

Experimental Unicode mathematical typesetting: The unicode-math package

Will Robertson, Philipp Stephani and Khaled Hosny
`will.robertson@latex-project.org`

2012/05/30 v0.7

Abstract

This is the first incarnation of the unicode-math package, which is intended to be a complete implementation of Unicode maths for \LaTeX using the \XeTeX and \LuaTeX typesetting engines. With this package, changing maths fonts is as easy as changing text fonts — and there are more and more maths fonts appearing now. Maths input can also be simplified with Unicode since literal glyphs may be entered instead of control sequences in your document source.

The package provides support for both \XeTeX and \LuaTeX . The different engines provide differing levels of support for Unicode maths. Please let us know of any troubles.

Alongside this documentation file, you should be able to find a minimal example demonstrating the use of the package, ‘unimath-example.ltx’. It also comes with a separate document, ‘unimath-symbols.pdf’, containing a complete listing of mathematical symbols defined by unicode-math, including comparisons between different fonts.

Finally, while the STIX fonts may be used with this package, accessing their alphabets in their ‘private user area’ is not yet supported. (Of these additional alphabets there is a separate caligraphic design distinct to the script design already included.) Better support for the STIX fonts is planned for an upcoming revision of the package after any problems have been ironed out with the initial version.

Part I

User documentation

Table of Contents

1	Introduction	2
2	Acknowledgements	2
3	Getting started	3
3.1	Package options	3
3.2	Known issues	4
4	Unicode maths font setup	4
4.1	Using multiple fonts	5
4.2	Script and scriptscript fonts/features	6
4.3	Maths ‘versions’	6
5	Maths input	6
5.1	Math ‘style’	6
5.2	Bold style	7
5.3	Sans serif style	8
5.4	All (the rest) of the mathematical alphabets	9
5.5	Miscellanea	10
6	Advanced	16
6.1	Warning messages	16
6.2	Programmer’s interface	17

1 Introduction

This document describes the unicode-math package, which is an *experimental* implementation of a macro to Unicode glyph encoding for mathematical characters.

Users who desire to specify maths alphabets only (Greek and Latin letters, and Arabic numerals) may wish to use Andrew Moschou’s mathspec package instead. (X_YTeX-only at time of writing.)

2 Acknowledgements

Many thanks to: Microsoft for developing the mathematics extension to OpenType as part of Microsoft Office 2007; Jonathan Kew for implementing Unicode math support in X_YTeX; Taco Hoekwater for implementing Unicode math support in LuaTeX; Barbara Beeton for her prodigious effort compiling the definitive list of

Unicode math glyphs and their L^AT_EX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions; Philipp Stephani for extending the package to support LuaT_EX. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use T_EX in the future. Apostolos Syropoulos, Joel Salomon, Khaled Hosny, and Mariusz Wodzicki have been fantastic beta testers.

3 Getting started

Load unicode-math as a regular L^AT_EX package. It should be loaded after any other maths or font-related package in case it needs to overwrite their definitions. Here's an example:

```
\usepackage{amsmath} % if desired
\usepackage{unicode-math}
\setmathfont{Asana-Math.otf}
```

Three OpenType maths fonts are included by default in T_EX Live 2011: Latin Modern Math, Asana Math, and XITS Math. These can be loaded directly with their filename with both X_YL^AT_EX and LuaL^AT_EX; resp.,

```
\setmathfont{lmmath-regular.otf}
\setmathfont{Asana-Math.otf}
\setmathfont{xits-math.otf}
```

Other OpenType maths fonts may be loaded in the usual way; please see the fontspec documentation for more information.

Once the package is loaded, traditional TFM-based fonts are not supported any more; you can only switch to a different OpenType math font using the `\setmathfont` command. If you do not load an OpenType maths font before `\begin{document}`, Latin Modern Math (see above) will be loaded automatically.

3.1 Package options

Package options may be set when the package is loaded or at any later stage with the `\unimathsetup` command. Therefore, the following two examples are equivalent:

```
\usepackage[math-style=TeX]{unicode-math}
% OR
\usepackage{unicode-math}
\unimathsetup{math-style=TeX}
```

Note, however, that some package options affect how maths is initialised and changing an option such as `math-style` will not take effect until a new maths font is set up.

Table 1: Package options.

Option	Description	See...
<code>math-style</code>	Style of letters	section §5.1
<code>bold-style</code>	Style of bold letters	section §5.2
<code>sans-style</code>	Style of sans serif letters	section §5.3
<code>nabla</code>	Style of the nabla symbol	section §5.5.1
<code>partial</code>	Style of the partial symbol	section §5.5.2
<code>vargreek-shape</code>	Style of phi and epsilon	section §5.5.3
<code>colon</code>	Behaviour of <code>\colon</code>	section §5.5.6
<code>slash-delimiter</code>	Glyph to use for ‘stretchy’ slash	section §5.5.7

Package options may *also* be used when declaring new maths fonts, passed via options to the `\setmathfont` command. Therefore, the following two examples are equivalent:

```
\unimathsetup{math-style=TeX}
\setmathfont{Cambria Math}
% OR
\setmathfont[math-style=TeX]{Cambria Math}
```

A short list of package options is shown in table 1. See following sections for more information.

3.2 Known issues

In some cases, X_YTeX’s math support is either missing or I have not discovered how to access features for various types of maths construct. An example of this are horizontal extensible symbols, such as arrows that can grow longer if necessary. Behaviour with such symbols is not necessarily going to be consistent; please report problem areas to me.

Symbols for maths characters have been inherited from the STIX project and may change slightly in the long term. We have tried to preserve backwards compatibility with L^AT_EX conventions as best as possible; again, please report areas of concern.

4 Unicode maths font setup

In the ideal case, a single Unicode font will contain all maths glyphs we need. The file `unicode-math-table.tex` (based on Barbara Beeton’s `stix` table) provides the mapping between Unicode maths glyphs and macro names (all 3298 — or however many — of them!). A single command

```
\setmathfont[{font features}]{{font name}}
```

implements this for every every symbol and alphabetic variant. That means `x` to `x`, `\xi` to ξ , `\leq` to \leq , etc., `\mathscr{H}` to \mathcal{H} and so on, all for Unicode glyphs

Table 2: Maths font options.

Option	Description	See...
<code>range</code>	Style of letters	section §4.1
<code>script-font</code>	Font to use for sub- and super-scripts	section §4.2
<code>script-features</code>	Font features for sub- and super-scripts	section §4.2
<code>sscript-font</code>	Font to use for nested sub- and super-scripts	section §4.2
<code>sscript-features</code>	Font features for nested sub- and super-scripts	section §4.2

within a single font.

This package deals well with Unicode characters for maths input. This includes using literal Greek letters in formulae, resolving to upright or italic depending on preference.

Font features specific to `unicode-math` are shown in table 2. Package options (see table 1) may also be used. Other `fontspec` features are also valid.

4.1 Using multiple fonts

There will probably be few cases where a single Unicode maths font suffices (simply due to glyph coverage). The `stix` font comes to mind as a possible exception. It will therefore be necessary to delegate specific Unicode ranges of glyphs to separate fonts:

```
\setmathfont[range=<unicode range>,<font features>]{<font name>}
```

where *<unicode range>* is a comma-separated list of Unicode slots and ranges such as `{"27D0-"27EB,"27FF,"295B-"297F}`. You may also use the macro for accessing the glyph, such as `\int`, or whole collection of symbols with the same math type, such as `\mathopen`, or complete math styles such as `\mathbb`. (Only numerical slots, however, can be used in ranged declarations.)

4.1.1 Control over maths alphabets

Exact control over maths alphabets can be somewhat involved. Here is the current plan.

- `[range=\mathbb]` to use the font for ‘bb’ letters only.
- `[range=\mathbfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (also with `latin`, `Latin`, `num` as possible options for Latin lower-/upper-case and numbers, resp.).
- `[range=\mathsfit->\mathbfsfit]` to map to different output alphabet(s) (which is rather useless right now but will become less useless in the future).

And now the trick. If a particular math alphabet is not defined in the font, fall back onto the lower-base plane (i.e., upright) glyphs. Therefore, to use an ASCII-encoded fractur font, for example, write

```
\setmathfont[range=\mathfrak]{SomeFrakturFont}
```

and because the math plane fractur glyphs will be missing, unicode-math will know to use the ASCII ones instead. If necessary this behaviour can be forced with `[range=\mathfrac->\mathup]`.

4.2 Script and scriptscript fonts/features

Cambria Math uses OpenType font features to activate smaller optical sizes for scriptsize and scriptscriptsize symbols (the B and C , respectively, in A_{BC}). Other fonts will possibly use entirely separate fonts.

The features `script-font` and `sscript-font` allow alternate fonts to be selected for the script and scriptscript sizes, and `script-features` and `sscript-features` to apply different OpenType features to them.

By default `script-features` is defined as `Style=MathScript` and `sscript-features` is `Style=MathScriptScript`. These correspond to the two levels of OpenType’s `ssty` feature tag. If the `(s)script-features` options are specified manually, you must additionally specify the `Style` options as above.

4.3 Maths ‘versions’

L^AT_EX uses a concept known as ‘maths versions’ to switch math fonts mid-document. This is useful because it is more efficient than loading a complete maths font from scratch every time—especially with thousands of glyphs in the case of Unicode maths! The canonical example for maths versions is to select a ‘bold’ maths font which might be suitable for section headings, say. (Not everyone agrees with this typesetting choice, though; be careful.)

To select a new maths font in a particular version, use the syntax

```
\setmathfont[version=(version name),<font features>]{<font name>}
```

and to switch between maths versions mid-document use the standard L^AT_EX command `\mathversion{<version name>}`.

5 Maths input

X_YL^AT_EX’s Unicode support allows maths input through two methods. Like classical T_EX, macros such as `\alpha`, `\sum`, `\pm`, `\leq`, and so on, provide verbose access to the entire repertoire of characters defined by Unicode. The literal characters themselves may be used instead, for more readable input files.

5.1 Math ‘style’

Classically, T_EX uses italic lowercase Greek letters and *upright* uppercase Greek letters for variables in mathematics. This is contrary to the ISO standards of using italic forms for both upper- and lowercase. Furthermore, the French have been known to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

Table 3: Effects of the `math-style` package option.

Package option	Example	
	Latin	Greek
<code>math-style=ISO</code>	(a, z, B, X)	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=TeX</code>	(a, z, B, X)	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=french</code>	(a, z, B, X)	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=upright</code>	(a, z, B, X)	$(\alpha, \beta, \Gamma, \Xi)$

The `unicode-math` package accommodates these possibilities with an interface heavily inspired by Walter Schmidt’s `lucimatx` package: a package option `math-style` that takes one of four arguments: `TeX`, `ISO`, `french`, or `upright` (case sensitive).

The philosophy behind the interface to the mathematical alphabet symbols lies in \LaTeX ’s attempt of separating content and formatting. Because input source text may come from a variety of places, the `upright` and ‘mathematical’ italic Latin and Greek alphabets are *unified* from the point of view of having a specified meaning in the source text. That is, to get a mathematical ‘ x ’, either the `ascii` (‘keyboard’) letter `x` may be typed, or the actual Unicode character may be used. Similarly for Greek letters. The `upright` or italic forms are then chosen based on the `math-style` package option.

If glyphs are desired that do not map as per the package option (for example, an upright ‘ g ’ is desired but typing `g` yields ‘ g ’), `markup` is required to specify this; to follow from the example: `\mathup{g}`. Maths alphabets commands such as `\mathup` are detailed later.

Alternative interface However, some users may not like this convention of normalising their input. For them, an upright `x` is an upright ‘ x ’ and that’s that. (This will be the case when obtaining source text from copy/pasting PDF or Microsoft Word documents, for example.) For these users, the `literal` option to `math-style` will effect this behaviour.

The `math-style` options’ effects are shown in brief in table 3.

5.2 Bold style

Similar as in the previous section, ISO standards differ somewhat to \TeX ’s conventions (and classical typesetting) for ‘boldness’ in mathematics. In the past, it has been customary to use bold *upright* letters to denote things like vectors and matrices. For example, $\mathbf{M} = (M_x, M_y, M_z)$. Presumably, this was due to the relatively scarcity of bold italic fonts in the pre-digital typesetting era. It has been suggested that *italic* bold symbols are used nowadays instead.

Bold Greek letters have simply been bold variant glyphs of their regular weight, as in $\xi = (\xi_r, \xi_\phi, \xi_\theta)$. Confusingly, the syntax in \LaTeX has been different for these two examples: `\mathbf` in the former (‘ \mathbf{M} ’), and `\bm` (or `\boldsymbol`, deprecated) in the latter (‘ ξ ’).

Table 4: Effects of the `bold-style` package option.

Package option	Example	
	Latin	Greek
<code>bold-style=ISO</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$
<code>bold-style=TeX</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$
<code>bold-style=upright</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$

In `unicode-math`, the `\mathbf` command works directly with both Greek and Latin maths alphabet characters and depending on package option either switches to upright for Latin letters (`bold-style=TeX`) as well or keeps them italic (`bold-style=ISO`).

To match the package options for non-bold characters, for `bold-style=upright` all bold characters are upright, and `bold-style=literal` does not change the upright/italic shape of the letter.

Upright and italic bold mathematical letters input as direct Unicode characters are normalised with the same rules. For example, with `bold-style=TeX`, a literal bold italic latin character will be typeset upright.

Note that `bold-style` is independent of `math-style`, although if the former is not specified then sensible defaults are chosen based on the latter.

The `bold-style` options' effects are shown in brief in table 4.

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical alphabet characters. These may be explicitly selected with the `\mathsfup`, `\mathsf`, `\mathbfsfup`, and `\mathbfsf` commands discussed in section 5.4.

How should the generic `\mathsf` behave? Unlike bold, sans serif is used much more sparingly in mathematics. I've seen recommendations to typeset tensors in sans serif italic or sans serif italic bold (e.g., examples in the `isomath` and `mattens` packages). But \LaTeX 's `\mathsf` is *upright* sans serif.

Therefore I reluctantly add the package options [`sans-style=upright`] and [`sans-style=italic`] to control the behaviour of `\mathsf`. The `upright` style sets up the command to use upright sans serif, including Greek; the `italic` style switches to using italic in both Latin and Greek alphabets. In other words, this option simply changes the meaning of `\mathsf` to either `\mathsfup` or `\mathsf`, respectively. Please let me know if more granular control is necessary here.

There is also a [`sans-style=literal`] setting, set automatically with [`math-style=literal`], which retains the uprightness of the input characters used when selecting the sans serif output.

5.3.1 What about bold sans serif?

While you might want your bold upright and your sans serif italic, I don't believe you'd also want your bold sans serif upright (or all vice versa, if that's even con-

Table 5: Mathematical alphabets defined in Unicode. Black dots indicate an alphabet exists in the font specified; blue dots indicate shapes that should always be taken from the upright font even in the italic style. See main text for description of `\mathbbit`.

Font				Alphabet		
Style	Shape	Series	Switch	Latin	Greek	Numerals
Serif	Upright	Normal	<code>\mathup</code>	•	•	•
		Bold	<code>\mathbfup</code>	•	•	•
	Italic	Normal	<code>\mathit</code>	•	•	•
		Bold	<code>\mathbfit</code>	•	•	•
Sans serif	Upright	Normal	<code>\mathsfup</code>	•		•
	Italic	Normal	<code>\mathsfit</code>	•		•
	Upright	Bold	<code>\mathbfsfup</code>	•	•	•
	Italic	Bold	<code>\mathbfsfit</code>	•	•	•
Typewriter	Upright	Normal	<code>\mathtt</code>	•		•
Double-struck	Upright	Normal	<code>\mathbb</code>	•		•
	Italic	Normal	<code>\mathbbit</code>	•		
Script	Upright	Normal	<code>\mathscr</code>	•		
		Bold	<code>\mathbfscr</code>	•		
Fraktur	Upright	Normal	<code>\mathfrak</code>	•		
		Bold	<code>\mathbffrac</code>	•		

ceivable). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

In other words, `\mathbfsf` is `\mathbfsfup` or `\mathbfsfit` based on `[sans-style=upright]` or `[sans-style=italic]`, respectively. And `[sans-style=literal]` causes `\mathbfsf` to retain the same italic or upright shape as the input, and turns it bold sans serif.

Note well! There is no medium-weight sans serif Greek alphabet in Unicode; therefore, `\mathsf{\alpha}` does not make sense (simply produces ‘ α ’) while `\mathbfsf{\alpha}` gives ‘ α ’.

5.4 All (the rest) of the mathematical alphabets

Unicode contains separate codepoints for most if not all variations of alphabet shape one may wish to use in mathematical notation. The complete list is shown in table 5. Some of these have been covered in the previous sections.

At present, the math font switching commands do not nest; therefore if you want sans serif bold, you must write `\mathsfbf{...}` rather than `\mathbf{\mathsf{...}}`. This may change in the future.

5.4.1 Double-struck

The double-struck alphabet (also known as ‘blackboard bold’) consists of upright Latin letters $\{\mathbb{a}-\mathbb{z}, \mathbb{A}-\mathbb{Z}\}$, numerals $\mathbb{0}-\mathbb{9}$, summation symbol $\mathbb{\Sigma}$, and four Greek letters only: $\{\mathbb{\gamma}, \mathbb{\Gamma}, \mathbb{\Pi}, \mathbb{\Omega}\}$.

While `\mathbb{\sum}` does produce a double-struck summation symbol, its limits aren’t properly aligned. Therefore, either the literal character or the control sequence `\Bbbsum` are recommended instead.

There are also five Latin *italic* double-struck letters: $\mathbb{D}, \mathbb{d}, \mathbb{E}, \mathbb{e}, \mathbb{J}$. These can be accessed (if not with their literal characters or control sequences) with the `\mathbb{it}` alphabet switch, but note that only those five letters will give the expected output.

5.4.2 Caligraphic vs. Script variants

The Unicode maths encoding contains an alphabet style for ‘Script’ letters, and while by default `\mathcal` and `\mathscr` are synonyms, there are some situations when a separate ‘Caligraphic’ style is needed as well.

If a font contains alternate glyphs for a separate caligraphic style, they can be selected explicitly as shown below. This feature is currently only supported by the XITS Math font, where the caligraphic letters are accessed with the same glyph slots as the script letters but with the first stylistic set feature (`ss01`) applied.

```
\setmathfont[range={\mathcal,\mathbfcal},StylisticSet=1]{XITS Math}
```

An example is shown below.

The Script style (`\mathscr`) in XITS Math is: $\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{X}\mathscr{Y}\mathscr{Z}$

The Caligraphic style (`\mathcal`) in XITS Math is: $\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{X}\mathcal{Y}\mathcal{Z}$

5.5 Miscellanea

5.5.1 Nabla

The symbol ∇ comes in the six forms shown in table 6. We want an individual option to specify whether we want upright or italic nabla by default (when either upright or italic nabla is used in the source). $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ classically uses an upright nabla, and ISO standards agree with this convention. The package options `nabla=upright` and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. This is then inherited through `\mathbf{\nabla}`; `\mathit{\nabla}` and `\mathup{\nabla}` can be used to force one way or the other.

`nabla=italic` is the default. `nabla=literal` is activated automatically after `math-style=literal`.

5.5.2 Partial

The same applies to the symbols ∂ partial differential and ∂ italic partial differential.

Table 6: The various forms of nabla.

Description		Glyph
Upright	Serif	∇
	Bold serif	∇
	Bold sans	∇
Italic	Serif	<i>∇</i>
	Bold serif	<i>∇</i>
	Bold sans	<i>∇</i>

Table 7: The various forms of the partial differential. Note that in the fonts used to display these glyphs, the first upright partial is incorrectly shown in an italic style.

Description		Glyph
Regular	Upright	∂
	Italic	<i>∂</i>
Bold	Upright	∂
	Italic	<i>∂</i>
Sans bold	Upright	∂
	Italic	<i>∂</i>

At time of writing, both the Cambria Math and STIX fonts display these two glyphs in the same italic style, but this is hopefully a bug that will be corrected in the future — the ‘plain’ partial differential should really have an upright shape.

Use the `partial=upright` or `partial=italic` package options to specify which one you would like, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests and argues otherwise) `partial=italic`.¹ `partial=literal` is activated following `math-style=literal`.

See table 7 for the variations on the partial differential symbol.

5.5.3 Epsilon and phi: ϵ vs. ε and ϕ vs. φ

\TeX defines `\epsilon` to look like ϵ and `\varepsilon` to look like ε . By contrast, the Unicode glyph directly after delta and before zeta is ‘epsilon’ and looks like ε ; there is a subsequent variant of epsilon that looks like ϵ . This creates a problem. People who use Unicode input won’t want their glyphs transforming; \TeX users will be confused that what they think as ‘normal epsilon’ is actual the ‘variant epsilon’. And the same problem exists for ‘phi’.

We have a package option to control this behaviour. With `vargreek-shape=TeX`, `\phi` and `\epsilon` produce ϕ and ϵ and `\varphi` and `\varepsilon` produce φ and ε .

¹A good argument would revolve around some international standards body recommending upright over italic. I just don’t have the time right now to look it up.

and ε . With `vargreek-shape=unicode`, these symbols are swapped. Note, however, that Unicode characters are not affected by this option. That is, no remapping occurs of the characters/glyphs, only the control sequences.

The package default is to use `vargreek-shape=TeX`.

5.5.4 Primes

Primes (x') may be input in several ways. You may use any combination the ASCII straight quote (') or the Unicode prime U+2032 ('); when multiple primes occur next to each other, they chain together to form double, triple, or quadruple primes if the font contains pre-drawn glyphs. The individual prime glyphs are accessed, as usual, with the `\prime` command, and the double-, triple-, and quadruple-prime glyphs are available with `\dprime`, `\trprime`, and `\qprime`, respectively.

If the font does not contain the pre-drawn glyphs or more than four primes are used, the single prime glyph is used multiple times with a negative kern to get the spacing right. There is no user interface to adjust this negative kern yet (because I haven't decided what it should look like); if you need to, write something like this:

```
\ExplSyntaxOn
\muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }
\ExplSyntaxOff
```

Backwards or reverse primes behave in exactly the same way; use the ASCII back tick (`) or the Unicode reverse prime U+2035 ('). The command to access the backprime is `\backprime`, and multiple backwards primes can be accessed with `\backdprime`, `\backtrprime`, and `\backqprime`.

In all cases above, no error checking is performed if you attempt to access a multi-prime glyph in a font that doesn't contain one. For this reason, it may be safer to write `x''''` instead of `x\qprime` in general.

If you ever need to enter the straight quote ' or the backtick ` in maths mode, these glyphs can be accessed with `\mathstraightquote` and `\mathbacktick`.

5.5.5 Unicode subscripts and superscripts

You may, if you wish, use Unicode subscripts and superscripts in your source document. For basic expressions, the use of these characters can make the input more readable. Adjacent sub- or super-scripts will be concatenated into a single expression.

The range of subscripts and superscripts supported by this package are shown in figures 1 and 2. Please request more if you think it is appropriate.

5.5.6 Colon

The colon is one of the few confusing characters of Unicode maths. In \TeX , `:` is defined as a colon with relation spacing: ' $a : b$ '. While `\colon` is defined as a colon with punctuation spacing: ' $a:b$ '.

A 0 1 2 3 4 5 6 7 8 9 + - = () i n Z

Figure 1: The Unicode superscripts supported as input characters. These are the literal glyphs from Charis SIL, not the output seen when used for maths input. The ‘A’ and ‘Z’ are to provide context for the size and location of the superscript glyphs.

A 0 1 2 3 4 5 6 7 8 9 + - = () a e i o r u v x β γ ρ ϕ χ Z

Figure 2: The Unicode subscripts supported as input characters. See note from figure 1.

In Unicode, U+003A colon is defined as a punctuation symbol, while U+2236 ratio is the colon-like symbol used in mathematics to denote ratios and other things.

This breaks the usual straightforward mapping from control sequence to Unicode input character to (the same) Unicode glyph.

To preserve input compatibility, we remap the ASCII input character ‘:’ to U+2236. Typing a literal U+2236 char will result in the same output. If `amsmath` is loaded, then the definition of `\colon` is inherited from there (it looks like a punctuation colon with additional space around it). Otherwise, `\colon` is made to output a colon with `\mathpunct` spacing.

The package option `colon=literal` forces ASCII input ‘:’ to be printed as `\mathcolon` instead.

5.5.7 Slashes and backslashes

There are several slash-like symbols defined in Unicode. The complete list is shown in table 8.

In regular \LaTeX we can write `\left\slash...\right\backslash` and so on and obtain extensible delimiter-like symbols. Not all of the Unicode slashes are suitable for this (and do not have the font support to do it).

Table 8: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	<code>\slash</code>
U+2044	FRACTION SLASH	/	<code>\fracslash</code>
U+2215	DIVISION SLASH	/	<code>\divslash</code>
U+29F8	BIG SOLIDUS	/	<code>\xsol</code>
U+005C	REVERSE SOLIDUS	\	<code>\backslash</code>
U+2216	SET MINUS	\	<code>\smallsetminus</code>
U+29F5	REVERSE SOLIDUS OPERATOR	\	<code>\setminus</code>
U+29F9	BIG REVERSE SOLIDUS	\	<code>\xbsol</code>

Slash Of U+2044 fraction slash, TR25 says that it is:

...used to build up simple fractions in running text...however parsers of mathematical texts should be prepared to handle fraction slash when it is received from other sources.

U+2215 division slash should be used when division is represented without a built-up fraction; $\pi \approx 22/7$, for example.

U+29F8 big solidus is a ‘big operator’ (like Σ).

Backslash The U+005C reverse solidus character `\backslash` is used for denoting double cosets: $A \backslash B$. (So I’m led to believe.) It may be used as a ‘stretchy’ delimiter if supported by the font.

MathML uses U+2216 set minus like this: $A \setminus B$.² The \LaTeX command name `\smallsetminus` is used for backwards compatibility.

Presumably, U+29F5 reverse solidus operator is intended to be used in a similar way, but it could also (perhaps?) be used to represent ‘inverse division’: $\pi \approx 7 \setminus 22$.³ The \LaTeX name for this character is `\setminus`.

Finally, U+29F9 big reverse solidus is a ‘big operator’ (like Σ).

How to use all of these things Unfortunately, font support for the above characters/glyphs is rather inconsistent. In Cambria Math, the only slash that grows (say when writing

$$\left[\begin{array}{cc} a & b \\ c & d \end{array} \right] \bigg/ \left[\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right])$$

is the FRACTION SLASH, which we just established above is sort of only supposed to be used in text.

Of the above characters, the following are allowed to be used after `\left`, `\middle`, and `\right`:

- `\solidus`;
- `\fracslash`;
- `\slash`; and,
- `\backslash` (the only reverse slash).

However, we assume that there is only *one* stretchy slash in the font; this is assumed by default to be U+002F solidus. Writing `\left/` or `\left\slash` or `\leftfracslash` will all result in the same stretchy delimiter being used.

The delimiter used can be changed with the `slash-delimiter` package option. Allowed values are `ascii`, `frac`, and `div`, corresponding to the respective Unicode slots.

²§4.4.5.11 <http://www.w3.org/TR/MathML3/>

³This is valid syntax in the Octave and Matlab programming languages, in which it means matrix inverse pre-multiplication. I.e., $A \setminus B \equiv A^{-1}B$.

For example: as mentioned above, Cambria Math’s stretchy slash is `U+2044` fraction slash. When using Cambria Math, then `unicode-math` should be loaded with the `slash-delimiter=frac` option. (This should be a font option rather than a package option, but it will change soon.)

5.5.8 Growing and non-growing accents

There are a few accents for which \TeX has both non-growing and growing versions. Among these are `\hat` and `\tilde`; the corresponding growing versions are called `\widehat` and `\widetilde`, respectively.

Older versions of \XTeX and $\text{Lua}\TeX$ did not support this distinction, however, and *all* accents there were growing automatically. (I.e., `\hat` and `\widehat` are equivalent.) As of $\text{Lua}\TeX$ v0.65 and \XTeX v0.9998, these wide/non-wide commands will again behave in their expected manner.

5.5.9 Pre-drawn fraction characters

Pre-drawn fractions `U+00BC–U+00BE`, `U+2150–U+215E` are not suitable for use in mathematics output. However, they can be useful as input characters to abbreviate common fractions.

$\frac{1}{4}$ $\frac{1}{2}$ $\frac{3}{4}$ $\frac{0}{3}$ $\frac{1}{7}$ $\frac{1}{9}$ $\frac{1}{10}$ $\frac{1}{3}$ $\frac{2}{3}$ $\frac{1}{5}$ $\frac{2}{5}$ $\frac{3}{5}$ $\frac{4}{5}$ $\frac{1}{6}$ $\frac{5}{6}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{5}{8}$ $\frac{7}{8}$

For example, instead of writing `\tfrac{12}{x}`, you may consider it more readable to have `\%x` in the source instead.

If the `\tfrac` command exists (i.e., if `amsmath` is loaded or you have specially defined `\tfrac` for this purpose), it will be used to typeset the fractions. If not, regular `\frac` will be used. The command to use (`\tfrac` or `\frac`) can be forced either way with the package option `active-frac=small` or `active-frac=normalsize`, respectively.

5.5.10 Circles

Unicode defines a large number of different types of circles for a variety of mathematical purposes. There are thirteen alone just considering the all white and all black ones, shown in table 9.

