

Experimental unicode mathematical typesetting: The unicode-math package

Will Robertson

2009/10/22 v0.4

Abstract

Warning! This package is experimental and subject to change without regard for backwards compatibility. Performance issues may be encountered until algorithms are refined.

This package is intended to be a complete implementation of unicode maths for \LaTeX using the \XeTeX (and later, \LuaTeX) typesetting engines. With this package, changing maths fonts will be as easy as changing text fonts — not that there are many unicode maths fonts yet.

Maths input is simplified with unicode since literal glyphs may be entered instead of control sequences.

Contents

1	Introduction	3			
2	Acknowledgements	3			
3	Getting started	3			
3.1	Package options	3			
4	Unicode maths font setup	4			
4.1	Using multiple fonts	5			
4.2	Script and scriptscript fonts/features	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			
I	The unicode-math package	16			
6	Things we need	16			
6.1	Options	23			
6.2	Overcoming <code>\@on-</code> preamble	27			
6.3	Other things	27			
7	Fundamentals	28			
7.1	Enlarging the number of maths families	28			
			7.2	<code>\DeclareMathSymbol</code> for unicode ranges	28
			7.3	The main <code>\setmathfont</code> macro	30
			7.4	(Big) operators	37
			7.5	Radicals	40
			7.6	Delimiters	40
			7.7	Maths accents	42
			8	Font features	43
			8.1	OpenType maths font features	44
			8.2	Script and scriptscript font options	44
			8.3	Range processing	44
			8.4	Resolving Greek symbol name control sequences	51
			9	Maths alphabets mapping definitions	51
			9.1	Alphabets	56
			10	Definitions of the math symbols	68
			11	Epilogue	69
			12	stix table data extraction	80
			A	Documenting maths support in the NFSS	81
			B	X_YTeX math font dimensions	83

1 Introduction

This document describes the unicode-math package, which is an *experimental* implementation of a macro to unicode glyph encoding for mathematical characters. Its intended use is for \XeTeX , although it is conjectured that some effort could be spent to create a cross-format package that would also work with $\text{Lua}\TeX$.

Users who desire to specify maths alphabets only (Greek and Latin letters) may wish to use Andrew Moschou’s mathspec package instead.

2 Acknowledgements

Many thanks to Microsoft for developing OpenType math as part of Office 2007; Jonathan Kew for implementing unicode math support in $\text{Xe}\TeX$; Barbara Beeton for her prodigious effort compiling the definitive list of unicode math glyphs and their $\text{L}\text{A}\text{T}\text{E}\text{X}$ names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions. 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 TEX in the future. Apostolos Syropoulos, Joel Salomon, and Khaled Hosny have been fantastic beta testers.

3 Getting started

Load unicode-math as a regular $\text{L}\text{A}\text{T}\text{E}\text{X}$ 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{Cambria Math}
```

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}
```

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

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

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.

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., `\mathcal{H}` to \mathcal{H} and so on, all for unicode glyphs 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.

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

4.1 Using multiple fonts

There will probably be few cases where a single unicode maths font suffices (simply due to glyph coverage). The upcoming `srix` 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 alphabets 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=\mathbfssfit/{greek,Greek}]` for Greek lowercase and uppercase only (with `latin`, `Latin`, `num` as well for Latin lower-/upper-case and numbers).
- `[range=\mathsf->\mathbfssfit]` 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 (but why?) 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_{B_C}). Other fonts will possibly use entirely separate fonts.

Not yet implemented: Both of these options must be taken into account. I hope this will be mostly automatic from the users' points of view. The `+ssty` feature can be detected and applied automatically, and appropriate optical size information embedded in the fonts will ensure this latter case. Fine tuning should be possible automatically with `fontspec` options. We might have to wait until MnMath, for example, before we really know.

5 Maths input

X_YTeX's unicode support allows maths input through two methods. Like classical TeX, 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, TeX 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 (contrary again, *quelle surprise*) 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.

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 insensitive).

The philosophy behind the interface to the mathematical alphabet symbols lies in L^ATeX'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.

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)	$(\alpha, \beta, \Gamma, \Xi)$

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 $\xi = (\xi_r, \xi_\varphi, \xi_\theta)$. Confusingly, the syntax in \LaTeX has been different for these two examples: `\mathbf{f}` in the former (‘ \mathbf{M} ’), and `\bm` (or `\boldsymbol`, deprecated) in the latter (‘ ξ ’).

In `unicode-math`, the `\mathbf{f}` 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.

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

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

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical alphabet characters. These may be explicitly selected with the `\mathsfup`, `\mathsfit`, `\mathbfsup`, and `\mathbfsfit` 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 L^AT_EX’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 the seemingly-useless 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 `\mathsfit`, 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 conceivable). 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 `\mathbfsup` 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 ’.

Table 5: Mathematical alphabets defined in unicode. Black dots indicate an alphabet exists in the font specified; grey 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>\mathsfbfup</code>	•	•	•
	Italic	Bold	<code>\mathsfbfit</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>\matbfscr</code>	•		
Fraktur	Upright	Normal	<code>\mathfrak</code>	•		
		Bold	<code>\mathbffrac</code>	•		

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 $\{a-z, A-Z\}$, numerals $\{0-9\}$, summation symbol Σ , and four Greek letters only: $\{\gamma, \Omega, \Gamma, \Pi\}$.

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

There are also five Latin *italic* double-struck letters: $\mathbb{D}, \mathbb{d}, \mathbb{e}, \mathbb{i}, \mathbb{j}$. These can be accessed (if not with their literal characters or control sequences) with the `\mathbbit`

Table 6: The various forms of nabla.

Description		Glyph
Upright	Serif	∇
	Bold serif	∇
	Bold sans	∇
Italic	Serif	∇
	Bold serif	∇
	Bold sans	∇

alphabet switch, but note that only those five letters will give the expected output.

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). \TeX classically uses an upright nabla, but iso standards differ (I think). The package options `nabla=upright` and `nabla=italic` switch between the two choices. This is then inherited through `\mathbf`; `\mathit` and `\mathup` can be used to force one way or the other.

`nabla=italic` is implicit when using `math-style=ISO` and `nabla=upright` follows both `math-style=TeX` and `math-style=French`.

5.5.2 Partial

The same applies to the symbols `u+2202` partial differential and `u+1D715` math italic partial differential.

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. The default is (always, unless someone requests and argues otherwise) `partial=italic`.¹

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

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.

