which are causing all sorts of trouble to designers or implementors of math fonts intended for use with TEX as well as other typesetting systems.

This document attempts to summarize these points and to assess in how far some of these issues may be addressed in the context of various ongoing projects to develop a successor to TEX.

1 Overview of projects to extend TEX

The $\mathcal{NTS}$ project. The $\mathcal{NTS}$ project is a long-term project established in 1992 to develop a New Typesetting System (NTS) as a successor to TEX. This may ultimately amount to rewriting TEX from scratch, hopefully in a more modular fashion, so that individual components or algorithms could easily be replaced by alternative implementations.

At present, the $\mathcal{NTS}$ project is still in the early conceptual stages and it is difficult to tell when it will eventually make any concrete steps towards an actual implementation.

From the point of view of math font issues, $\mathcal{NTS}$ may offer a great potential to reconsider many of the assumptions that are taken for granted in the context of TEX, including changes to the math typesetting algorithms or the way font metric information is stored and represented.

The ε-TEX project. The ε-TEX project is an offspring of the $\mathcal{NTS}$ project. It is a medium-term project that aims to take an evolutionary path to develop incremental extensions to TEX while remaining 100% compatible with the original TEX when all the additional features are turned off.

More precisely, ε-TEX actually provides two different modes of operation: a compatibility mode, in which ε-TEX behaves exactly like the original TEX, and an extended mode, in which a certain number of extra features termed *extensions* are enabled while the behavior of the existing TEX commands remains unchanged. Moreover, ε-TEX also provides another kind of additions termed *enhancements* that have to be enabled explicitly since they imply incompatible changes to the typesetting algorithms.

The majority of features presently implemented in ε-TEX Version 2.0 (released in March 1998) fall in the category of extensions and are mostly related to programming and diagnostics features. The only enhancements implemented so far are the TEX--XET features to support switching between left-to-right and right-to-left typesetting.

As far as math typesetting is concerned, it seems plausible that it may be possible to implement a certain number of very specific changes within the scope of ε-TEX, some of which would fall in the category of enhancements leading to incompatible changes. It should be clear, however, that it will not be possible in this context to address issues that would require far-reaching changes such as modifications to the well-established file formats.

The Ω project. The Ω project is another project to extend TEX that has developed independently of the other two. It focuses primarily on multi-lingual issues such as making it possible to use 16-bit characters sets, switching the typesetting direction, or simplifying the input and output conversion between different characters sets by means of filter programs, so-called OTPs (Omega translation processes).

Due to the nature of the extensions addressed, Ω necessarily requires the use of an extended font file format called OFM. This is realized by simply doubling every table size in the TFM format, so that it becomes possible to handle 16-bit character sets with 256 different heights or depths, 4096(?) different italic corrections, and 65536 different widths. At the same time, the number of math families has also been increased from 16 to 256.

If Ω could be assumed to be widely available, the ability to access 16-bit character sets could potentially be very useful in math fonts, since it would allow to accommodate a much larger symbol complement without having to worry about technical limitations in the design of 8-bit font tables.

2 Math font issues to be addressed

Limitations on font metrics. Unlike Adobe's AFM files, TEX's TFM files do not store the character metrics directly with each characters. Instead, the TFM file format uses a lookup table to store all the character widths, heights, depths, and italic corrections used in a font in one place. The number of entries in these tables is limited to no more than 16 different heights and depths, 64 different italic corrections, and 256 different widths.

While these table sizes may be generally sufficient for text fonts consisting mostly of characters of similar size (e. g. *x_height*, *asc_height* or *cap_height*), the limitation on the number of heights and depths does turn out to be a real problem in the implementation of math extension fonts. This is actually not surprising considering that such a font happens to contain big delimiters and radicals in many different sizes, which necessarily must live together in the same font, so that they can be linked through the `charlist` mechanism.

The problem is made worse by the requirement that the heights and depths of the glyphs in a math extension font must be represented accurately without rounding, so that extensible glyphs constructed by stacking several pieces on top of each other will be rendered properly without overlaps or gaps.

It should be clear from the discussion in the previous section that changes to the TFM file format to overcome these limitations are out of the question in the context of ε-TEX, but it may well be an issue to consider for $\mathcal{NTS}$. In Ω many of the most pressing constraints have already been lifted by doubling every table size, but the limitations have not been removed in principle.

Character metrics. Another problem regarding font metric information is the peculiar way in which TEX (ab)uses the character widths and italic corrections in math fonts to convey information about the subscript and superscript positions.

