Pushing math forward with ConTEXt lmtx

Hans Hagen, Mikael P. Sundqvist

Editor's note: For enlarged views of many of the small details presented here, please also see the slide presentation, available from `tug.org/tug2022`.

Abstract

We report on some recent work on mathematical typesetting. Our main purpose has been to make both the input and output of math cleaner and more structured. Among the many enhancements, we mention here the introduction of new atom classes that has given better control over many details. We also cover the unboxing of fenced material which, together with improved line-breaking and more flexible multiline display math, has created a coherent way to produce displayed formulas that split over lines.

1 Introduction

When creating ConTEXt MkIV, the ConTEXt version that is based on LuaTEX, we only had Cambria available as an OpenType math font with a complete math table, so for the others we started out with runtime virtual math fonts assembled from traditional TEX fonts. Stepwise, more fonts became available. Not all fonts conform to the way Cambria does things. In order to deal with the inconsistencies in these fonts and because the specification was vague — it is better now — and had to be derived from e.g. how Microsoft Word does things, we ended up with a mix of generic runtime fixes to fonts and font-specific corrections done in what we call goodie files. However, given the amount of work involved, it never became complete.

That all changed when we became aware of Lansburgh's 1964 book [1]. This book predates TEX, and it is, with its more than 400 pages of well-motivated typesetting rules — the majority of them about mathematics — the most comprehensive guide we are aware of. The book, written in Swedish, was originally used as a typesetting guide for the publisher Almqvist & Wiksell, in particular for the highly respected mathematics journal *Acta Mathematica*. For this reason Lansburgh discusses the rules for typesetting mathematics in great detail.

The question became: why can't we do now what was recommended 50 years ago? The LuaMetaTEX math engine was already at this point partially redone and more configurable, but why not go further? We knew it would take much time to get all done, and it did (basically years of full time), but here we are. In the process we looked over the goodie files (they are now organized tweaks) and all fonts, by now stable but flawed, were studied in detail. A direct consequence was rewriting the LuaTEX math engine to permit more control. So, in some ways, one has to thank Lansburgh for our work.

In this article we discuss only some of what the end user sees. At a lower level it all boils down to configuring the many (also new) font parameters, selectively fixing properties of glyphs, adding additional properties such as staircase-like kerns for some, setting up lots of pairwise spacing and penalties (we inherit where we can, so that saves some effort), defining rules that influence the inter-atom handling, etc. The LuaMetaTEX engine is completely configurable, meaning that we have more variables that can be set, and one can even change the styling rules, of which many were hard-coded. This is why a project like this takes much time and dedication but is also much fun.

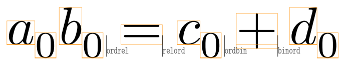
One note has to be made: Don Knuth did a tremendous job on TEX and the math engine, and only by working with the code can one realize how quickly it was all achieved: we're baffled. It does what was possible within the constraints of hardware and fonts, and it does it well. For instance, when we mention the `\nulldelimiterspace` parameter that we try to avoid, it doesn't mean that it was not there for a reason: there is a subtle interplay between fonts, where characters have italic corrections as the means for spacing and attachments of sub/superscripts, and a zero-ordered spacing that then cooperates nicely with the few relatively unknown spacing parameters. As with everything TEX: it all makes sense when you see it in perspective and there are excellent tricks to be found in there. That said: we took advantage of today's faster processors, plenty of memory, fonts that collect all shapes into one with more properties per font and shape, and in the end "time", as we were under no pressure to finish this soon.

To date, enough has been done to fill a whole issue of *TUGboat*. Maybe we will wrap up some more in articles in due time. After all, we also have an additional wishlist to fulfill.

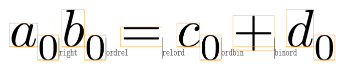
2 Math microtypography

By math microtypography we mean the fine-tuning of small details in mathematical formulas. Let us give an example. When you type `a_{0}b` in math mode you get a_0b. Have you ever noticed that there is a small space automatically inserted between the 0 and the b? If the space is not there, as in a_0b, it is no longer clear if the 0 belongs to the a or to the b.

There are occasions when this space is unwanted. For example, we usually expect a symmetric space around relations (as in $a = b$) and binary symbols (as in $a + b$). If, however, there is a subscript (or a superscript) just to the left of such a symbol, the surrounding space becomes uneven because of the inserted extra space.