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	∂
Bold	Upright	∂
	Italic	∂
Sans bold	Upright	
	Italic	

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 . 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 . 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 of ASCII straight quote (’), unicode prime `\u2032` (’), and `\prime`; 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. These may also be accessed 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
```

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.

Backwards or reverse primes behave in exactly the same way; use any of `ASCII` back tick (```), unicode reverse prime `u+2035` (`'`), or `\backprime` to access it. Multiple backwards primes can also be called with `\backdprime`, `\backtrprime`, and `\backqprime`.

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'`.

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.

Table 8: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	<code>\solidus</code>
U+2044	FRACTION SLASH	/	<code>\fracslash</code>
U+2215	DIVISION SLASH	/	<code>\slash</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>

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 L^AT_EX 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).

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 L^AT_EX command name `\smallsetminus` is used for backwards compatibility.

²§4.4.5.11 <http://www.w3.org/TR/2000/REC-MATHML3/>

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 L^AT_EX 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

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

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`;
- `\frac slash`;
- `\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 `\left\frac slash` 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.

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 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.

L^AT_EX defines considerably fewer: `\circ` and `\bigcirc` for white; `\bullet` for black. This package maps those commands to `\vysmwhtcircle`, `\mdlgwhtcircle`, and `\smblkcircle`, respectively.

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

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdotp</code>	.			
U+22C5	<code>\cdot</code>	.			
U+2219	<code>\vysmbllkcircle</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>\mdbllkcircle</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.

5.5.9 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. Namely, Δ and \vartriangle and \triangle and Δ . 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.

But you’re better off if you’re using the latter form to indicate an increment to use the glyph intended for this purpose: Δx .

Finally, given that Δ and \vartriangle 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.

5.5.10 Normalising some input characters

I believe all variant forms should be used as legal input that is normalised to a consistent output glyph, because we want to be fault-tolerant in the input. Here are the duplicates:

u+251 latin small letter alpha
 u+25B latin small letter epsilon
 u+263 latin small letter gamma
 u+269 latin small letter iota
 u+278 latin small letter phi
 u+28A latin small letter upsilon
 u+190 latin capital letter epsilon
 u+194 latin capital letter gamma
 u+196 latin capital letter iota
 u+1B1 latin capital letter upsilon

(Not yet implemented.)

File I

The unicode-math package

This is the package.

```

1 \ProvidesPackage{unicode-math}
2 [2009/10/22 v0.4 Unicode maths in XeLaTeX]

```

6 Things we need

Packages

```

3 \RequirePackage{expl3}[2009/08/12]
4 \RequirePackage{xparse}[2009/08/31]
5 \RequirePackage{fontspec}

```

Start using L^AT_EX3 — finally!

```

6 \ExplSyntaxOn

```

Extras we need to define:

```

7 \cs_generate_variant:Nn \tl_put_right:Nn {cx}
8 \cs_generate_variant:Nn \seq_if_in:NnTF {NV}
9 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
10 \cs_generate_variant:Nn \prop_get:NnN {cxN}
11 \cs_generate_variant:Nn \prop_if_in:NnTF {cx}

```


Counters and conditionals

```
12 \int_new:N \g_um_fam_int
13 \bool_new:N \l_um_fontspec_feature_bool
14 \bool_new:N \l_um_ot_math_bool
15 \bool_new:N \l_um_init_bool
```

For math-style:

```
16 \bool_new:N \g_um_literal_bool
17 \bool_new:N \g_um_upLatin_bool
18 \bool_new:N \g_um_uplatin_bool
19 \bool_new:N \g_um_upGreek_bool
20 \bool_new:N \g_um_upgreek_bool
```

For bold-style:

```
21 \bool_new:N \g_um_bfliteral_bool
22 \bool_new:N \g_um_bfupLatin_bool
23 \bool_new:N \g_um_bfuplatin_bool
24 \bool_new:N \g_um_bfupGreek_bool
25 \bool_new:N \g_um_bfupgreek_bool
```

For nabla:

```
26 \bool_new:N \g_um_upNabla_bool
27 \bool_new:N \g_um_uppartial_bool
28 \bool_new:N \g_um_texgreek_bool
```

6.0.11 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.

```
29 \cs_new:Nn \usv_set:nnn {
30   \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}
31 }
32 \cs_new:Nn \um_to_usv:nn { g_um_#1_#2_usv }
33
34
35 \usv_set:nnn {up}{B}{`\B}
36 \usv_set:nnn {up}{C}{`\C}
37 \usv_set:nnn {up}{D}{`\D}
38 \usv_set:nnn {up}{E}{`\E}
39 \usv_set:nnn {up}{F}{`\F}
40 \usv_set:nnn {up}{H}{`\H}
41 \usv_set:nnn {up}{I}{`\I}
42 \usv_set:nnn {up}{L}{`\L}
43 \usv_set:nnn {up}{M}{`\M}
44 \usv_set:nnn {up}{N}{`\N}
```

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