At present, the nominal TFM width of a character in a math font is interpreted as the subscript position, while the italic correction is used to represent the offset between the subscript and superscript position. As a result of this, the nominal TFM width is usually smaller than the actual character advance width, which has to be calculated by adding the TFM width and the italic correction.

While this approach makes it possible to adjust the subscript position in cases like $\mathit{\Gamma}_2$ when a character is narrower at the base, there are no provisions to adjust the superscript position independent of the total width as would be needed in cases like $\mathit{\Delta}^2$ when a character is narrower at the top.

Ideally, one might wish to have an extended TFM file format which would allow to specify both the subscript and superscript positions independent of the width and italic correction fields.

If the latter fields could be used in the normal way, it would become much easier to derive a math italic font from a text italic fonts without having to determine appropriate adjustments to the character metrics by trial and error. However, it should be clear from the discussion in previous section that the chances for such kind of far-reaching changes are very slim before $\mathcal{NTS}$ will become a reality, since Ω has chosen to simply double the file formats while otherwise retaining the idiosyncrasies of TEX.

Accent positioning. TEX assumes that accents taken from text or math fonts are already properly positioned for use with lowercase letters that do not exceed *x_height*. For capital letters and lowercase letters with ascenders, the accents are automatically shifted upwards by the difference of the actual height of the character and *x_height*.

As for the horizontal positioning of accents, the accent is shifted by an appropriate amount to compensate the difference of the widths of the accentee and the accent glyph, but in the case of single letters an offset is added, taken from the pseudo-kern pairs between the accentee and the `\skewchar` of the font. If support for under-accents is added at the macro level, a similar scheme can be used, in which the offset is stored in the corresponding pseudo-kern pairs between the `\skewchar` and the accentee.

While it is obvious that the whole `\skewchar` business is just a result of having a counter-intuitive interpretation of the glyph metrics in the first place, the need for pseudo-kern pairs doesn't usually cause too many technical problems, as long as a special glyph can be set aside for use as the `\skewchar`, which doesn't need to be kerned with anything else. It would still be preferable to find a better solution to represent such offsets when it comes to $\mathcal{NTS}$.

Extensible wide accents. While TEX's `charlist` mechanism allows to link both vertical and horizontal objects of increasing size, such as big delimiters of different height or wide accents of different width, there is no horizontal counterpart of vertical extensible delimiters. Horizontal extensible delimiters such as over- and underbraces therefore have to be constructed using individual pieces for the ends and the middle and an `\hrulefill` to fill the gaps.

Since the use of TEX's `extensible` or `charlist` features only depends on the circumstances when the corresponding flags in the TFM file are looked at, it should be straight-forward to add support for extensible wide accents already in ε-TEX without requiring changes to the TFM file format at all. Once such a mechanism is in place, it should be possible to redefine over- and underbraces in terms of wide over- and underaccents.

Centering of big delimiters. TEX centers big and extensible delimiters on the math axis with respect to their total height and depth. The *axis_height* parameter is taken from the `\fontdimen`s of the math symbol font in family 2, and is used simultaneously for centering a variety of different objects, such as fractions, `\vcenter` atoms, big operators, and big delimiters.

While this does produce reasonable results for most font sets, it was pointed out by Berthold Horn that the design of certain typefaces, notably Lucida Bright, might actually require a secondary delimiter axis independent of the default math axis, on which only the delimiters would be centered.

While it is pretty clear that such an effect cannot easily be achieved within the context of the existing TEX, it seems plausible that this could easily be implemented in ε-TEX, simply by introducing an additional dedicated `\fontdimen` parameter for this purpose and slightly modifying the typesetting algorithms for the placement of delimiters.

If such a feature would be implemented as an enhancement that has to be activated explicitly, compatibility with existing fonts could be assured while offering a test bed for potential improvements for font sets taking advantage of this feature.

Centering of big radicals. TEX assumes that the glyphs containing radical signs are designed in a very special way. The height of the radical sign is assumed to be the same as the height of the horizontal rule to be put on top of the root with the consequence that the remainder of the radical sign will have to end up below the baseline.

This assumption has a number of rather unfortunate consequences and also implies that a number of criteria have to be fulfilled in math symbol fonts providing radical signs, as outlined in detail in another discussion paper.

While it is clear that the present situation of having the height of the rule depend on the glyph height rather than the *rule_thickness* parameter is very troublesome, it isn't clear whether a change of the typesetting algorithm could be implemented as an enhancement in ε-TEX or whether addressing this problem would have to wait for $\mathcal{NTS}$.

◇ Ulrik Vieth
Heinrich-Heine-Universität
Institut für Theoretische Physik II
Universitätsstraße 1
D-40225 Düsseldorf, Germany
`vieth@thphy.uni-duesseldorf.de`