\LaTeX defines considerably fewer: `\circ` and `csbigcirc` for white; `\bullet` for black. This package maps those commands to `\vysmwhtcircle`, `\mdlgwhtcircle`, and `\smbllkcircle`, respectively.

5.5.11 Triangles

While there aren’t as many different sizes of triangle as there are circle, there’s some important distinctions to make between a few similar characters. See table 10 for the full summary.

These triangles all have different intended meanings. Note for backwards compatibility with \TeX , `U+25B3` has *two* different mappings in `unicode-math`. `\bigtriangleup` is intended as a binary operator whereas `\triangle` is intended to be used as a letter-like symbol.

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdotp</code>	.			
U+22C5	<code>\cdot</code>	.			
U+2219	<code>\vysmblkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smbllkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbllkcircle</code>	●	◦	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdblkcircle</code>	●	◯	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgbllkcircle</code>	●	◯	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	◯	<code>\lgwhtcircle</code>	U+25EF

Table 9: Filled and hollow Unicode circles.

Slot	Command	Glyph	Class
U+25B5	<code>\vartriangle</code>	△	binary
U+25B3	<code>\bigtriangleup</code>	△	binary
U+25B3	<code>\triangle</code>	△	ordinary
U+2206	<code>\increment</code>	Δ	ordinary
U+0394	<code>\mathup\Delta</code>	Δ	ordinary

Table 10: Different upwards pointing triangles.

But you're better off if you're using the latter form to indicate an increment to use the glyph intended for this purpose, U+2206: Δx .

Finally, given that \triangle and Δ are provided for you already, it is better off to only use upright Greek Delta Δ if you're actually using it as a symbolic entity such as a variable on its own.

6 Advanced

6.1 Warning messages

This package can produce a number of informational messages to try and inform the user when something might be going wrong due to package conflicts or something else. As an experimental feature, these can be turned off on an individual basis with the package option `warnings-off` which takes a comma-separated list of warnings to suppress. A warning will give you its name when printed on the console output; e.g.,

```
* unicode-math warning: "mathtools-colon"
*
* ... <warning message> ...
```

This warning could be suppressed by loading the package as follows:

```
\usepackage[warnings-off={mathtools-colon}]{unicode-math}
```


6.2 Programmer's interface

(Tentative and under construction.) If you are writing some code that needs to know the current maths style (`\mathbf`, `\mathit`, etc.), you can query the variable `\l_um_mathstyle_t1`. It will contain the maths style without the leading 'math' string; for example, `\mathbf { \show \l_um_mathstyle_t1 }` will produce 'bf'.

Part II

Package implementation

Table of Contents

7	Header code	19
7.1	Extras	21
7.2	Function variants	21
7.3	Package options	21
8	Lua\LaTeX module	26
9	Bifurcation	27
9.1	Engine differences	27
9.2	Alphabet Unicode positions	28
9.3	STIX fonts	33
9.4	Overcoming <code>\@onlypreamble</code>	37
10	Fundamentals	38
10.1	Enlarging the number of maths families	38
10.2	Setting math chars, math codes, etc.	38
10.3	The main <code>\setmathfont</code> macro	41
10.4	(Big) operators	48
10.5	Radicals	48
10.6	Maths accents	49
10.7	Common interface for font parameters	49
11	Font features	53
11.1	Math version	54
11.2	Script and scriptscript font options	54
11.3	Range processing	54
11.4	Resolving Greek symbol name control sequences	58
12	Maths alphabets mapping definitions	59
12.1	Initialising math styles	59
12.2	Defining the math style macros	60
12.3	Defining the math alphabets per style	61
12.4	Mapping ‘naked’ math characters	63
12.5	Mapping chars inside a math style	65
12.6	Alphabets	67
13	A token list to contain the data of the math table	79
14	Definitions of the active math characters	80
15	Fall-back font	81
16	Epilogue	82

16.1 Primes	82
16.2 Unicode radicals	87
16.3 Unicode sub- and super-scripts	89
16.4 Synonyms and all the rest	93
16.5 Compatibility	94
17 Error messages	104
18 stix table data extraction	105
A Documenting maths support in the NFSS	106
B Legacy T_EX font dimensions	107
C X_YT_EX math font dimensions	107

7 Header code

We (later on) bifurcate the package based on the engine being used.

```

1 (*load)
2 \luatex_if_engine:T { \RequirePackage{unicode-math-luatex} \endinput }
3 \xetex_if_engine:T { \RequirePackage{unicode-math-xetex} \endinput }
4 (/load)

```

The shared part of the code starts here before the split above.

```

5 (*preamble&!XE&!LU)
6 \usepackage{ifxetex,ifluatex}
7 \ifxetex
8   \ifdim\number\XeTeXversion\XeTeXrevision in<0.9998in%
9     \PackageError{unicode-math}{%
10       Cannot run with this version of XeTeX!\MessageBreak
11       You need XeTeX 0.9998 or newer.%
12     }\@ehd
13   \fi
14 \else\ifluatex
15   \ifnum\luatexversion<64%
16     \PackageError{unicode-math}{%
17       Cannot run with this version of LuaTeX!\MessageBreak
18       You need LuaTeX 0.64 or newer.%
19     }\@ehd
20   \fi
21 \else
22   \PackageError{unicode-math}{%
23     Cannot be run with pdfLaTeX!\MessageBreak
24     Use XeLaTeX or LuaLaTeX instead.%
25   }\@ehd
26 \fi\fi

```

Packages

```
27 \RequirePackage{expl3}[2011/07/01]
28 \RequirePackage{xparse}[2009/08/31]
29 \RequirePackage{l3keys2e}
30 \RequirePackage{fontspec}[2010/10/25]
31 \RequirePackage{catchfile}
32 \RequirePackage{fix-cm} % avoid some warnings
33 \RequirePackage{filehook}[2011/01/03]
```

Start using L^AT_EX3 — finally!

```
34 \ExplSyntaxOn
```

Extra expl3 variants

```
35 \cs_generate_variant:Nn \tl_put_right:Nn {cx}
36 \cs_generate_variant:Nn \seq_if_in:NnTF {NV}
37 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
38 \cs_generate_variant:Nn \prop_get:NnN {cxN}
39 \cs_generate_variant:Nn \prop_if_in:NnTF {cx}
```

Extra expansion command:

```
40 \cs_set:Npn \exp_args:NNcc #1#2#3#4 {
41   \exp_after:wN #1 \exp_after:wN #2
42   \cs:w #3 \exp_after:wN \cs_end:
43   \cs:w #4 \cs_end:
44 }
```

Conditionals

```
45 \bool_new:N \l_um_ot_math_bool
46 \bool_new:N \l_um_init_bool
47 \bool_new:N \l_um_implicit_alph_bool
48 \bool_new:N \g_um_mainfont_already_set_bool
```

For math-style:

```
49 \bool_new:N \g_um_literal_bool
50 \bool_new:N \g_um_upLatin_bool
51 \bool_new:N \g_um_uplatin_bool
52 \bool_new:N \g_um_upGreek_bool
53 \bool_new:N \g_um_upgreek_bool
```

For bold-style:

```
54 \bool_new:N \g_um_bfliteral_bool
55 \bool_new:N \g_um_bfupLatin_bool
56 \bool_new:N \g_um_bfuplatin_bool
57 \bool_new:N \g_um_bfupGreek_bool
58 \bool_new:N \g_um_bfupgreek_bool
```

For sans-style:

```
59 \bool_new:N \g_um_upsans_bool
60 \bool_new:N \g_um_sfliteral_bool
```

For assorted package options:

```

61 \bool_new:N \g_um_upNabla_bool
62 \bool_new:N \g_um_uppartial_bool
63 \bool_new:N \g_um_literal_Nabla_bool
64 \bool_new:N \g_um_literal_partial_bool
65 \bool_new:N \g_um_texgreek_bool
66 \bool_set_true:N \g_um_texgreek_bool
67 \bool_new:N \l_um_smallfrac_bool
68 \bool_new:N \g_um_literal_colon_bool

```

Variables

```

69 \int_new:N \g_um_fam_int

70 \tl_const:Nn \c_um_math_alphabet_name_latin_tl {Latin,~lowercase}
71 \tl_const:Nn \c_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
72 \tl_const:Nn \c_um_math_alphabet_name_greek_tl {Greek,~lowercase}
73 \tl_const:Nn \c_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
74 \tl_const:Nn \c_um_math_alphabet_name_num_tl {Numerals}
75 \tl_const:Nn \c_um_math_alphabet_name_misc_tl {Misc.}

```

7.1 Extras

`\um_glyph_if_exist:nTF` : TODO: Generalise for arbitrary fonts! `\l_um_font` is not always the one used for a specific glyph!!

```

76 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
77   \iffontchar \l_um_font #1 \scan_stop:
78   \prg_return_true:
79   \else:
80   \prg_return_false:
81   \fi:
82 }
83 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
84 \cs_generate_variant:Nn \um_glyph_if_exist_nTF {c}
85 \cs_generate_variant:Nn \um_glyph_if_exist_nT {c}
86 \cs_generate_variant:Nn \um_glyph_if_exist_nF {c}

```

7.2 Function variants

```

87 \cs_generate_variant:Nn \fontspec_set_family:Nnn {Nx}
88 \cs_generate_variant:Nn \fontspec_set_fontface:NNnn {NNx}

```

7.3 Package options

`\unimathsetup` This macro can be used in lieu of or later to override options declared when the package is loaded.

```

89 \DeclareDocumentCommand \unimathsetup {m} {
90   \keys_set:nn {unicode-math} {#1}
91 }

```

$$\mathbf{math-style}$$

```

92 \keys_define:nn {unicode-math} {
93   normal-style .choice_code:n =
94   {
95     \bool_set_false:N \g_um_literal_bool
96     \ifcase \l_keys_choice_int
97       \bool_set_false:N \g_um_upGreek_bool
98       \bool_set_false:N \g_um_upgreek_bool
99       \bool_set_false:N \g_um_upLatin_bool
100      \bool_set_false:N \g_um_uplatin_bool
101     \or
102       \bool_set_true:N \g_um_upGreek_bool
103       \bool_set_false:N \g_um_upgreek_bool
104       \bool_set_false:N \g_um_upLatin_bool
105       \bool_set_false:N \g_um_uplatin_bool
106     \or
107       \bool_set_true:N \g_um_upGreek_bool
108       \bool_set_true:N \g_um_upgreek_bool
109       \bool_set_true:N \g_um_upLatin_bool
110       \bool_set_false:N \g_um_uplatin_bool
111     \or
112       \bool_set_true:N \g_um_upGreek_bool
113       \bool_set_true:N \g_um_upgreek_bool
114       \bool_set_true:N \g_um_upLatin_bool
115       \bool_set_true:N \g_um_uplatin_bool
116     \or
117       \bool_set_true:N \g_um_literal_bool
118     \fi
119   } ,
120   normal-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
121 }
122 \keys_define:nn {unicode-math} {
123   math-style .choice_code:n =
124   {
125     \ifcase \l_keys_choice_int
126       \unimathsetup {
127         normal-style=ISO,
128         bold-style=ISO,
129         sans-style=italic,
130         nabla=upright,
131         partial=italic,
132       }
133     \or
134       \unimathsetup {
135         normal-style=TeX,
136         bold-style=TeX,
137         sans-style=upright,
138         nabla=upright,
139         partial=italic,
140       }

```

```

141 \or
142 \unimathsetup {
143     normal-style=french,
144     bold-style=upright,
145     sans-style=upright,
146     nabla=upright,
147     partial=upright,
148 }
149 \or
150 \unimathsetup {
151     normal-style=upright,
152     bold-style=upright,
153     sans-style=upright,
154     nabla=upright,
155     partial=upright,
156 }
157 \or
158 \unimathsetup {
159     normal-style=literal,
160     bold-style=literal,
161     sans-style=literal,
162     colon=literal,
163     nabla=literal,
164     partial=literal,
165 }
166 \fi
167 } ,
168 math-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
169 }

```

bold-style

```

170 \keys_define:nn {unicode-math} {
171     bold-style .choice_code:n = {
172         \bool_set_false:N \g_um_bfliteral_bool
173         \ifcase \l_keys_choice_int
174             \bool_set_false:N \g_um_bfupgreek_bool
175             \bool_set_false:N \g_um_bfupgreek_bool
176             \bool_set_false:N \g_um_bfuplatin_bool
177             \bool_set_false:N \g_um_bfuplatin_bool
178         \or
179             \bool_set_true:N \g_um_bfupgreek_bool
180             \bool_set_false:N \g_um_bfupgreek_bool
181             \bool_set_true:N \g_um_bfuplatin_bool
182             \bool_set_true:N \g_um_bfuplatin_bool
183         \or
184             \bool_set_true:N \g_um_bfupgreek_bool
185             \bool_set_true:N \g_um_bfupgreek_bool
186             \bool_set_true:N \g_um_bfuplatin_bool
187             \bool_set_true:N \g_um_bfuplatin_bool
188         \or

```

```

189     \bool_set_true:N \g_um_bfliteral_bool
190   \fi
191 } ,
192 bold-style .generate_choices:n = {ISO,TeX,upright,literal} ,
193 }

```

sans-style

```

194 \keys_define:nn {unicode-math} {
195   sans-style .choice_code:n = {
196     \ifcase \l_keys_choice_int
197       \bool_set_false:N \g_um_upsans_bool
198     \or
199       \bool_set_true:N \g_um_upsans_bool
200     \or
201       \bool_set_true:N \g_um_sfliteral_bool
202     \fi
203   } ,
204   sans-style .generate_choices:n = {italic,upright,literal} ,
205 }

```

Nabla and partial

```

206 \keys_define:nn {unicode-math} {
207   nabla .choice_code:n = {
208     \bool_set_false:N \g_um_literal_Nabla_bool
209     \ifcase \l_keys_choice_int
210       \bool_set_true:N \g_um_upNabla_bool
211     \or
212       \bool_set_false:N \g_um_upNabla_bool
213     \or
214       \bool_set_true:N \g_um_literal_Nabla_bool
215     \fi
216   } ,
217   nabla .generate_choices:n = {upright,italic,literal} ,
218 }

219 \keys_define:nn {unicode-math} {
220   partial .choice_code:n = {
221     \bool_set_false:N \g_um_literal_partial_bool
222     \ifcase \l_keys_choice_int
223       \bool_set_true:N \g_um_uppartial_bool
224     \or
225       \bool_set_false:N \g_um_uppartial_bool
226     \or
227       \bool_set_true:N \g_um_literal_partial_bool
228     \fi
229   } ,
230   partial .generate_choices:n = {upright,italic,literal} ,
231 }

```


Epsilon and phi shapes

```
232 \keys_define:nn {unicode-math} {  
233   vargreek-shape .choice: ,  
234   vargreek-shape / unicode .code:n = {  
235     \bool_set_false:N \g_um_texgreek_bool  
236   } ,  
237   vargreek-shape / TeX .code:n = {  
238     \bool_set_true:N \g_um_texgreek_bool  
239   }  
240 }
```

Colon style

```
241 \keys_define:nn {unicode-math} {  
242   colon .choice: ,  
243   colon / literal .code:n = {  
244     \bool_set_true:N \g_um_literal_colon_bool  
245   } ,  
246   colon / TeX .code:n = {  
247     \bool_set_false:N \g_um_literal_colon_bool  
248   }  
249 }
```

Slash delimiter style

```
250 \keys_define:nn {unicode-math} {  
251   slash-delimiter .choice: ,  
252   slash-delimiter / ascii .code:n = {  
253     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}  
254   } ,  
255   slash-delimiter / frac .code:n = {  
256     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}  
257   } ,  
258   slash-delimiter / div .code:n = {  
259     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}  
260   }  
261 }
```

Active fraction style

```
262 \keys_define:nn {unicode-math} {  
263   active-frac .choice: ,  
264   active-frac / small .code:n = {  
265     \cs_if_exist:NTF \tfrac {  
266       \bool_set_true:N \l_um_smallfrac_bool  
267     } {  
268       \um_warning:n {no-tfrac}  
269       \bool_set_false:N \l_um_smallfrac_bool  
270     }  
271     \use:c{um_setup_active_frac:}  
272   } ,  
273   active-frac / normalsize .code:n = {
```

```

274     \bool_set_false:N \l_um_smallfrac_bool
275     \use:c{um_setup_active_frac:}
276   }
277 }

```

Debug/tracing

```

278 \keys_define:nn {unicode-math}
279 {
280   warnings-off .code:n =
281   {
282     \clist_map_inline:nn {#1}
283     { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
284   }
285 }
286 \keys_define:nn {unicode-math} {
287   trace .choice: ,
288   trace / debug .code:n = {
289     \msg_redirect_module:nnn { unicode-math } { log } { warning }
290   } ,
291   trace / on .code:n = {
292     % default
293   } ,
294   trace / off .code:n = {
295     \msg_redirect_module:nnn { unicode-math } { log } { none }
296   } ,
297 }
298 \unimathsetup {math-style=TeX}
299 \unimathsetup {slash-delimiter=ascii}
300 \unimathsetup {trace=off}
301 \cs_if_exist:NT \tfrac {
302   \unimathsetup {active-frac=small}
303 }
304 \ProcessKeysOptions {unicode-math}

```

8 Lua^AT_EX module

We create a luatexbase module that contains Lua functions for use with Lua^AT_EX.

```

305 </preamble&!XE&!LU>
306 {*lua}
307 local err, warn, info, log = luatexbase.provides_module({
308   name      = "unicode-math",
309   date      = "2012/04/23",
310   version   = 0.1,
311   description = "Unicode math typesetting for LuaLaTeX",
312   author    = "Khaled Hosny, Will Robertson, Philipp Stephani",
313   licence   = "LPPL v1.3+"
314 })

```

LuaTeX does not provide interface to accessing (Script)ScriptPercentScaleDown math constants, so we emulate XeTeX behaviour by setting `\fontdimen10` and `\fontdimen11`.

```

315 local function set_sscale_dims(fontdata)
316   local mc = fontdata.MathConstants
317   if mc then
318     fontdata.parameters[10] = mc.ScriptPercentScaleDown or 70
319     fontdata.parameters[11] = mc.ScriptScriptPercentScaleDown or 50
320   end
321 end
322 luatexbase.add_to_callback("luaotfload.patch_font", set_sscale_dims, "uni-
code_math.set_sscale_dims")

```

Cambria Math has too small DisplayOperatorMinHeight constant, so we patch it to amore accebttable value.

```

323 local function patch_cambria_domh(fontdata)
324   local mc = fontdata.MathConstants
325   local mh = 2800 / fontdata.units * fontdata.size
326   if fontdata.pname == "CambriaMath" and mc then
327     if mc.DisplayOperatorMinHeight < mh then
328       mc.DisplayOperatorMinHeight = mh
329     end
330   end
331 end
332 luatexbase.add_to_callback("luaotfload.patch_font", patch_cambria_domh, "cam-
bria.domh")
333 /lua

```

(Error messages and warning definitions go here from the msg chunk defined in section §17 on page 104.)

9 Bifurcation

And here the split begins. Most of the code is still shared, but code for LuaTeX uses the ‘LU’ prefix and code for XeTeX uses ‘XE’.

```

334 {*package & (XE j LU)}
335 \ExplSyntaxOn

```

9.1 Engine differences

```

336 \cs_new:Nn \um_cs_compat:n
337 (XE) { \cs_set_eq:cc {U#1} {XeTeX#1} }
338 (LU) { \cs_set_eq:cc {U#1} {LuTeX#1} }
339 \um_cs_compat:n {mathcode}
340 \um_cs_compat:n {delcode}
341 \um_cs_compat:n {mathcodenum}
342 \um_cs_compat:n {mathcharnum}
343 \um_cs_compat:n {mathchardef}
344 \um_cs_compat:n {radical}

```

```

345 \um_cs_compat:n {mathaccent}
346 \um_cs_compat:n {delimiter}

347 (*LU)
348 \RequirePackage { LuaLatex-math } [ 2011/08/07 ]
349 \RequirePackage { Luatexbase }
350 \RequirePackage { Luaotfload } [ 2010/11/26 ]
351 \RequireLuaModule { unicode-math } [ 2012/04/23 ]
352 (/LU)

```

9.2 Alphabet Unicode positions

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.⁴

Rather than 'readable', in the end, this makes the code more extensible.

```

353 \cs_new:Nn \usv_set:nnn {
354   \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}
355 }
356 \cs_new:Nn \um_to_usv:nn { g_um_#1_#2_usv }

```

Alphabets

```

357 \usv_set:nnn {up}{num}{48}
358 \usv_set:nnn {up}{Latin}{65}
359 \usv_set:nnn {up}{latin}{97}
360 \usv_set:nnn {up}{Greek}{391}
361 \usv_set:nnn {up}{greek}{381}
362 \usv_set:nnn {it}{Latin}{1D434}
363 \usv_set:nnn {it}{latin}{1D44E}
364 \usv_set:nnn {it}{Greek}{1D6E2}
365 \usv_set:nnn {it}{greek}{1D6FC}
366 \usv_set:nnn {bb}{num}{1D7D8}
367 \usv_set:nnn {bb}{Latin}{1D538}
368 \usv_set:nnn {bb}{latin}{1D552}
369 \usv_set:nnn {scr}{Latin}{1D49C}
370 \usv_set:nnn {cal}{Latin}{1D49C}
371 \usv_set:nnn {scr}{latin}{1D4B6}
372 \usv_set:nnn {frak}{Latin}{1D504}
373 \usv_set:nnn {frak}{latin}{1D51E}
374 \usv_set:nnn {sf}{num}{1D7E2}
375 \usv_set:nnn {sfup}{num}{1D7E2}
376 \usv_set:nnn {sfit}{num}{1D7E2}
377 \usv_set:nnn {sfup}{Latin}{1D5A0}
378 \usv_set:nnn {sf}{Latin}{1D5A0}
379 \usv_set:nnn {sfup}{latin}{1D5BA}
380 \usv_set:nnn {sf}{latin}{1D5BA}
381 \usv_set:nnn {sfit}{Latin}{1D608}
382 \usv_set:nnn {sfit}{latin}{1D622}
383 \usv_set:nnn {tt}{num}{1D7F6}
384 \usv_set:nnn {tt}{Latin}{1D670}
385 \usv_set:nnn {tt}{latin}{1D68A}

```

⁴'u.s.v.' stands for 'Unicode scalar value'.

Bold:

```
386 \usv_set:nnn {bf}{num}{ "1D7CE}
387 \usv_set:nnn {bfup}{num}{ "1D7CE}
388 \usv_set:nnn {bfit}{num}{ "1D7CE}
389 \usv_set:nnn {bfup}{Latin}{ "1D400}
390 \usv_set:nnn {bfup}{latin}{ "1D41A}
391 \usv_set:nnn {bfup}{Greek}{ "1D6A8}
392 \usv_set:nnn {bfup}{greek}{ "1D6C2}
393 \usv_set:nnn {bfit}{Latin}{ "1D468}
394 \usv_set:nnn {bfit}{latin}{ "1D482}
395 \usv_set:nnn {bfit}{Greek}{ "1D71C}
396 \usv_set:nnn {bfit}{greek}{ "1D736}
397 \usv_set:nnn {bffrak}{Latin}{ "1D56C}
398 \usv_set:nnn {bffrak}{latin}{ "1D586}
399 \usv_set:nnn {bfscr}{Latin}{ "1D4D0}
400 \usv_set:nnn {bfcal}{Latin}{ "1D4D0}
401 \usv_set:nnn {bfscr}{latin}{ "1D4EA}
402 \usv_set:nnn {bfsf}{num}{ "1D7EC}
403 \usv_set:nnn {bfsfup}{num}{ "1D7EC}
404 \usv_set:nnn {bfsfit}{num}{ "1D7EC}
405 \usv_set:nnn {bfsfup}{Latin}{ "1D5D4}
406 \usv_set:nnn {bfsfup}{latin}{ "1D5EE}
407 \usv_set:nnn {bfsfup}{Greek}{ "1D756}
408 \usv_set:nnn {bfsfup}{greek}{ "1D770}
409 \usv_set:nnn {bfsfit}{Latin}{ "1D63C}
410 \usv_set:nnn {bfsfit}{latin}{ "1D656}
411 \usv_set:nnn {bfsfit}{Greek}{ "1D790}
412 \usv_set:nnn {bfsfit}{greek}{ "1D7AA}

413 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsfi
414 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsfi
415 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsfi
416 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsfi
417 \usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_La
418 \usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_la
419 \usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_Gr
420 \usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_gr
```

Greek variants:

```
421 \usv_set:nnn {up}{varTheta}{ "3F4}
422 \usv_set:nnn {up}{Digamma}{ "3DC}
423 \usv_set:nnn {up}{varepsilon}{ "3F5}
424 \usv_set:nnn {up}{vartheta}{ "3D1}
425 \usv_set:nnn {up}{varkappa}{ "3F0}
426 \usv_set:nnn {up}{varphi}{ "3D5}
427 \usv_set:nnn {up}{varrho}{ "3F1}
428 \usv_set:nnn {up}{varpi}{ "3D6}
429 \usv_set:nnn {up}{digamma}{ "3DD}
```

Bold:

```
430 \usv_set:nnn {bfup}{varTheta}{ "1D6B9}
431 \usv_set:nnn {bfup}{Digamma}{ "1D7CA}
```

432 \usv_set:nnn {bfup}{varepsilon}{1D6DC}
 433 \usv_set:nnn {bfup}{vartheta}{1D6DD}
 434 \usv_set:nnn {bfup}{varkappa}{1D6DE}
 435 \usv_set:nnn {bfup}{varphi}{1D6DF}
 436 \usv_set:nnn {bfup}{varrho}{1D6E0}
 437 \usv_set:nnn {bfup}{varpi}{1D6E1}
 438 \usv_set:nnn {bfup}{digamma}{1D7CB}

Italic Greek variants:

439 \usv_set:nnn {it}{varTheta}{1D6F3}
 440 \usv_set:nnn {it}{varepsilon}{1D716}
 441 \usv_set:nnn {it}{vartheta}{1D717}
 442 \usv_set:nnn {it}{varkappa}{1D718}
 443 \usv_set:nnn {it}{varphi}{1D719}
 444 \usv_set:nnn {it}{varrho}{1D71A}
 445 \usv_set:nnn {it}{varpi}{1D71B}

Bold italic:

446 \usv_set:nnn {bfit}{varTheta}{1D72D}
 447 \usv_set:nnn {bfit}{varepsilon}{1D750}
 448 \usv_set:nnn {bfit}{vartheta}{1D751}
 449 \usv_set:nnn {bfit}{varkappa}{1D752}
 450 \usv_set:nnn {bfit}{varphi}{1D753}
 451 \usv_set:nnn {bfit}{varrho}{1D754}
 452 \usv_set:nnn {bfit}{varpi}{1D755}

Bold sans:

453 \usv_set:nnn {bfsfup}{varTheta}{1D767}
 454 \usv_set:nnn {bfsfup}{varepsilon}{1D78A}
 455 \usv_set:nnn {bfsfup}{vartheta}{1D78B}
 456 \usv_set:nnn {bfsfup}{varkappa}{1D78C}
 457 \usv_set:nnn {bfsfup}{varphi}{1D78D}
 458 \usv_set:nnn {bfsfup}{varrho}{1D78E}
 459 \usv_set:nnn {bfsfup}{varpi}{1D78F}

Bold sans italic:

460 \usv_set:nnn {bfsfit}{varTheta} {1D7A1}
 461 \usv_set:nnn {bfsfit}{varepsilon}{1D7C4}
 462 \usv_set:nnn {bfsfit}{vartheta} {1D7C5}
 463 \usv_set:nnn {bfsfit}{varkappa} {1D7C6}
 464 \usv_set:nnn {bfsfit}{varphi} {1D7C7}
 465 \usv_set:nnn {bfsfit}{varrho} {1D7C8}
 466 \usv_set:nnn {bfsfit}{varpi} {1D7C9}

Nabla:

467 \usv_set:nnn {up} {Nabla}{02207}
 468 \usv_set:nnn {it} {Nabla}{1D6FB}
 469 \usv_set:nnn {bfup} {Nabla}{1D6C1}
 470 \usv_set:nnn {bfit} {Nabla}{1D735}
 471 \usv_set:nnn {bfsfup}{Nabla}{1D76F}
 472 \usv_set:nnn {bfsfit}{Nabla}{1D7A9}

Partial:

```

473 \usv_set:nnn {up} {partial}{02202}
474 \usv_set:nnn {it} {partial}{1D715}
475 \usv_set:nnn {bfup} {partial}{1D6DB}
476 \usv_set:nnn {bfit} {partial}{1D74F}
477 \usv_set:nnn {bfsfup}{partial}{1D789}
478 \usv_set:nnn {bfsfit}{partial}{1D7C3}