```

45 \usv_set:nnn {up}{P}{`\P}
46 \usv_set:nnn {up}{Q}{`\Q}
47 \usv_set:nnn {up}{R}{`\R}
48 \usv_set:nnn {up}{Z}{`\Z}
49
50 \usv_set:nnn {it}{B}{"1D435}
51 \usv_set:nnn {it}{C}{"1D436}
52 \usv_set:nnn {it}{D}{"1D437}
53 \usv_set:nnn {it}{E}{"1D438}
54 \usv_set:nnn {it}{F}{"1D439}
55 \usv_set:nnn {it}{H}{"1D43B}
56 \usv_set:nnn {it}{I}{"1D43C}
57 \usv_set:nnn {it}{L}{"1D43F}
58 \usv_set:nnn {it}{M}{"1D440}
59 \usv_set:nnn {it}{N}{"1D441}
60 \usv_set:nnn {it}{P}{"1D443}
61 \usv_set:nnn {it}{Q}{"1D444}
62 \usv_set:nnn {it}{R}{"1D445}
63 \usv_set:nnn {it}{Z}{"1D44D}
64
65 \usv_set:nnn {up}{d}{`\d}
66 \usv_set:nnn {up}{e}{`\e}
67 \usv_set:nnn {up}{g}{`\g}
68 \usv_set:nnn {up}{h}{"0068}
69 \usv_set:nnn {up}{i}{`\i}
70 \usv_set:nnn {up}{j}{`\j}
71 \usv_set:nnn {up}{o}{`\o}
72
73 \usv_set:nnn {it}{d}{"1D451}
74 \usv_set:nnn {it}{e}{"1D452}
75 \usv_set:nnn {it}{g}{"1D454}
76 \usv_set:nnn {it}{h}{"210E}
77 \usv_set:nnn {it}{i}{"1D456}
78 \usv_set:nnn {it}{j}{"1D457}
79 \usv_set:nnn {it}{o}{"1D45C}
80
81 \usv_set:nnn {up}{num}{48}
82 \usv_set:nnn {up}{Latin}{65}
83 \usv_set:nnn {up}{latin}{97}
84 \usv_set:nnn {up}{Greek}{"391}
85 \usv_set:nnn {up}{greek}{"3B1}
86 \usv_set:nnn {it}{Latin}{"1D434}
87 \usv_set:nnn {it}{latin}{"1D44E}
88 \usv_set:nnn {it}{Greek}{"1D6E2}
89 \usv_set:nnn {it}{greek}{"1D6FC}
90 \usv_set:nnn {bb}{num}{"1D7D8}

```

91 \usv_set:nnn {bb}{Latin}{ "1D538}
 92 \usv_set:nnn {bb}{latin}{ "1D552}
 93 \usv_set:nnn {scr}{Latin}{ "1D49C}
 94 \usv_set:nnn {scr}{latin}{ "1D4B6}
 95 \usv_set:nnn {frak}{Latin}{ "1D504}
 96 \usv_set:nnn {frak}{latin}{ "1D51E}
 97 \usv_set:nnn {sf}{num}{ "1D7E2}
 98 \usv_set:nnn {sfup}{num}{ "1D7E2}
 99 \usv_set:nnn {sfit}{num}{ "1D7E2}
 100 \usv_set:nnn {sfup}{Latin}{ "1D5A0}
 101 \usv_set:nnn {sf}{Latin}{ "1D5A0}
 102 \usv_set:nnn {sfup}{latin}{ "1D5BA}
 103 \usv_set:nnn {sf}{latin}{ "1D5BA}
 104 \usv_set:nnn {sfit}{Latin}{ "1D608}
 105 \usv_set:nnn {sfit}{latin}{ "1D622}
 106 \usv_set:nnn {tt}{num}{ "1D7F6}
 107 \usv_set:nnn {tt}{Latin}{ "1D670}
 108 \usv_set:nnn {tt}{latin}{ "1D68A}

Bold:

109 \usv_set:nnn {bf}{num}{ "1D7CE}
 110 \usv_set:nnn {bfup}{num}{ "1D7CE}
 111 \usv_set:nnn {bfit}{num}{ "1D7CE}
 112 \usv_set:nnn {bfup}{Latin}{ "1D400}
 113 \usv_set:nnn {bfup}{latin}{ "1D41A}
 114 \usv_set:nnn {bfup}{Greek}{ "1D6A8}
 115 \usv_set:nnn {bfup}{greek}{ "1D6C2}
 116 \usv_set:nnn {bfit}{Latin}{ "1D468}
 117 \usv_set:nnn {bfit}{latin}{ "1D482}
 118 \usv_set:nnn {bfit}{Greek}{ "1D71C}
 119 \usv_set:nnn {bfit}{greek}{ "1D736}
 120 \usv_set:nnn {bffrak}{Latin}{ "1D56C}
 121 \usv_set:nnn {bffrak}{latin}{ "1D586}
 122 \usv_set:nnn {bfscr}{Latin}{ "1D4D0}
 123 \usv_set:nnn {bfscr}{latin}{ "1D4EA}
 124 \usv_set:nnn {bfsf}{num}{ "1D7EC}
 125 \usv_set:nnn {bfsfup}{num}{ "1D7EC}
 126 \usv_set:nnn {bfsfit}{num}{ "1D7EC}
 127 \usv_set:nnn {bfsfup}{Latin}{ "1D5D4}
 128 \usv_set:nnn {bfsfup}{latin}{ "1D5EE}
 129 \usv_set:nnn {bfsfup}{Greek}{ "1D756}
 130 \usv_set:nnn {bfsfup}{greek}{ "1D770}
 131 \usv_set:nnn {bfsfit}{Latin}{ "1D63C}
 132 \usv_set:nnn {bfsfit}{latin}{ "1D656}
 133 \usv_set:nnn {bfsfit}{Greek}{ "1D790}
 134 \usv_set:nnn {bfsfit}{greek}{ "1D7AA}

135 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsf

\backslash usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsf
 \backslash usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsf
 \backslash usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsf
 \backslash usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_L
 \backslash usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_L
 \backslash usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_G
 \backslash usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_g

Greek variants:

\backslash usv_set:nnn {up}{varTheta}{ "3F4}
 \backslash usv_set:nnn {up}{Digamma}{ "3DC}
 \backslash usv_set:nnn {up}{varepsilon}{ "3F5}
 \backslash usv_set:nnn {up}{vartheta}{ "3D1}
 \backslash usv_set:nnn {up}{varkappa}{ "3F0}
 \backslash usv_set:nnn {up}{varphi}{ "3D5}
 \backslash usv_set:nnn {up}{varrho}{ "3F1}
 \backslash usv_set:nnn {up}{varpi}{ "3D6}
 \backslash usv_set:nnn {up}{digamma}{ "3DD}

Bold:

\backslash usv_set:nnn {bfup}{varTheta}{ "1D6B9}
 \backslash usv_set:nnn {bfup}{Digamma}{ "1D7CA}
 \backslash usv_set:nnn {bfup}{varepsilon}{ "1D6DC}
 \backslash usv_set:nnn {bfup}{vartheta}{ "1D6DD}
 \backslash usv_set:nnn {bfup}{varkappa}{ "1D6DE}
 \backslash usv_set:nnn {bfup}{varphi}{ "1D6DF}
 \backslash usv_set:nnn {bfup}{varrho}{ "1D6E0}
 \backslash usv_set:nnn {bfup}{varpi}{ "1D6E1}
 \backslash usv_set:nnn {bfup}{digamma}{ "1D7CB}

Italic Greek variants:

\backslash usv_set:nnn {it}{varTheta}{ "1D6F3}
 \backslash usv_set:nnn {it}{varepsilon}{ "1D716}
 \backslash usv_set:nnn {it}{vartheta}{ "1D717}
 \backslash usv_set:nnn {it}{varkappa}{ "1D718}
 \backslash usv_set:nnn {it}{varphi}{ "1D719}
 \backslash usv_set:nnn {it}{varrho}{ "1D71A}
 \backslash usv_set:nnn {it}{varpi}{ "1D71B}

Bold italic:

\backslash usv_set:nnn {bfit}{varTheta}{ "1D72D}
 \backslash usv_set:nnn {bfit}{varepsilon}{ "1D750}
 \backslash usv_set:nnn {bfit}{vartheta}{ "1D751}
 \backslash usv_set:nnn {bfit}{varkappa}{ "1D752}
 \backslash usv_set:nnn {bfit}{varphi}{ "1D753}
 \backslash usv_set:nnn {bfit}{varrho}{ "1D754}
 \backslash usv_set:nnn {bfit}{varpi}{ "1D755}

Bold sans:

175 \usv_set:nnn {bfsfup}{varTheta}{1D767}
 176 \usv_set:nnn {bfsfup}{varepsilon}{1D78A}
 177 \usv_set:nnn {bfsfup}{vartheta}{1D78B}
 178 \usv_set:nnn {bfsfup}{varkappa}{1D78C}
 179 \usv_set:nnn {bfsfup}{varphi}{1D78D}
 180 \usv_set:nnn {bfsfup}{varrho}{1D78E}
 181 \usv_set:nnn {bfsfup}{varpi}{1D78F}

Bold sans italic:

182 \usv_set:nnn {bfsfit}{varTheta}{1D7A1}
 183 \usv_set:nnn {bfsfit}{varepsilon}{1D7C4}
 184 \usv_set:nnn {bfsfit}{vartheta}{1D7C5}
 185 \usv_set:nnn {bfsfit}{varkappa}{1D7C6}
 186 \usv_set:nnn {bfsfit}{varphi}{1D7C7}
 187 \usv_set:nnn {bfsfit}{varrho}{1D7C8}
 188 \usv_set:nnn {bfsfit}{varpi}{1D7C9}

Nabla:

189 \usv_set:nnn {up}{Nabla}{2207}
 190 \usv_set:nnn {it}{Nabla}{1D6FB}
 191 \usv_set:nnn {bfup}{Nabla}{1D6C1}
 192 \usv_set:nnn {bfit}{Nabla}{1D735}
 193 \usv_set:nnn {bfsfup}{Nabla}{1D76F}
 194 \usv_set:nnn {bfsfit}{Nabla}{1D7A9}

Partial:

195 \usv_set:nnn {up}{partial}{2202}
 196 \usv_set:nnn {it}{partial}{1D715}
 197 \usv_set:nnn {bfup}{partial}{1D6DB}
 198 \usv_set:nnn {bfit}{partial}{1D74F}
 199 \usv_set:nnn {bfsfup}{partial}{1D789}
 200 \usv_set:nnn {bfsfit}{partial}{1D7C3}

Latin 'h':

201 \usv_set:nnn {bb}{h}{1D559}
 202 \usv_set:nnn {tt}{h}{1D691}
 203 \usv_set:nnn {scr}{h}{1D4BD}
 204 \usv_set:nnn {frak}{h}{1D525}
 205 \usv_set:nnn {bfup}{h}{1D421}
 206 \usv_set:nnn {bfit}{h}{1D489}
 207 \usv_set:nnn {sfup}{h}{1D5C1}
 208 \usv_set:nnn {sfit}{h}{1D629}
 209 \usv_set:nnn {bffrak}{h}{1D58D}
 210 \usv_set:nnn {bfscr}{h}{1D4F1}
 211 \usv_set:nnn {bfsfup}{h}{1D5F5}
 212 \usv_set:nnn {bfsfit}{h}{1D65D}

Blackboard:

213 \usv_set:nnn {bb}{C}{2102}

214 \usv_set:nnn {bb}{H}{ "210D}
 215 \usv_set:nnn {bb}{N}{ "2115}
 216 \usv_set:nnn {bb}{P}{ "2119}
 217 \usv_set:nnn {bb}{Q}{ "211A}
 218 \usv_set:nnn {bb}{R}{ "211D}
 219 \usv_set:nnn {bb}{Z}{ "2124}
 220 \usv_set:nnn {up}{Pi} {"03A0}
 221 \usv_set:nnn {up}{pi} {"03C0}
 222 \usv_set:nnn {up}{Gamma} {"0393}
 223 \usv_set:nnn {up}{gamma} {"03B3}
 224 \usv_set:nnn {up}{summation}{ "2211}
 225 \usv_set:nnn {it}{Pi} {"1D6F1}
 226 \usv_set:nnn {it}{pi} {"1D70B}
 227 \usv_set:nnn {it}{Gamma} {"1D6E4}
 228 \usv_set:nnn {it}{gamma} {"1D6FE}
 229 \usv_set:nnn {bb}{Pi} {"213F}
 230 \usv_set:nnn {bb}{pi} {"213C}
 231 \usv_set:nnn {bb}{Gamma} {"213E}
 232 \usv_set:nnn {bb}{gamma} {"213D}
 233 \usv_set:nnn {bb}{summation}{ "2140}

Italic blackboard:

234 \usv_set:nnn {bbi}{D}{ "2145}
 235 \usv_set:nnn {bbi}{d}{ "2146}
 236 \usv_set:nnn {bbi}{e}{ "2147}
 237 \usv_set:nnn {bbi}{i}{ "2148}
 238 \usv_set:nnn {bbi}{j}{ "2149}

Script exceptions:

239 \usv_set:nnn {scr}{B}{ "212C}
 240 \usv_set:nnn {scr}{E}{ "2130}
 241 \usv_set:nnn {scr}{F}{ "2131}
 242 \usv_set:nnn {scr}{H}{ "210B}
 243 \usv_set:nnn {scr}{I}{ "2110}
 244 \usv_set:nnn {scr}{L}{ "2112}
 245 \usv_set:nnn {scr}{M}{ "2133}
 246 \usv_set:nnn {scr}{R}{ "211B}
 247 \usv_set:nnn {scr}{e}{ "212F}
 248 \usv_set:nnn {scr}{g}{ "210A}
 249 \usv_set:nnn {scr}{o}{ "2134}

Fraktur exceptions:

250 \usv_set:nnn {frak}{C}{ "212D}
 251 \usv_set:nnn {frak}{H}{ "210C}
 252 \usv_set:nnn {frak}{I}{ "2111}
 253 \usv_set:nnn {frak}{R}{ "211C}
 254 \usv_set:nnn {frak}{Z}{ "2128}

6.1 Options

xkeyval's package support is used here. I'll switch over to l3keys2e at some stage.

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