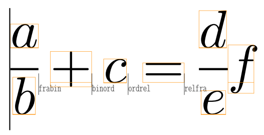
The space is specified by the `\scriptspace` parameter. Don Knuth set it to 0.5pt in plain TEX (likely a choice that looked good with his 10pt bodyfont size). The `\scriptspace` parameter, and its particular value, has survived several decades, formats, body font sizes and engines. In ConTEXt lmtx we have introduced several options for the different math atom classes. One of these class options is `\nopostslackclassoptioncode`, and if it is set for a class then any inserted `\scriptspace` will be removed. Looking at the example above we see that the unwanted extra space is present before = and +. And indeed, both the relation class and the binary class do have this option set. Thus, when we typeset the formula above in ConTEXt lmtx we get the following.



The space between the a_0 and the b is in fact no longer a `\scriptspace`, but we instead rely on the font parameter `SpaceAfterScript`.

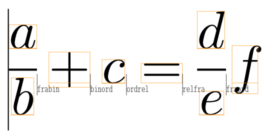
The situation with `\nulldelimiterspace` is a bit similar. It is traditionally used as a kind of side bearing in fences and fractions. Its value was in plain TEX set to 1.2pt, and that has also stayed. In the formula $\frac{1}{2}a$ the space is inserted between the $\frac{1}{2}$ and the a and without it the formula would look bad: $\frac{1}{2}a$.

The `\nulldelimiterspace` is, however, also inserted *before* the fraction $\frac{1}{2}$, making the space before the formula slightly (1.2pt) larger than the space after it. This means that the margin will not be perfectly aligned if the fraction is located at the beginning or at the end of a line.



In addition to the extra space at the left margin, the spaces around the + and the = above have become asymmetrical due to the inserted 1.2pt space.

In ConTEXt lmtx we use new atom classes to control the spacing around fractions. One of the new atom classes is the fraction class. Thus, we set the `\nulldelimiterspace` value to 0pt.



Observe that no space is inserted to the left of the first fraction.

In the examples above we have used one of the many ConTEXt helpers (`\showmakeup[mathglue]`) to visualize the inserted spaces. For instance, to the right of the fraction we see `frabin`, which means that the classes that meet are fraction and binary; the space between them is set up to be a `\medmuskip`. We also used `\showglyphs` to draw the bounding boxes in orange (grayscaled in print). These and other helpers have been indispensable for our work.

3 A more general spacing model

In traditional TEX the spaces between atoms have traditionally been set to one of the following muskips.

```
\thickmuskip  5mu plus 5mu
\medmuskip    4mu plus 2mu minus 4mu
\thinmuskip   3mu
\zeromuskip   0mu
```

For example, between an ordinary and a binary atom, TEX inserts a `\medmuskip`. It has not been possible to set up the space between a single pair of atoms without altering the spaces between others.

In ConTEXt lmtx the inter-atom spaces are no longer hard-coded to `\thickmuskip`, `\medmuskip` and `\thinmuskip`. Users are free to define new muskips and to use them between any atom pair. After a lot of testing, we decided to alter the old muskips just a little, and added two new ones.

```
\thickmuskip  5mu plus 3mu minus 1mu
\medmuskip    4mu plus 2mu minus 2mu
\thinmuskip   3mu
\tinymuskip   2mu minus 1mu
\pettymuskip  1mu minus 0.5mu
\zeromuskip   0mu
```

We use the `\tinymuskip` for example between the radical and ordinary atoms, and between ordinary and fraction atoms. Traditionally, there is no space inserted in the first case.

$$a\sqrt{b}c + d\frac{e}{f}g$$

This is how it looks in ConTEXt lmtx:

$$a\sqrt{b}\,c + d\,\frac{e}{f}\,g$$

Note that there is a space between the $\sqrt{b}$ and the c.