```

Exceptions These are need for mapping with the exceptions in other alphabets:
(coming up)

```

479 \usv_set:nnn {up}{B}{\B}
480 \usv_set:nnn {up}{C}{\C}
481 \usv_set:nnn {up}{D}{\D}
482 \usv_set:nnn {up}{E}{\E}
483 \usv_set:nnn {up}{F}{\F}
484 \usv_set:nnn {up}{H}{\H}
485 \usv_set:nnn {up}{I}{\I}
486 \usv_set:nnn {up}{L}{\L}
487 \usv_set:nnn {up}{M}{\M}
488 \usv_set:nnn {up}{N}{\N}
489 \usv_set:nnn {up}{P}{\P}
490 \usv_set:nnn {up}{Q}{\Q}
491 \usv_set:nnn {up}{R}{\R}
492 \usv_set:nnn {up}{Z}{\Z}

493 \usv_set:nnn {it}{B}{1D435}
494 \usv_set:nnn {it}{C}{1D436}
495 \usv_set:nnn {it}{D}{1D437}
496 \usv_set:nnn {it}{E}{1D438}
497 \usv_set:nnn {it}{F}{1D439}
498 \usv_set:nnn {it}{H}{1D43B}
499 \usv_set:nnn {it}{I}{1D43C}
500 \usv_set:nnn {it}{L}{1D43F}
501 \usv_set:nnn {it}{M}{1D440}
502 \usv_set:nnn {it}{N}{1D441}
503 \usv_set:nnn {it}{P}{1D443}
504 \usv_set:nnn {it}{Q}{1D444}
505 \usv_set:nnn {it}{R}{1D445}
506 \usv_set:nnn {it}{Z}{1D44D}

507 \usv_set:nnn {up}{d}{\d}
508 \usv_set:nnn {up}{e}{\e}
509 \usv_set:nnn {up}{g}{\g}
510 \usv_set:nnn {up}{h}{\h}
511 \usv_set:nnn {up}{i}{\i}
512 \usv_set:nnn {up}{j}{\j}
513 \usv_set:nnn {up}{o}{\o}

514 \usv_set:nnn {it}{d}{1D451}
515 \usv_set:nnn {it}{e}{1D452}
516 \usv_set:nnn {it}{g}{1D454}
517 \usv_set:nnn {it}{h}{0210E}
518 \usv_set:nnn {it}{i}{1D456}

```

519 \usv_set:nnn {it}{j}{ "1D457}
 520 \usv_set:nnn {it}{o}{ "1D45C}

Latin ‘h’:

521 \usv_set:nnn {bb} {h}{ "1D559}
 522 \usv_set:nnn {tt} {h}{ "1D691}
 523 \usv_set:nnn {scr} {h}{ "1D4BD}
 524 \usv_set:nnn {frak} {h}{ "1D525}
 525 \usv_set:nnn {bfup} {h}{ "1D421}
 526 \usv_set:nnn {bfit} {h}{ "1D489}
 527 \usv_set:nnn {sfup} {h}{ "1D5C1}
 528 \usv_set:nnn {sfit} {h}{ "1D629}
 529 \usv_set:nnn {bffrak}{h}{ "1D58D}
 530 \usv_set:nnn {bfscr} {h}{ "1D4F1}
 531 \usv_set:nnn {bfsfup}{h}{ "1D5F5}
 532 \usv_set:nnn {bfsfit}{h}{ "1D65D}

Dotless ‘i’ and ‘j’:

533 \usv_set:nnn {up}{dotlessi}{ "00131}
 534 \usv_set:nnn {up}{dotlessj}{ "00237}
 535 \usv_set:nnn {it}{dotlessi}{ "1D6A4}
 536 \usv_set:nnn {it}{dotlessj}{ "1D6A5}

Blackboard:

537 \usv_set:nnn {bb}{C}{ "2102}
 538 \usv_set:nnn {bb}{H}{ "210D}
 539 \usv_set:nnn {bb}{N}{ "2115}
 540 \usv_set:nnn {bb}{P}{ "2119}
 541 \usv_set:nnn {bb}{Q}{ "211A}
 542 \usv_set:nnn {bb}{R}{ "211D}
 543 \usv_set:nnn {bb}{Z}{ "2124}
 544 \usv_set:nnn {up}{Pi} { "003A0}
 545 \usv_set:nnn {up}{pi} { "003C0}
 546 \usv_set:nnn {up}{Gamma} { "00393}
 547 \usv_set:nnn {up}{gamma} { "003B3}
 548 \usv_set:nnn {up}{summation}{ "02211}
 549 \usv_set:nnn {it}{Pi} { "1D6F1}
 550 \usv_set:nnn {it}{pi} { "1D70B}
 551 \usv_set:nnn {it}{Gamma} { "1D6E4}
 552 \usv_set:nnn {it}{gamma} { "1D6FE}
 553 \usv_set:nnn {bb}{Pi} { "0213F}
 554 \usv_set:nnn {bb}{pi} { "0213C}
 555 \usv_set:nnn {bb}{Gamma} { "0213E}
 556 \usv_set:nnn {bb}{gamma} { "0213D}
 557 \usv_set:nnn {bb}{summation}{ "02140}

Italic blackboard:

558 \usv_set:nnn {bbit}{D}{ "2145}
 559 \usv_set:nnn {bbit}{d}{ "2146}
 560 \usv_set:nnn {bbit}{e}{ "2147}
 561 \usv_set:nnn {bbit}{i}{ "2148}
 562 \usv_set:nnn {bbit}{j}{ "2149}

Script exceptions:

```

563 \usv_set:nnn {scr}{B}{ "212C}
564 \usv_set:nnn {scr}{E}{ "2130}
565 \usv_set:nnn {scr}{F}{ "2131}
566 \usv_set:nnn {scr}{H}{ "210B}
567 \usv_set:nnn {scr}{I}{ "2110}
568 \usv_set:nnn {scr}{L}{ "2112}
569 \usv_set:nnn {scr}{M}{ "2133}
570 \usv_set:nnn {scr}{R}{ "211B}
571 \usv_set:nnn {scr}{e}{ "212F}
572 \usv_set:nnn {scr}{g}{ "210A}
573 \usv_set:nnn {scr}{o}{ "2134}

574 \usv_set:nnn {cal}{B}{ "212C}
575 \usv_set:nnn {cal}{E}{ "2130}
576 \usv_set:nnn {cal}{F}{ "2131}
577 \usv_set:nnn {cal}{H}{ "210B}
578 \usv_set:nnn {cal}{I}{ "2110}
579 \usv_set:nnn {cal}{L}{ "2112}
580 \usv_set:nnn {cal}{M}{ "2133}
581 \usv_set:nnn {cal}{R}{ "211B}

```

Fraktur exceptions:

```

582 \usv_set:nnn {frak}{C}{ "212D}
583 \usv_set:nnn {frak}{H}{ "210C}
584 \usv_set:nnn {frak}{I}{ "2111}
585 \usv_set:nnn {frak}{R}{ "211C}
586 \usv_set:nnn {frak}{Z}{ "2128}

```

9.3 STIX fonts

Version 1.0.0 of the STIX fonts contains a number of alphabets in the private use area of Unicode; i.e., it contains many math glyphs that have not (yet or if ever) been accepted into the Unicode standard.

But we still want to be able to use them if possible.

```

587 </package & (X E j L U)>
588 <*stix>

```

Upright

```

589 \usv_set:nnn {stixsfup}{partial}{ "E17C}
590 \usv_set:nnn {stixsfup}{Greek}{ "E17D}
591 \usv_set:nnn {stixsfup}{greek}{ "E196}
592 \usv_set:nnn {stixsfup}{varTheta}{ "E18E}
593 \usv_set:nnn {stixsfup}{varepsilon}{ "E1AF}
594 \usv_set:nnn {stixsfup}{vartheta}{ "E1B0}
595 \usv_set:nnn {stixsfup}{varkappa}{0000} % ???
596 \usv_set:nnn {stixsfup}{varphi}{ "E1B1}
597 \usv_set:nnn {stixsfup}{varrho}{ "E1B2}
598 \usv_set:nnn {stixsfup}{varpi}{ "E1B3}
599 \usv_set:nnn {stixupslash}{Greek}{ "E2FC}

```

Italic

```

600 \usv_set:nnn {stixbbit}{A}{E154}
601 \usv_set:nnn {stixbbit}{B}{E155}
602 \usv_set:nnn {stixbbit}{E}{E156}
603 \usv_set:nnn {stixbbit}{F}{E157}
604 \usv_set:nnn {stixbbit}{G}{E158}
605 \usv_set:nnn {stixbbit}{I}{E159}
606 \usv_set:nnn {stixbbit}{J}{E15A}
607 \usv_set:nnn {stixbbit}{K}{E15B}
608 \usv_set:nnn {stixbbit}{L}{E15C}
609 \usv_set:nnn {stixbbit}{M}{E15D}
610 \usv_set:nnn {stixbbit}{O}{E15E}
611 \usv_set:nnn {stixbbit}{S}{E15F}
612 \usv_set:nnn {stixbbit}{T}{E160}
613 \usv_set:nnn {stixbbit}{U}{E161}
614 \usv_set:nnn {stixbbit}{V}{E162}
615 \usv_set:nnn {stixbbit}{W}{E163}
616 \usv_set:nnn {stixbbit}{X}{E164}
617 \usv_set:nnn {stixbbit}{Y}{E165}

618 \usv_set:nnn {stixbbit}{a}{E166}
619 \usv_set:nnn {stixbbit}{b}{E167}
620 \usv_set:nnn {stixbbit}{c}{E168}
621 \usv_set:nnn {stixbbit}{f}{E169}
622 \usv_set:nnn {stixbbit}{g}{E16A}
623 \usv_set:nnn {stixbbit}{h}{E16B}
624 \usv_set:nnn {stixbbit}{k}{E16C}
625 \usv_set:nnn {stixbbit}{l}{E16D}
626 \usv_set:nnn {stixbbit}{m}{E16E}
627 \usv_set:nnn {stixbbit}{n}{E16F}
628 \usv_set:nnn {stixbbit}{o}{E170}
629 \usv_set:nnn {stixbbit}{p}{E171}
630 \usv_set:nnn {stixbbit}{q}{E172}
631 \usv_set:nnn {stixbbit}{r}{E173}
632 \usv_set:nnn {stixbbit}{s}{E174}
633 \usv_set:nnn {stixbbit}{t}{E175}
634 \usv_set:nnn {stixbbit}{u}{E176}
635 \usv_set:nnn {stixbbit}{v}{E177}
636 \usv_set:nnn {stixbbit}{w}{E178}
637 \usv_set:nnn {stixbbit}{x}{E179}
638 \usv_set:nnn {stixbbit}{y}{E17A}
639 \usv_set:nnn {stixbbit}{z}{E17B}

640 \usv_set:nnn {stixsfit}{Numerals}{E1B4}
641 \usv_set:nnn {stixsfit}{partial}{E1BE}
642 \usv_set:nnn {stixsfit}{Greek}{E1BF}
643 \usv_set:nnn {stixsfit}{greek}{E1D8}
644 \usv_set:nnn {stixsfit}{varTheta}{E1D0}
645 \usv_set:nnn {stixsfit}{varepsilon}{E1F1}
646 \usv_set:nnn {stixsfit}{vartheta}{E1F2}
647 \usv_set:nnn {stixsfit}{varkappa}{0000} % ???
648 \usv_set:nnn {stixsfit}{varphi}{E1F3}

```

649 \usv_set:nnn {stixsfit}{varrho}{E1F4}
 650 \usv_set:nnn {stixsfit}{varpi}{E1F5}
 651 \usv_set:nnn {stixcal}{Latin}{E22D}
 652 \usv_set:nnn {stixcal}{num}{E262}
 653 \usv_set:nnn {scr}{num}{48}
 654 \usv_set:nnn {it}{num}{48}
 655 \usv_set:nnn {stixsfitslash}{Latin}{E294}
 656 \usv_set:nnn {stixsfitslash}{latin}{E2C8}
 657 \usv_set:nnn {stixsfitslash}{greek}{E32C}
 658 \usv_set:nnn {stixsfitslash}{varepsilon}{E37A}
 659 \usv_set:nnn {stixsfitslash}{vartheta}{E35E}
 660 \usv_set:nnn {stixsfitslash}{varkappa}{E374}
 661 \usv_set:nnn {stixsfitslash}{varphi}{E360}
 662 \usv_set:nnn {stixsfitslash}{varrho}{E376}
 663 \usv_set:nnn {stixsfitslash}{varpi}{E362}
 664 \usv_set:nnn {stixsfitslash}{digamma}{E36A}

Bold

665 \usv_set:nnn {stixbfupslash}{Greek}{E2FD}
 666 \usv_set:nnn {stixbfupslash}{Digamma}{E369}
 667 \usv_set:nnn {stixbfbb}{A}{E38A}
 668 \usv_set:nnn {stixbfbb}{B}{E38B}
 669 \usv_set:nnn {stixbfbb}{E}{E38D}
 670 \usv_set:nnn {stixbfbb}{F}{E38E}
 671 \usv_set:nnn {stixbfbb}{G}{E38F}
 672 \usv_set:nnn {stixbfbb}{I}{E390}
 673 \usv_set:nnn {stixbfbb}{J}{E391}
 674 \usv_set:nnn {stixbfbb}{K}{E392}
 675 \usv_set:nnn {stixbfbb}{L}{E393}
 676 \usv_set:nnn {stixbfbb}{M}{E394}
 677 \usv_set:nnn {stixbfbb}{O}{E395}
 678 \usv_set:nnn {stixbfbb}{S}{E396}
 679 \usv_set:nnn {stixbfbb}{T}{E397}
 680 \usv_set:nnn {stixbfbb}{U}{E398}
 681 \usv_set:nnn {stixbfbb}{V}{E399}
 682 \usv_set:nnn {stixbfbb}{W}{E39A}
 683 \usv_set:nnn {stixbfbb}{X}{E39B}
 684 \usv_set:nnn {stixbfbb}{Y}{E39C}
 685 \usv_set:nnn {stixbfbb}{a}{E39D}
 686 \usv_set:nnn {stixbfbb}{b}{E39E}
 687 \usv_set:nnn {stixbfbb}{c}{E39F}
 688 \usv_set:nnn {stixbfbb}{f}{E3A2}
 689 \usv_set:nnn {stixbfbb}{g}{E3A3}
 690 \usv_set:nnn {stixbfbb}{h}{E3A4}
 691 \usv_set:nnn {stixbfbb}{k}{E3A7}
 692 \usv_set:nnn {stixbfbb}{l}{E3A8}
 693 \usv_set:nnn {stixbfbb}{m}{E3A9}
 694 \usv_set:nnn {stixbfbb}{n}{E3AA}
 695 \usv_set:nnn {stixbfbb}{o}{E3AB}

696 \usv_set:nnn {stixbfbb}{p}{E3AC}
 697 \usv_set:nnn {stixbfbb}{q}{E3AD}
 698 \usv_set:nnn {stixbfbb}{r}{E3AE}
 699 \usv_set:nnn {stixbfbb}{s}{E3AF}
 700 \usv_set:nnn {stixbfbb}{t}{E3B0}
 701 \usv_set:nnn {stixbfbb}{u}{E3B1}
 702 \usv_set:nnn {stixbfbb}{v}{E3B2}
 703 \usv_set:nnn {stixbfbb}{w}{E3B3}
 704 \usv_set:nnn {stixbfbb}{x}{E3B4}
 705 \usv_set:nnn {stixbfbb}{y}{E3B5}
 706 \usv_set:nnn {stixbfbb}{z}{E3B6}
 707 \usv_set:nnn {stixbfsfup}{Numerals}{E3B7}

Bold Italic

708 \usv_set:nnn {stixbfsfit}{Numerals}{E1F6}
 709 \usv_set:nnn {stixbfbbi}{A}{E200}
 710 \usv_set:nnn {stixbfbbi}{B}{E201}
 711 \usv_set:nnn {stixbfbbi}{E}{E203}
 712 \usv_set:nnn {stixbfbbi}{F}{E204}
 713 \usv_set:nnn {stixbfbbi}{G}{E205}
 714 \usv_set:nnn {stixbfbbi}{I}{E206}
 715 \usv_set:nnn {stixbfbbi}{J}{E207}
 716 \usv_set:nnn {stixbfbbi}{K}{E208}
 717 \usv_set:nnn {stixbfbbi}{L}{E209}
 718 \usv_set:nnn {stixbfbbi}{M}{E20A}
 719 \usv_set:nnn {stixbfbbi}{O}{E20B}
 720 \usv_set:nnn {stixbfbbi}{S}{E20C}
 721 \usv_set:nnn {stixbfbbi}{T}{E20D}
 722 \usv_set:nnn {stixbfbbi}{U}{E20E}
 723 \usv_set:nnn {stixbfbbi}{V}{E20F}
 724 \usv_set:nnn {stixbfbbi}{W}{E210}
 725 \usv_set:nnn {stixbfbbi}{X}{E211}
 726 \usv_set:nnn {stixbfbbi}{Y}{E212}
 727 \usv_set:nnn {stixbfbbi}{a}{E213}
 728 \usv_set:nnn {stixbfbbi}{b}{E214}
 729 \usv_set:nnn {stixbfbbi}{c}{E215}
 730 \usv_set:nnn {stixbfbbi}{e}{E217}
 731 \usv_set:nnn {stixbfbbi}{f}{E218}
 732 \usv_set:nnn {stixbfbbi}{g}{E219}
 733 \usv_set:nnn {stixbfbbi}{h}{E21A}
 734 \usv_set:nnn {stixbfbbi}{k}{E21D}
 735 \usv_set:nnn {stixbfbbi}{l}{E21E}
 736 \usv_set:nnn {stixbfbbi}{m}{E21F}
 737 \usv_set:nnn {stixbfbbi}{n}{E220}
 738 \usv_set:nnn {stixbfbbi}{o}{E221}
 739 \usv_set:nnn {stixbfbbi}{p}{E222}
 740 \usv_set:nnn {stixbfbbi}{q}{E223}
 741 \usv_set:nnn {stixbfbbi}{r}{E224}
 742 \usv_set:nnn {stixbfbbi}{s}{E225}

```

743 \usv_set:nnn {stixbfbbbit}{t}{ "E226}
744 \usv_set:nnn {stixbfbbbit}{u}{ "E227}
745 \usv_set:nnn {stixbfbbbit}{v}{ "E228}
746 \usv_set:nnn {stixbfbbbit}{w}{ "E229}
747 \usv_set:nnn {stixbfbbbit}{x}{ "E22A}
748 \usv_set:nnn {stixbfbbbit}{y}{ "E22B}
749 \usv_set:nnn {stixbfbbbit}{z}{ "E22C}

750 \usv_set:nnn {stixbfcal}{Latin}{ "E247}

751 \usv_set:nnn {stixbfitslash}{Latin}{ "E295}
752 \usv_set:nnn {stixbfitslash}{latin}{ "E2C9}
753 \usv_set:nnn {stixbfitslash}{greek}{ "E32D}
754 \usv_set:nnn {stixsfitslash}{varepsilon}{ "E37B}
755 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35F}
756 \usv_set:nnn {stixsfitslash}{varkappa}{ "E375}
757 \usv_set:nnn {stixsfitslash}{varphi}{ "E361}
758 \usv_set:nnn {stixsfitslash}{varrho}{ "E377}
759 \usv_set:nnn {stixsfitslash}{varpi}{ "E363}
760 \usv_set:nnn {stixsfitslash}{digamma}{ "E36B}

761 </stix>
762 {*package & (XE j LU)}

```

9.4 Overcoming \@onlypreamble

The requirement of only setting up the maths fonts in the preamble is now removed. The following list might be overly ambitious.

```

763 \tl_map_inline:nn {
764   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
765   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
766   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
767   \version@list\version@elt\alpha@list\alpha@elt
768   \restore@mathversion\init@restore@version\dorestore@version\process@table
769   \new@mathversion\DeclareSymbolFont\group@list\group@elt
770   \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
771   \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
772   \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
773   \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
774   \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
775   \set@mathdelimiter\DeclareMathRadical\mathchar@type
776   \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
777 }{
778   \tl_remove_once:Nn \@preamblecmds {\do#1}
779 }

```

10 Fundamentals

10.1 Enlarging the number of maths families

To start with, we've got a power of two as many `\fams` as before. So (from `lftfssbas.dtx`) we want to redefine

```
780 (*XE)
781 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
782 \let\newfam\new@mathgroup
783 \end{XE}
```

This is sufficient for L^AT_EX's `\DeclareSymbolFont`-type commands to be able to define 256 named maths fonts. For LuaL^AT_EX, this is handled by the `lualatex-math` package.

10.2 Setting math chars, math codes, etc.

`\um_set_mathsymbol:nNNn` #1 : A L^AT_EX symbol font, e.g., operators
#2 : Symbol macro, e.g., `\alpha`
#3 : Type, e.g., `\mathalpha`
#4 : Slot, e.g., "221E

There are a bunch of tests to perform to process the various characters. The following assignments should all be fairly straightforward.

```
784 \cs_set:Nn \um_set_mathsymbol:nNNn {
785   \prg_case_tl:Nnn #3 {
786     \mathop { \um_set_big_operator:nnn {#1} {#2} {#4} }
787     \mathopen
788     {
789       \tl_if_in:NnTF \l_um_radicals_tl {#2}
790       {
791         \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
792         { \um_radical:nn {#1} {#4} }
793         \tl_set:cn {\l_um_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #4}
794       }
795       {
796         \um_set_delcode:nnn {#1} {#4} {#4}
797         \um_set_mathcode:nnn {#4} \mathopen {#1}
798         \cs_gset_protected_nopar:Npx #2
799         { \um_delimiter:Nnn \mathopen {#1} {#4} }
800       }
801     }
802   \mathclose
803   {
804     \um_set_delcode:nnn {#1} {#4} {#4}
805     \um_set_mathcode:nnn {#4} \mathclose {#1}
806     \cs_gset_protected_nopar:Npx #2
807     { \um_delimiter:Nnn \mathclose {#1} {#4} }
808   }
809   \mathfence
810   {
```

```

811 \um_set_mathcode:nnn {#4} {#3} {#1}
812 \um_set_delcode:nnn {#1} {#4} {#4}
813 \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
814 { \um_delimiter:Nnn \mathopen {#1} {#4} }
815 \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
816 { \um_delimiter:Nnn \mathclose {#1} {#4} }
817 }
818 \mathaccent
819 { \cs_gset_protected_nopar:Npx #2 { \um_accnt:nnn {fixed} {#1} {#4} } }
820 \mathbotaccent
821 { \cs_gset_protected_nopar:Npx #2 { \um_accnt:nnn {bot-
tom~ fixed} {#1} {#4} } }
822 \mathover
823 {
824 \cs_set_protected_nopar:Npx #2 ##1
825 { \mathop { \um_accnt:nnn {} {#1} {#4} {##1} } \limits }
826 }
827 \mathunder
828 {
829 \cs_set_protected_nopar:Npx #2 ##1
830 { \mathop { \um_accnt:nnn {bottom} {#1} {#4} {##1} } \limits }
831 }
832 }{
833 \um_set_mathcode:nnn {#4} {#3} {#1}
834 }
835 }

836 \edef\mathfence{\string\mathfence}
837 \edef\mathover{\string\mathover}
838 \edef\mathunder{\string\mathunder}
839 \edef\mathbotaccent{\string\mathbotaccent}

```

`\um_set_big_operator:nnn` #1 : Symbol font name

#2 : Macro to assign

#3 : Glyph slot

In the examples following, say we're defining for the symbol `\sum` (Σ). In order for literal Unicode characters to be used in the source and still have the correct limits behaviour, big operators are made math-active. This involves three steps:

- The active math char is defined to expand to the macro `\sum_sym`. (Later, the control sequence `\sum` will be assigned the math char.)
- Declare the plain old `mathchardef` for the control sequence `\sumop`. (This follows the convention of $\text{\LaTeX}/\text{amsmath}$.)
- Define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

Whether the `\nolimits` suffix is inserted is controlled by the token list `\l_um_nolimits_tl`, which contains a list of such characters. This list is checked dynamically to allow it to be updated mid-document.

Examples of expansion, by default, for two big operators:

(\sum →) \sum → \sum_sym → \sumop\nolimits

(\int →) \int → \int_sym → \intop

```

840 \cs_new:Nn \um_set_big_operator:nnn {
841   \group_begin:
842     \char_set_catcode_active:n {#3}
843     \char_gmake_mathactive:n {#3}
844     \um_active_char_set:wc #3 \q_nil { \cs_to_str:N #2 _sym }
845   \group_end:
846   \um_set_mathchar:cNnn { \cs_to_str:N #2 op } \mathop {#1} {#3}
847   \cs_gset:cpx { \cs_to_str:N #2 _sym } {
848     \exp_not:c { \cs_to_str:N #2 op }
849     \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
850   }
851 }

```

\um_set_mathcode:nnnn These are all wrappers for the primitive commands that take numerical input only.

```

\um_set_mathcode:nnn 852 \cs_set:Npn \um_set_mathcode:nnnn #1#2#3#4 {
\um_set_mathchar:NNnn 853   \Umathcode \int_eval:n {#1} =
\um_set_mathchar:cNnn 854   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
\um_set_delcode:nnn 855 }
\um_radical:nn 856 \cs_set:Npn \um_set_mathcode:nnn #1#2#3 {
\um_delimiter:Nnn 857   \Umathcode \int_eval:n {#1} =
\um_accent:nnn 858   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
\um_accent_keyword: 859 }
860 \cs_set:Npn \um_set_mathchar:NNnn #1#2#3#4 {
861   \Umathchardef #1 =
862   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
863 }
864 \cs_new:Nn \um_set_delcode:nnn {
865   \Udelcode#2 = \csname sym#1\endcsname #3
866 }
867 \cs_new:Nn \um_radical:nn {
868   \Uradical \csname sym#1\endcsname #2 \scan_stop:
869 }
870 \cs_new:Nn \um_delimiter:Nnn {
871   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
872 }
873 \cs_new:Nn \um_accent:nnn {
874   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
875 }
876 \cs_generate_variant:Nn \um_set_mathchar:NNnn {c}

```

\char_gmake_mathactive:N

\char_gmake_mathactive:n

```

877 \cs_new:Nn \char_gmake_mathactive:N {
878   \global\mathcode `#1 = "8000 \scan_stop:
879 }
880 \cs_new:Nn \char_gmake_mathactive:n {
881   \global\mathcode #1 = "8000 \scan_stop:
882 }

```


10.3 The main `\setmathfont` macro

Using a range including large character sets such as `\mathrel`, `\mathalpha`, *etc.*, is *very slow*! I hope to improve the performance somehow.