```
255 \DeclareDocumentCommand \unimathsetup {m} {  
256   \setkeys{unicode-math.sty}{#1}  
257 }
```

math-style

```
258 \define@choicekey*{unicode-math.sty}  
259   {math-style}[\@tempa\@tempb]{iso,tex,french,upright,literal}{  
260   \ifcase\@tempb\relax  
261     \bool_set_false:N \g_um_upGreek_bool  
262     \bool_set_false:N \g_um_upgreek_bool  
263     \bool_set_false:N \g_um_upLatin_bool  
264     \bool_set_false:N \g_um_uplatin_bool  
265     \bool_set_false:N \g_um_bfupGreek_bool  
266     \bool_set_false:N \g_um_bfupgreek_bool  
267     \bool_set_false:N \g_um_bfupLatin_bool  
268     \bool_set_false:N \g_um_bfuplatin_bool  
269     \bool_set_false:N \g_um_upNabla_bool  
270     \bool_set_false:N \g_um_uppartial_bool  
271     \bool_set_false:N \g_um_upsans_bool  
272     \bool_set_false:N \g_um_texgreek_bool  
273     \bool_set_false:N \g_um_literal_bool  
274   \or  
275     \bool_set_true:N \g_um_upGreek_bool  
276     \bool_set_false:N \g_um_upgreek_bool  
277     \bool_set_false:N \g_um_upLatin_bool  
278     \bool_set_false:N \g_um_uplatin_bool  
279     \bool_set_true:N \g_um_bfupGreek_bool  
280     \bool_set_false:N \g_um_bfupgreek_bool  
281     \bool_set_true:N \g_um_bfupLatin_bool  
282     \bool_set_true:N \g_um_bfuplatin_bool  
283     \bool_set_true:N \g_um_upNabla_bool  
284     \bool_set_false:N \g_um_uppartial_bool  
285     \bool_set_true:N \g_um_upsans_bool  
286     \bool_set_false:N \g_um_texgreek_bool  
287     \bool_set_false:N \g_um_literal_bool  
288   \or  
289     \bool_set_true:N \g_um_upGreek_bool  
290     \bool_set_true:N \g_um_upgreek_bool  
291     \bool_set_true:N \g_um_upLatin_bool  
292     \bool_set_false:N \g_um_uplatin_bool
```