The `\pettymuskip` is mostly used in scriptstyle, in sub- and superscripts, where TEX traditionally inserts no space. We don't know why, but it might be that one simply wants the formulas to take less space. It might also be that the smallest available non-zero muskip, the `\tinymuskip`, was too big.

$$\sum_{k=0}^{j+n} a_k = e^{a+b-c}$$

With the `\pettymuskip` added, it looks like below, and you can judge for yourself whether it looks better or not.

$$\sum_{k=0}^{j+n} a_k = e^{a+b-c}$$

The observant reader has now realized that the spacing between atoms not only can be set to values other than the traditional four, but also they can also be different in different math styles. Indeed, when we do the setups we have access to the following keywords.

```
\alldisplaystyles
\alltextstyles
\allscriptstyles
\allscriptscriptstyles
\allmathstyles
\allsplitstyles
\alluncrampedstyles
\allcrampedstyles
```

Let us show examples with one of the new classes, the exponential class. This is a very small class, with currently only one member, the exponential e, accessed via `\ee`. This class is set up to inherit the inter-atom spaces from the ordinary atom class.

```
\setnewconstant \mathexponentialcode
    \mathclassvalue exponential

\copymathspacing \mathexponentialcode
    \mathordinarycode
```

Thus, if we type

```
\dm{rs \ee^{-rs \ee^{st} tu} tu}
```

in math mode, we get

$$rse^{-rse^{st}tu}tu$$

Lansburgh suggests that a small space should be inserted between exponentials and other symbols, in particular if it carries exponents. We obtain that with the code below (and similar for the ordinary exponential combination).

```
\setmathspacing
    \mathexponentialcode \mathordinarycode
    \allsplitstyles \tinymuskip
\setmathspacing
    \mathexponentialcode \mathordinarycode
    \allscriptstyles \pettymuskip
```

This results in some extra space around the e.

$$rs\,e^{-rs\,e^{st}\,tu}\,tu$$

4 New atom classes

The classes defined in the LuaMetaTeX engine are ordinary, operator, binary, relation, open, close, punctuation, variable, active, inner, under, over, radical, fraction, middle, accent, fenced, ghost, vcenter.

The classes defined so far in ConTeXt lmtx are imaginary, differential, exponential, ellipsis, function, digit, explicit, division, factorial, wrapped, construct, mathpunctuation, dimension, unspaced, begin, end, all and unary.

You probably recognize many of the engine classes from classical TeX. We felt that it would be good to convert some standard constructions, like fractions and radicals, to their own classes. Once we decided to open up for more classes, we rapidly found a use for several new ones, some with just a few members, and some of a more technical nature. Let us give some comments.

Fractions and radicals are now their own classes, and since fenced material inherits their class structure from the content, there is currently no use of the inner class in ConTeXt lmtx.

The middle class, introduced in ε-TeX, was more like a technical hack built on top of the open class. In ConTeXt lmtx it is a true atom class.

The imaginary, differential, exponential, ellipsis and factorial classes have only a few members each. The differential class is perhaps the most interesting among them. The macro `\dd` yields a differential d with an adapted spacing; that is, the code `\int_{a}^{b} f(x) \dd x` in math mode gives $\int_a^b f(x)\,dx$. With

```
\setupmathematics[differentiald=upright]
```

the `\dd` gives an upright d instead, $\int_a^b f(x)\,\mathrm{d}x$.

The factorial class consists of only one character, the exclamation mark. It is merely there to automatically get a small space between the exclamation mark and an ordinary symbol. Thus, if we type `\binom{n}{k} = \frac{n!}{(n-k)!k!}` in display math mode it comes out as follows.

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

Observe the extra space between the)! and the k.

5 Tweaking fonts

Take a look at the following formula, set with TeX Gyre Bonum Math:

$$\mathscr{q}[f] = \int_0^\pi [f'(t)]^2\,dt$$

In the so-called “goodie files” we have collected fixes for the various math fonts. Let us look at the same formula, with the fixes in the goodie file applied.

$$\mathfrak{q}[f] = \int_0^\pi [f'(t)]^2\,dt$$

We fix most issues in so-called tweaks, but some are also done with the help of font parameters, some of which are our own. We used

- the dimension tweak to scale the whole fraktur lowercase alphabet.
- the same tweak to modify the bounding box and italic correction of lower case italic f so that it does not clash with other letters.
- the font parameter `DisplayOperatorMinHeight` to increase the size of the integral sign, and `NoLimitSubFactor` to move the lower limit closer to the integral sign.
- the fixprimes tweak to move the prime down (see further discussion below).

