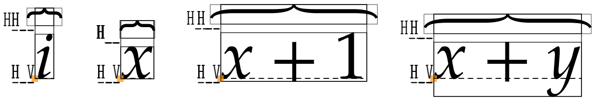
OpenType extensible brace debugging

Hans Hagen, Mikael P. Sundqvist

When you combine writing a new math manual with development of the math typesetting subsystem, you can run into surprising buglets.

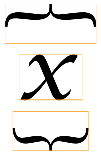
A valid traditional TEX approach to putting braces over or under (a bit of) formula is to assemble such a brace from five snippets, where the left, middle and right snippet are characters and the "even" ones are rules that can stretch. An OpenType math font can have line segments that are used instead of rules. But before that assembly happens, one can first check if there are precomposed wider variants in the font.

Consider the following formula elements (typeset with simply `$\overbrace{i}$`, etc.):

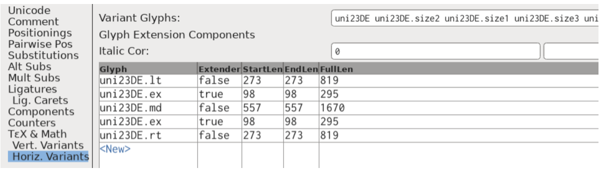


When you test examples like this you would expect the overbrace on the i to be narrower than on the x. But in the above, using Pagella (its design makes the problem apparent), we see they are the same size. Also, the braces above $x+1$ and $x+y$ are slightly wider than they need to be.

Another issue is that we expect braces above and below (with `\underbrace`) to have the same size. Switching to Lucida for the example below, we can see that they don't; the brace on top is noticeably bigger:



Of course you will seldom put a brace both on top of and below a single character, and that is likely why this went unnoticed for quite a while. And, because normally code involved in handling this is kind of symmetric, one starts wondering about the font. And indeed, when we looked into the font, we found that `uni23DE` (TOP CURLY BRACKET) had an error in the variant list: `.size1` and `.size2` were swapped. In FontForge (the variant list is given in the box at the upper right of the screenshot):



Variant Glyphs: uni23DE uni23DE.size2 uni23DE.size1 uni23DE.size3 uni

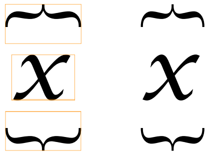
Glyph Extension Components

Italic Cor: 0

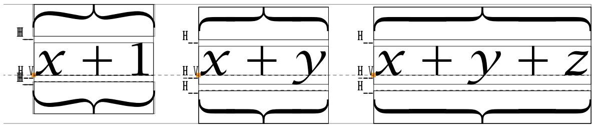
Glyph	Extender	StartLen	EndLen	FullLen
uni23DE.lt	false	273	273	819
uni23DE.ex	true	98	98	295
uni23DE.md	false	557	557	1670
uni23DE.ex	true	98	98	295
uni23DE.rt	false	273	273	819
<New>				

Because we had just updated this font[1] we immediately thought that we had messed up, but a look at the old version showed us that this swap was already there: it just went unnoticed! The simple fact that we showed both braces made it prominent. How likely is it to have two braces on a single character?

Sorting the sizes correctly results in braces of the same size (left below), but you may notice that the braces are still wider than the x. We can fix this by scaling the brace slightly (right below).

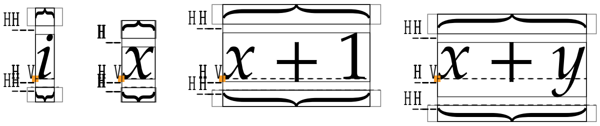


While looking closer at this, we realized that another problem exists: Lucida's widest precomposed brace character is much smaller than the narrowest extensible brace. Also, the precomposed braces are naturally "curvier" than the extensible ones, so the braces for $x+1$ are quite different compared to $x+y$:



Since the different curviness is by design it is nothing to fix, but in a future version we might need to add a size or modify the extensible recipe.

Since the problems have gone unnoticed, there is no hurry to push another release, though. Meanwhile, in ConTEXt, we can easily fix the swap with a tweak in the goodie file (by sorting variants on size), and users will not notice the gap between the sizes, since we stretch or shrink the result. Here we show how it looks for Pagella, to be able to compare with the first figure, with stretching and shrinking enabled:



Note that the sizes of all braces now match their content.

Font debugging grows ever more complex ...

◇ Hans Hagen
Pragma ADE

◇ Mikael P. Sundqvist
Department of Mathematics
Lund University
mickep (at) gmail dot com

[1] tug.org/TUGboat/tb43-3/tb135hagen-lucida.pdf

doi.org/10.47397/tb/44-1/tb136hagen-extensible