```

293 \bool_set_true:N \g_um_bfupGreek_bool
294 \bool_set_true:N \g_um_bfupgreek_bool
295 \bool_set_true:N \g_um_bfupLatin_bool
296 \bool_set_true:N \g_um_bfuplatin_bool
297 \bool_set_true:N \g_um_upNabla_bool
298 \bool_set_true:N \g_um_uppartial_bool
299 \bool_set_true:N \g_um_upsans_bool
300 \bool_set_false:N \g_um_texgreek_bool
301 \bool_set_false:N \g_um_literal_bool
302 \or
303 \bool_set_true:N \g_um_upGreek_bool
304 \bool_set_true:N \g_um_upgreek_bool
305 \bool_set_true:N \g_um_upLatin_bool
306 \bool_set_true:N \g_um_uplatin_bool
307 \bool_set_true:N \g_um_bfupGreek_bool
308 \bool_set_true:N \g_um_bfupgreek_bool
309 \bool_set_true:N \g_um_bfupLatin_bool
310 \bool_set_true:N \g_um_bfuplatin_bool
311 \bool_set_true:N \g_um_upNabla_bool
312 \bool_set_true:N \g_um_uppartial_bool
313 \bool_set_true:N \g_um_upsans_bool
314 \bool_set_false:N \g_um_texgreek_bool
315 \bool_set_false:N \g_um_literal_bool
316 \or
317 \bool_set_true:N \g_um_literal_bool
318 \bool_set_true:N \g_um_bfliteral_bool
319 \bool_set_true:N \g_um_sfliteral_bool
320 \bool_set_false:N \g_um_texgreek_bool
321 \fi
322 }

```

bold-style

```

323 \define@choicekey*{unicode-math.sty}{bold-style}[\@tempa\@tempb]{iso,tex,upright,literal}{
324 \ifcase\@tempb\relax
325 \bool_set_false:N \g_um_bfliteral_bool
326 \bool_set_false:N \g_um_bfupGreek_bool
327 \bool_set_false:N \g_um_bfupgreek_bool
328 \bool_set_false:N \g_um_bfupLatin_bool
329 \bool_set_false:N \g_um_bfuplatin_bool
330 \or
331 \bool_set_false:N \g_um_bfliteral_bool
332 \bool_set_true:N \g_um_bfupGreek_bool
333 \bool_set_false:N \g_um_bfupgreek_bool
334 \bool_set_true:N \g_um_bfupLatin_bool
335 \bool_set_true:N \g_um_bfuplatin_bool
336 \or

```



```

337 \bool_set_false:N \g_um_bfliteral_bool
338 \bool_set_true:N \g_um_bfupGreek_bool
339 \bool_set_true:N \g_um_bfupgreek_bool
340 \bool_set_true:N \g_um_bfupLatin_bool
341 \bool_set_true:N \g_um_bfuplatin_bool
342 \or
343 \bool_set_true:N \g_um_bfliteral_bool
344 \fi
345 }

```

sans-style

```

346 \bool_new:N \g_um_upsans_bool
347 \bool_new:N \g_um_sfliteral_bool
348 \define@choicekey*{unicode-math.sty}
349 {sans-style}[\@tempa\@tempb]{italic,upright,literal}{
350 \ifcase\@tempb\relax
351 \bool_set_false:N \g_um_upsans_bool
352 \or
353 \bool_set_true:N \g_um_upsans_bool
354 \or
355 \bool_set_true:N \g_um_sfliteral_bool
356 \fi
357 }

```

Symbol obliqueness

```

358 \define@choicekey*{unicode-math.sty}{nabla}[\@tempa\@tempb]{upright,italic}{
359 \ifcase\@tempb
360 \bool_set_true:N \g_um_upNabla_bool
361 \or
362 \bool_set_false:N \g_um_upNabla_bool
363 \fi
364 }
365 \cs_set:Nn \um_setup_nabla: {
366 \bool_if:NTF \g_um_upNabla_bool {
367 \tl_set:Nn \g_um_Nabla_up_or_it_usv { \g_um_up_Nabla_usv }
368 \tl_set:Nn \g_um_bfNabla_up_or_it_usv { \g_um_bfup_Nabla_usv }
369 \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfup_Nabla_usv }
370 }{
371 \tl_set:Nn \g_um_Nabla_up_or_it_usv { \g_um_it_Nabla_usv }
372 \tl_set:Nn \g_um_bfNabla_up_or_it_usv { \g_um_bfit_Nabla_usv }
373 \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfit_Nabla_usv }
374 }
375 }
376 \define@choicekey*{unicode-math.sty}{partial}[\@tempa\@tempb]{upright,italic}{
377 \ifcase\@tempb

```

```

378     \bool_set_true:N \g_um_uppartial_bool
379   \or
380     \bool_set_false:N \g_um_uppartial_bool
381   \fi
382 }
383 \cs_set:Nn \um_setup_partial: {
384   \bool_if:NTF \g_um_uppartial_bool {
385     \tl_set:Nn \g_um_partial_up_or_it_usv { \g_um_up_partial_usv }
386     \tl_set:Nn \g_um_bfpartial_up_or_it_usv { \g_um_bfup_partial_usv }
387     \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfup_partial_usv }
388   }{
389     \tl_set:Nn \g_um_partial_up_or_it_usv { \g_um_it_partial_usv }
390     \tl_set:Nn \g_um_bfpartial_up_or_it_usv { \g_um_bfit_partial_usv }
391     \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfit_partial_usv }
392   }
393 }

```

Epsilon and phi shapes

```

394 \define@choicekey*{unicode-math.sty}{vargreek-shape}[\@tempa\@tempb]{unicode,TeX}{
395   \ifcase\@tempb
396     \bool_set_false:N \g_um_texgreek_bool
397   \or
398     \bool_set_true:N \g_um_texgreek_bool
399   \fi
400 }

```

Colon style

```

401 \bool_new:N \g_um_literal_colon_bool
402 \define@choicekey*{unicode-math.sty}{colon}[\@tempa\@tempb]{literal,TeX}{
403   \ifcase\@tempb
404     \bool_set_true:N \g_um_literal_colon_bool
405   \or
406     \bool_set_false:N \g_um_literal_colon_bool
407   \fi
408 }

```

Slash delimiter style

```

409 \define@choicekey*{unicode-math.sty}{slash-delimiter}[\@tempa\@tempb]{ascii,frac,div}{
410   \ifcase\@tempb
411     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
412   \or
413     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
414   \or
415     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
416   \fi

```

```

417 }
418 \ExecuteOptionsX{math-style=TeX,slash-delimiter=ascii}
419 \ProcessOptionsX

```

6.2 Overcoming \@onlypreamble

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

```

420 \tl_map_inline:nn {
421 \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
422 \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
423 \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
424 \version@list\version@elt\alpha@list\alpha@elt
425 \restore@mathversion\init@restore@version\dorestore@version\process@table
426 \new@mathversion\DeclareSymbolFont\group@list\group@elt
427 \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
428 \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
429 \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
430 \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter\@DeclareMathDelimiter
431 \@xDeclareMathDelimiter\set@mathdelimiter\set@@@mathdelimiter\DeclareMathRadical
432 \mathchar@type\DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
433 }{
434 \tl_remove_in:Nn \@preamblecmds {\do#1}
435 }

```

6.3 Other things

`\um_fontdimen_to_percent:nn` #1 : Font dimen number
`\fontdimens` 10, 11, and 65 aren't actually dimensions, they're percentage values given in units of sp. This macro takes a font dimension number and outputs the decimal value of the associated parameter.

```

436 \def\um_fontdimen_to_percent:nn#1#2{
437   0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
438 }

```

`\um@scaled@apply` #1 : A math style
 #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.