The details of these fixes, as well as others, can be found in the goodie file `bonum-math.lfg`.

In order to solve the persistent issues with primes (fonts differ widely in that) the engine now supports primes natively. This means that every atom can have a super- and subscript, a super- and subprescript as well as a prime attached. Optionally, scripts can be shifted to behave like an index. The fact that we need to deal with all four corners of a nucleus also means that we need to make sure that the glyphs behave well at both ends. That gave us some extra work. There are additional parameters to control the relative positioning of primes and superscripts.

Some Unicode math fonts, including Bonum, have several sizes of the integral sign, and we can use `\startintegral` and `\stopintegral` to make them grow as delimiters. This means we can write

```
\startintegral[bottom={a},top={b}]
\frac{1 + \frac{f_1(x)}{f_2(x)}}
     {1 + \frac{f_3(x)}{f_4(x)}} \dd x
\stopintegral
```

in math mode, to get

$$\int_a^b \frac{1+\frac{f_1(x)}{f_2(x)}}{1+\frac{f_3(x)}{f_4(x)}}\,dx$$

Technically, the integral sign works as the left part of a paired delimiter. Thus, we see an example of a paired delimiter where the sub- and superscript are placed on the left delimiter. It is also possible to set the size of the integral sign manually. You can play with `\int[size=50pt]` if you need specific sizes.

6 Math macrotypography

ConTEXt has in the past had good support for typesetting displayed equations, and there has been rather complete support for different types of alignments, numbering of equations, and so on [2].

Multiline formulas have historically been set in TEX via the `\halign` primitive. As a consequence these formulas have in fact been an array of math mode cells.

In ConTEXt there has for some time existed partial support for displayed formulas typeset as paragraphs, but they were not configured for real usage. When we opened up the set of atom classes, and introduced the unboxing of subformulas, it was also a good time to set this up and extend the functionality.

We build the formulas as one long formula, and do the layout mainly with the `split` and `align` keys. The user can also insert manual formatting with `\breakhere`, `\skiphere` and `\alignhere`.

The default value for the `split` key is `text`, and that means that formulas can split over lines, but not over pages. If we also want them to split over pages, we set `split` to `page`. The only difference between these two settings is the setup of penalties, and it is possible for the user to define their own. For formulas that fit on a line it does not matter.

$$\|P(\lambda) - P(\lambda_0)\|_{L^2(\Gamma)\to H^{5/2}(\Omega)} \leq C|\lambda - \lambda_0|$$

Longer formulas automatically split over lines.

```
\startformula
    \iint K(xy) f(x) g(y) \dd x \dd y
    \leq \phi(p^{-1})
    \left[\int x^{p-2}f(x)^p \dd x\right]^{1/p}
    \left[\int g(y)^q \dd y \right]^{1/q}
\stopformula
```

$$\iint K(xy)\,f(x)\,g(y)\,dx\,dy \leq$$

$$\phi(p^{-1})\left[\int x^{p-2}f(x)^p\,dx\right]^{1/p}\left[\int g(y)^q\,dy\right]^{1/q}$$

The splitting of the formula can be prohibited by adding `split=no` as an argument to `\startformula`. We get a formula that is set in a box (here we clip the formula so as not to mess up the formatting of this article).

$$\iint K(xy)\,f(x)\,g(y)\,dx\,dy \leq \phi(p^{-1})\left[\int x^{p-2}f(x)^p\,dx\right.$$

If we instead add `align=slanted`, and also insert a `\breakhere` just before the `\leq`, we get

$$\iint K(xy)\,f(x)\,g(y)\,dx\,dy$$

$$\leq \phi(p^{-1})\left[\int x^{p-2}f(x)^p\,dx\right]^{1/p}\left[\int g(y)^q\,dy\right]^{1/q}$$

The `align=slanted` flushes the first line left, the last line right, and midaligns the other lines. This key can be given any of the values `middle` (default), `flushleft`, `flushright` and `slanted`.