`\setmathfont` [#1]: font features

#2 : font name

```
883 \cs_new:Nn \um_init: {
884   \bool_set_true:N \l_um_ot_math_bool
```

- Erase any conception \LaTeX has of previously defined math symbol fonts; this allows `\DeclareSymbolFont` at any point in the document.

```
885   \cs_set_eq:NN \glb@currsz \scan_stop:
```

- To start with, assume we’re defining the font for every math symbol character.

```
886   \bool_set_true:N \l_um_init_bool
887   \seq_clear:N \l_um_char_range_seq
888   \clist_clear:N \l_um_char_num_range_clist
889   \seq_clear:N \l_um_mathalph_seq
890   \seq_clear:N \l_um_missing_alph_seq
```

- By default use the ‘normal’ math version

```
891   \tl_set:Nn \l_um_mversion_tl {normal}
```

- Other range initialisations

```
892   \tl_set:Nn \um_symfont_tl {operators}
893   \cs_set_eq:NN \um_sym:nnn \um_process_symbol_noparse:nnn
894   \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
895   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
896   \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
897   \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
898   \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_noparse:nn
899   \cs_set_eq:NN \um_make_mathactive:nNN \um_make_mathactive_noparse:nNN
```

- Define default font features for the script and scriptscript font.

```
900   \tl_set:Nn \l_um_script_features_tl {Style=MathScript}
901   \tl_set:Nn \l_um_sscript_features_tl {Style=MathScriptScript}
902   \tl_set_eq:NN \l_um_script_font_tl \l_um_fontname_tl
903   \tl_set_eq:NN \l_um_sscript_font_tl \l_um_fontname_tl
904 }
905 \DeclareDocumentCommand \setmathfont { O{} m } {
906   \tl_set:Nn \l_um_fontname_tl {#2}
907   \um_init:
```

Grab the current size information: (is this robust enough? Maybe it should be preceded by `\normalsize`). The macro `\S@<size>` contains the definitions of the sizes used for maths letters, subscripts and subsubscripts in `\tf@size`, `\sf@size`, and `\ssf@size`, respectively.

```
908 \cs_if_exist:cF { S@ \f@size } { \calculate@math@sizes }
909 \csname S@\f@size\endcsname
```

Parse options and tell people what's going on:

```
910 \keys_set_known:nnN {unicode-math} {#1} \l_um_unknown_keys_clist
911 \bool_if:NT \l_um_init_bool { \um_log:n {default-math-font} }
```

Use fontspec to select a font to use.

```
912 \um_fontspec_select_font:
```

Now define `\um_symfont_tl` as the L^AT_EX math font to access everything:

```
913 \cs_if_exist:cF { sym \um_symfont_tl }
914 {
915     \DeclareSymbolFont{\um_symfont_tl}
916     {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
917 }
918 \SetSymbolFont{\um_symfont_tl}{\l_um_mversion_tl}
919 {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
```

Set the bold math version.

```
920 \tl_set:Nn \l_um_tmpa_tl {normal}
921 \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
922 {
923     \SetSymbolFont{\um_symfont_tl}{bold}
924     {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
925 }
```

Declare the math sizes (i.e., scaling of superscripts) for the specific values for this font, and set defaults for math fams two and three for legacy compatibility:

```
926 \bool_if:NT {\l_um_ot_math_bool && !\g_um_mainfont_already_set_bool} {
927     \bool_set_true:N \g_um_mainfont_already_set_bool
928     \um_declare_math_sizes:
929     \um_setup_legacy_fam_two:
930     \um_setup_legacy_fam_three:
931 }
```

And now we input every single maths char.

```
932 \um_input_math_symbol_table:
```

Finally,

- Remap symbols that don't take their natural mathcode
- Activate any symbols that need to be math-active
- Enable wide/narrow accents
- Assign delimiter codes for symbols that need to grow
- Setup the maths alphabets (`\mathbf` etc.)

```

933 \um_remap_symbols:
934 \um_setup_mathactives:
935 \um_setup_accents:
936 \um_setup_delcodes:
937 \um_setup_alphabets:
938 \um_setup_negations:

```

Prevent spaces, and that's it:

```

939 \ignorespaces
940 }

```

`\um_declare_math_sizes:` Set the math sizes according to the recommend font parameters:

```

941 \cs_new:Nn \um_declare_math_sizes:
942 {
943   \dim_compare:nF { \fontdimen 10 \l_um_font == 0pt }
944   {
945     \DeclareMathSizes { \f@size } { \f@size }
946     { \um_fontdimen_to_scale:nn {10} { \l_um_font } }
947     { \um_fontdimen_to_scale:nn {11} { \l_um_font } }
948   }
949 }

```

`\um_setup_legacy_fam_two:`

```

950 \cs_new:Nn \um_setup_legacy_fam_two:
951 {
952   \fontspec_set_family:Nxn \l_um_family_tl
953   {
954     \l_um_font_keyval_tl,
955     Scale=1.00001,
956     FontAdjustment={
957       \fontdimen8\font= \um_get_fontparam:nn {43} {FractionNumeratorDis-
958         playStyleShiftUp}\relax
959       \fontdimen9\font= \um_get_fontparam:nn {42} {FractionNumerator-
960         ShiftUp}\relax
961       \fontdimen10\font=\um_get_fontparam:nn {32} {StackTopShiftUp}\relax
962       \fontdimen11\font=\um_get_fontparam:nn {45} {FractionDenomina-
963         torDisplayStyleShiftDown}\relax
964       \fontdimen12\font=\um_get_fontparam:nn {44} {FractionDenomina-
965         torShiftDown}\relax
966       \fontdimen13\font=\um_get_fontparam:nn {21} {Superscript-
967         ShiftUp}\relax
968       \fontdimen14\font=\um_get_fontparam:nn {21} {Superscript-
969         ShiftUp}\relax
970       \fontdimen15\font=\um_get_fontparam:nn {22} {SuperscriptShif-
971         tUpCramped}\relax
972       \fontdimen16\font=\um_get_fontparam:nn {18} {SubscriptShift-
973         Down}\relax
974       \fontdimen17\font=\um_get_fontparam:nn {18} {SubscriptShiftDown-
975         WithSuperscript}\relax
976       \fontdimen18\font=\um_get_fontparam:nn {24} {SuperscriptBaseline-
977         DropMax}\relax

```

```

968      \fontdimen19\font=\um_get_fontparam:nn {20} {SubscriptBaseline-
DropMin}\relax
969      \fontdimen20\font=0pt\relax % delim1 = FractionDelimiterDisplaySize
970      \fontdimen21\font=0pt\relax % delim2 = FractionDelimiterSize
971      \fontdimen22\font=\um_get_fontparam:nn {15} {AxisHeight}\relax
972    }
973    } {\l_um_fontname_tl}
974    \SetSymbolFont{symbols}{\l_um_mversion_tl}
975      {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
976
977    \tl_set:Nn \l_um_tmpa_tl {normal}
978    \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
979      {
980        \SetSymbolFont{symbols}{bold}
981          {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
982      }
983  }

```

\um_setup_legacy_fam_three:

```

984 \cs_new:Nn \um_setup_legacy_fam_three:
985 {
986   \fontspec_set_family:Nxn \l_um_family_tl
987   {
988     \l_um_font_keyval_tl,
989     Scale=0.99999,
990     FontAdjustment={
991       \fontdimen8\font= \um_get_fontparam:nn {48} {FractionRuleThick-
ness}\relax
992       \fontdimen9\font= \um_get_fontparam:nn {28} {UpperLimitGap-
Min}\relax
993       \fontdimen10\font=\um_get_fontparam:nn {30} {LowerLimitGap-
Min}\relax
994       \fontdimen11\font=\um_get_fontparam:nn {29} {UpperLimitBaselineR-
iseMin}\relax
995       \fontdimen12\font=\um_get_fontparam:nn {31} {LowerLimitBaseline-
DropMin}\relax
996       \fontdimen13\font=0pt\relax
997     }
998   } {\l_um_fontname_tl}
999   \SetSymbolFont{largesymbols}{\l_um_mversion_tl}
1000     {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
1001
1002   \tl_set:Nn \l_um_tmpa_tl {normal}
1003   \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
1004     {
1005       \SetSymbolFont{largesymbols}{bold}
1006         {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
1007     }
1008 }
1009 \cs_new:Nn \um_get_fontparam:nn

```

```

1010 (XE) { \the\fontdimen#1\l_um_font\relax }
1011 (LU) { \directlua{fontspec.mathfontdimen("l_um_font", "#2")} }

```

Backward compatibility alias.

```

1012 \cs_set_eq:NN \resetmathfont \setmathfont

```

`\um_fontspec_select_font:` Select the font with `\fontspec` and define `\l_um_font` from it.

```

1013 \cs_new:Nn \um_fontspec_select_font: {
1014   \tl_set:Nx \l_um_font_keyval_tl {
1015     (LU) Renderer = Basic,
1016     BoldItalicFont = {}, ItalicFont = {},
1017     Script = Math,
1018     SizeFeatures = {
1019       {Size = \tf@size-} ,
1020       {Size = \sf@size-\tf@size ,
1021         Font = \l_um_script_font_tl ,
1022         \l_um_script_features_tl
1023       } ,
1024       {Size = -\sf@size ,
1025         Font = \l_um_sscript_font_tl ,
1026         \l_um_sscript_features_tl
1027       }
1028     },
1029     \l_um_unknown_keys_clist
1030   }
1031   \fontspec_set_fontface:NNxn \l_um_font \l_um_family_tl
1032   {\l_um_font_keyval_tl} {\l_um_fontname_tl}

```

Check whether we're using a real maths font:

```

1033 \group_begin:
1034   \fontfamily{\l_um_family_tl}\selectfont
1035   \fontspec_if_script:nF {math} {\bool_gset_false:N \l_um_ot_math_bool}
1036 \group_end:
1037 }

```

10.3.1 Functions for setting up symbols with mathcodes

`\um_process_symbol_noparse:nnn` If the range font feature has been used, then only a subset of the Unicode glyphs are to be defined. See section §11.3 for the code that enables this.

```

1038 \cs_set:Npn \um_process_symbol_noparse:nnn #1#2#3 {
1039   \um_set_mathsymbol:nNNn {\um_syfont_tl} #2#3{#1}
1040 }
1041 \cs_set:Npn \um_process_symbol_parse:nnn #1#2#3 {
1042   \um_if_char_spec:nNNT{#1}{#2}{#3}{
1043     \um_process_symbol_noparse:nnn {#1}{#2}{#3}
1044   }
1045 }

```

`\um_remap_symbols:` This function is used to define the mathcodes for those chars which should be mapped to a different glyph than themselves.

`\um_remap_symbol_noparse:nnn`

`\um_remap_symbol_parse:nnn`

```

1046 \cs_new:Npn \um_remap_symbols: {
1047   \um_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
1048   \um_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "cen-
      tred asterisk"
1049   \bool_if:NF \g_um_literal_colon_bool {
1050     \um_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
1051   }
1052 }

```

Where `\um_remap_symbol:nnn` is defined to be one of these two, depending on the range setup:

```

1053 \cs_new:Nn \um_remap_symbol_parse:nnn {
1054   \um_if_char_spec:nNTT {#3} {\@nil} {#2} {
1055     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
1056   }
1057 }
1058 \cs_new:Nn \um_remap_symbol_noparse:nnn {
1059   \clist_map_inline:nn {#1} {
1060     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
1061   }
1062 }

```

10.3.2 Active math characters

There are more math active chars later in the subscript/superscript section. But they don't need to be able to be typeset directly.

`\um_setup_mathactives:`

```

1063 \cs_new:Npn \um_setup_mathactives: {
1064   \um_make_mathactive:nNN {"2032} \um_prime_single_mchar \mathord
1065   \um_make_mathactive:nNN {"2033} \um_prime_double_mchar \mathord
1066   \um_make_mathactive:nNN {"2034} \um_prime_triple_mchar \mathord
1067   \um_make_mathactive:nNN {"2057} \um_prime_quad_mchar \mathord
1068   \um_make_mathactive:nNN {"2035} \um_backprime_single_mchar \mathord
1069   \um_make_mathactive:nNN {"2036} \um_backprime_double_mchar \mathord
1070   \um_make_mathactive:nNN {"2037} \um_backprime_triple_mchar \mathord
1071   \um_make_mathactive:nNN {\`'} \mathstraightquote \mathord
1072   \um_make_mathactive:nNN {\`'} \mathbacktick \mathord
1073 }

```

`\um_make_mathactive:nNN` Makes #1 a mathactive char, and gives cs #2 the meaning of mathchar #1 with class #3. You are responsible for giving active #1 a particular meaning!

```

1074 \cs_new:Nn \um_make_mathactive_parse:nNN
1075 {
1076   \um_if_char_spec:nNTT {#1} #2 #3
1077   { \um_make_mathactive_noparse:nNN {#1} #2 #3 }
1078 }
1079 \cs_new:Nn \um_make_mathactive_noparse:nNN
1080 {
1081   \um_set_mathchar:NNnn #2 #3 {\um_symfont_tl} {#1}
1082   \char_gmake_mathactive:n {#1}

```

```
1083 }
```

10.3.3 Delimiter codes

`\um_assign_delcode:nn`

```
1084 \cs_new:Nn \um_assign_delcode_noparse:nn {
1085   \um_set_delcode:nnn \um_symfont_tl {#1} {#2}
1086 }
1087 \cs_new:Nn \um_assign_delcode_parse:nn {
1088   \um_if_char_spec:nNT {#2}{\@nil}{\@nil} {
1089     \um_assign_delcode_noparse:nn {#1} {#2}
1090   }
1091 }
```

`\um_assign_delcode:n` Shorthand.

```
1092 \cs_new:Nn \um_assign_delcode:n { \um_assign_delcode:nn {#1} {#1} }
```

Some symbols that aren't `mathopen`/`mathclose` still need to have delimiter codes assigned. The list of vertical arrows may be incomplete. On the other hand, many fonts won't support them all being stretchy. And some of them are probably not meant to stretch, either. But adding them here doesn't hurt.

`\um_setup_delcodes:`

```
1093 \cs_new:Npn \um_setup_delcodes: {
1094   \um_assign_delcode:nn {\.} {\c_zero} % ensure \left. and \right. work
1095   \um_assign_delcode:nn {\./} {\g_um_slash_delimiter_usv}
1096   \um_assign_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
1097   \um_assign_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
1098   \um_assign_delcode:n {"005C} % backslash
1099   \um_assign_delcode:nn {\<} {"27E8} % angle brackets with ascii notation
1100   \um_assign_delcode:nn {\>} {"27E9} % angle brackets with ascii notation
1101   \um_assign_delcode:n {"2191} % up arrow
1102   \um_assign_delcode:n {"2193} % down arrow
1103   \um_assign_delcode:n {"2195} % updown arrow
1104   \um_assign_delcode:n {"219F} % up arrow twohead
1105   \um_assign_delcode:n {"21A1} % down arrow twohead
1106   \um_assign_delcode:n {"21A5} % up arrow from bar
1107   \um_assign_delcode:n {"21A7} % down arrow from bar
1108   \um_assign_delcode:n {"21A8} % updown arrow from bar
1109   \um_assign_delcode:n {"21BE} % up harpoon right
1110   \um_assign_delcode:n {"21BF} % up harpoon left
1111   \um_assign_delcode:n {"21C2} % down harpoon right
1112   \um_assign_delcode:n {"21C3} % down harpoon left
1113   \um_assign_delcode:n {"21C5} % arrows up down
1114   \um_assign_delcode:n {"21F5} % arrows down up
1115   \um_assign_delcode:n {"21C8} % arrows up up
1116   \um_assign_delcode:n {"21CA} % arrows down down
1117   \um_assign_delcode:n {"21D1} % double up arrow
1118   \um_assign_delcode:n {"21D3} % double down arrow
1119   \um_assign_delcode:n {"21D5} % double updown arrow
```

```

1120 \um_assign_delcode:n {"21DE} % up arrow double stroke
1121 \um_assign_delcode:n {"21DF} % down arrow double stroke
1122 \um_assign_delcode:n {"21E1} % up arrow dashed
1123 \um_assign_delcode:n {"21E3} % down arrow dashed
1124 \um_assign_delcode:n {"21E7} % up white arrow
1125 \um_assign_delcode:n {"21E9} % down white arrow
1126 \um_assign_delcode:n {"21EA} % up white arrow from bar
1127 \um_assign_delcode:n {"21F3} % updown white arrow
1128 }

```

10.4 (Big) operators

Turns out that \LaTeX is clever enough to deal with big operators for us automatically with `\Umathchardef`. Amazing!

However, the limits aren't set automatically; that is, we want to define, a la Plain \TeX *etc.*, `\def\int{\intop\nolimits}`, so there needs to be a transformation from `\int` to `\intop` during the expansion of `_um_sym:nnn` in the appropriate contexts.

`\l_um_nolimits_tl` This macro is a sequence containing those maths operators that require a `\nolimits` suffix. This list is used when processing `unicode-math-table.tex` to define such commands automatically (see the macro `\um_set_mathsymbol:nNNn`). I've chosen essentially just the operators that look like integrals; hopefully a better mathematician can help me out here. I've a feeling that it's more useful *not* to include the multiple integrals such as \int , but that might be a matter of preference.

```

1129 \tl_new:N \l\_um\_nolimits\_tl
1130 \tl_set:Nn \l\_um\_nolimits\_tl {
1131   \int\iint\iiint\iiint\oint\oiint\oiint
1132   \intclockwise\varointclockwise\ointctrclockwise\sumint
1133   \intbar\intBar\oint\cirfnint\awint\rppoint
1134   \scpolint\mpoint\pointint\sqint\intlarhk\intx
1135   \intcap\intcup\upint\lowint
1136 }

```

`\addnolimits` This macro appends material to the macro containing the list of operators that don't take limits.

```

1137 \DeclareDocumentCommand \addnolimits {m} {
1138   \tl_put_right:Nn \l\_um\_nolimits\_tl {#1}
1139 }

```

`\removenolimits` Can this macro be given a better name? It removes an item from the `nolimits` list.

```

1140 \DeclareDocumentCommand \removenolimits {m} {
1141   \tl_remove_all:Nn \l\_um\_nolimits\_tl {#1}
1142 }

```

10.5 Radicals

The radical for square root is organised in `\um_set_mathsymbol:nNNn`. I think it's the only radical ever. (Actually, there is also `\cuberoot` and `\fourthroot`, but they don't seem to behave as proper radicals.)

Also, what about right-to-left square roots?

`\l_um_radicals_tl` We organise radicals in the same way as nolimits-operators.

```
1143 \tl_new:N \l_um_radicals_tl
1144 \tl_set:Nn \l_um_radicals_tl {\sqrt \longdivision}
```

10.6 Maths accents

Maths accents should just work *if they are available in the font*.

10.7 Common interface for font parameters

X_YTeX and LuaTeX have different interfaces for math font parameters. We use LuaTeX's interface because it's much better, but rename the primitives to be more L^AT_EX3-like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the LuaTeX names, with underscores inserted between words. For every parameter `\Umath{LuaTeX name}`, we define an expandable getter command `\um_{LATEX3 name}:N` and a protected setter command `\um_set_{LATEX3 name}:Nn`. The getter command takes one of the style primitives (`\displaystyle` etc.) and expands to the font parameter, which is a *⟨dimension⟩*. The setter command takes a style primitive and a dimension expression, which is parsed with `\dim_eval:n`.

Often, the mapping between font dimensions and font parameters is bijective, but there are cases which require special attention:

- Some parameters map to different dimensions in display and non-display styles.
- Likewise, one parameter maps to different dimensions in non-cramped and cramped styles.
- There are a few parameters for which X_YTeX doesn't seem to provide `\font-dimens`; in this case the getter and setter commands are left undefined.

Cramped style tokens LuaTeX has `\crampeddisplaystyle` etc., but they are loaded as `\luatexcrampeddisplaystyle` etc. by the `luatextra` package. X_YTeX, however, doesn't have these primitives, and their syntax cannot really be emulated. Nevertheless, we define these commands as quarks, so they can be used as arguments to the font parameter commands (but nowhere else). Making these commands available is necessary because we need to make a distinction between cramped and non-cramped styles for one font parameter.

`\um_new_cramped_style:N` **#1** : command

Define *⟨command⟩* as a new cramped style switch. For LuaTeX, simply rename the corresponding primitive. For X_YTeX, define *⟨command⟩* as a new quark.

```
1145 \cs_new_protected_nopar:Nn \um_new_cramped_style:N
1146 (XE) { \quark_new:N #1 }
1147 (LU) { \cs_new_eq:Nc #1 { Luatex \cs_to_str:N #1 } }
```

```

\crampeddisplaystyle The cramped style commands.
\crampedtextstyle    1148 \um_new_cramped_style:N \crampeddisplaystyle
\crampedscriptstyle  1149 \um_new_cramped_style:N \crampedtextstyle
\crampedscriptscriptstyle 1150 \um_new_cramped_style:N \crampedscriptstyle
                        1151 \um_new_cramped_style:N \crampedscriptscriptstyle

```

Font dimension mapping Font parameters may differ between the styles. LuaTeX accounts for this by having the parameter primitives take a style token argument. To replicate this behavior in XeTeX, we have to map style tokens to specific combinations of font dimension numbers and math fonts (`\textfont` etc.).

```

\um_font_dimen:Nnnnn #1 : style token
                     #2 : font dimen for display style
                     #3 : font dimen for cramped display style
                     #4 : font dimen for non-display styles
                     #5 : font dimen for cramped non-display styles

```

Map math style to XeTeX math font dimension. `<style token>` must be one of the style switches (`\displaystyle`, `\crampeddisplaystyle`, ...). The other parameters are integer constants referring to font dimension numbers. The macro expands to a dimension which contains the appropriate font dimension.

```

1152 (*XE)
1153 \cs_new_nopar:Npn \um_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
1154   \fontdimen
1155   \cs_if_eq:NNTF #1 \displaystyle {
1156     #2 \textfont
1157   } {
1158     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
1159       #3 \textfont
1160     } {
1161       \cs_if_eq:NNTF #1 \textstyle {
1162         #4 \textfont
1163       } {
1164         \cs_if_eq:NNTF #1 \crampedtextstyle {
1165           #5 \textfont
1166         } {
1167           \cs_if_eq:NNTF #1 \scriptstyle {
1168             #4 \scriptfont
1169           } {
1170             \cs_if_eq:NNTF #1 \crampedscriptstyle {
1171               #5 \scriptfont
1172             } {
1173               \cs_if_eq:NNTF #1 \scriptscriptstyle {
1174                 #4 \scriptscriptfont
1175               } {
1176                 #5 \scriptscriptfont
1177               }
1178             }

```

Should we check here if the style is invalid?

```

1179     }
1180   }
1181 }
1182 }
1183 }

```

Which family to use?

```

1184   \c_two
1185 }
1186 \>XE

```

Font parameters This paragraph contains macros for defining the font parameter interface, as well as the definition for all font parameters known to LuaTeX.

`\um_font_param:nnnnn` #1 : name
 #2 : font dimension for non-cramped display style
 #3 : font dimension for cramped display style
 #4 : font dimension for non-cramped non-display styles
 #5 : font dimension for cramped non-display styles
 This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XeTeX font dimension numbers must be integer constants.

```

1187 \cs_new_protected_nopar:Nn \um_font_param:nnnnn
1188 {*XE}
1189 {
1190   \um_font_param_aux:ccnnnn { um_ #1 :N } { um_set_ #1 :N }
1191     { #2 } { #3 } { #4 } { #5 }
1192 }
1193 \>XE}
1194 {*LU}
1195 {
1196   \tl_set:Nn \l_um_tmpa_tl { #1 }
1197   \tl_remove_all:Nn \l_um_tmpa_tl { _ }
1198   \um_font_param_aux:ccc { um_ #1 :N } { um_set_ #1 :N }
1199     { \luatexUmath \l_um_tmpa_tl }
1200 }
1201 \>LU}

```

`\um_font_param:nnn` #1 : name
 #2 : font dimension for display style
 #3 : font dimension for non-display styles
 This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XeTeX font dimension numbers must be integer constants.

```

1202 \cs_new_protected_nopar:Npn \um_font_param:nnn #1 #2 #3 {
1203   \um_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
1204 }

```

`\um_font_param:nn` **#1** : name
#2 : font dimension
This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XeTeX font dimension number must be an integer constant.

```

1205 \cs_new_protected_nopar:Npn \um_font_param:nn #1 #2 {
1206   \um_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
1207 }

```

`\um_font_param:n` **#1** : name
This macro defines getter and setter functions for the font parameter $\langle name \rangle$, which is considered unavailable in XeTeX. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`.

```

1208 \cs_new_protected_nopar:Nn \um_font_param:n
1209 (XE) { }
1210 (LU) { \um_font_param:nnnnn { #1 } { 0 } { 0 } { 0 } { 0 } }

```

`\um_font_param_aux:NNnnnn` Auxiliary macros for generating font parameter accessor macros.

```

\um_font_param_aux:NNN
1211 (*XE)
1212 \cs_new_protected_nopar:Nn \um_font_param_aux:NNnnnn
1213 {
1214   \cs_new_nopar:Npn #1 ##1 {
1215     \um_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
1216   }
1217   \cs_new_protected_nopar:Npn #2 ##1 ##2 {
1218     #1 ##1 \dim_eval:n { ##2 }
1219   }
1220 }
1221 \cs_generate_variant:Nn \um_font_param_aux:NNnnnn { cc }
1222 (/XE)
1223 (*LU)
1224 \cs_new_protected_nopar:Nn \um_font_param_aux:NNN
1225 {
1226   \cs_new_nopar:Npn #1 ##1 {
1227     #3 ##1
1228   }
1229   \cs_new_protected_nopar:Npn #2 ##1 ##2 {
1230     #3 ##1 \dim_eval:n { ##2 }
1231   }
1232 }
1233 \cs_generate_variant:Nn \um_font_param_aux:NNN { ccc }
1234 (/LU)

```

Now all font parameters that are listed in the LuaTeX reference follow.

```

1235 \um_font_param:nn { axis } { 15 }
1236 \um_font_param:nn { operator_size } { 13 }
1237 \um_font_param:n { fraction_del_size }
1238 \um_font_param:nnn { fraction_denom_down } { 45 } { 44 }
1239 \um_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }

```

```

1240 \um_font_param:nnn { fraction_num_up } { 43 } { 42 }
1241 \um_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
1242 \um_font_param:nn { fraction_rule } { 48 }
1243 \um_font_param:nn { limit_above_bgap } { 29 }
1244 \um_font_param:n { limit_above_kern }
1245 \um_font_param:nn { limit_above_vgap } { 28 }
1246 \um_font_param:nn { limit_below_bgap } { 31 }
1247 \um_font_param:n { limit_below_kern }
1248 \um_font_param:nn { limit_below_vgap } { 30 }
1249 \um_font_param:nn { over_delimiter_vgap } { 41 }
1250 \um_font_param:nn { over_delimiter_bgap } { 38 }
1251 \um_font_param:nn { under_delimiter_vgap } { 40 }
1252 \um_font_param:nn { under_delimiter_bgap } { 39 }
1253 \um_font_param:nn { overbar_kern } { 55 }
1254 \um_font_param:nn { overbar_rule } { 54 }
1255 \um_font_param:nn { overbar_vgap } { 53 }
1256 \um_font_param:n { quad }
1257 \um_font_param:nn { radical_kern } { 62 }
1258 \um_font_param:nn { radical_rule } { 61 }
1259 \um_font_param:nnn { radical_vgap } { 60 } { 59 }
1260 \um_font_param:nn { radical_degree_before } { 63 }
1261 \um_font_param:nn { radical_degree_after } { 64 }
1262 \um_font_param:nn { radical_degree_raise } { 65 }
1263 \um_font_param:nn { space_after_script } { 27 }
1264 \um_font_param:nnn { stack_denom_down } { 35 } { 34 }
1265 \um_font_param:nnn { stack_num_up } { 33 } { 32 }
1266 \um_font_param:nnn { stack_vgap } { 37 } { 36 }
1267 \um_font_param:nn { sub_shift_down } { 18 }
1268 \um_font_param:nn { sub_shift_drop } { 20 }
1269 \um_font_param:n { subsup_shift_down }
1270 \um_font_param:nn { sub_top_max } { 19 }
1271 \um_font_param:nn { subsup_vgap } { 25 }
1272 \um_font_param:nn { sup_bottom_min } { 23 }
1273 \um_font_param:nn { sup_shift_drop } { 24 }
1274 \um_font_param:nnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
1275 \um_font_param:nn { supsub_bottom_max } { 26 }
1276 \um_font_param:nn { underbar_kern } { 58 }
1277 \um_font_param:nn { underbar_rule } { 57 }
1278 \um_font_param:nn { underbar_vgap } { 56 }
1279 \um_font_param:n { connector_overlap_min }

```

11 Font features

`\new@mathversion` Fix bug in the L^AT_EX version. (Fixed upstream, too, but unsure when that will propagate.)

```

1280 \def\new@mathversion#1{%
1281   \expandafter\in@\expandafter#1\expandafter{\version@list}%
1282   \ifin@
1283     \@font@info{Redeclaring math version

```

```

1284         \expandafter \@gobblefour \string#1'%
1285     \else
1286         \expandafter \newcount \csname c@\expandafter
1287             \@gobble \string#1 \endcsname
1288         \def \version@elt {\noexpand \version@elt \noexpand}%
1289         \edef \version@list {\version@list \version@elt#1}%
1290     \fi
1291     \toks@{}%
1292     \count@ \z@
1293     \def \group@elt##1##2{%
1294         \advance \count@ \@ne
1295         \addto@hook \toks@ {\getanddefine@fonts##1##2}%
1296     }%
1297     \group@list
1298     \global \csname c@\expandafter \@gobble \string#1 \endcsname \count@
1299     \def \alpha@elt##1##2##3{%
1300         \ifx##2 \no@alphabet@error
1301             \toks@ \expandafter {\the \toks@ \install@mathalphabet##1%
1302                 {\no@alphabet@error##1}}%
1303         \else
1304             \toks@ \expandafter {\the \toks@ \install@mathalphabet##1%
1305                 {\select@group##1##2##3}}%
1306         \fi
1307     }%
1308     \alpha@list
1309     \xdef#1 {\the \toks@}%
1310 }

```

11.1 Math version

```

1311 \keys_define:nn {unicode-math}
1312 {
1313     version .code:n =
1314     {
1315         \tl_set:Nn \l_um_mversion_tl {#1}
1316         \DeclareMathVersion{\l_um_mversion_tl}
1317     }
1318 }

```

11.2 Script and scriptscript font options

```

1319 \keys_define:nn {unicode-math}
1320 {
1321     script-features .tl_set:N = \l_um_script_features_tl ,
1322     sscript-features .tl_set:N = \l_um_sscript_features_tl ,
1323     script-font .tl_set:N = \l_um_script_font_tl ,
1324     sscript-font .tl_set:N = \l_um_sscript_font_tl ,
1325 }

```

11.3 Range processing

```

1326 \seq_new:N \l_um_mathalph_seq

```

```

1327 \seq_new:N \l_um_char_range_seq
1328 \seq_new:N \l_um_mclass_range_seq
1329 \seq_new:N \l_um_cmd_range_seq
1330 \keys_define:nn {unicode-math} {
1331   range .code:n = {
1332     \bool_set_false:N \l_um_init_bool

```

Set processing functions if we're not defining the full Unicode math repertoire. Math symbols are defined with `_um_sym:nnn`; see section §10.3.1 for the individual definitions

```

1333   \int_incr:N \g_um_fam_int
1334   \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
1335   \cs_set_eq:NN \_um_sym:nnn \um_process_symbol_parse:nnn
1336   \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_parse:Nnn
1337   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
1338   \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
1339   \cs_set_eq:NN \um_map_char_single:nn \um_map_char_parse:nn
1340   \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_parse:nn
1341   \cs_set_eq:NN \um_make_mathactive:nnn \um_make_mathactive_parse:nnn

```

Proceed by filling up the various ‘range’ seqs according to the user options.

```

1342   \seq_clear:N \l_um_char_range_seq
1343   \seq_clear:N \l_um_mclass_range_seq
1344   \seq_clear:N \l_um_cmd_range_seq
1345   \seq_clear:N \l_um_mathalph_seq
1346   \clist_map_inline:nn {#1} {
1347     \um_if_mathalph_decl:nTF {##1} {
1348       \seq_put_right:Nx \l_um_mathalph_seq {
1349         { \exp_not:V \l_um_tmppa_tl }
1350         { \exp_not:V \l_um_tmppb_tl }
1351         { \exp_not:V \l_um_tmppc_tl }
1352       }
1353     }{

```

Four cases: math class matching the known list; single item that is a control sequence—command name; single item that isn't—edge case, must be 0–9; none of the above—char range.

```

1354       \seq_if_in:NnTF \g_um_mathclasses_seq {##1}
1355       { \seq_put_right:Nn \l_um_mclass_range_seq {##1} }
1356       {
1357         \bool_if:nTF { \tl_if_single_p:n {##1} && \token_if_cs_p:N ##1 }
1358         { \seq_put_right:Nn \l_um_cmd_range_seq {##1} }
1359         { \seq_put_right:Nn \l_um_char_range_seq {##1} }
1360       }
1361     }
1362   }
1363 }
1364 }
1365 \seq_new:N \g_um_mathclasses_seq
1366 \seq_set_from_clist:Nn \g_um_mathclasses_seq
1367 {
1368   \mathord,\mathalpha,\mathop,\mathbin,\mathrel,

```

```

1369 \mathopen,\mathclose,\mathpunct,\mathaccent,
1370 \mathfence,\mathover,\mathunder,\mathbotaccent
1371 }

```

`\um_if_mathalph_decl:nTF` Possible forms of input:

```

\mathscr
\mathscr->\mathup
\mathscr/{Latin}
\mathscr/{Latin}->\mathup

```

Outputs:

tmpa: math style (e.g., `\mathscr`)

tmpb: alphabets (e.g., Latin)

tmpc: remap style (e.g., `\mathup`). Defaults to tmpa.