```

439 \def\um@scaled@apply#1#2#3{
440   \ifx#1\scriptstyle
441     #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
442   \else

```

```

443 \ifx#1\scriptscriptstyle
444   #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
445 \else
446   #2#3%
447 \fi
448 \fi
449 }

```

7 Fundamentals

7.1 Enlarging the number of maths families

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

```

450 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
451 \let\newfam\new@mathgroup

```

This is sufficient for L^AT_EX’s `\DeclareSymbolFont`-type commands to be able to define 256 named maths fonts. Now we need a new `\DeclareMathSymbol`.

7.2 `\DeclareMathSymbol` for unicode ranges

This command is a bit funny at the moment; it doesn’t define the actual macro for almost all of the symbols passed to it, but it does assign the `\XeTeXmathchar`.

The final macros that actually define the maths symbol with X_ET_EX primitives.

```

\um_set_mathsymbol:nNNn #1 : Symbol font number, e.g., \symoperators
                        #2 : Symbol macro, e.g., \alpha
                        #3 : Type, e.g., \mathalpha
                        #4 : Slot, e.g., "221E

```

If the symbol definition is for a macro. There are a bunch of tests to perform to process the various characters.

```

452 \cs_set:Nn \um_set_mathsymbol:nNNn {

```

Operators In the examples following, say we’re defining for the symbol `\sum` (Σ).

```

453 \ifx\mathop#3\relax

```

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.

The active math char is defined to expand to the macro `\sum_sym`.

```

454 \begingroup
455   \char_make_active:n {#4}
456   \global\mathcode#4="8000\relax
457   \um@scanactivedef #4 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }
458 \endgroup

```

Some of these require a `\nolimits` suffix. This is controlled by the `\um@nolimits` macro, which contains a list of such characters. This list is checked dynamically because we're not interested in efficiency. Or something. This allows the list to be updated in the middle of a document.

Declare the plain old `mathchardef` for the control sequence `\sumop`.

```
459 \expandafter\global\expandafter\XeTeXmathchardef
460 \csname\cs_to_str:N #2 op\endcsname = "\mathchar@type#3 #1 #4\relax
```

Now define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

```
461 \cs_gset:cpx { \cs_to_str:N #2 _sym } {
462   \exp_not:c { \cs_to_str:N #2 op }
463   \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
464 }
```

Don't forget that the actual `\sum` macro is simply defined in terms of the literal unicode symbol!

```
465 \else
```

Delimiters and radicals Sqrt radical is defined as a `csmathopen`.

```
466 \ifx\mathopen#3\relax
467   \tl_if_in:NnTF \l_um_radicals_tl #2 {
468     \cs_gset:cpn { \cs_to_str:N #2 sign } { \XeTeXradical #1 #4 \relax }
469   }{
470     \cs_gset:Npn #2 { \XeTeXdelimiter "\mathchar@type#3 #1 #4 \relax }
471     \global\XeTeXdelcode#4=#1 #4 \relax
472     \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
473   }
474 \else
475   \ifx\mathclose#3\relax
476     \cs_gset:Npn #2 { \XeTeXdelimiter "\mathchar@type#3 #1 #4 \relax }
477     \global\XeTeXdelcode#4=#1 #4 \relax
478     \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
479   \else
```

Fences

```
480 \ifx\mathfence#3
481   \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
482   \global\XeTeXdelcode#4=#1 #4 \relax
483   \cs_gset:cpn {l \cs_to_str:N #2} { \XeTeXdelimiter "\math-
484     char@type\mathopen #1 #4 \relax }
485   \cs_gset:cpn {r \cs_to_str:N #2} { \XeTeXdelimiter "\math-
    char@type\mathclose #1 #4 \relax }
486 \else
```

Accents

```

486         \ifx\mathaccent#3\relax
487         \cs_gset:Npx #2 {\XeTeXmathaccent "\mathchar@type#3 #1 #4\relax}
488         \else

```

And finally, the general case. We define the unicode mathcode for the character. The macro is defined later on generically in terms of the unicode character.

```

489         \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4\relax
490     \fi
491 \fi
492 \fi
493 \fi
494 \fi
495 }

```

`\um_set_mathcode:nnnn` Note that this declaration *isn't* global so that it can be constrained by grouping inside math alphabet switches.

```

\um_set_mathchar:Nnnn
496 \cs_set:Nn \um_set_mathcode:nnnn {
497     \XeTeXmathcode#1="\mathchar@type#2 \csname sym#3\endcsname #4\relax
498 }
499 \cs_set:Nn \um_set_mathchar:Nnnn {
500     \XeTeXmathchardef #1 = "\mathchar@type#2 \csname sym#3\endcsname #4\relax
501 }

```

7.3 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

```

502 \DeclareDocumentCommand \setmathfont { O{} m } {

```

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

```

503     \let\glb@currsizel\relax

```

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

```

504     \bool_set_true:N \l_um_init_bool
505     \seq_clear:N \l_um_char_range_seq
506     \let\um@char@num@range\@empty

```

- Grab the current size information (is this robust enough? Maybe it should be preceded by `\normalsize`).

```
507 \csname S@\f@size\endcsname
```

- Set the name of the math version being defined. (obviously more needs to be done here!)

```
508 \tl_set:Nn \l_um_mversion_tf {normal}
509 \DeclareMathVersion{\l_um_mversion_tf}
```

Define default font features for the script and scriptscript font.

```
510 \tl_set:Nn \l_um_script_features_tl {ScriptStyle}
511 \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
512 \tl_set:Nn \l_um_script_font_tl {#2}
513 \tl_set:Nn \l_um_sscript_font_tl {#2}
```

Use `fontspec` to select a font to use. 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.

```
514 \setkeys*{unicode-math.sty}{#1}
515 \cs_set:Npx \um_tmp: {
516   \exp_not:N \setkeys*[um]{options}{\exp_not:V \XKV@rm}
517 }
518 \um_tmp:
519 \cs_set:Npx \um_tmp: {
520   \exp_not:N \zf@fontspec {
521     BoldFont = {}, ItalicFont = {},
522     Script = Math,
523     SizeFeatures = {
524       {Size = \tf@size-} ,
525       {Size = \sf@size-\tf@size ,
526         Font = \l_um_script_font_tl ,
527         \l_um_script_features_tl
528       } ,
529       {Size = -\sf@size ,
530         Font = \l_um_sscript_font_tl ,
531         \l_um_sscript_features_tl
532       }
533     },
534     \XKV@rm
535   }{#2}
536 }
537 \bool_set_true:N \l_um_fontspec_feature_bool
538 \um_tmp:
539 \bool_set_false:N \l_um_fontspec_feature_bool
```

Check for the correct number of `\fontdimens`:

```

540 \font\l_um_font="#2"\relax
541 %% \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
542 %% \bool_set_true:N \l_um_ot_math_bool
543 %% \else
544 %% \bool_set_false:N \l_um_ot_math_bool
545 %% \PackageWarningNoLine{unicode-math}{
546 %% The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
547 %% Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
548 %% in~ a~ substandard~ manner
549 %% }
550 %% \fi

```

If we're defining the full unicode math repertoire, then we skip all the parsing processing needed if we're only defining a subset.

- Math symbols are defined with `\UnicodeMathSymbol`; see section §7.3.1 for the individual definitions

```

551 \bool_if:NTF \l_um_init_bool {
552   \tl_set:Nn \um_symfont_tl {um_allsym}
553   \PackageInfo{unicode-math}{Defining~ the~ default~ maths~ font~ as~ '#2'}
554   \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_noparse:nnnn
555   \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
556   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
557   \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
558   \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_noparse:nn
559 }{
560   \int_incr:N \g_um_fam_int
561   \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
562   \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_parse:nnnn
563   \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_parse:Nnn
564   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
565   \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
566   \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_parse:nn
567 }

```

Now defined `\um_symfont_tl` as the \LaTeX math font to access everything:

```

568 \DeclareSymbolFont{\um_symfont_tl}
569   {\encodingdefault}{\zf@family}{\mddefault}{\updefault}

```

And now we input every single maths char. See File 12 for the source to `unicode-math.tex` which is used to create `unicode-math-table.tex`.