With the default values of `split` and `align` we can easily add an align point with `\alignhere`.

```
\startformula
    \tfrac{1}{2}( p^2 \abs{x} + \abs{x} p^2 )
    \alignhere
    = \abs{x} p \abs{x}^{-1} p \abs{x}
    - \tfrac{1}{2} \abs{x}
      ( \laplace \abs{x}^{-1} ) \abs{x}
    \breakhere
    = \abs{x} p \abs{x}^{-1} p \abs{x}
    - \tfrac{1}{2} \abs{x} 4 \pi \delta(0)
      \abs{x}
    \breakhere
    = \abs{x} p \abs{x}^{-1} p \abs{x}
\stopformula
```

Observe the `\breakhere` where we want new lines.

$$\begin{aligned}\tfrac{1}{2}(p^2|x| + |x|p^2) &= |x|\,p|x|^{-1}\,p|x| - \tfrac{1}{2}|x|\,(\Delta|x|^{-1})\,|x| \\ &= |x|\,p|x|^{-1}\,p|x| - \tfrac{1}{2}|x|\,4\pi\delta(0)\,|x| \\ &= |x|\,p|x|^{-1}\,p|x|\end{aligned}$$

The careful reader also notes that there is a space after the close atoms, something that was suggested in [1]. Compare the final term on the right-hand side with $|x|p|x|^{-1}p|x|$, with no such space inserted.

We end with a slightly more advanced chain formula.

$$\begin{aligned}&\text{A}P'(iy_1, \ldots, iy_n, i\eta_k) \\ &\qquad \text{B}= (ir)^{k-1}\left[P'_k\left(z_1, \ldots, z_n, \frac{\eta_k}{r}\right)\right. \\ &\qquad \text{B} \qquad\qquad \text{S 3}+\frac{1}{ir}P'_{k-1}\left(z_1, \ldots, z_n, \frac{\eta_k}{r}\right) + \ldots\left.\right] \\ &\qquad \text{B}= (ir)^{k-1}\,P'_k\left(z_1, \ldots, z_n, \frac{\eta_k}{r}\right) + O(r^{k-2})\end{aligned}$$

Here we have marked the align point with an `A`. Its position might at first glance be a bit surprising. To the formula we have added `textdistance=2em`. This is the space that is automatically added at each `\breakhere`, the extra horizontal shift you see at the `B`, compared to the `A`. At one row we have in addition added `\skiphere[3]`, that adds an extra space of 6em (the configurable unit is by default 2em). This is shown as `S 3`. This is how we typed the formula:

```
\startformula[textdistance=2em]
    \alignhere
    P'(iy_1, \ldots, iy_n, i\eta_k)
    \breakhere
    = (ir)^{k-1} \left[
    P_k' \left( z_ 1, \ldots, z_n,
                \frac{\eta_k}{r} \right)
    \breakhere
    \skiphere[3]
    + \frac{1}{ir}P_{k-1}'
        \left( z_ 1, \ldots, z_n,
                \frac{\eta_k}{r} \right)
    + \ldots
    \right]
    \breakhere
    = (ir)^{k-1}
        P_k'\left( z_ 1, \ldots, z_n,
                    \frac{\eta_k}{r} \right)
    + O(r^{k-2})
\stopformula
```

We emphasize that the formula above is broken inside the `\left[` and `\right]` fences. This is thanks to the possibility to unpack and repack subformulas.

One of the things we're currently experimenting with is carrying over kerns at the corners of nested subformulas. For instance, when a fenced formula, fraction, radical or any composed atom is prepared, it happens in a nested call to the mlist-to-hlist converter. The content is sort of abstract and wrapped in an atom of some class (say fenced) that determines spacing. In that case, anchoring a superscript cannot be related to the shape of, for instance, the right fence, which can have some extreme inward bending shape (as in Cambria). Dealing with that is not entirely trivial, but we managed to get it working. Of course, we then need to add shape-related kerning information to the goodie files because it is not part of the OpenType math concept. It is all about look and feel here.

References

[1] W.N. Lansburgh. *Almqvist & Wiksells sättningsregler*. Almqvist & Wiksell, 1964.

[2] A. Mahajan. My way: Using `\startalign` and friends, 2006. `dl.contextgarden.net/myway/mathalign.pdf`

◇ Hans Hagen
`https://pragma-ade.nl`

◇ Mikael P. Sundqvist
Department of Mathematics
Lund University
Box 118
221 00 Lund
Sweden
`mickep (at) gmail dot com`