The remap style can also be `\mathcal->stixcal`, which I marginally prefer in the general case.

```

1372 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
1373   \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {#1} }
1374   \tl_clear:N \l_um_tmpb_tl
1375   \tl_clear:N \l_um_tmpc_tl
1376   \tl_if_in:NnT \l_um_tmpa_tl {->} {
1377     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
1378   }
1379   \tl_if_in:NnT \l_um_tmpa_tl {/} {
1380     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
1381   }
1382   \tl_if_empty:NT \l_um_tmpc_tl { \tl_set_eq:NN \l_um_tmpc_tl \l_um_tmpa_tl }
1383   \seq_if_in:NVTF \g_um_mathstyles_seq \l_um_tmpa_tl {
1384     \prg_return_true:
1385   }{
1386     \prg_return_false:
1387   }
1388 }
1389 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
1390   \tl_set:Nn \l_um_tmpa_tl {#1}
1391   \tl_if_single:nTF {#2}
1392     { \tl_set:Nn \l_um_tmpc_tl {#2} }
1393     { \exp_args:NNc \tl_set:Nn \l_um_tmpc_tl {math#2} }
1394 }
1395 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
1396   \tl_set:Nn \l_um_tmpa_tl {#1}
1397   \tl_set:Nn \l_um_tmpb_tl {#2}
1398 }

```

Pretty basic comma separated range processing. Donald Arseneau's `selectp` package has a cleverer technique.

```

\um_if_char_spec:nNNT #1 : Unicode character slot
                      #2 : control sequence (character macro)
                      #3 : control sequence (math class)
                      #4 : code to execute

```


This macro expands to #4 if any of its arguments are contained in \l_um_char_range_seq. This list can contain either character ranges (for checking with #1) or control sequences. These latter can either be the command name of a specific character, or the math type of one (e.g., \mathbin).

Character ranges are passed to \um@parse@range, which accepts input in the form shown in table 11.

Table 11: Ranges accepted by \um@parse@range.

Input	Range
x	$r = x$
$x-$	$r \geq x$
$-y$	$r \leq y$
$x-y$	$x \leq r \leq y$

We have three tests, performed sequentially in order of execution time. Any test finding a match jumps directly to the end.

```

1399 \cs_new:Nn \um_if_char_spec:nNNT
1400 {
1401
1402   % math class:
1403   \seq_if_in:NnT \l_um_mclass_range_seq {#3}
1404     { \use_none_delimit_by_q_nil:w }
1405
1406   % command name:
1407   \seq_if_in:NnT \l_um_cmd_range_seq {#2}
1408     { \use_none_delimit_by_q_nil:w }
1409
1410   % character slot:
1411   \seq_map_inline:Nn \l_um_char_range_seq
1412     {
1413       \um_int_if_slot_in_range:nnT {#1} {##1}
1414       { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
1415     }
1416
1417   % this executes if no match was found:
1418   \use_none:nnn
1419   \q_nil
1420   \use:n
1421   {
1422     \clist_put_right:Nx \l_um_char_num_range_clist { \int_eval:n {#1} }
1423     #4
1424   }
1425 }
```

\um_int_if_slot_in_range:nnT A ‘numrange’ is like -2,5-8,12,17- (can be unsorted).

Four cases, four argument types:

```

% input      #2      #3      #4
% "1 "      [ 1] - [qn] - [ ] qs
```

```

% "1- " [ 1] - [ ] - [qn-] qs
% " -3" [ ] - [ 3] - [qn-] qs
% "1-3" [ 1] - [ 3] - [qn-] qs

1426 \cs_new:Nn \um_int_if_slot_in_range:nnT
1427 { \um_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
1428 \cs_set:Npn \um_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
1429 {
1430   \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
1431   {
1432     \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
1433     {
1434       \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
1435       {
1436         \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
1437         } } }
1438   }

```

11.4 Resolving Greek symbol name control sequences

`\um_resolve_greek:` This macro defines `\Alpha...``\omega` as their corresponding Unicode (mathematical italic) character. Remember that the mapping to upright or italic happens with the mathcode definitions, whereas these macros just stand for the literal Unicode characters.

```

1439 \AtBeginDocument{\um_resolve_greek:}
1440 \cs_new:Npn \um_resolve_greek: {
1441   \clist_map_inline:nn {
1442     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1443     alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
1444     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1445     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,    chi,psi,omega,
1446     varTheta,
1447     varsigma,vartheta,varkappa,varrho,varpi
1448   }{
1449     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
1450   }
1451   \tl_set:Nn \epsilon {
1452     \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
1453   }
1454   \tl_set:Nn \phi {
1455     \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
1456   }
1457   \tl_set:Nn \varepsilon {
1458     \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
1459   }
1460   \tl_set:Nn \varphi {
1461     \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
1462   }
1463 }

```

12 Maths alphabets mapping definitions

Algorithm for setting alphabet fonts. By default, when `range` is empty, we are in *implicit* mode. If `range` contains the name of the math alphabet, we are in *explicit* mode and do things slightly differently.

Implicit mode:

- Try and set all of the alphabet shapes.
- Check for the first glyph of each alphabet to detect if the font supports each alphabet shape.
- For alphabets that do exist, overwrite whatever's already there.
- For alphabets that are not supported, *do nothing*. (This includes leaving the old alphabet definition in place.)

Explicit mode:

- Only set the alphabets specified.
- Check for the first glyph of the alphabet to detect if the font contains the alphabet shape in the Unicode math plane.
- For Unicode math alphabets, overwrite whatever's already there.
- Otherwise, use the ASCII letters instead.

12.1 Initialising math styles

`\um_new_mathstyle:N` This function defines a new command like `\mathfrak`.

```
1464 \cs_new:Nn \um_new_mathstyle:N {
1465   \um_prepare_mathstyle:f {\exp_after:wN \use_none:nnnnn \token_to_str:N #1}
1466   \seq_put_right:Nn \g_um_mathstyles_seq {#1}
1467 }
```

`\g_um_default_mathalph_seq` This sequence stores the alphabets in each math style.

```
1468 \seq_new:N \g_um_default_mathalph_seq
```

`\g_um_mathstyles_seq` This is every math style known to unicode-math.

```
1469 \seq_new:N \g_um_mathstyles_seq
```

```
1470 \AtEndOfPackage{
1471 \clist_map_inline:nn {
1472   {\mathup    } {latin, Latin, greek, Greek, num, misc} {\mathup    } ,
1473   {\mathit    } {latin, Latin, greek, Greek, misc}      {\mathit    } ,
1474   {\mathbb    } {latin, Latin, num, misc}                {\mathbb    } ,
1475   {\mathbbbit } {misc}                                   {\mathbbbit } ,
1476   {\mathscr   } {latin, Latin}                           {\mathscr   } ,
1477   {\mathcal   } {Latin}                                   {\mathcal   } ,
1478   {\mathbfcal } {Latin}                                   {\mathbfscr } ,
1479   {\mathfrak  } {latin, Latin}                           {\mathfrak  } ,
```

```

1480 {\mathtt } {latin,Latin,num} {\mathtt } ,
1481 {\mathsfup } {latin,Latin,num} {\mathsfup } ,
1482 {\mathsfit } {latin,Latin} {\mathsfit } ,
1483 {\mathbfup } {latin,Latin,greek,Greek,num,misc} {\mathbfup } ,
1484 {\mathbfit } {latin,Latin,greek,Greek,misc} {\mathbfit } ,
1485 {\mathbfscr } {latin,Latin} {\mathbfscr } ,
1486 {\mathbffrak } {latin,Latin} {\mathbffrak } ,
1487 {\mathbfsfup } {latin,Latin,greek,Greek,num,misc} {\mathbfsfup } ,
1488 {\mathbfsfit } {latin,Latin,greek,Greek,misc} {\mathbfsfit }
1489 }{
1490 \seq_put_right:Nn \g_um_default_mathalph_seq {#1}
1491 \exp_after:wN \um_new_mathstyle:N \use_i:nnn #1
1492 }

```

These are ‘false’ mathstyles that inherit other definitions:

```

1493 \um_new_mathstyle:N \mathsf
1494 \um_new_mathstyle:N \mathbf
1495 \um_new_mathstyle:N \mathbfsf
1496 }

```

12.2 Defining the math style macros

We call the different shapes that a math alphabet can be a ‘math style’. Note that different alphabets can exist within the same math style. E.g., we call ‘bold’ the math style `bf` and within it there are upper and lower case Greek and Roman alphabets and Arabic numerals.

`\um_prepare_mathstyle:n` #1 : math style name (e.g., `it` or `bb`)

Define the high level math alphabet macros (`\mathit`, etc.) in terms of unicode-math definitions. Use `\bgroup`/`\egroup` so s’scripts scan the whole thing.

The flag `\l_um_mathstyle_tl` is for other applications to query the current math style.

```

1497 \cs_new:Nn \um_prepare_mathstyle:n {
1498   \um_init_alphabet:x {#1}
1499   \cs_set:cpn {_um_math#1_aux:n} ##1 {
1500     \use:c {um_switchto_math#1:} ##1 \egroup
1501   }
1502   \cs_set_protected:cpn {math#1} {
1503     \exp_not:n{
1504       \bgroup
1505       \mode_if_math:F
1506       {
1507         \egroup\expandafter
1508         \non@alpherr\expandafter{\csname math#1\endcsname\space}
1509       }
1510       \tl_set:Nn \l_um_mathstyle_tl {#1}
1511     }
1512     \exp_not:c {_um_math#1_aux:n}
1513   }
1514 }

```

```

1515 \tl_new:N \l_um_mathstyle_tl
1516 \cs_generate_variant:Nn \um_prepare_mathstyle:n {f}

```

`\um_init_alphabet:n` **#1** : math alphabet name (e.g., it or bb)

This macro initialises the macros used to set up a math alphabet. First used with the math alphabet macro is first defined, but then used later when redefining a particular maths alphabet.

```

1517 \cs_set:Npn \um_init_alphabet:n #1 {
1518   \um_log:nx {alph-initialise} {#1}
1519   \cs_set_eq:cN {um_switchto_math#1:} \prg_do_nothing:
1520 }
1521 \cs_generate_variant:Nn \um_init_alphabet:n {x}

```

Variants (cannot use `\cs_generate_variant:Nn` because the base function is defined dynamically.)

```

1522 \cs_new:Npn \um_maybe_init_alphabet:V {
1523   \exp_args:NV \um_maybe_init_alphabet:n
1524 }

```

12.3 Defining the math alphabets per style

Variables:

```

1525 \seq_new:N \l_um_missing_alph_seq

```

`\um_setup_alphabets:` This function is called within `\setmathfont` to configure the mapping between characters inside math styles.

```

1526 \cs_new:Npn \um_setup_alphabets: {

```

If `range=` has been used to configure styles, those choices will be in `\l_um_mathalph_seq`. If not, set up the styles implicitly:

```

1527   \seq_if_empty:NTF \l_um_mathalph_seq {
1528     \um_log:n {setup-implicit}
1529     \seq_set_eq:NN \l_um_mathalph_seq \g_um_default_mathalph_seq
1530     \bool_set_true:N \l_um_implicit_alph_bool
1531     \um_maybe_init_alphabet:n {sf}
1532     \um_maybe_init_alphabet:n {bf}
1533     \um_maybe_init_alphabet:n {bfsf}
1534   }

```

If `range=` has been used then we're in explicit mode:

```

1535   {
1536     \um_log:n {setup-explicit}
1537     \bool_set_false:N \l_um_implicit_alph_bool
1538     \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
1539     \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
1540   }

```

Now perform the mapping:

```

1541   \seq_map_inline:Nn \l_um_mathalph_seq {
1542     \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
1543     \tl_set:No \l_um_tmpb_tl { \use_ii:nnn ##1 }

```

```

1544 \tl_set:No \l_um_remap_style_tl { \use_iii:nnn ##1 }
1545 \tl_set:Nx \l_um_remap_style_tl {
1546   \exp_after:wN \exp_after:wN \exp_after:wN \use_none:nnnnn
1547   \exp_after:wN \token_to_str:N \l_um_remap_style_tl
1548 }
1549 \tl_if_empty:NT \l_um_tmpb_tl {
1550   \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
1551   \tl_set:Nn \l_um_tmpb_tl { latin, Latin, greek, Greek, num, misc }
1552 }
1553 \um_setup_math_alphabet:VVV
1554   \l_um_tmpa_tl \l_um_tmpb_tl \l_um_remap_style_tl
1555 }
1556 \seq_if_empty:NF \l_um_missing_alph_seq { \um_log:n { missing-alphabets } }
1557 }

```

`\um_setup_math_alphabet:Nnn` **#1** : Math font style command (e.g., `\mathbb`)
#2 : Math alphabets, comma separated of {latin, Latin, greek, Greek, num}
#3 : Name of the output math style (usually same as input `bb`)

```

1558 \cs_new:Nn \um_setup_math_alphabet:Nnn {
1559   \tl_set:Nx \l_um_style_tl {
1560     \exp_after:wN \use_none:nnnnn \token_to_str:N #1
1561   }

```

First check that at least one of the alphabets for the font shape is defined...

```

1562 \clist_map_inline:nn {#2} {
1563   \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {##1} }
1564   \cs_if_exist:cT {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {
1565     \str_if_eq:xxTF {\l_um_tmpa_tl}{misc} {
1566       \um_maybe_init_alphabet:V \l_um_style_tl
1567       \clist_map_break:
1568     }{
1569       \um_glyph_if_exist:cT { \um_to_usv:nn {#3}{\l_um_tmpa_tl} }{
1570         \um_maybe_init_alphabet:V \l_um_style_tl
1571         \clist_map_break:
1572       }
1573     }
1574   }
1575 }

```

...and then loop through them defining the individual ranges:

```

1576 \clist_map_inline:nn {#2} {
1577   \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {##1} }
1578   \cs_if_exist:cT {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {
1579     \str_if_eq:xxTF {\l_um_tmpa_tl}{misc} {
1580       \um_log:nx {setup-alph} {math \l_um_style_tl~(\l_um_tmpa_tl)}
1581       \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {#3}
1582     }{
1583       \um_glyph_if_exist:cTF { \um_to_usv:nn {#3}{\l_um_tmpa_tl} } {
1584         \um_log:nx {setup-alph} {math \l_um_style_tl~(\l_um_tmpa_tl)}
1585         \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {#3}
1586       }{

```

```

1587         \bool_if:NTF \l_um_implicit_alph_bool {
1588             \seq_put_right:Nx \l_um_missing_alph_seq {
1589                 \@backslashchar math \l_um_style_tl \space
1590                 (\tl_use:c{c_um_math_alphabet_name_ \l_um_tmpa_tl _tl})
1591             }
1592         }{
1593             \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {up}
1594         }
1595     }
1596 }
1597 }
1598 }
1599 }
1600 \cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {VVV}

```

12.4 Mapping ‘naked’ math characters

Before we show the definitions of the alphabet mappings using the functions `\um_config_\l_um_style_tl_##1:n`, we first want to define some functions to be used inside them to actually perform the character mapping.

12.4.1 Functions

`\um_map_char_single:nn` Wrapper for `\um_map_char_noparse:nn` or `\um_map_char_parse:nn` depending on the context. Cannot use `\cs_generate_variant:Nn` because the base function is defined dynamically.

```

1601 \cs_new:Npn \um_map_char_single:cc { \exp_args:Ncc \um_map_char_single:nn }

```

`\um_map_char_noparse:nn`

```

\um_map_char_parse:nn
1602 \cs_new:Nn \um_map_char_noparse:nn {
1603     \um_set_mathcode:nnnn {#1}{\mathalpha}{\um_symfont_tl}{#2}
1604 }
1605 \cs_new:Nn \um_map_char_parse:nn {
1606     \um_if_char_spec:nNT {#1} {\@nil} {\mathalpha} {
1607         \um_map_char_noparse:nn {#1}{#2}
1608     }
1609 }

```

`\um_map_single:nnn` #1 : char name (‘dotlessi’)
#2 : from alphabet(s)
#3 : to alphabet

```

1610 \cs_new:Nn \um_map_char_single:nnn {
1611     \um_map_char_single:cc { \um_to_usv:nn {#1}{#3} }
1612     { \um_to_usv:nn {#2}{#3} }
1613 }
1614 \cs_set:Npn \um_map_single:nnn #1#2#3 {
1615     \cs_if_exist:cT { \um_to_usv:nn {#3} {#1} }
1616     {
1617         \clist_map_inline:nn {#2} {

```

```

1618         \um_map_char_single:nnn {##1} {#3} {#1}
1619     }
1620 }
1621 }

```

\um_map_chars_range:nnnn #1 : Number of chars (26)
#2 : From style, one or more (it)
#3 : To style (up)
#4 : Alphabet name (Latin)
First the function with numbers:

```

1622 \cs_set:Npn \um_map_chars_range:nnn #1#2#3 {
1623   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {
1624     \um_map_char_single:nn {#2+##1}{#3+##1}
1625   }
1626 }
1627 \cs_generate_variant:Nn \um_map_chars_range:nnn {ncc}

```

And the wrapper with names:

```

1628 \cs_new:Nn \um_map_chars_range:nnnn {
1629   \um_map_chars_range:ncc {#1} { \um_to_usv:nn {#2}{#4} }
1630   { \um_to_usv:nn {#3}{#4} }
1631 }

```

12.4.2 Functions for alphabets

```

1632 \cs_new:Nn \um_map_chars_Latin:nn {
1633   \clist_map_inline:nn {#1} {
1634     \um_map_chars_range:nnnn {26} {##1} {#2} {Latin}
1635   }
1636 }

1637 \cs_new:Nn \um_map_chars_latin:nn {
1638   \clist_map_inline:nn {#1} {
1639     \um_map_chars_range:nnnn {26} {##1} {#2} {latin}
1640   }
1641 }

1642 \cs_new:Nn \um_map_chars_greek:nn {
1643   \clist_map_inline:nn {#1} {
1644     \um_map_chars_range:nnnn {25} {##1} {#2} {greek}
1645     \um_map_char_single:nnn {##1} {#2} {\varepsilon}
1646     \um_map_char_single:nnn {##1} {#2} {\vartheta}
1647     \um_map_char_single:nnn {##1} {#2} {\varkappa}
1648     \um_map_char_single:nnn {##1} {#2} {\varphi}
1649     \um_map_char_single:nnn {##1} {#2} {\varrho}
1650     \um_map_char_single:nnn {##1} {#2} {\varpi}
1651   }
1652 }

1653 \cs_new:Nn \um_map_chars_Greek:nn {
1654   \clist_map_inline:nn {#1} {
1655     \um_map_chars_range:nnnn {25} {##1} {#2} {Greek}

```



```

1656 \um_map_char_single:nnn {##1} {#2} {varTheta}
1657 }
1658 }
1659 \cs_new:Nn \um_map_chars_numbers:nn {
1660 \um_map_chars_range:nnnn {10} {#1} {#2} {num}
1661 }

```

12.5 Mapping chars inside a math style

12.5.1 Functions for setting up the maths alphabets

`\um_set_mathalphabet_char:Nnn` This is a wrapper for either `\um_mathmap_noparse:Nnn` or `\um_mathmap_parse:Nnn`, depending on the context. Cannot use `\cs_generate_variant:Nn` because the base function is defined dynamically.

```

1662 \cs_new:Npn \um_set_mathalphabet_char:Ncc {
1663 \exp_args:NNcc \um_set_mathalphabet_char:Nnn
1664 }

```

`\um_mathmap_noparse:Nnn` **#1** : Maths alphabet, *e.g.*, `\mathbb`
#2 : Input slot(s), *e.g.*, the slot for ‘A’ (comma separated)
#3 : Output slot, *e.g.*, the slot for ‘A’
Adds `\um_set_mathcode:nnnn` declarations to the specified maths alphabet’s definition.

```

1665 \cs_new:Nn \um_mathmap_noparse:Nnn {
1666 \clist_map_inline:nn {#2} {
1667 \tl_put_right:cx {um_switchto_\cs_to_str:N #1:} {
1668 \um_set_mathcode:nnnn{##1}{\mathalpha}{\um_symfont_tl}{#3}
1669 }
1670 }
1671 }

```

`\um_mathmap_parse:Nnn` **#1** : Maths alphabet, *e.g.*, `\mathbb`
#2 : Input slot(s), *e.g.*, the slot for ‘A’ (comma separated)
#3 : Output slot, *e.g.*, the slot for ‘A’
When `\um_if_char_spec:nNNT` is executed, it populates the `\l_um_char_num_range_clist` macro with slot numbers corresponding to the specified range. This range is used to conditionally add `\um_set_mathcode:nnnn` declarations to the maths alphabet definition.

```

1672 \cs_new:Nn \um_mathmap_parse:Nnn {
1673 \clist_if_in:NnT \l_um_char_num_range_clist {#3} {
1674 \um_mathmap_noparse:Nnn {#1}{#2}{#3}
1675 }
1676 }

```

`\um_set_mathalphabet_char:Nnnn` **#1** : math style command
#2 : input math alphabet name
#3 : output math alphabet name
#4 : char name to map

```

1677 \cs_new:Npn \um_set_mathalphabet_char:Nnnn #1#2#3#4 {

```

```

1678 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {#2} {#4} }
1679 { \um_to_usv:nn {#3} {#4} }
1680 }

```

`\um_set_mathalph_range:nNnn` #1 : Number of iterations
#2 : Maths alphabet
#3 : Starting input char (single)
#4 : Starting output char
Loops through character ranges setting `\mathcode`. First the version that uses numbers:

```

1681 \cs_new:Npn \um_set_mathalph_range:nNnn #1#2#3#4 {
1682 \prg_stepwise_inline:nnnn {0}{1}{#1-1}
1683 { \um_set_mathalphabet_char:Nnn {#2} { ##1 + #3 } { ##1 + #4 } }
1684 }
1685 \cs_generate_variant:Nn \um_set_mathalph_range:nNnn {nNcc}

```

Then the wrapper version that uses names:

```

1686 \cs_new:Npn \um_set_mathalph_range:nNnnn #1#2#3#4#5 {
1687 \um_set_mathalph_range:nNcc {#1} #2 { \um_to_usv:nn {#3} {#5} }
1688 { \um_to_usv:nn {#4} {#5} }
1689 }

```

12.5.2 Individual mapping functions for different alphabets

```

1690 \cs_new:Npn \um_set_mathalphabet_pos:Nnnn #1#2#3#4 {
1691 \cs_if_exist:cT { \um_to_usv:nn {#4}{#2} } {
1692 \clist_map_inline:nn {#3}
1693 { \um_set_mathalphabet_char:Nnnn #1 {##1} {#4} {#2} }
1694 }
1695 }

1696 \cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
1697 \clist_map_inline:nn {#2}
1698 { \um_set_mathalph_range:nNnnn {10} #1 {##1} {#3} {num} }
1699 }

1700 \cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
1701 \clist_map_inline:nn {#2}
1702 { \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {Latin} }
1703 }

1704 \cs_new:Nn \um_set_mathalphabet_latin:Nnn {
1705 \clist_map_inline:nn {#2} {
1706 \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {latin}
1707 \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {h}
1708 }
1709 }

1710 \cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
1711 \clist_map_inline:nn {#2} {
1712 \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {Greek}
1713 \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varTheta}
1714 }

```

```

1715 }
1716 \cs_new:Nn \um_set_mathalphabet_greek:Nnn {
1717   \clist_map_inline:nn {#2} {
1718     \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {greek}
1719     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varepsilon}
1720     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\vartheta}
1721     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varkappa}
1722     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varphi}
1723     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varrho}
1724     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {\varpi}
1725   }
1726 }

```

12.6 Alphabets

12.6.1 Upright: `\mathup`

```

1727 \cs_new:Nn \um_config_up_num:n {
1728   \um_map_chars_numbers:nn {up}{#1}
1729   \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}
1730 }
1731 \cs_new:Nn \um_config_up_Latin:n
1732 {
1733   \bool_if:NTF \g_um_literal_bool { \um_map_chars_Latin:nn {up} {#1} }
1734   {
1735     \bool_if:NT \g_um_upLatin_bool { \um_map_chars_Latin:nn {up,it} {#1} }
1736   }
1737   \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
1738 }
1739 \cs_new:Nn \um_config_up_latin:n {
1740   \bool_if:NTF \g_um_literal_bool { \um_map_chars_latin:nn {up} {#1} }
1741   {
1742     \bool_if:NT \g_um_uplatin_bool {
1743       \um_map_chars_latin:nn {up,it} {#1}
1744       \um_map_single:nnn {h} {up,it} {#1}
1745       \um_map_single:nnn {dotlessi} {up,it} {#1}
1746       \um_map_single:nnn {dotlessj} {up,it} {#1}
1747     }
1748   }
1749   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
1750 }
1751 \cs_new:Nn \um_config_up_Greek:n {
1752   \bool_if:NTF \g_um_literal_bool { \um_map_chars_Greek:nn {up}{#1} }
1753   {
1754     \bool_if:NT \g_um_upGreek_bool { \um_map_chars_Greek:nn {up,it}{#1} }
1755   }
1756   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
1757 }
1758 \cs_new:Nn \um_config_up_greek:n {
1759   \bool_if:NTF \g_um_literal_bool { \um_map_chars_greek:nn {up} {#1} }
1760   {

```

```

1761 \bool_if:NT \g_um_upgreek_bool {
1762   \um_map_chars_greek:nn {up,it} {#1}
1763 }
1764 }
1765 \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
1766 }
1767 \cs_new:Nn \um_config_up_misc:n {
1768   \bool_if:NTF \g_um_literal_Nabla_bool {
1769     \um_map_single:nnn {Nabla}{up}{up}
1770   }{
1771     \bool_if:NT \g_um_upNabla_bool {
1772       \um_map_single:nnn {Nabla}{up,it}{up}
1773     }
1774   }
1775   \bool_if:NTF \g_um_literal_partial_bool {
1776     \um_map_single:nnn {partial}{up}{up}
1777   }{
1778     \bool_if:NT \g_um_uppartial_bool {
1779       \um_map_single:nnn {partial}{up,it}{up}
1780     }
1781   }
1782   \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it} {#1}
1783   \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it} {#1}
1784   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessi} {up,it} {#1}
1785   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessj} {up,it} {#1}
1786 }

```

12.6.2 Italic: \mathit

```

1787 \cs_new:Nn \um_config_it_Latin:n {
1788   \bool_if:NTF \g_um_literal_bool { \um_map_chars_Latin:nn {it} {#1} }
1789   {
1790     \bool_if:NF \g_um_upLatin_bool { \um_map_chars_Latin:nn {up,it} {#1} }
1791   }
1792   \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
1793 }
1794 \cs_new:Nn \um_config_it_latin:n {
1795   \bool_if:NTF \g_um_literal_bool {
1796     \um_map_chars_latin:nn {it} {#1}
1797     \um_map_single:nnn {h}{it}{#1}
1798   }{
1799     \bool_if:NF \g_um_uplatin_bool {
1800       \um_map_chars_latin:nn {up,it} {#1}
1801       \um_map_single:nnn {h}{up,it}{#1}
1802       \um_map_single:nnn {dotlessi}{up,it}{#1}
1803       \um_map_single:nnn {dotlessj}{up,it}{#1}
1804     }
1805   }
1806   \um_set_mathalphabet_latin:Nnn \mathit {up,it} {#1}
1807   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessi} {up,it} {#1}
1808   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessj} {up,it} {#1}
1809 }