```

570 \@input{unicode-math-table.tex}

```

Finally,

- Set up shapes for italic/upright or ordinary/var symbols as per package options.

- Remap symbols that don't take their natural mathcode
- Activate any symbols that need to be math-active
- Assign delimiter codes for symbols that need to grow
- Setup the maths alphabets (\mathbf{b} etc.)

```

571 \um_setup_nabla:
572 \um_setup_partial:
573 \um_remap_symbols:
574 \um_setup_mathactives:
575 \um_setup_delcodes:
576 \um_setup_alphabets:
577 }

```

7.3.1 Functions for setting up symbols with mathcodes

`\um_process_symbol_noparse:nnnn` If the range font feature has been used, then only a subset of the unicode glyphs are to be defined. See section §8.3 for the code that enables this.

```

578 \cs_set:Nn \um_process_symbol_noparse:nnnn {
579   \exp_args:Nc \um_set_mathsymbol:nNNn {sym\um_symfont_tl}#2#3{#1}
580 }
581 \cs_set:Nn \um_process_symbol_parse:nnnn {
582   \um@parse@term{#1}{#2}{#3}{
583     \um_process_symbol_noparse:nnnn{#1}{#2}{#3}{#4}
584   }
585 }

```

`\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
586 \cs_new:Nn \um_remap_symbols: {
587   \um_remap_symbol:nnn{`-}{\mathbin}{"02212}% hyphen to minus
588   \um_remap_symbol:nnn{`*}{\mathbin}{"02217}% text asterisk to "cen-
      tred asterisk"
589   \bool_if:NF \g_um_literal_colon_bool {
590     \um_remap_symbol:nnn{`:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
591   }
592   \bool_if:NTF \g_um_literal_bool {
593     \um_remap_symbol:nnn {\g_um_up_Nabla_usv}{\mathord}{\g_um_up_Nabla_usv}
594     \um_remap_symbol:nnn {\g_um_it_Nabla_usv}{\mathord}{\g_um_it_Nabla_usv}
595     \um_remap_symbol:nnn {\g_um_up_partial_usv}{\mathord}{\g_um_up_partial_usv}
596     \um_remap_symbol:nnn {\g_um_it_partial_usv}{\mathord}{\g_um_it_partial_usv}
597   }{
598     \um_remap_symbol:nnn {\g_um_up_Nabla_usv,\g_um_it_Nabla_usv}{\mathord}{\g_um_Nabla_up_or_
599     \um_remap_symbol:nnn {\g_um_up_partial_usv,\g_um_it_partial_usv}{\mathord}{\g_um_partial_
600   }

```

Some of these in the `bfliteral` block may be redundant, but that's okay:

```

601 \bool_if:NTF \g_um_bfliteral_bool {
602   \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv} {\mathord}{\g_um_bfup_Nabla_usv}
603   \um_remap_symbol:nnn {\g_um_bfit_Nabla_usv} {\mathord}{\g_um_bfit_Nabla_usv}
604   \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv} {\mathord}{\g_um_bfsfup_Nabla_usv}
605   \um_remap_symbol:nnn {\g_um_bfsfit_Nabla_usv} {\mathord}{\g_um_bfsfit_Nabla_usv}
606   \um_remap_symbol:nnn {\g_um_bfup_partial_usv} {\mathord}{\g_um_bfup_partial_usv}
607   \um_remap_symbol:nnn {\g_um_bfit_partial_usv} {\mathord}{\g_um_bfit_partial_usv}
608   \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv} {\mathord}{\g_um_bfsfup_partial_usv}
609   \um_remap_symbol:nnn {\g_um_bfsfit_partial_usv} {\mathord}{\g_um_bfsfit_partial_usv}
610 }{
611   \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv,\g_um_bfit_Nabla_usv} {\mathord}{\g_um_bfNabla_
612   \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv,\g_um_bfsfit_Nabla_usv} {\mathord}{\g_um_bfsf
613   \um_remap_symbol:nnn {\g_um_bfup_partial_usv,\g_um_bfit_partial_usv} {\mathord}{\g_um_bfpa
614   \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv,\g_um_bfsfit_partial_usv} {\mathord}{\g_um_
615 }
616 }

```

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

```

617 \cs_new:Nn \um_remap_symbol_parse:nnn {
618   \um@parse@term {#3} {\@nil} {#2} {
619     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
620   }
621 }
622 \cs_new:Nn \um_remap_symbol_noparse:nnn {
623   \clist_map_inline:nn {#1} {
624     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
625   }
626 }

```

7.3.2 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:`

```

627 \cs_new:Nn \um_setup_mathactives: {
628   \um_make_mathactive:nNN {"2032} \um_prime_single_mchar \mathord
629   \um_make_mathactive:nNN {"2033} \um_prime_double_mchar \mathord
630   \um_make_mathactive:nNN {"2034} \um_prime_triple_mchar \mathord
631   \um_make_mathactive:nNN {"2057} \um_prime_quad_mchar \mathord
632   \um_make_mathactive:nNN {"2035} \um_backprime_single_mchar \mathord
633   \um_make_mathactive:nNN {"2036} \um_backprime_double_mchar \mathord
634   \um_make_mathactive:nNN {"2037} \um_backprime_triple_mchar \mathord
635   \XeTeXmathcodenum `` = "1FFFFF \scan_stop:
636 }

```

`\um_make_mathactive:nNN` : TODO : hook into range feature 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!

```

637 \cs_new:Nn \um_make_mathactive:nNN {
638   \um_set_mathchar:Nnn #2 {#3} {\um_symfont_tl} {#1}
639   \XeTeXmathcodenum #1 = "1FFFFF \scan_stop:
640 }

