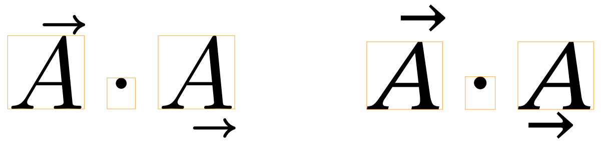
On bottom accents in OpenType math

Hans Hagen, Mikael P. Sundqvist

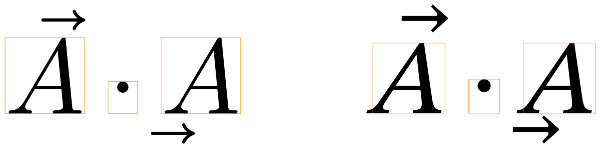
We recently worked on accents in math in ConTEXt. While looking at the bottom accents, we realized that there was work to be done, since Microsoft did not specify how to deal with them (there is support for bottom accents built in to LuaMetaTEX). While discussing and examining examples, we noticed that fonts behave differently, once again.

Let's take the `\wideunderrightarrow` (or simply `\underrightarrow`) as an example. This glyph (`U+20EF`) is missing in the reference font Cambria, but it is available in Latin Modern Math and STIX Two Math, among others. We were surprised to find that the glyph had no width and was positioned with the arrow tip pointing at the x-coordinate zero, but maybe that reflects some previous "standard" on typesetting them. We also saw that in Latin Modern, it had no accent anchor point set, but in STIX Two, it did. Anchor points on the base glyph and the accent are to be aligned.

Until recently, this image shows how it came out in ConTEXt. We show Latin Modern to the left and STIX Two to the right.



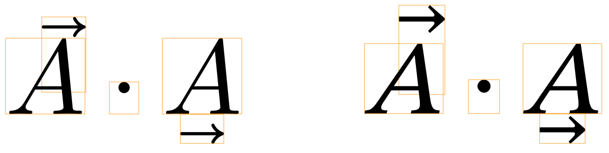
The horizontal location of the top accent is controlled by the anchors of the base character and the accent: they align. We have not changed this behavior because it is, after all, part of the specification. The location of the bottom accent was never specified by Microsoft. Some OpenType fonts mimic old TEX fonts, others mimic Cambria. We wanted a simple model that works well with all fonts. Notice that none of the arrows have orange (grayscaled for print) boxes around them, meaning that they do not carry real widths. The first attempt was to simply mid-align the bottom accents. This came out as follows:



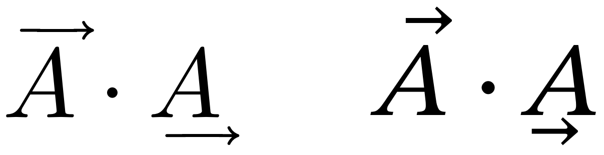
The problem that the arrows do not have a bounding box is now apparent. Looking in FontForge, we found that the tips of the arrows are located at x-coordinate zero. We thus needed to force the arrows

doi.org/10.47397/tb/44-2/tb137hagen-bottomaccent

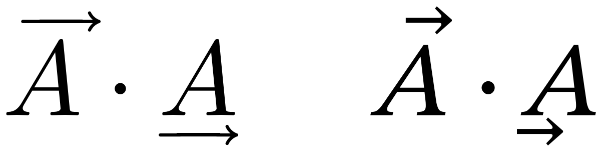
to get their true widths. This was done, and then the centering worked better:



During the process, we wondered if we were making any obvious mistakes. We compared with the LaTeX output of the same examples. With LuaLaTeX we got:



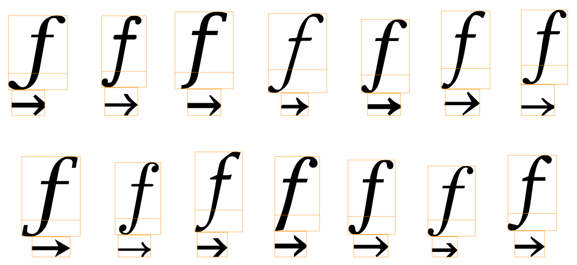
and with XeLaTeX we got:



We notice that Latin Modern (left) worked well in XeLaTeX, but did not look great in LuaLaTeX. On the other hand, STIX Two (right) worked well in LuaLaTeX but not in XeLaTeX. There can be several reasons for this: one can use a traditional TeX engine setup and map OpenType functionality, fonts and parameters to that (maybe that is what XeTeX does) or one can take the traditional fonts, parameters and expectations and translate these to OpenType math rendering (which is what LuaTeX does). Mix that with fonts that are predominantly traditional (Latin Modern) or standard (like Cambria) and you start to see the confusing picture.

For these reasons, the LuaMetaTeX engine adds a lot of detailed control in order to deal with a mismatch. However, the fact we still get unexpected outcomes also points to possible issues (inconsistencies) in fonts. When a designer makes a new math font, a lot of how it behaves depends on what font was taken as its reference.

In the process of getting the best possible output we decided to only use anchor points for the top accents and simply center the bottom anchors under the original box of the glyph. We have discussed elsewhere our getting rid of italic correction (by changing the bounding box and introducing a lower right corner kern). We show here the math italic f in many fonts; it's often one of the more problematic characters, since it sticks out from its box (before we tweak it).



To sum up, there is a problem with how to place bottom accents in Unicode math. The fonts seem to suggest different approaches but the underlying problem lies in the absence of a standardized approach. In light of this, we propose a solution that we hope will effectively address this issue for our users. Depending on the font, an accent has a width or not. When it doesn't have one, we see the mentioned horizontal displacements combined with strange anchors. The displacement sort of positions at the bottom, and the anchor aligns with the character. Because we don't want to rely on side effects we calculate the width from the bounding box and recalculate the anchors. Once more it is more reliable to simply ignore one aspect of OpenType math and individual font implementations.

Our discussion above considers accents below single characters. For multiple characters, we use the variants and extensibles to try to match the total width.

Let us mention one more odd thing that we noticed in connection with this. In STIX Two Math, the bottom accent arrow in fact has five variants and then an extensible recipe. It would have sufficed to provide an extensible recipe and no variants. Moreover, the extensible recipe does not use the base character but the first variant, and that complicates matters if one wants to wipe the variants and go directly to the extensible recipe.

◇ Hans Hagen
Pragma ADE

◇ Mikael P. Sundqvist
Department of Mathematics
Lund University