```

```

1810 \cs_new:Nn \um_config_it_Greek:n {
1811   \bool_if:NTF \g_um_literal_bool { \um_map_chars_Greek:nn {it}{#1}
1812   }{
1813     \bool_if:NF \g_um_upGreek_bool { \um_map_chars_Greek:nn {up,it}{#1} }
1814   }
1815   \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
1816 }
1817 \cs_new:Nn \um_config_it_greek:n {
1818   \bool_if:NTF \g_um_literal_bool { \um_map_chars_greek:nn {it} {#1} }
1819   {
1820     \bool_if:NF \g_um_upgreek_bool { \um_map_chars_greek:nn {it,up} {#1} }
1821   }
1822   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
1823 }
1824 \cs_new:Nn \um_config_it_misc:n {
1825   \bool_if:NTF \g_um_literal_Nabla_bool {
1826     \um_map_single:nnn {Nabla}{it}{it}
1827   }{
1828     \bool_if:NF \g_um_upNabla_bool {
1829       \um_map_single:nnn {Nabla}{up,it}{it}
1830     }
1831   }
1832   \bool_if:NTF \g_um_literal_partial_bool {
1833     \um_map_single:nnn {partial}{it}{it}
1834   }{
1835     \bool_if:NF \g_um_uppartial_bool {
1836       \um_map_single:nnn {partial}{up,it}{it}
1837     }
1838   }
1839   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
1840   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{#1}
1841 }

```

12.6.3 Blackboard or double-struck: `\mathbb` and `\mathbbbit`

```

1842 \cs_new:Nn \um_config_bb_latin:n {
1843   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
1844 }
1845 \cs_new:Nn \um_config_bb_Latin:n {
1846   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
1847   \um_set_mathalphabet_pos:Nnnn \mathbb {C} {up,it} {#1}
1848   \um_set_mathalphabet_pos:Nnnn \mathbb {H} {up,it} {#1}
1849   \um_set_mathalphabet_pos:Nnnn \mathbb {N} {up,it} {#1}
1850   \um_set_mathalphabet_pos:Nnnn \mathbb {P} {up,it} {#1}
1851   \um_set_mathalphabet_pos:Nnnn \mathbb {Q} {up,it} {#1}
1852   \um_set_mathalphabet_pos:Nnnn \mathbb {R} {up,it} {#1}
1853   \um_set_mathalphabet_pos:Nnnn \mathbb {Z} {up,it} {#1}
1854 }
1855 \cs_new:Nn \um_config_bb_num:n {
1856   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
1857 }
1858 \cs_new:Nn \um_config_bb_misc:n {

```

```

1859 \um_set_mathalphabet_pos:Nnnn \mathbb      {\Pi} {up,it} {#1}
1860 \um_set_mathalphabet_pos:Nnnn \mathbb      {\pi} {up,it} {#1}
1861 \um_set_mathalphabet_pos:Nnnn \mathbb      {\Gamma} {up,it} {#1}
1862 \um_set_mathalphabet_pos:Nnnn \mathbb      {\gamma} {up,it} {#1}
1863 \um_set_mathalphabet_pos:Nnnn \mathbb      {\sum} {up} {#1}
1864 }
1865 \cs_new:Nn \um_config_bbit_misc:n {
1866 \um_set_mathalphabet_pos:Nnnn \mathbbbit {D} {up,it} {#1}
1867 \um_set_mathalphabet_pos:Nnnn \mathbbbit {d} {up,it} {#1}
1868 \um_set_mathalphabet_pos:Nnnn \mathbbbit {e} {up,it} {#1}
1869 \um_set_mathalphabet_pos:Nnnn \mathbbbit {i} {up,it} {#1}
1870 \um_set_mathalphabet_pos:Nnnn \mathbbbit {j} {up,it} {#1}
1871 }

```

12.6.4 Script and caligraphic: `\mathscr` and `\mathcal`

```

1872 \cs_new:Nn \um_config_scr_Latin:n {
1873 \um_set_mathalphabet_Latin:Nnn \mathscr    {up,it}{#1}
1874 \um_set_mathalphabet_pos:Nnnn \mathscr    {B}{up,it}{#1}
1875 \um_set_mathalphabet_pos:Nnnn \mathscr    {E}{up,it}{#1}
1876 \um_set_mathalphabet_pos:Nnnn \mathscr    {F}{up,it}{#1}
1877 \um_set_mathalphabet_pos:Nnnn \mathscr    {H}{up,it}{#1}
1878 \um_set_mathalphabet_pos:Nnnn \mathscr    {I}{up,it}{#1}
1879 \um_set_mathalphabet_pos:Nnnn \mathscr    {L}{up,it}{#1}
1880 \um_set_mathalphabet_pos:Nnnn \mathscr    {M}{up,it}{#1}
1881 \um_set_mathalphabet_pos:Nnnn \mathscr    {R}{up,it}{#1}
1882 }
1883 \cs_new:Nn \um_config_scr_latin:n {
1884 \um_set_mathalphabet_latin:Nnn \mathscr    {up,it}{#1}
1885 \um_set_mathalphabet_pos:Nnnn \mathscr    {e}{up,it}{#1}
1886 \um_set_mathalphabet_pos:Nnnn \mathscr    {g}{up,it}{#1}
1887 \um_set_mathalphabet_pos:Nnnn \mathscr    {o}{up,it}{#1}
1888 }

```

These are by default synonyms for the above, but with the STIX fonts we want to use the alternate alphabet.

```

1889 \cs_new:Nn \um_config_cal_Latin:n {
1890 \um_set_mathalphabet_Latin:Nnn \mathcal    {up,it}{#1}
1891 \um_set_mathalphabet_pos:Nnnn \mathcal    {B}{up,it}{#1}
1892 \um_set_mathalphabet_pos:Nnnn \mathcal    {E}{up,it}{#1}
1893 \um_set_mathalphabet_pos:Nnnn \mathcal    {F}{up,it}{#1}
1894 \um_set_mathalphabet_pos:Nnnn \mathcal    {H}{up,it}{#1}
1895 \um_set_mathalphabet_pos:Nnnn \mathcal    {I}{up,it}{#1}
1896 \um_set_mathalphabet_pos:Nnnn \mathcal    {L}{up,it}{#1}
1897 \um_set_mathalphabet_pos:Nnnn \mathcal    {M}{up,it}{#1}
1898 \um_set_mathalphabet_pos:Nnnn \mathcal    {R}{up,it}{#1}
1899 }

```

12.6.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1900 \cs_new:Nn \um_config_frak_Latin:n {
1901 \um_set_mathalphabet_Latin:Nnn \mathfrak    {up,it}{#1}
1902 \um_set_mathalphabet_pos:Nnnn \mathfrak    {C}{up,it}{#1}

```

```

1903 \um_set_mathalphabet_pos:Nnnn \mathfrak {H}{up,it}{#1}
1904 \um_set_mathalphabet_pos:Nnnn \mathfrak {I}{up,it}{#1}
1905 \um_set_mathalphabet_pos:Nnnn \mathfrak {R}{up,it}{#1}
1906 \um_set_mathalphabet_pos:Nnnn \mathfrak {Z}{up,it}{#1}
1907 }
1908 \cs_new:Nn \um_config_frak_latin:n {
1909 \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{#1}
1910 }

```

12.6.6 Sans serif upright: \mathsfup

```

1911 \cs_new:Nn \um_config_sfup_num:n {
1912 \um_set_mathalphabet_numbers:Nnn \mathsf {up}{#1}
1913 \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
1914 }
1915 \cs_new:Nn \um_config_sfup_Latin:n {
1916 \bool_if:NTF \g_um_sfliteral_bool {
1917 \um_map_chars_Latin:nn {sfup} {#1}
1918 \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
1919 }{
1920 \bool_if:NT \g_um_upsans_bool {
1921 \um_map_chars_Latin:nn {sfup,sfit} {#1}
1922 \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1923 }
1924 }
1925 \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}
1926 }
1927 \cs_new:Nn \um_config_sfup_latin:n {
1928 \bool_if:NTF \g_um_sfliteral_bool {
1929 \um_map_chars_latin:nn {sfup} {#1}
1930 \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
1931 }{
1932 \bool_if:NT \g_um_upsans_bool {
1933 \um_map_chars_latin:nn {sfup,sfit} {#1}
1934 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1935 }
1936 }
1937 \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
1938 }

```

12.6.7 Sans serif italic: \mathsfit

```

1939 \cs_new:Nn \um_config_sfit_Latin:n {
1940 \bool_if:NTF \g_um_sfliteral_bool {
1941 \um_map_chars_Latin:nn {sfit} {#1}
1942 \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
1943 }{
1944 \bool_if:NF \g_um_upsans_bool {
1945 \um_map_chars_Latin:nn {sfup,sfit} {#1}
1946 \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1947 }
1948 }
1949 \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{#1}

```

```

1950 }
1951 \cs_new:Nn \um_config_sfit_latin:n {
1952   \bool_if:NTF \g_um_sfliteral_bool {
1953     \um_map_chars_latin:nn {sfit} {#1}
1954     \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
1955   }{
1956     \bool_if:NF \g_um_upsans_bool {
1957       \um_map_chars_latin:nn {sfup,sfit} {#1}
1958       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1959     }
1960   }
1961   \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1962 }

```

12.6.8 Typewriter or monospaced: `\mathtt`

```

1963 \cs_new:Nn \um_config_tt_num:n {
1964   \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
1965 }
1966 \cs_new:Nn \um_config_tt_Latin:n {
1967   \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
1968 }
1969 \cs_new:Nn \um_config_tt_latin:n {
1970   \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
1971 }

```

12.6.9 Bold Italic: `\mathbfit`

```

1972 \cs_new:Nn \um_config_bfit_Latin:n {
1973   \bool_if:NF \g_um_bfupLatin_bool {
1974     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1975   }
1976   \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
1977   \bool_if:NTF \g_um_bfliteral_bool {
1978     \um_map_chars_Latin:nn {bfit} {#1}
1979     \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
1980   }{
1981     \bool_if:NF \g_um_bfupLatin_bool {
1982       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1983       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1984     }
1985   }
1986 }
1987 \cs_new:Nn \um_config_bfit_latin:n {
1988   \bool_if:NF \g_um_bfuplatin_bool {
1989     \um_map_chars_latin:nn {bfup,bfit} {#1}
1990   }
1991   \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
1992   \bool_if:NTF \g_um_bfliteral_bool {
1993     \um_map_chars_latin:nn {bfit} {#1}
1994     \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
1995   }{
1996     \bool_if:NF \g_um_bfuplatin_bool {

```



```

1997     \um_map_chars_latin:nn {bfup,bfit} {#1}
1998     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1999   }
2000 }
2001 }
2002 \cs_new:Nn \um_config_bfit_Greek:n {
2003   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
2004   \bool_if:NTF \g_um_bfliteral_bool {
2005     \um_map_chars_Greek:nn {bfit}{#1}
2006     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
2007   }{
2008     \bool_if:NF \g_um_bfupGreek_bool {
2009       \um_map_chars_Greek:nn {bfup,bfit}{#1}
2010       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
2011     }
2012   }
2013 }
2014 \cs_new:Nn \um_config_bfit_greek:n {
2015   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
2016   \bool_if:NTF \g_um_bfliteral_bool {
2017     \um_map_chars_greek:nn {bfit} {#1}
2018     \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
2019   }{
2020     \bool_if:NF \g_um_bfupgreek_bool {
2021       \um_map_chars_greek:nn {bfit,bfup} {#1}
2022       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
2023     }
2024   }
2025 }
2026 \cs_new:Nn \um_config_bfit_misc:n {
2027   \bool_if:NTF \g_um_literal_Nabla_bool {
2028     \um_map_single:nnn {Nabla}{bfit}{#1}
2029   }{
2030     \bool_if:NF \g_um_upNabla_bool {
2031       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
2032     }
2033   }
2034   \bool_if:NTF \g_um_literal_partial_bool {
2035     \um_map_single:nnn {partial}{bfit}{#1}
2036   }{
2037     \bool_if:NF \g_um_uppartial_bool {
2038       \um_map_single:nnn {partial}{bfup,bfit}{#1}
2039     }
2040   }
2041   \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{#1}
2042   \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{#1}
2043   \bool_if:NTF \g_um_literal_partial_bool {
2044     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{#1}
2045   }{
2046     \bool_if:NF \g_um_uppartial_bool {
2047       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}

```

```

2048     }
2049   }
2050   \bool_if:NTF \g_um_literal_Nabla_bool {
2051     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla}   {it}{#1}
2052   }{
2053     \bool_if:NF \g_um_upNabla_bool {
2054       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla}   {up,it}{#1}
2055     }
2056   }
2057 }

```

12.6.10 Bold Upright: `\mathbfup`

```

2058 \cs_new:Nn \um_config_bfup_num:n {
2059   \um_set_mathalphabet_numbers:Nnn \mathbf   {up}{#1}
2060   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
2061 }
2062 \cs_new:Nn \um_config_bfup_Latin:n {
2063   \bool_if:NT \g_um_bfupLatin_bool {
2064     \um_map_chars_Latin:nn {bfup,bfit} {#1}
2065   }
2066   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
2067   \bool_if:NTF \g_um_bfliteral_bool {
2068     \um_map_chars_Latin:nn {bfup} {#1}
2069     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
2070   }{
2071     \bool_if:NT \g_um_bfupLatin_bool {
2072       \um_map_chars_Latin:nn {bfup,bfit} {#1}
2073       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
2074     }
2075   }
2076 }
2077 \cs_new:Nn \um_config_bfup_latin:n {
2078   \bool_if:NT \g_um_bfuplatin_bool {
2079     \um_map_chars_latin:nn {bfup,bfit} {#1}
2080   }
2081   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
2082   \bool_if:NTF \g_um_bfliteral_bool {
2083     \um_map_chars_latin:nn {bfup} {#1}
2084     \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
2085   }{
2086     \bool_if:NT \g_um_bfuplatin_bool {
2087       \um_map_chars_latin:nn {bfup,bfit} {#1}
2088       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
2089     }
2090   }
2091 }
2092 \cs_new:Nn \um_config_bfup_Greek:n {
2093   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
2094   \bool_if:NTF \g_um_bfliteral_bool {
2095     \um_map_chars_Greek:nn {bfup}{#1}
2096     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}

```

```

2097 }{
2098   \bool_if:NT \g_um_bfupGreek_bool {
2099     \um_map_chars_greek:nn {bfup,bfit}{#1}
2100     \um_set_mathalphabet_greek:Nnn \mathbf {up,it}{#1}
2101   }
2102 }
2103 }
2104 \cs_new:Nn \um_config_bfup_greek:n {
2105   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
2106   \bool_if:NTF \g_um_bfliteral_bool {
2107     \um_map_chars_greek:nn {bfup} {#1}
2108     \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
2109   }{
2110     \bool_if:NT \g_um_bfupgreek_bool {
2111       \um_map_chars_greek:nn {bfup,bfit} {#1}
2112       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
2113     }
2114   }
2115 }
2116 \cs_new:Nn \um_config_bfup_misc:n {
2117   \bool_if:NTF \g_um_literal_Nabla_bool {
2118     \um_map_single:nnn {Nabla}{bfup}{#1}
2119   }{
2120     \bool_if:NT \g_um_upNabla_bool {
2121       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
2122     }
2123   }
2124   \bool_if:NTF \g_um_literal_partial_bool {
2125     \um_map_single:nnn {partial}{bfup}{#1}
2126   }{
2127     \bool_if:NT \g_um_uppartial_bool {
2128       \um_map_single:nnn {partial}{bfup,bfit}{#1}
2129     }
2130   }
2131   \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{#1}
2132   \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{#1}
2133   \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{#1}
2134   \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{#1}
2135   \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{#1}
2136   \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{#1}
2137   \bool_if:NTF \g_um_literal_partial_bool {
2138     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{#1}
2139   }{
2140     \bool_if:NT \g_um_uppartial_bool {
2141       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
2142     }
2143   }
2144   \bool_if:NTF \g_um_literal_Nabla_bool {
2145     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{#1}
2146   }{
2147     \bool_if:NT \g_um_upNabla_bool {

```

```

2148     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
2149   }
2150 }
2151 }

```

12.6.11 Bold fractur or fraktur or blackletter: `\mathbffrak`

```

2152 \cs_new:Nn \um_config_bffrak_Latin:n {
2153   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
2154 }
2155 \cs_new:Nn \um_config_bffrak_latin:n {
2156   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
2157 }

```

12.6.12 Bold script or calligraphic: `\mathbfscr`

```

2158 \cs_new:Nn \um_config_bfscr_Latin:n {
2159   \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{#1}
2160 }
2161 \cs_new:Nn \um_config_bfscr_latin:n {
2162   \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{#1}
2163 }
2164 \cs_new:Nn \um_config_bfcal_Latin:n {
2165   \um_set_mathalphabet_Latin:Nnn \mathbfcal {up,it}{#1}
2166 }

```

12.6.13 Bold upright sans serif: `\mathbfsfup`

```

2167 \cs_new:Nn \um_config_bfsfup_num:n {
2168   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{#1}
2169   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
2170 }
2171 \cs_new:Nn \um_config_bfsfup_Latin:n {
2172   \bool_if:NTF \g_um_sfliteral_bool {
2173     \um_map_chars_Latin:nn {bfsfup} {#1}
2174     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
2175   }{
2176     \bool_if:NT \g_um_upsans_bool {
2177       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
2178       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
2179     }
2180   }
2181   \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
2182 }
2183 \cs_new:Nn \um_config_bfsfup_latin:n {
2184   \bool_if:NTF \g_um_sfliteral_bool {
2185     \um_map_chars_latin:nn {bfsfup} {#1}
2186     \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
2187   }{
2188     \bool_if:NT \g_um_upsans_bool {
2189       \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
2190       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
2191     }
2192   }

```

```

2192 }
2193 \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
2194 }
2195 \cs_new:Nn \um_config_bfsfup_greek:n {
2196   \bool_if:NTF \g_um_sfliteral_bool {
2197     \um_map_chars_greek:nn {bfsfup}{#1}
2198     \um_set_mathalphabet_greek:Nnn \mathbfsf {up}{#1}
2199   }{
2200     \bool_if:NT \g_um_upsans_bool {
2201       \um_map_chars_greek:nn {bfsfup,bfsfit}{#1}
2202       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it}{#1}
2203     }
2204   }
2205   \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it}{#1}
2206 }
2207 \cs_new:Nn \um_config_bfsfup_greek:n {
2208   \bool_if:NTF \g_um_sfliteral_bool {
2209     \um_map_chars_greek:nn {bfsfup} {#1}
2210     \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
2211   }{
2212     \bool_if:NT \g_um_upsans_bool {
2213       \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
2214       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
2215     }
2216   }
2217   \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
2218 }
2219 \cs_new:Nn \um_config_bfsfup_misc:n {
2220   \bool_if:NTF \g_um_literal_Nabla_bool {
2221     \um_map_single:nnn {Nabla}{bfsfup}{#1}
2222   }{
2223     \bool_if:NT \g_um_upNabla_bool {
2224       \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
2225     }
2226   }
2227   \bool_if:NTF \g_um_literal_partial_bool {
2228     \um_map_single:nnn {partial}{bfsfup}{#1}
2229   }{
2230     \bool_if:NT \g_um_uppartial_bool {
2231       \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
2232     }
2233   }
2234   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{#1}
2235   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{#1}
2236   \bool_if:NTF \g_um_literal_partial_bool {
2237     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{#1}
2238   }{
2239     \bool_if:NT \g_um_uppartial_bool {
2240       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
2241     }
2242   }

```

```

2243 \bool_if:NTF \g_um_literal_Nabla_bool {
2244   \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{#1}
2245 }{
2246   \bool_if:NT \g_um_upNabla_bool {
2247     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
2248   }
2249 }
2250 }

```

12.6.14 Bold italic sans serif: \mathbfsfit

```

2251 \cs_new:Nn \um_config_bfsfit_Latin:n {
2252   \bool_if:NTF \g_um_sfliteral_bool {
2253     \um_map_chars_Latin:nn {bfsfit} {#1}
2254     \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
2255   }{
2256     \bool_if:NF \g_um_upsans_bool {
2257       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
2258       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
2259     }
2260   }
2261   \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
2262 }
2263 \cs_new:Nn \um_config_bfsfit_latin:n {
2264   \bool_if:NTF \g_um_sfliteral_bool {
2265     \um_map_chars_latin:nn {bfsfit} {#1}
2266     \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
2267   }{
2268     \bool_if:NF \g_um_upsans_bool {
2269       \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
2270       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
2271     }
2272   }
2273   \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{#1}
2274 }
2275 \cs_new:Nn \um_config_bfsfit_Greek:n {
2276   \bool_if:NTF \g_um_sfliteral_bool {
2277     \um_map_chars_Greek:nn {bfsfit}{#1}
2278     \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{#1}
2279   }{
2280     \bool_if:NF \g_um_upsans_bool {
2281       \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
2282       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
2283     }
2284   }
2285   \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{#1}
2286 }
2287 \cs_new:Nn \um_config_bfsfit_greek:n {
2288   \bool_if:NTF \g_um_sfliteral_bool {
2289     \um_map_chars_greek:nn {bfsfit} {#1}
2290     \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
2291   }{

```

```

2292 \bool_if:NF \g_um_upsans_bool {
2293   \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
2294   \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
2295 }
2296 }
2297 \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
2298 }
2299 \cs_new:Nn \um_config_bfsfit_misc:n {
2300   \bool_if:NTF \g_um_literal_Nabla_bool {
2301     \um_map_single:nnn {Nabla}{bfsfit}{#1}
2302   }{
2303     \bool_if:NF \g_um_upNabla_bool {
2304       \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
2305     }
2306   }
2307   \bool_if:NTF \g_um_literal_partial_bool {
2308     \um_map_single:nnn {partial}{bfsfit}{#1}
2309   }{
2310     \bool_if:NF \g_um_uppartial_bool {
2311       \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
2312     }
2313   }
2314   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{#1}
2315   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{#1}
2316   \bool_if:NTF \g_um_literal_partial_bool {
2317     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{#1}
2318   }{
2319     \bool_if:NF \g_um_uppartial_bool {
2320       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
2321     }
2322   }
2323   \bool_if:NTF \g_um_literal_Nabla_bool {
2324     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{#1}
2325   }{
2326     \bool_if:NF \g_um_upNabla_bool {
2327       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
2328     }
2329   }
2330 }

```

13 A token list to contain the data of the math table

Instead of \input-ing the unicode math table every time we want to re-read its data, we save it within a macro. This has two advantages: 1. it should be slightly faster, at the expense of memory; 2. we don't need to worry about catcodes later, since they're frozen at this point.

In time, the case statement inside `set_mathsymbol` will be moved in here to avoid re-running it every time.

```

2331 \cs_new:Npn \um_symbol_setup: {
2332   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4 {

```

```

2333     \exp_not:n {\_um_sym:nnn{##1}{##2}{##3}}
2334   }
2335 }
2336 \CatchFileEdef \g_um_mathtable_tl {unicode-math-table.tex} {\_um_symbol_setup:}

\_um_input_math_symbol_table: This function simply expands to the token list containing all the data.
2337 \cs_new:Nn \um_input_math_symbol_table: {\g_um_mathtable_tl}

```

14 Definitions of the active math characters

Here we define every Unicode math codepoint an equivalent macro name. The two are equivalent, in a `\let\xyz=^^^1234` kind of way.

`\um_cs_set_eq_active_char:Nw` We need to do some trickery to transform the `_um_sym:nnn` argument "ABCDEF into the \TeX ‘caret input’ form `^^^^abcdef`. It is *very important* that the argument has five characters. Otherwise we need to change the number of `^` chars.

`\um_active_char_set:wc` To do this, turn `^` into a regular ‘other’ character and define the macro to perform the lowercasing and `\let`. `\scantokens` changes the carets back into their original meaning after the group has ended and `^`’s catcode returns to normal.

```

2338 \group_begin:
2339   \char_set_catcode_other:N \^
2340   \cs_gset:Npn \um_cs_set_eq_active_char:Nw #1 = "#2 \q_nil {
2341     \tex_lowercase:D {
2342       \tl_rescan:nn {
2343         \ExplSyntaxOn
2344         \char_set_catcode_other:N \{
2345         \char_set_catcode_other:N \}
2346         \char_set_catcode_other:N \&
2347         \char_set_catcode_other:N \%
2348         \char_set_catcode_other:N \$
2349       }{
2350         \cs_gset_eq:NN #1 ^^^^#2
2351       }
2352     }
2353   }

```

Making `^` the right catcode isn’t strictly necessary right now but it helps to future proof us with, e.g., `breqn`. Because we’re inside a `\tl_rescan:nn`, use plain old \TeX syntax to avoid any catcode problems.

```

2354   \cs_new:Npn \um_active_char_set:wc "#1 \q_nil #2 {
2355     \tex_lowercase:D {
2356       \tl_rescan:nn { \ExplSyntaxOn }
2357       { \cs_gset_protected_nopar:Npx ^^^^#1 { \exp_not:c {#2} } }
2358     }
2359   }
2360 \group_end:

```

Now give `_um_sym:nnn` a definition in terms of `\um_cs_set_eq_active_char:Nw` and we’re good to go.

Ensure catcodes are appropriate; make sure # is an ‘other’ so that we don’t get confused with \mathoctothorpe.

```

2361 \AtBeginDocument{\um_define_math_chars:}
2362 \cs_new:Nn \um_define_math_chars: {
2363   \group_begin:
2364     \char_set_catcode_math_superscript:N \^
2365     \cs_set:Npn \um_sym:nnn ##1##2##3 {
2366       \bool_if:nF { \cs_if_eq_p:NN ##3 \mathaccent ||
2367                     \cs_if_eq_p:NN ##3 \mathopen  ||
2368                     \cs_if_eq_p:NN ##3 \mathclose ||
2369                     \cs_if_eq_p:NN ##3 \mathover  ||
2370                     \cs_if_eq_p:NN ##3 \mathunder ||
2371                     \cs_if_eq_p:NN ##3 \mathbotaccent } {
2372         \um_cs_set_eq_active_char:Nw ##2 = ##1 \q_nil \ignorespaces
2373       }
2374     }
2375     \char_set_catcode_other:N \#
2376     \um_input_math_symbol_table:
2377   \group_end:
2378 }

```

Fix \backslash, which is defined as the escape char character above:

```

2379 \group_begin:
2380   \lccode`\*=\ \
2381   \char_set_catcode_escape:N \
2382   \char_set_catcode_other:N \
2383   |lowercase{
2384     |AtBeginDocument{
2385       |let|backslash=*
2386     }
2387   }
2388 |group_end:

```

Fix \backslash:

15 Fall-back font

Want to load Latin Modern Math if nothing else.

```

2389 \AtBeginDocument { \um_load_lm_if_necessary: }
2390 \cs_new:Nn \um_load_lm_if_necessary:
2391 {
2392   \cs_if_exist:NF \l_um_fontname_tl
2393   {
2394     % XXX: update this when lmmath-bold.otf is released
2395     \setmathfont[BoldFont={lmmath-regular.otf}]{lmmath-regular.otf}
2396   }
2397 }

```

16 Epilogue

Lots of little things to tidy up.

16.1 Primes

We need a new ‘prime’ algorithm. Unicode math has four pre-drawn prime glyphs.

```
u+2032 prime (\prime):  $x'$ 
u+2033 double prime (\dprime):  $x''$ 
u+2034 triple prime (\trprime):  $x'''$ 
u+2057 quadruple prime (\qprime):  $x''''$ 
```

As you can see, they’re all drawn at the correct height without being superscripted. However, in a correctly behaving OpenType font, we also see different behaviour after the `ssty` feature is applied:

$x' \ x'' \ x''' \ x''''$

The glyphs are now ‘full size’ so that when placed inside a superscript, their shape will match the originally sized ones. Many thanks to Ross Mills of Tiro Typeworks for originally pointing out this behaviour.

In regular \LaTeX , primes can be entered with the straight quote character `'`, and multiple straight quotes chain together to produce multiple primes. Better results can be achieved in unicode-math by chaining multiple single primes into a pre-drawn multi-prime glyph; consider x''' vs. x''' .

For Unicode maths, we wish to conserve this behaviour and augment it with the possibility of adding any combination of Unicode prime or any of the n -prime characters. E.g., the user might copy-paste a double prime from another source and then later type another single prime after it; the output should be the triple prime.

Our algorithm is:

- Prime encountered; pcount=1.
- Scan ahead; if prime: pcount:=pcount+1; repeat.
- If not prime, stop scanning.
- If pcount=1, `\prime`, end.
- If pcount=2, check `\dprime`; if it exists, use it, end; if not, goto last step.
- Ditto pcount=3 & `\trprime`.
- Ditto pcount=4 & `\qprime`.
- If pcount>4 or the glyph doesn’t exist, insert pcount `\primes` with `\primekern` between each.

This is a wrapper to insert a superscript; if there is a subsequent trailing superscript, then it is included within the insertion.

