Math:

progress or standing still

Hans Hagen

TUG Conference

Tokyo, October 2013

Math as script	Alphabets	Heavy bold	Radicals
Primes	Accents	Stackers	Fences
Directions	Structure	Italic correction	Big
Macros	Unscripting	Combining fonts	Tracing

Math as script

- math can be input using the TEX syntax, MATHML, calculator like sequences, ...
- but apart from content MATHML all stay close to good old TEX
- although not officially a script, OPENTYPE treats it as such, but without control

```
$ ( (x + 1) / a + 1 )^2  = (x - 1) / b $

$ \left( \frac{x + 1}{a}  + 1 \right)^2  = \frac{x - 1}{b} $

<mfenced open="(" close = ")">
  <mfrac>...</mfrac> <mo>+</mo> <mn>1</mn>
</mfenced>

<mrow>
  <mo>(</mo> <mfrac>...</mfrac> <mo>+</mo> <mn>1</mn> <mo>)</mo>
</mrow>
```

There is recognition of math as a proper (but not standardized) script.

Alphabets

- the shape (style) of a character determines its meaning
- but in most cases an type a is entered as ASCII character
- and tagged with some rendering directive, often indicating a font style
- in traditional TEX we have alphabets in different fonts, so we're talking switches
- in UNICODE and OPENTYPE we have alphabets with standardized code points (but gaps too)
- this has big advantages for communicating, transferring data etc
- but a math engine still has to deal with ASCII input as well
- multiple axis: types, alphabets, styles, variants, shapes, modifiers

We're off better but the gaps are an anomality.

Heavy bold

- for titles and captions we might need bolder math
- bold symbols in math have special meaning
- so when going full bold they should become heavy
- heavy math involves boldening everything, including extensibles
- there are currently no fonts that have such complete heavy companions

We need proper bold fonts, but they need to be relatively complete.

Radicals

- this always has been (and still is) a combination of vertical extensibles and horizontal rules
- it is the only two dimensional extensible so always a bit of an exception
- in the wide engines we now have more direct support primitive for that (no macro needed)
- in practice (at least in MkIV) we still use macros because we want control

Native support for radicals is nice to have and makes coding cleaner (**ex**).

Primes

- this is a special case as we (sort of) have upto two superscripts
- and also need to handle an optional subscript of the base symbol
- and in order to be visually okay, we need to collect multiple primes
- some fonts have primes raised, some have them flying high
- maybe at some point the upcoming math pre- and postscripts will help

Supporting primes will always be a bit of a pain but I stay on top of it.

Accents

- they can go on top or below one or more characters (also in combination)
- accents have some hard codes positional properties
- the wide engines have more direct support for this
- fonts provide a limited set of sizes, such accents cannot extend (by design)

Engine support for accents is better now but maybe fonts need to have more sizes (**ex**).

Stackers

- arrows (and other horizontal extensibles) traditionally were made from snippets
- we need them also for chemistry, in rather flexible ways
- in upcoming math fonts they are become real extensibles
- but then we still need to deal with existing fonts that lack them (one font in the end)
- there will be native support for so called character leaders

Stackers are more easily implemented although fonts pose some challenges.

Fences

- these go left and right (or in the middle) of things
- there need to be a matching pair else we get an error
- they have to adapt their size to what they wrap
- TEXies can take care of that in their input
- but in for instance MATHML checking all this is a bit of a pain
- this is still the domain of macros
- but we could make the engines a bit more tolerant (hard to do)

Matching fences will always be a bit of a problem.

Directions

- bidirectional math is mostly a matter of the availability of fonts
- there need to be some agreement (at the macro package level) of control
- it's (for me) a visually interesting challenge
- there are some TEXies working on these matters (quite some research is done already)

Right to left math will show up thanks to pioneers.

Structure

- demand for tagging also means that we need to carry a bit more info around
- this puts a little more burden on the user
- in the end it largely is a macro package issue
- better tagging of input can also help rendering
- detailed control at the TEX level makes that users can spoil the game

In these times structure gets more important so minimal coding is less an option.

Italic correction

- in traditional TEX fonts this was used for spacing as well as special purposed
- across fonts there was never much correction
- OpenType doesn't have this concept
- OpenType math has some of if but also more powerful kerning
- generally speaking: we can ignore italic corrections

We need to accept that old concepts die and new onces show up.

Big

- normally extensible fences are chosen automatically
- but macro packages provide tricks to choose a size
- extensible steps are unpredictable but still several mechanisms can be (and are) provided

Users will always want control and no engine can provide that but macros can.

Macros

- some special symbols were constructed by macros (and using special font properties)
- these are mostly gone (like the diagonal dots)
- if it is ever needed again, we should extend the fonts

Thanks to new font technologies and wide engines need less dirty tricks.

Unscripting

- you can bet on those funny Unicode super and subscripts showing up in input
- it's a somewhat limited and unuseable lot for math (a modifier would have made more sense)
- it's one of these legacies that we need to deal with
- so the macro package needs to intercept them and map them onto proper math

We always need to deal with weird input, if only because standards lack.

Combining fonts

- we can expect math fonts to be rather complete and if not, you should choose another one
- but sometimes (for simple math) you want to swap in alphabets and digits that match the text font
- given that we talk of ranges this is easy to support at the macro package level

Although fonts are more complete, occasional combinations should remain possible.

Tracing

- there are lots of symbols involved
- and we have those extensibles too
- the larger the fonts get the more checking we need to do
- so macro packages need to provide some tracing options (or tables in print)

We keep an eye on things.

example-root.tex

```
\starttext

\startuniqueMPgraphic{math:radical:extra}
  draw
    math_radical_simple(OverlayWidth,OverlayHeight,OverlayDepth,OverlayOffset)
    withpen pencircle
      xscaled (2 * OverlayLineWidth)
      yscaled (3 * OverlayLineWidth / 4)
      rotated 30
%     dashed evenly
    withcolor OverlayLineColor ;
\stopuniqueMPgraphic

\setupmathradical
  [sqrt]
  [alternative=mp,
   mp=math:radical:extra,
   color=darkred]

\startTEXpage[offset=10pt]
    $ y = \sqrt   { x^2 + ax + b } $            \blank
    $ y = \sqrt[2]{ x^2 + ax + b } $            \blank
    $ y = \sqrt[3]{ \frac{x^2 + ax + b }{c} } $
\stopTEXpage

\stoptext
```

example-accent.tex

```
\starttext

\startMPextensions
    vardef math_ornament_hat(expr w,h,d,o,l) text t =
        image (
            fill
                (w/2,10l) -- (w + o/2,o/2) --
                (w/2, 7l) -- (  - o/2,o/2) --
                cycle shifted (0,h-o) t ;
            setbounds
                currentpicture
            to
                unitsquare xysized(w,h) enlarged (o/2,0)
        )
    enddef ;
\stopMPextensions

\startuniqueMPgraphic{math:ornament:hat}
    draw
        math_ornament_hat(
            OverlayWidth,
            OverlayHeight,
            OverlayDepth,
            OverlayOffset,
            OverlayLineWidth
        )
    withpen
        pencircle
            xscaled (2 * OverlayLineWidth)
            yscaled (3 * OverlayLineWidth / 4)
            rotated 30
    withcolor
        OverlayLineColor ;
\stopuniqueMPgraphic

\definemathornament [mathhat] [mp=math:ornament:hat,color=darkred]

\startTEXpage[offset=10pt]
    \dorecurse{8}{$\mathhat{\blackrule[width=#1ex,color=gray]}$ }
\stopTEXpage

\stoptext
```