```
2398 \cs_new:Nn \um_arg_i_before_egroup:n {#1\egroup}
2399 \cs_new:Nn \um_superscript:n {
2400   ^\bgroup #1
2401   \peek_meaning_remove:NTF ^ \um_arg_i_before_egroup:n \egroup
2402 }
```

```

2403 \muskip_new:N \g_um_primekern_muskip
2404 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
2405 \int_new:N \l_um_primecount_int
2406 \cs_new:Nn \um_nprimes:Nn {
2407   \um_superscript:n {
2408     #1
2409     \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
2410   }
2411 }
2412 \cs_new:Nn \um_nprimes_select:nn {
2413   \prg_case_int:nnn {#2}{
2414     {1} { \um_superscript:n {#1} }
2415     {2} {
2416       \um_glyph_if_exist:nTF {"2033}
2417       { \um_superscript:n {\um_prime_double_mchar} }
2418       { \um_nprimes:Nn #1 {#2} }
2419     }
2420     {3} {
2421       \um_glyph_if_exist:nTF {"2034}
2422       { \um_superscript:n {\um_prime_triple_mchar} }
2423       { \um_nprimes:Nn #1 {#2} }
2424     }
2425     {4} {
2426       \um_glyph_if_exist:nTF {"2057}
2427       { \um_superscript:n {\um_prime_quad_mchar} }
2428       { \um_nprimes:Nn #1 {#2} }
2429     }
2430   }{
2431     \um_nprimes:Nn #1 {#2}
2432   }
2433 }
2434 \cs_new:Nn \um_nbackprimes_select:nn {
2435   \prg_case_int:nnn {#2}{
2436     {1} { \um_superscript:n {#1} }
2437     {2} {
2438       \um_glyph_if_exist:nTF {"2036}
2439       { \um_superscript:n {\um_backprime_double_mchar} }
2440       { \um_nprimes:Nn #1 {#2} }
2441     }
2442     {3} {
2443       \um_glyph_if_exist:nTF {"2037}
2444       { \um_superscript:n {\um_backprime_triple_mchar} }
2445       { \um_nprimes:Nn #1 {#2} }
2446     }
2447   }{
2448     \um_nprimes:Nn #1 {#2}
2449   }
2450 }

```

Scanning is annoying because I'm too lazy to do it for the general case.

```

2451 \cs_new:Npn \um_scan_prime: {
2452   \cs_set_eq:NN \um_superscript:n \use:n
2453   \int_zero:N \l_um_primecount_int
2454   \um_scanprime_collect:N \um_prime_single_mchar
2455 }
2456 \cs_new:Npn \um_scan_dprime: {
2457   \cs_set_eq:NN \um_superscript:n \use:n
2458   \int_set:Nn \l_um_primecount_int {1}
2459   \um_scanprime_collect:N \um_prime_single_mchar
2460 }
2461 \cs_new:Npn \um_scan_trprime: {
2462   \cs_set_eq:NN \um_superscript:n \use:n
2463   \int_set:Nn \l_um_primecount_int {2}
2464   \um_scanprime_collect:N \um_prime_single_mchar
2465 }
2466 \cs_new:Npn \um_scan_qprime: {
2467   \cs_set_eq:NN \um_superscript:n \use:n
2468   \int_set:Nn \l_um_primecount_int {3}
2469   \um_scanprime_collect:N \um_prime_single_mchar
2470 }
2471 \cs_new:Npn \um_scan_sup_prime: {
2472   \int_zero:N \l_um_primecount_int
2473   \um_scanprime_collect:N \um_prime_single_mchar
2474 }
2475 \cs_new:Npn \um_scan_sup_dprime: {
2476   \int_set:Nn \l_um_primecount_int {1}
2477   \um_scanprime_collect:N \um_prime_single_mchar
2478 }
2479 \cs_new:Npn \um_scan_sup_trprime: {
2480   \int_set:Nn \l_um_primecount_int {2}
2481   \um_scanprime_collect:N \um_prime_single_mchar
2482 }
2483 \cs_new:Npn \um_scan_sup_qprime: {
2484   \int_set:Nn \l_um_primecount_int {3}
2485   \um_scanprime_collect:N \um_prime_single_mchar
2486 }
2487 \cs_new:Nn \um_scanprime_collect:N {
2488   \int_incr:N \l_um_primecount_int
2489   \peek_meaning_remove:NTF ' {
2490     \um_scanprime_collect:N #1
2491   }{
2492     \peek_meaning_remove:NTF \um_scan_prime: {
2493       \um_scanprime_collect:N #1
2494     }{
2495       \peek_meaning_remove:NTF ^^^^2032 {
2496         \um_scanprime_collect:N #1
2497       }{
2498         \peek_meaning_remove:NTF \um_scan_dprime: {
2499           \int_incr:N \l_um_primecount_int
2500           \um_scanprime_collect:N #1
2501         }{

```

```

2502     \peek_meaning_remove:NTF ^^^^2033 {
2503         \int_incr:N \l_um_primecount_int
2504         \um_scanprime_collect:N #1
2505     }{
2506         \peek_meaning_remove:NTF \um_scan_trprime: {
2507             \int_add:Nn \l_um_primecount_int {2}
2508             \um_scanprime_collect:N #1
2509         }{
2510             \peek_meaning_remove:NTF ^^^^2034 {
2511                 \int_add:Nn \l_um_primecount_int {2}
2512                 \um_scanprime_collect:N #1
2513             }{
2514                 \peek_meaning_remove:NTF \um_scan_qprime: {
2515                     \int_add:Nn \l_um_primecount_int {3}
2516                     \um_scanprime_collect:N #1
2517                 }{
2518                     \peek_meaning_remove:NTF ^^^^2057 {
2519                         \int_add:Nn \l_um_primecount_int {3}
2520                         \um_scanprime_collect:N #1
2521                     }{
2522                         \um_nprimes_select:nn {#1} {\l_um_primecount_int}
2523                     }
2524                 }
2525             }
2526         }
2527     }
2528 }
2529 }
2530 }
2531 }
2532 }
2533 \cs_new:Npn \um_scan_backprime: {
2534     \cs_set_eq:NN \um_superscript:n \use:n
2535     \int_zero:N \l_um_primecount_int
2536     \um_scanbackprime_collect:N \um_backprime_single_mchar
2537 }
2538 \cs_new:Npn \um_scan_backdprime: {
2539     \cs_set_eq:NN \um_superscript:n \use:n
2540     \int_set:Nn \l_um_primecount_int {1}
2541     \um_scanbackprime_collect:N \um_backprime_single_mchar
2542 }
2543 \cs_new:Npn \um_scan_backtrprime: {
2544     \cs_set_eq:NN \um_superscript:n \use:n
2545     \int_set:Nn \l_um_primecount_int {2}
2546     \um_scanbackprime_collect:N \um_backprime_single_mchar
2547 }
2548 \cs_new:Npn \um_scan_sup_backprime: {
2549     \int_zero:N \l_um_primecount_int
2550     \um_scanbackprime_collect:N \um_backprime_single_mchar
2551 }
2552 \cs_new:Npn \um_scan_sup_backdprime: {

```

```

2553 \int_set:Nn \l_um_primecount_int {1}
2554 \um_scanbackprime_collect:N \um_backprime_single_mchar
2555 }
2556 \cs_new:Npn \um_scan_sup_backtrprime: {
2557 \int_set:Nn \l_um_primecount_int {2}
2558 \um_scanbackprime_collect:N \um_backprime_single_mchar
2559 }
2560 \cs_new:Nn \um_scanbackprime_collect:N {
2561 \int_incr:N \l_um_primecount_int
2562 \peek_meaning_remove:NTF ` {
2563 \um_scanbackprime_collect:N #1
2564 }{
2565 \peek_meaning_remove:NTF \um_scan_backprime: {
2566 \um_scanbackprime_collect:N #1
2567 }{
2568 \peek_meaning_remove:NTF ^^^^2035 {
2569 \um_scanbackprime_collect:N #1
2570 }{
2571 \peek_meaning_remove:NTF \um_scan_backdprime: {
2572 \int_incr:N \l_um_primecount_int
2573 \um_scanbackprime_collect:N #1
2574 }{
2575 \peek_meaning_remove:NTF ^^^^2036 {
2576 \int_incr:N \l_um_primecount_int
2577 \um_scanbackprime_collect:N #1
2578 }{
2579 \peek_meaning_remove:NTF \um_scan_backtrprime: {
2580 \int_add:Nn \l_um_primecount_int {2}
2581 \um_scanbackprime_collect:N #1
2582 }{
2583 \peek_meaning_remove:NTF ^^^^2037 {
2584 \int_add:Nn \l_um_primecount_int {2}
2585 \um_scanbackprime_collect:N #1
2586 }{
2587 \um_nbackprimes_select:nn {#1} {\l_um_primecount_int}
2588 }
2589 }
2590 }
2591 }
2592 }
2593 }
2594 }
2595 }
2596 \AtBeginDocument{\um_define_prime_commands: \um_define_prime_chars:}
2597 \cs_new:Nn \um_define_prime_commands: {
2598 \cs_set_eq:NN \prime \um_prime_single_mchar
2599 \cs_set_eq:NN \dprime \um_prime_double_mchar
2600 \cs_set_eq:NN \trprime \um_prime_triple_mchar
2601 \cs_set_eq:NN \qprime \um_prime_quad_mchar
2602 \cs_set_eq:NN \backprime \um_backprime_single_mchar

```

```

2603 \cs_set_eq:NN \backdprime \um_backprime_double_mchar
2604 \cs_set_eq:NN \backtrprime \um_backprime_triple_mchar
2605 }
2606 \group_begin:
2607 \char_set_catcode_active:N \'
2608 \char_set_catcode_active:N `
2609 \char_set_catcode_active:n {"2032}
2610 \char_set_catcode_active:n {"2033}
2611 \char_set_catcode_active:n {"2034}
2612 \char_set_catcode_active:n {"2057}
2613 \char_set_catcode_active:n {"2035}
2614 \char_set_catcode_active:n {"2036}
2615 \char_set_catcode_active:n {"2037}
2616 \cs_gset:Nn \um_define_prime_chars: {
2617   \cs_set_eq:NN ' \um_scan_sup_prime:
2618   \cs_set_eq:NN ^^^^2032 \um_scan_sup_prime:
2619   \cs_set_eq:NN ^^^^2033 \um_scan_sup_dprime:
2620   \cs_set_eq:NN ^^^^2034 \um_scan_sup_trprime:
2621   \cs_set_eq:NN ^^^^2057 \um_scan_sup_qprime:
2622   \cs_set_eq:NN ` \um_scan_sup_backprime:
2623   \cs_set_eq:NN ^^^^2035 \um_scan_sup_backprime:
2624   \cs_set_eq:NN ^^^^2036 \um_scan_sup_backdprime:
2625   \cs_set_eq:NN ^^^^2037 \um_scan_sup_backtrprime:
2626 }
2627 \group_end:

```

16.2 Unicode radicals

```

2628 \AtBeginDocument{\um_redefine_radical:}
2629 \cs_new:Nn \um_redefine_radical:
2630 {*XE}
2631 {
2632   \@ifpackageloaded { amsmath } { } { } {

```

\r@@t #1 : A mathstyle (for \mathpalette)

#2 : Leading superscript for the sqrt sign

A re-implementation of L^AT_EX's hard-coded n-root sign using the appropriate \fontdimens.

```

2633   \cs_set_nopar:Npn \r@@t ##1 ##2 {
2634     \hbox_set:Nn \l_tmpa_box {
2635       \c_math_toggle_token
2636       \m@th
2637       ##1
2638       \sqrtsign { ##2 }
2639       \c_math_toggle_token
2640     }
2641     \um_mathstyle_scale:Nnn ##1 { \kern } {
2642       \fontdimen 63 \l_um_font
2643     }
2644     \box_move_up:nn {
2645       (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)

```

```

2646         * \number \fontdimen 65 \L_um_font / 100
2647     } {
2648         \box_use:N \rootbox
2649     }
2650     \um_mathstyle_scale:Nnn ##1 { \kern } {
2651         \fontdimen 64 \L_um_font
2652     }
2653     \box_use_clear:N \L_tmpa_box
2654 }

2655 }
2656 }
2657 </XE>
2658 <*LU>
2659 {
2660     \@ifpackageloaded { amsmath } { } {

```

\root Redefine this macro for LuaTeX, which provides us a nice primitive to use.

```

2661     \cs_set:Npn \root ##1 \of ##2 {
2662         \luatexUroot \L_um_radical_sqrt_tl { ##1 } { ##2 }
2663     }

2664 }
2665 }
2666 </LU>

```

\um_fontdimen_to_percent:nn #1 : Font dimen number
 \um_fontdimen_to_scale:nn #2 : Font ‘variable’

\fontdimens 10, 11, and 65 aren’t actually dimensions, they’re percentage values given in units of sp. \um_fontdimen_to_percent:nn takes a font dimension number and outputs the decimal value of the associated parameter. \um_fontdimen_to_scale:nn returns a dimension correspond to the current font size relative proportion based on that percentage.

```

2667 \cs_new:Nn \um_fontdimen_to_percent:nn {
2668     \strip@pt\dimexpr\fontdimen#1#2*65536/100\relax
2669 }
2670 \cs_new:Nn \um_fontdimen_to_scale:nn
2671 {
2672     \um_fontdimen_to_percent:nn {#1} {#2} \dimexpr \f@size pt\relax
2673 }

```

\um_mathstyle_scale:Nnn #1 : A math style (\scriptstyle, say)
 #2 : Macro that takes a non-delimited length argument (like \kern)
 #3 : Length control sequence to be scaled according to the math style
 This macro is used to scale the lengths reported by \fontdimen according to the scale factor for script- and scriptscript-size objects.

```

2674 \cs_new:Nn \um_mathstyle_scale:Nnn {
2675     \ifx#1\scriptstyle
2676         #2\um_fontdimen_to_percent:nn{10}\L_um_font#3
2677     \else

```



```

2678 \ifx#1\scriptscriptstyle
2679 #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
2680 \else
2681 #2#3
2682 \fi
2683 \fi
2684 }

```

16.3 Unicode sub- and super-scripts

The idea here is to enter a scanning state after a superscript or subscript is encountered. If subsequent superscripts or subscripts (resp.) are found, they are lumped together. Each sub/super has a corresponding regular size glyph which is used by X_YTeX to typeset the results; this means that the actual subscript/superscript glyphs are never seen in the output document — they are only used as input characters.

Open question: should the superscript-like ‘modifiers’ (U+1D2C modifier capital letter a and on) be included here?

```

2685 \prop_new:N \g_um_supers_prop
2686 \prop_new:N \g_um_subs_prop
2687 \group_begin:

```

Superscripts Populate a property list with superscript characters; their meaning as their key, for reasons that will become apparent soon, and their replacement as each key’s value. Then make the superscript active and bind it to the scanning function.

\scantokens makes this process much simpler since we can activate the char and assign its meaning in one step.

```

2688 \cs_new:Nn \um_setup_active_superscript:nn {
2689 \prop_gput:Nxn \g_um_supers_prop {\meaning #1} {#2}
2690 \char_set_catcode_active:N #1
2691 \char_gmake_mathactive:N #1
2692 \scantokens{
2693 \cs_gset:Npn #1 {
2694 \tl_set:Nn \l_um_ss_chain_tl {#2}
2695 \cs_set_eq:NN \um_sub_or_super:n \sp
2696 \tl_set:Nn \l_um_tmpa_tl {supers}
2697 \um_scan_sscript:
2698 }
2699 }
2700 }

```

Bam:

```

2701 \um_setup_active_superscript:nn {^^^2070} {0}
2702 \um_setup_active_superscript:nn {^^^00b9} {1}
2703 \um_setup_active_superscript:nn {^^^00b2} {2}
2704 \um_setup_active_superscript:nn {^^^00b3} {3}
2705 \um_setup_active_superscript:nn {^^^2074} {4}
2706 \um_setup_active_superscript:nn {^^^2075} {5}

```

```

2707 \um_setup_active_superscript:nn {^^^^2076} {6}
2708 \um_setup_active_superscript:nn {^^^^2077} {7}
2709 \um_setup_active_superscript:nn {^^^^2078} {8}
2710 \um_setup_active_superscript:nn {^^^^2079} {9}
2711 \um_setup_active_superscript:nn {^^^^207a} {+}
2712 \um_setup_active_superscript:nn {^^^^207b} {-}
2713 \um_setup_active_superscript:nn {^^^^207c} {=}
2714 \um_setup_active_superscript:nn {^^^^207d} {(}
2715 \um_setup_active_superscript:nn {^^^^207e} {)}
2716 \um_setup_active_superscript:nn {^^^^207f} {i}
2717 \um_setup_active_superscript:nn {^^^^207f} {n}

```

Subscripts Ditto above.

```

2718 \cs_new:Nn \um_setup_active_subscript:nn {
2719   \prop_gput:Nxn \g_um_subs_prop  {\meaning #1} {#2}
2720   \char_set_catcode_active:N #1
2721   \char_gmake_mathactive:N #1
2722   \scantokens{
2723     \cs_gset:Npn #1 {
2724       \tl_set:Nn \l_um_ss_chain_tl {#2}
2725       \cs_set_eq:NN \um_sub_or_super:n \sb
2726       \tl_set:Nn \l_um_tmpa_tl {subs}
2727       \um_scan_sscript:
2728     }
2729   }
2730 }

```

A few more subscripts than superscripts:

```

2731 \um_setup_active_subscript:nn {^^^^2080} {0}
2732 \um_setup_active_subscript:nn {^^^^2081} {1}
2733 \um_setup_active_subscript:nn {^^^^2082} {2}
2734 \um_setup_active_subscript:nn {^^^^2083} {3}
2735 \um_setup_active_subscript:nn {^^^^2084} {4}
2736 \um_setup_active_subscript:nn {^^^^2085} {5}
2737 \um_setup_active_subscript:nn {^^^^2086} {6}
2738 \um_setup_active_subscript:nn {^^^^2087} {7}
2739 \um_setup_active_subscript:nn {^^^^2088} {8}
2740 \um_setup_active_subscript:nn {^^^^2089} {9}
2741 \um_setup_active_subscript:nn {^^^^208a} {+}
2742 \um_setup_active_subscript:nn {^^^^208b} {-}
2743 \um_setup_active_subscript:nn {^^^^208c} {=}
2744 \um_setup_active_subscript:nn {^^^^208d} {(}
2745 \um_setup_active_subscript:nn {^^^^208e} {)}
2746 \um_setup_active_subscript:nn {^^^^2090} {a}
2747 \um_setup_active_subscript:nn {^^^^2091} {e}
2748 \um_setup_active_subscript:nn {^^^^1d62} {i}
2749 \um_setup_active_subscript:nn {^^^^2092} {o}
2750 \um_setup_active_subscript:nn {^^^^1d63} {r}
2751 \um_setup_active_subscript:nn {^^^^1d64} {u}
2752 \um_setup_active_subscript:nn {^^^^1d65} {v}
2753 \um_setup_active_subscript:nn {^^^^2093} {x}

```

```

2754 \um_setup_active_subscript:nn {^^^^1d66} {\beta}
2755 \um_setup_active_subscript:nn {^^^^1d67} {\gamma}
2756 \um_setup_active_subscript:nn {^^^^1d68} {\rho}
2757 \um_setup_active_subscript:nn {^^^^1d69} {\phi}
2758 \um_setup_active_subscript:nn {^^^^1d6a} {\chi}
2759 \group_end:

```

The scanning command, evident in its purpose:

```

2760 \cs_new:Npn \um_scan_sscript: {
2761   \um_scan_sscript:TF {
2762     \um_scan_sscript:
2763   }{
2764     \um_sub_or_super:n {\l_um_ss_chain_tl}
2765   }
2766 }

```

The main theme here is stolen from the source to the various `\peek_` functions. Consider this function as simply boilerplate: TODO: move all this to `expl3`, and don't use internal `expl3` macros.

```

2767 \cs_new:Npn \um_scan_sscript:TF #1#2 {
2768   \tl_set:Nx \peek_true_aux:w { \exp_not:n{ #1 } }
2769   \tl_set_eq:NN \peek_true:w \peek_true_remove:w
2770   \tl_set:Nx \peek_false:w { \exp_not:n { \group_align_safe_end: #2 } }
2771   \group_align_safe_begin:
2772   \peek_after:Nw \um_peek_execute_branches_ss:
2773 }

```

We do not skip spaces when scanning ahead, and we explicitly wish to bail out on encountering a space or a brace.

```

2774 \cs_new:Npn \um_peek_execute_branches_ss: {
2775   \bool_if:nTF {
2776     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2777     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2778     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2779   }
2780   { \peek_false:w }
2781   { \um_peek_execute_branches_ss_aux: }
2782 }

```

This is the actual comparison code. Because the peeking has already tokenised the next token, it's too late to extract its charcode directly. Instead, we look at its meaning, which remains a 'character' even though it is itself math-active. If the character is ever made fully active, this will break our assumptions!

If the char's meaning exists as a property list key, we build up a chain of sub-/superscripts and iterate. (If not, exit and typeset what we've already collected.)

```

2783 \cs_new:Npn \um_peek_execute_branches_ss_aux: {
2784   \prop_if_in:cxTF
2785     {g_um_\l_um_tmpa_tl _prop} {\meaning\l_peek_token}
2786   {
2787     \prop_get:cxN
2788     {g_um_\l_um_tmpa_tl _prop} {\meaning\l_peek_token} \l_um_tmpb_tl

```

```

2789     \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
2790     \peek_true:w
2791   }
2792   { \peek_false:w }
2793 }

```

16.3.1 Active fractions

Active fractions can be setup independently of any maths font definition; all it requires is a mapping from the Unicode input chars to the relevant L^AT_EX fraction declaration.

```

2794 \cs_new:Npn \um_define_active_frac:Nw #1 #2/#3 {
2795   \char_set_catcode_active:N #1
2796   \char_gmake_mathactive:N #1
2797   \tl_rescan:nn {
2798     \catcode`\_ =11\relax
2799     \catcode`\:=11\relax
2800   }{
2801     \cs_gset:Npx #1 {
2802       \bool_if:NTF \l_um_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2803       {#2} {#3}
2804     }
2805   }
2806 }

```

These are redefined for each math font selection in case the active-frac feature changes.

```

2807 \cs_new:Npn \um_setup_active_frac: {
2808   \group_begin:
2809   \um_define_active_frac:Nw ^^2189 0/3
2810   \um_define_active_frac:Nw ^^2152 1/{10}
2811   \um_define_active_frac:Nw ^^2151 1/9
2812   \um_define_active_frac:Nw ^^215b 1/8
2813   \um_define_active_frac:Nw ^^2150 1/7
2814   \um_define_active_frac:Nw ^^2159 1/6
2815   \um_define_active_frac:Nw ^^2155 1/5
2816   \um_define_active_frac:Nw ^^00bc 1/4
2817   \um_define_active_frac:Nw ^^2153 1/3
2818   \um_define_active_frac:Nw ^^215c 3/8
2819   \um_define_active_frac:Nw ^^2156 2/5
2820   \um_define_active_frac:Nw ^^00bd 1/2
2821   \um_define_active_frac:Nw ^^2157 3/5
2822   \um_define_active_frac:Nw ^^215d 5/8
2823   \um_define_active_frac:Nw ^^2154 2/3
2824   \um_define_active_frac:Nw ^^00be 3/4
2825   \um_define_active_frac:Nw ^^2158 4/5
2826   \um_define_active_frac:Nw ^^215a 5/6
2827   \um_define_active_frac:Nw ^^215e 7/8
2828   \group_end:
2829 }

```

```
2830 \um_setup_active_frac:
```

16.4 Synonyms and all the rest

These are symbols with multiple names. Eventually to be taken care of automatically by the maths characters database.

```
2831 \def\to{\rightarrow}
2832 \def\le{\leq}
2833 \def\ge{\geq}
2834 \def\neq{\neq}
2835 \def\triangle{\mathord{\bigtriangleup}}
2836 \def\bigcirc{\mdlgwhtcircle}
2837 \def\circ{\vysmwhtcircle}
2838 \def\bullet{\smbllkcircle}
2839 \def\mathyen{\yen}
2840 \def\mathsterling{\sterling}
2841 \def\diamond{\smwhtdiamond}
2842 \def\emptyset{\varnothing}
2843 \def\hbar{\hslash}
2844 \def\land{\wedge}
2845 \def\lor{\vee}
2846 \def\owns{\ni}
2847 \def\gets{\leftarrow}
2848 \def\mathring{\ocirc}
2849 \def\lnot{\neg}
2850 \def\longdivision{\longdivisionsign}
```

These are somewhat odd: (and their usual Unicode uprightness does not match their amssymb glyphs)

```
2851 \def\backepsilon{\upbackepsilon}
2852 \def\eth{\matheth}
```

Due to the magic of OpenType math, big operators are automatically enlarged when necessary. Since there isn't a separate unicode glyph for 'small integral', I'm not sure if there is a better way to do this:

```
2853 \def\smallint{{\textstyle\int}\limits}
```

`\colon` Define `\colon` as a mathpunct `':'`. This is wrong: it should be U+003A colon instead! We hope no-one will notice.

```
2854 \@ifpackageloaded{amsmath}{
2855   % define their own colon, perhaps I should just steal it. (It does look much bet-
2856   % ter.)
2857   }{
2858   \cs_set_protected:Npn \colon {
2859     \bool_if:NTF \g_um_literal_colon_bool {:} { \mathpunct{:} }
2860   }
2861 }
```

`\mathrm`

```
2861 \def\mathrm{\mathup}
2862 \let\mathfence\mathord
```

`\digamma` I might end up just changing these in the table.

`\Digamma` 2863 `\def\digamma{\updigamma}`

2864 `\def\Digamma{\upDigamma}`

16.5 Compatibility

We need to change L^AT_EX's idea of the font used to typeset things like `\sin` and `\cos`:

2865 `\def\operator@font{\um_switchto_mathup:}`

`\um_tmpa:w` A scratch macro.

2866 `\chk_if_free_cs:N \um_tmpa:w`

`\um_check_and_fix:NNnnnn` **#1** : command

#2 : factory command

#3 : parameter text

#4 : expected replacement text

#5 : new replacement text for Lua_T_EX

#6 : new replacement text for X_Y_T_EX

Tries to patch *⟨command⟩*. If *⟨command⟩* is undefined, do nothing. Otherwise it must be a macro with the given *⟨parameter text⟩* and *⟨expected replacement text⟩*, created by the given *⟨factory command⟩* or equivalent. In this case it will be overwritten using the *⟨parameter text⟩* and the *⟨new replacement text for Lua_T_EX⟩* or the *⟨new replacement text for X_Y_T_EX⟩*, depending on the engine. Otherwise issue a warning and don't overwrite.

```
2867 \cs_new_protected_nopar:Npn \um_check_and_fix:NNnnnn #1 #2 #3 #4 #5 #6 {
2868   \cs_if_exist:NT #1 {
2869     \token_if_macro:NTF #1 {
2870       \group_begin:
2871       #2 \um_tmpa:w #3 { #4 }
2872       \cs_if_eq:NNTF #1 \um_tmpa:w {
2873         \msg_info:nxx { unicode-math } { patch-macro }
2874         { \token_to_str:N #1 }
2875         \group_end:
2876         #2 #1 #3
2877         (XE)      { #6 }
2878         (LU)      { #5 }
2879       } {
2880         \msg_warning:nnxxx { unicode-math } { wrong-meaning }
2881         { \token_to_str:N #1 } { \token_to_meaning:N #1 }
2882         { \token_to_meaning:N \um_tmpa:w }
2883         \group_end:
2884       }
2885     } {
2886       \msg_warning:nxx { unicode-math } { macro-expected }
2887       { \token_to_str:N #1 }
2888     }
2889   }
2890 }
```

```

\um_check_and_fix:NNnnn #1 : command
                        #2 : factory command
                        #3 : parameter text
                        #4 : expected replacement text
                        #5 : new replacement text
Tries to patch  $\langle command \rangle$ . If  $\langle command \rangle$  is undefined, do nothing. Otherwise it must
be a macro with the given  $\langle parameter text \rangle$  and  $\langle expected replacement text \rangle$ , created by
the given  $\langle factory command \rangle$  or equivalent. In this case it will be overwritten using
the  $\langle parameter text \rangle$  and the  $\langle new replacement text \rangle$ . Otherwise issue a warning and
don't overwrite.
2891 \cs_new_protected_nopar:Npn \um_check_and_fix:NNnnn #1 #2 #3 #4 #5 {
2892   \um_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
2893 }

\um_check_and_fix_luatex:NNnnn #1 : command
\um_check_and_fix_luatex:cNnnn #2 : factory command
                        #3 : parameter text
                        #4 : expected replacement text
                        #5 : new replacement text
Tries to patch  $\langle command \rangle$ . If XeLaTeX is the current engine or  $\langle command \rangle$  is unde-
fined, do nothing. Otherwise it must be a macro with the given  $\langle parameter text \rangle$ 
and  $\langle expected replacement text \rangle$ , created by the given  $\langle factory command \rangle$  or equiv-
alent. In this case it will be overwritten using the  $\langle parameter text \rangle$  and the  $\langle new$ 
 $\langle replacement text \rangle$ . Otherwise issue a warning and don't overwrite.
2894 \cs_new_protected_nopar:Npn \um_check_and_fix_luatex:NNnnn #1 #2 #3 #4 #5 {
2895   \luatex_if_engine:T {
2896     \um_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
2897   }
2898 }
2899 \cs_generate_variant:Nn \um_check_and_fix_luatex:NNnnn { c }

```

url Simply need to get url in a state such that when it switches to math mode and enters `ASCII` characters, the maths setup (i.e., `unicode-math`) doesn't remap the symbols into Plane 1. Which is, of course, what `\mathup` is doing.

This is the same as writing, e.g., `\def\UrlFont{\ttfamily\um_switchto_mathup:}` but activates automatically so old documents that might change the `\url` font still work correctly.

```

2900 \AtEndOfPackageFile * {url} {
2901   \tl_put_left:Nn \Url@FormatString { \um_switchto_mathup: }
2902   \tl_put_right:Nn \Url@Specials {
2903     \do\{\mathchar`\` }
2904     \do\'\{\mathchar`\' }
2905     \do\$\{\mathchar`\$ }
2906     \do\&\{\mathchar`\& }
2907   }
2908 }

```

amsmath Since the mathcode of ``\-` is greater than eight bits, this piece of `\AtBeginDocument` code from `amsmath` dies if we try and set the maths font in the preamble:

```

2909 \AtEndOfPackageFile * {amsmath} {
2910 (*XE)
2911   \tl_remove_once:Nn \@begindocumenthook {
2912     \mathchardef\std@minus\mathcode`\-\relax
2913     \mathchardef\std@equal\mathcode`\=\relax
2914   }
2915   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
2916   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
2917 (/XE)

2918 \cs_set:Npn \@cdots {\mathinner{\cdots}}
2919 \cs_set_eq:NN \dotso \cdots

```

This isn't as clever as the `amsmath` definition but I think it works:

```

2920 (*XE)
2921   \def \resetMathstrut@ {%
2922     \setbox\z@\hbox{${$}%}
2923     \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
2924   }

```

The subarray environment uses inappropriate font dimensions.

```

2925   \um_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 } {
2926     \vcenter
2927     \bgroup
2928     \Let@
2929     \restore@math@cr
2930     \default@tag
2931     \baselineskip \fontdimen 10~ \scriptfont \tw@
2932     \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
2933     \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
2934     \lineskiplimit \lineskip
2935     \ialign
2936     \bgroup
2937     \ifx c #1 \hfil \fi
2938     $ \m@th \scriptstyle ## $
2939     \hfil
2940     \crrr
2941   } {
2942     \vcenter
2943     \c_group_begin_token
2944     \Let@
2945     \restore@math@cr
2946     \default@tag
2947     \skip_set:Nn \baselineskip {

```

Here we use stack top shift + stack bottom shift, which sounds reasonable.

```

2948       \um_stack_num_up:N \scriptstyle
2949       + \um_stack_denom_down:N \scriptstyle
2950     }

```


Here we use the minimum stack gap.

```

2951     \lineskip \um_stack_vgap:N \scriptstyle
2952     \lineskiplimit \lineskip
2953     \ialign
2954     \c_group_begin_token
2955     \token_if_eq_meaning:NNT c #1 { \hfil }
2956     \c_math_toggle_token
2957     \m@th
2958     \scriptstyle
2959     \c_parameter_token \c_parameter_token
2960     \c_math_toggle_token
2961     \hfil
2962     \c_rcr
2963   }
2964 </XE>

```

The roots need a complete rework.

```

2965 \um_check_and_fix_luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 } {
2966   \setbox \rootbox \hbox {
2967     $ \m@th \scriptscriptstyle { #1 } $
2968   }
2969   \mathchoice
2970     { \r@@t \displaystyle { #2 } }
2971     { \r@@t \textstyle { #2 } }~
2972     { \r@@t \scriptstyle { #2 } }
2973     { \r@@t \scriptscriptstyle { #2 } }
2974   \egroup
2975 } {
2976   \bool_if:nTF {
2977     \int_compare_p:nNn { \uproot@ } = { \c_zero }
2978     && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
2979   } {
2980     \luatexUroot \l_um_radical_sqrt_tl { #1 } { #2 }
2981   } {
2982     \hbox_set:Nn \rootbox {
2983       \c_math_toggle_token
2984       \m@th
2985       \scriptscriptstyle { #1 }
2986       \c_math_toggle_token
2987     }
2988     \mathchoice
2989       { \r@@t \displaystyle { #2 } }
2990       { \r@@t \textstyle { #2 } }
2991       { \r@@t \scriptstyle { #2 } }
2992       { \r@@t \scriptscriptstyle { #2 } }
2993   }
2994   \c_group_end_token
2995 }
2996 \um_check_and_fix:NNnnnn \r@@t \cs_set_nopar:Npn { #1 #2 } {
2997   \setboxz@h { $ \m@th #1 \sqrt{sign { #2 } } $ }
2998   \dimen@ \ht\z@

```

```

2999 \advance \dimen@ -\dp\z@
3000 \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
3001 \advance \dimen@ by 1.667 \wd\@ne
3002 \mkern -\leftroot@ mu
3003 \mkern 5mu
3004 \raise .6\dimen@ \copy\rootbox
3005 \mkern -10mu
3006 \mkern \leftroot@ mu
3007 \boxz@
3008 } {
3009 \hbox_set:Nn \l_tmpa_box {
3010   \c_math_toggle_token
3011   \m@th
3012   #1
3013   \mskip \uproot@ mu
3014   \c_math_toggle_token
3015 }
3016 \luatexUroot \l_um_radical_sqrt_tl {
3017   \box_move_up:nn { \box_wd:N \l_tmpa_box } {
3018     \hbox:n {
3019       \c_math_toggle_token
3020       \m@th
3021       \mkern -\leftroot@ mu
3022       \box_use:N \rootbox
3023       \mkern \leftroot@ mu
3024       \c_math_toggle_token
3025     }
3026   }
3027 } {
3028   #2
3029 }
3030 } {
3031 \hbox_set:Nn \l_tmpa_box {
3032   \c_math_toggle_token
3033   \m@th
3034   #1
3035   \sqrtsign { #2 }
3036   \c_math_toggle_token
3037 }
3038 \hbox_set:Nn \l_tmpb_box {
3039   \c_math_toggle_token
3040   \m@th
3041   #1
3042   \mskip \uproot@ mu
3043   \c_math_toggle_token
3044 }
3045 \mkern -\leftroot@ mu
3046 \um_mathstyle_scale:Nnn #1 { \kern } {
3047   \fontdimen 63 \l_um_font
3048 }
3049 \box_move_up:nn {

```

```

3050     \box_wd:N \l_tmpb_box
3051     + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
3052     * \number \fontdimen 65 \l_um_font / 100
3053   } {
3054     \box_use:N \rootbox
3055   }
3056   \um_mathstyle_scale:Nnn #1 { \kern } {
3057     \fontdimen 64 \l_um_font
3058   }
3059   \mkern \leftroot@ mu
3060   \box_use_clear:N \l_tmpa_box
3061 }
3062 }

```

amsopn This code is to improve the output of alphabetic symbols in text of operator names (`\sin`, `\cos`, etc.). Just comment out the offending lines for now:

```

3063 \AtEndOfPackageFile * {amsopn} {
3064   \cs_set:Npn \newmcodes@ {
3065     \mathcode`\'39\scan_stop:
3066     \mathcode`\'*42\scan_stop:
3067     \mathcode`\'."613A\scan_stop:
3068     %% \ifnum\mathcode`\'=-45 \else
3069     %%   \mathchardef\std@minus\mathcode`\'-\relax
3070     %% \fi
3071     \mathcode`\'-45\scan_stop:
3072     \mathcode`\'/47\scan_stop:
3073     \mathcode`\':"603A\scan_stop:
3074   }
3075 }

```

Symbols

```

3076 \cs_set:Npn \l { \Vert }
\mathinner items:
3077 \cs_set:Npn \mathellipsis { \mathinner { \unicodeellipsis } }
3078 \cs_set:Npn \cdots { \mathinner { \unicodecdots } }

```

Accents

```

3079 \cs_new_protected_nopar:Nn \um_setup_accents: {
3080   \cs_gset_protected_nopar:Npx \widehat {
3081     \um_accent:nnn {} { \um_symfont_t1 } { "0302 }
3082   }
3083   \cs_gset_protected_nopar:Npx \widetilde {
3084     \um_accent:nnn {} { \um_symfont_t1 } { "0303 }
3085   }
3086   \cs_gset_protected_nopar:Npx \overleftarrow {
3087     \um_accent:nnn {} { \um_symfont_t1 } { "20D6 }
3088   }
3089   \cs_gset_protected_nopar:Npx \overrightarrow {

```

```

3090 \um_accent:nnn {} { \um_symfont_t1 } { "20D7 }
3091 }
3092 \cs_gset_protected_nopar:Npx \overleftarrow {
3093 \um_accent:nnn {} { \um_symfont_t1 } { "20E1 }
3094 }
3095 \cs_gset_protected_nopar:Npx \wideutilde {
3096 \um_accent:nnn {bottom} { \um_symfont_t1 } { "0330 }
3097 }
3098 \cs_gset_protected_nopar:Npx \underrightharpoonowdown {
3099 \um_accent:nnn {bottom} { \um_symfont_t1 } { "20EC }
3100 }
3101 \cs_gset_protected_nopar:Npx \underleftharpoonowdown {
3102 \um_accent:nnn {bottom} { \um_symfont_t1 } { "20ED }
3103 }
3104 \cs_gset_protected_nopar:Npx \underleftarrow {
3105 \um_accent:nnn {bottom} { \um_symfont_t1 } { "20EE }
3106 }
3107 \cs_gset_protected_nopar:Npx \underrightarrow {
3108 \um_accent:nnn {bottom} { \um_symfont_t1 } { "20EF }
3109 }
3110 \cs_gset_protected_nopar:Npx \underleftrightharpoonow {
3111 \um_accent:nnn {bottom} { \um_symfont_t1 } { "034D }
3112 }
3113 }
3114 \cs_set_eq:NN \um_text_slash: \slash
3115 \cs_set_protected:Npn \slash {
3116 \mode_if_math:TF {\mathslash} {\um_text_slash:}
3117 }

```

\not The situation of `\not` symbol is currently messy, in Unicode it is defined as a combining mark so naturally it should be treated as a math accent, however neither Lua \TeX nor Xe \TeX correctly place it as it needs special treatment compared to other accents, furthermore a math accent changes the spacing of its nucleus, so `\not=` will be spaced as an ordinary not relational symbol, which is undesired.

Here modify `\not` to a macro that tries to use predefined negated symbols, which would give better results in most cases, until there is more robust solution in the engines.

This code is based on an answer to a TeX – Stack Exchange question by Enrico Gregorio⁵.

```

3118 \tl_new:N \l_not_token_name_tl
3119
3120 \cs_new:Npn \not_newnot:N #1 {
3121 \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
3122 \tl_if_empty:xF { \tl_tail:V \l_not_token_name_tl } {
3123 \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }
3124 }
3125 \cs_if_exist:cTF { n \l_not_token_name_tl } {

```

⁵<http://tex.stackexchange.com/a/47260/729>

```

3126 \use:c { n \l_not_token_name_tl }
3127 } {
3128 \cs_if_exist:cTF { not \l_not_token_name_tl } {
3129 \use:c { not \l_not_token_name_tl }
3130 } {
3131 \not_oldnot: #1 %\l_not_token_name_tl
3132 }
3133 }
3134 }
3135
3136 \cs_new_protected_nopar:Nn \um_setup_negations: {
3137 \cs_set_eq:NN \not_oldnot: \not
3138 \cs_set_eq:NN \not_newnot:N
3139
3140 \cs_gset:cpn { not= } { \neq }
3141 \cs_gset:cpn { not< } { \nless }
3142 \cs_gset:cpn { not> } { \ngtr }
3143 \cs_gset:Npn \ngets { \nleftarrow }
3144 \cs_gset:Npn \nsimeq { \nsime }
3145 \cs_gset:Npn \nequal { \ne }
3146 \cs_gset:Npn \nle { \nleq }
3147 \cs_gset:Npn \nge { \ngeq }
3148 \cs_gset:Npn \ngreater { \ngtr }
3149 \cs_gset:Npn \nforksnot { \forks }
3150 }

```

mathtools mathtools's `\cramped` command and others that make use of its internal version use an incorrect font dimension.

```

3151 \AtEndOfPackageFile * { mathtools } {
3152 (*XE)
3153 \chk_if_free_cs:N \g_um_empty_fam
3154 \newfam \g_um_empty_fam
3155 \um_check_and_fix:NNnnn
3156 \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
3157 {
3158 \sbox \z@ {
3159 $
3160 \m@th
3161 #1
3162 \nulldelimiterspace = \z@
3163 \radical \z@ { #2 }
3164 $
3165 }
3166 \ifx #1 \displaystyle
3167 \dimen@ = \fontdimen 8 \textfont 3
3168 \advance \dimen@ .25 \fontdimen 5 \textfont 2
3169 \else
3170 \dimen@ = 1.25 \fontdimen 8
3171 \ifx #1 \textstyle
3172 \textfont

```

```

3173     \else
3174       \ifx #1 \scriptstyle
3175         \scriptfont
3176       \else
3177         \scriptscriptfont
3178       \fi
3179     \fi
3180     3
3181   \fi
3182   \advance \dimen@ -\ht\z@
3183   \ht\z@ = -\dimen@
3184   \box\z@
3185 }

```

The X_YTeX version is pretty similar to the legacy version, only using the correct font dimensions. Note we used ‘XeTeXradical’ with a newly-allocated empty family to make sure that the radical rule width is not set.

```

3186 {
3187   \hbox_set:Nn \L_tmpa_box {
3188     \color@setgroup
3189     \c_math_toggle_token
3190     \m@th
3191     #1
3192     \dim_zero:N \nulldelimiterspace
3193     \XeTeXradical \g_um_empty_fam \c_zero { #2 }
3194     \c_math_toggle_token
3195     \color@endgroup
3196   }
3197   \box_set_ht:Nn \L_tmpa_box {
3198     \box_ht:N \L_tmpa_box

```

Here we use the radical vertical gap.

```

3199     - \um_radical_vgap:N #1
3200   }
3201   \box_use_clear:N \L_tmpa_box
3202 }
3203 \>/XE

```

`\overbracket` `\underbracket` mathtools’s `\overbracket` and `\underbracket` take optional arguments and are defined in terms of rules, so we keep them, and rename ours to `\Uoverbracket` and `\Underbracket`.

```

3204 \AtEndOfPackageFile * { mathtools } {
3205   \let\MToverbracket =\overbracket
3206   \let\MTunderbracket=\underbracket
3207
3208   \AtBeginDocument {
3209     \msg_warning:nn { unicode-math } { mathtools-overbracket }
3210
3211     \def\downbracketfill#1#2{%

```

Original definition used the height of `\bracehd` which is not available with Unicode fonts, so we are hard coding the 5/18ex suggested by mathtools’s documen-

tation.

```

3212         \edef\l_MT_bracketheight_fdim{.27ex}%
3213         \downbracketend{#1}{#2}
3214         \leaders \vrule \@height #1 \@depth \z@ \hfill
3215         \downbracketend{#1}{#2}%
3216     }
3217 \def\upbracketfill#1#2{%
3218     \edef\l_MT_bracketheight_fdim{.27ex}%
3219     \upbracketend{#1}{#2}
3220     \leaders \vrule \@height \z@ \@depth #1 \hfill
3221     \upbracketend{#1}{#2}%
3222 }
3223 \let\Uoverbracket =\overbracket
3224 \let\Uunderbracket=\underbracket
3225     \let\overbracket =\MToverbracket
3226     \let\underbracket =\MTunderbracket
3227 }
3228 }

```

`\dblcolon` mathtools defines several commands as combinations of colons and other characters, but with meanings incompatible to unicode-math. Thus we issue a warning. `\coloneqq` Because mathtools uses `\providecommand` `\AtBeginDocument`, we can just define the offending commands here.

```

3229 \msg_warning:nn { unicode-math } { mathtools-colon }
3230 \NewDocumentCommand \dblcolon { } { \Colon }
3231 \NewDocumentCommand \coloneqq { } { \coloneq }
3232 \NewDocumentCommand \Coloneqq { } { \Coloneq }
3233 \NewDocumentCommand \eqqcolon { } { \eqcolon }
3234 }

```

colonequals

`\ratio` Similarly to mathtools, the colonequals defines several colon combinations. Fortunately there are no name clashes, so we can just overwrite their definitions.

`\coloncolon`

`\minuscolon` 3235 `\AtEndOfPackageFile * { colonequals } {`

`\colonequals` 3236 `\msg_warning:nn { unicode-math } { colonequals }`

`\equalscolon` 3237 `\RenewDocumentCommand \ratio { } { \mathratio }`

`\coloncolonequals` 3238 `\RenewDocumentCommand \coloncolon { } { \Colon }`

3239 `\RenewDocumentCommand \minuscolon { } { \dashcolon }`

3240 `\RenewDocumentCommand \colonequals { } { \coloneq }`

3241 `\RenewDocumentCommand \equalscolon { } { \eqcolon }`

3242 `\RenewDocumentCommand \coloncolonequals { } { \Coloneq }`

3243 `}`

3244 `\ExplSyntaxOff`

3245 `\</package & (X E j L U)>`

17 Error messages

These are defined at the beginning of the package, but we leave their definition until now in the source to keep them out of the way.

```

3246 <*msg>

      Wrapper functions:
3247 \cs_new:Npn \um_warning:n { \msg_warning:nn {unicode-math} }
3248 \cs_new:Npn \um_log:n { \msg_log:nn {unicode-math} }
3249 \cs_new:Npn \um_log:nx { \msg_log:nxx {unicode-math} }

3250 \msg_new:nnn {unicode-math} {no-tfrac}
3251 {
3252   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\\
3253   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
3254 }
3255 \msg_new:nnn {unicode-math} {default-math-font}
3256 {
3257   Defining~ the~ default~ maths~ font~ as~ '\l_um_fontname_tl'.
3258 }
3259 \msg_new:nnn {unicode-math} {setup-implicit}
3260 {
3261   Setup~ alphabets:~ implicit~ mode.
3262 }
3263 \msg_new:nnn {unicode-math} {setup-explicit}
3264 {
3265   Setup~ alphabets:~ explicit~ mode.
3266 }
3267 \msg_new:nnn {unicode-math} {alph-initialise}
3268 {
3269   Initialising~ \@backslashchar math#1.
3270 }
3271 \msg_new:nnn {unicode-math} {setup-alph}
3272 {
3273   Setup~ alphabet:~ #1.
3274 }
3275 \msg_new:nnn { unicode-math } { missing-alphabets }
3276 {
3277   Missing~math~alphabets~in~font~ "\fontname\l_um_font" \\ \\
3278   \seq_map_function:NN \l_um_missing_alph_seq \um_print_indent:n
3279 }
3280 \cs_new:Nn \um_print_indent:n { \space\space\space\space #1 \\ }
3281 \msg_new:nnn {unicode-math} {macro-expected}
3282 {
3283   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
3284 }
3285 \msg_new:nnn {unicode-math} {wrong-meaning}
3286 {
3287   I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ mean-
      ing~ #2.
3288 }

```



```

3289 \msg_new:nnn {unicode-math} {patch-macro}
3290 {
3291   I'm~ going~ to~ patch~ macro~ #1.
3292 }
3293 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
3294   Using~ \token_to_str:N \overbracket\ and~
3295   \token_to_str:N \underbracket\ from~
3296   \mathtools'~ package.\\
3297   \\
3298   Use~ \token_to_str:N \Uoverbracket\ and~
3299   \token_to_str:N \Uunderbracket\ for~
3300   original~ \unicode-math'~ definition.
3301 }
3302 \msg_new:nnn { unicode-math } { mathtools-colon } {
3303   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
3304   the~ \mathtools'~ package: \\ \\
3305   \ \ \ \ \token_to_str:N \dblcolon,~
3306   \token_to_str:N \coloneqq,~
3307   \token_to_str:N \Coloneqq,~
3308   \token_to_str:N \eqqcolon. \\ \\
3309   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
3310   commands,~ using~ them~ will~ lead~ to~ inconsistencies.
3311 }
3312 \msg_new:nnn { unicode-math } { colonequals } {
3313   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
3314   the~ \colonequals'~ package: \\ \\
3315   \ \ \ \ \token_to_str:N \ratio,~
3316   \token_to_str:N \coloncolon,~
3317   \token_to_str:N \minuscolon, \\
3318   \ \ \ \ \token_to_str:N \colonequals,~
3319   \token_to_str:N \equalscolon,~
3320   \token_to_str:N \coloncolonequals. \\ \\
3321   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
3322   commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
3323   Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
3324   or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
3325   any~ effect~ on~ the~ re-defined~ commands.
3326 }
3327 </msg>

```

The end.

18 STIX table data extraction

The source for the T_EX names for the very large number of mathematical glyphs are provided via Barbara Beeton's table file for the STIX project (ams.org/STIX). A version is located at <http://www.ams.org/STIX/bnb/stix-tbl.asc> but check <http://www.ams.org/STIX/> for more up-to-date info.

This table is converted into a form suitable for reading by X_YT_EX. A single file

is produced containing all (more than 3298) symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols. Performance for now seems to be acceptable without such measures.

This file is currently developed outside this DTX file. It will be incorporated when the final version is ready. (I know this is not how things are supposed to work!)

3328 < See `stix-extract.sh` for now. >

A Documenting maths support in the NFSS

In the following, $\langle NFSS\ decl. \rangle$ stands for something like $\{\mathrm{T1}\}\{\mathrm{lmr}\}\{\mathrm{m}\}\{\mathrm{n}\}$.

Maths symbol fonts Fonts for symbols: $\alpha, \leq, \rightarrow$

`\DeclareSymbolFont{\langle name \rangle}\langle NFSS decl. \rangle`

Declares a named maths font such as operators from which symbols are defined with `\DeclareMathSymbol`.

Maths alphabet fonts Fonts for $ABC-xyz, \mathfrak{ABC}-\mathcal{XYZ}$, etc.

`\DeclareMathAlphabet{\langle cmd \rangle}\langle NFSS decl. \rangle`

For commands such as `\mathbf`, accessed through maths mode that are unaffected by the current text font, and which are used for alphabetic symbols in the ASCII range.

`\DeclareSymbolFontAlphabet{\langle cmd \rangle}\{\langle name \rangle\}`

Alternative (and optimisation) for `\DeclareMathAlphabet` if a single font is being used for both alphabetic characters (as above) and symbols.

Maths ‘versions’ Different maths weights can be defined with the following, switched in text with the `\mathversion{\langle maths version \rangle}` command.

`\SetSymbolFont{\langle name \rangle}\{\langle maths version \rangle}\langle NFSS decl. \rangle`

`\SetMathAlphabet{\langle cmd \rangle}\{\langle maths version \rangle}\langle NFSS decl. \rangle`

Maths symbols Symbol definitions in maths for both characters (=) and macros (`\eqdef`): `\DeclareMathSymbol{\langle symbol \rangle}\{\langle type \rangle\}\{\langle named font \rangle\}\{\langle slot \rangle\}` This is the macro that actually defines which font each symbol comes from and how they behave.

Delimiters and radicals use wrappers around TeX’s `\delimiter`/`\radical` primitives, which are re-designed in X_YTeX. The syntax used in L^ATeX’s NFSS is therefore not so relevant here.

Delimiters A special class of maths symbol which enlarge themselves in certain contexts.

`\DeclareMathDelimiter{\langle symbol \rangle}\{\langle type \rangle\}\{\langle sym. font \rangle\}\{\langle slot \rangle\}\{\langle sym. font \rangle\}\{\langle slot \rangle\}`

Radicals Similar to delimiters (`\DeclareMathRadical` takes the same syntax) but behave ‘weirdly’.

In those cases, glyph slots in *two* symbol fonts are required; one for the small (‘regular’) case, the other for situations when the glyph is larger. This is not the case in X_YTeX.

Accents are not included yet.

Summary For symbols, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathchardef#1"\mathchar@type#2
  \expandafter\hexnumber@\csname sym#2\endcsname
  {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

For characters, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathcode`#1"\mathchar@type#2
  \expandafter\hexnumber@\csname sym#2\endcsname
  {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

B Legacy T_EX font dimensions

Text fonts		Maths font, \fam2		Maths font, \fam3	
ϕ_1	slant per pt	σ_5	x height	ξ_8	default rule thickness
ϕ_2	interword space	σ_6	quad	ξ_9	big op spacing1
ϕ_3	interword stretch	σ_8	num1	ξ_{10}	big op spacing2
ϕ_4	interword shrink	σ_9	num2	ξ_{11}	big op spacing3
ϕ_5	x-height	σ_{10}	num3	ξ_{12}	big op spacing4
ϕ_6	quad width	σ_{11}	denom1	ξ_{13}	big op spacing5
ϕ_7	extra space	σ_{12}	denom2		
ϕ_8	cap height (X _Y TeX only)	σ_{13}	sup1		
		σ_{14}	sup2		
		σ_{15}	sup3		
		σ_{16}	sub1		
		σ_{17}	sub2		
		σ_{18}	sup drop		
		σ_{19}	sub drop		
		σ_{20}	delim1		
		σ_{21}	delim2		
		σ_{22}	axis height		

C X_YTeX math font dimensions

These are the extended \fontdimens available for suitable fonts in X_YTeX. Note that LuaTeX takes an alternative route, and this package will eventually provide a wrapper interface to the two (I hope).

\fontdimen	Dimension name	Description
10	SCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 1. Suggested value: 80%.
11	SCRIPTSCRIPTPERCENTSCALE-DOWN	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	DELIMITEDSUBFORMULAMIN-HEIGHT	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height \times 1.5.
13	DISPLAYOPERATORMINHEIGHT	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.
14	MATHLEADING	White space to be left between math formulas to ensure proper line spacing. For example, for applications that treat line gap as a part of line ascender, formulas with ink going above (os2.sTypoAscender + os2.sTypoLineGap – MathLeading) or with ink going below os2.sTypoDescender will result in increasing line height.
15	AXISHEIGHT	Axis height of the font.
16	ACCENTBASEHEIGHT	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (os2.sxHeight) plus any possible overshots.
17	FLATTENEDACCENTBASE-HEIGHT	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font (os2.sCapHeight).
18	SUBSCRIPTSHIFTDOWN	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: os2.ySubscriptYOffset.
19	SUBSCRIPTTOPMAX	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: $/5$ x-height.
20	SUBSCRIPTBASELINEDROPMIN	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.

\fontdimen	Dimension name	Description
21	SUPERSCRIPSHIFTUP	Standard shift up applied to superscript elements. Suggested: <code>os2.ySuperscriptYOffset</code> .
22	SUPERSCRIPSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPBTOMMIN	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height.
24	SUPERSCRIPBASELINEDROP-MAX	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SUBSUPERSCRIPGAPMIN	Minimum gap between the superscript and subscript ink. Suggested: $4 \times$ default rule thickness.
26	SUPERSCRIPBTOMMAX-WITHSUBSCRIPT	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: $\frac{1}{5}$ x-height.
27	SPACEAFTERSCRIPT	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	UPPERLIMITGAPMIN	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	UPPERLIMITBASELINERISEMIN	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	LOWERLIMITGAPMIN	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LOWERLIMITBASELINEDROP-MIN	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	STACKTOPSHIFTUP	Standard shift up applied to the top element of a stack.
33	STACKTOPDISPLAYSTYLESHIFT-UP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.

\fontdimen	Dimension name	Description
35	STACKBOTTOMDISPLAYSTYLE-SHIFTDOWN	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	STACKGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.
37	STACKDISPLAYSTYLEGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	STRETCHSTACKTOPSHIFTUP	Standard shift up applied to the top element of the stretch stack.
39	STRETCHSTACKBOTTOMSHIFT-DOWN	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	STRETCHSTACKGAPABOVEMIN	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: UpperLimitGapMin
41	STRETCHSTACKGAPBELOWMIN	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: LowerLimitGapMin.
42	FRACTIONNUMERATORSHIFTUP	Standard shift up applied to the numerator.
43	FRACTIONNUMERATOR-DISPLAYSTYLESHIFTUP	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FRACTIONDENOMINATORSHIFT-DOWN	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FRACTIONDENOMINATOR-DISPLAYSTYLESHIFTDOWN	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FRACTIONNUMERATORGAP-MIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness

\fontdimen	Dimension name	Description
47	FRACTIONNUMDISPLAYSTYLE-GAPMIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
48	FRACTIONRULETHICKNESS	Thickness of the fraction bar. Suggested: default rule thickness.
49	FRACTIONDENOMINATORGAP-MIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FRACTIONDENOMDISPLAY-STYLEGAPMIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
51	SKEWEDFRACTION-HORIZONTALGAP	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SKEWEDFRACTIONVERTICAL-GAP	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OVERBARVERTICALGAP	Distance between the overbar and the (ink) top of the base. Suggested: 3×default rule thickness.
54	OVERBARRULETHICKNESS	Thickness of overbar. Suggested: default rule thickness.
55	OVERBAREXTRAASCENDER	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UNDERBARVERTICALGAP	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UNDERBARRULETHICKNESS	Thickness of underbar. Suggested: default rule thickness.
58	UNDERBAREXTRADESCENDER	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RADICALVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RADICALDISPLAYSTYLE-VERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.

\fontdimen	Dimension name	Description
61	RADICALRULETHICKNESS	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RADICALEXTRAASCENDER	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.
63	RADICALKERNBEFOREDEGREE	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RADICALKERNAFTERDEGREE	Negative kern after the degree of a radical, if such is present. Suggested: -10/18 of em.
65	RADICALDEGREEBOTTOM-RAISEPERCENT	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.