```

7.3.3 Delimiter codes

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:`

```

641 \cs_new:Nn \um_setup_delcodes: {
642   \um_set_delcode:nn {\`\/} {\g_um_slash_delimiter_usv}
643   \um_set_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
644   \um_set_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
645   \um_set_delcode:n {"005C} % backslash
646   \um_set_delcode:nn {\`<} {"27E8} % angle brackets with ascii notation
647   \um_set_delcode:nn {\`>} {"27E9} % angle brackets with ascii notation
648   \um_set_delcode:n {"2191} % up arrow
649   \um_set_delcode:n {"2193} % down arrow
650   \um_set_delcode:n {"2195} % updown arrow
651   \um_set_delcode:n {"219F} % up arrow twohead
652   \um_set_delcode:n {"21A1} % down arrow twohead
653   \um_set_delcode:n {"21A5} % up arrow from bar
654   \um_set_delcode:n {"21A7} % down arrow from bar
655   \um_set_delcode:n {"21A8} % updown arrow from bar
656   \um_set_delcode:n {"21BE} % up harpoon right
657   \um_set_delcode:n {"21BF} % up harpoon left
658   \um_set_delcode:n {"21C2} % down harpoon right
659   \um_set_delcode:n {"21C3} % down harpoon left
660   \um_set_delcode:n {"21C5} % arrows up down
661   \um_set_delcode:n {"21F5} % arrows down up
662   \um_set_delcode:n {"21C8} % arrows up up
663   \um_set_delcode:n {"21CA} % arrows down down
664   \um_set_delcode:n {"21D1} % double up arrow
665   \um_set_delcode:n {"21D3} % double down arrow
666   \um_set_delcode:n {"21D5} % double updown arrow
667   \um_set_delcode:n {"21DE} % up arrow double stroke
668   \um_set_delcode:n {"21DF} % down arrow double stroke
669   \um_set_delcode:n {"21E1} % up arrow dashed
670   \um_set_delcode:n {"21E3} % down arrow dashed

```

```

671 \um_set_delcode:n {"21E7} % up white arrow
672 \um_set_delcode:n {"21E9} % down white arrow
673 \um_set_delcode:n {"21EA} % up white arrow from bar
674 \um_set_delcode:n {"21F3} % updown white arrow
675 }

```

`\um_set_delcode:nn` : TODO : hook into range feature

```

\um_set_delcode:n
676 \cs_new:Nn \um_set_delcode:nn {
677   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #2
678 }
679 \cs_new:Nn \um_set_delcode:n {
680   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #1
681 }

```

7.3.4 Maths alphabets' character mapping

7.3.5 Functions for setting up the maths alphabets

`\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.

```

682 \cs_set:Nn \um_mathmap_noparse:Nnn {
683   \clist_map_inline:nn {#2} {
684     \tl_put_right:cx {um_setup\cs_to_str:N #1:} {
685       \exp_not:N\um_set_mathcode:nnnn{##1}{\exp_not:N\mathalpha}{\um_symfont_tl}{#3}
686     }
687   }
688 }

```

`\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@parse@term` is executed, it populates the `\um@char@num@range` 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.

```

689 \cs_set:Nn \um_mathmap_parse:Nnn {
690   \clist_map_inline:Nn \um@char@num@range {
691     \ifnum##1=#3\relax
692       \um_mathmap_noparse:Nnn {#1}{#2}{#3}
693     \fi
694   }
695 }

```

7.4 (Big) operators
















Turns out that \XeTeX is clever enough to deal with big operators for us automatically with `\XeTeXmathchardef`. 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 `\UnicodeMathSymbol` in the appropriate contexts.

Following is a table of every math operator (`\mathop`) defined in `unicode-math-table.tex`, from which a subset need to be flagged for `\nolimits` adjustments. The limits behaviour as specified by `unicode-math` are shown (with grey 'scripts').

USV	Ex.	Macro	Description
U+02140	$\sum\limits_0^1$	<code>\Bbbsum</code>	DOUBLE-STRUCK N-ARY SUMMATION
U+0220F	$\prod\limits_0^1$	<code>\prod</code>	PRODUCT OPERATOR
U+02210	$\coprod\limits_0^1$	<code>\coprod</code>	COPRODUCT OPERATOR
U+02211	$\sum\limits_0^1$	<code>\sum</code>	SUMMATION OPERATOR
U+0222B	$\int\limits_0^1$	<code>\int</code>	INTEGRAL OPERATOR
U+0222C	$\iint\limits_0^1$	<code>\iint</code>	DOUBLE INTEGRAL OPERATOR
U+0222D	$\iiint\limits_0^1$	<code>\iiint</code>	TRIPLE INTEGRAL OPERATOR
U+0222E	$\oint\limits_0^1$	<code>\oint</code>	CONTOUR INTEGRAL OPERATOR
U+0222F	$\oiint\limits_0^1$	<code>\oiint</code>	DOUBLE CONTOUR INTEGRAL OPERATOR
U+02230	$\oiint\limits_0^1$	<code>\oiint</code>	TRIPLE CONTOUR INTEGRAL OPERATOR
U+02231	$\int\limits_0^1$	<code>\intclockwise</code>	CLOCKWISE INTEGRAL
U+02232	$\oint\limits_0^1$	<code>\varointclockwise</code>	CONTOUR INTEGRAL, CLOCKWISE
U+02233	$\oint\limits_0^1$	<code>\ointctrackwise</code>	CONTOUR INTEGRAL, ANTICLOCKWISE
U+022C0	$\bigwedge\limits_0^1$	<code>\bigwedge</code>	LOGICAL OR OPERATOR
U+022C1	$\bigvee\limits_0^1$	<code>\bigvee</code>	LOGICAL AND OPERATOR
U+022C2	$\bigcap\limits_0^1$	<code>\bigcap</code>	INTERSECTION OPERATOR
U+022C3	$\bigcup\limits_0^1$	<code>\bigcup</code>	UNION OPERATOR
U+027D5	$\left\lrcorner\limits_0^1$	<code>\leftouterjoin</code>	LEFT OUTER JOIN
U+027D6	$\right\lrcorner\limits_0^1$	<code>\rightouterjoin</code>	RIGHT OUTER JOIN

U+027D7		<code>\fullouterjoin</code>	FULL OUTER JOIN
U+027D8		<code>\bigbot</code>	LARGE UP TACK
U+027D9		<code>\bigtop</code>	LARGE DOWN TACK
U+029F8		<code>\xsol</code>	BIG SOLIDUS
U+029F9		<code>\xbsol</code>	BIG REVERSE SOLIDUS
U+02A00		<code>\bigodot</code>	N-ARY CIRCLED DOT OPERATOR
U+02A01		<code>\bigoplus</code>	N-ARY CIRCLED PLUS OPERATOR
U+02A02		<code>\bigotimes</code>	N-ARY CIRCLED TIMES OPERATOR
U+02A03		<code>\bigcupdot</code>	N-ARY UNION OPERATOR WITH DOT
U+02A04		<code>\biguplus</code>	N-ARY UNION OPERATOR WITH PLUS
U+02A05		<code>\bigsqcap</code>	N-ARY SQUARE INTERSECTION OPERATOR
U+02A06		<code>\bigsqcup</code>	N-ARY SQUARE UNION OPERATOR
U+02A07		<code>\conjquant</code>	TWO LOGICAL AND OPERATOR
U+02A08		<code>\disjquant</code>	TWO LOGICAL OR OPERATOR
U+02A09		<code>\bigtimes</code>	N-ARY TIMES OPERATOR
U+02A0B		<code>\sumint</code>	SUMMATION WITH INTEGRAL
U+02A0C		<code>\iiiiint</code>	QUADRUPLE INTEGRAL OPERATOR
U+02A0D		<code>\intbar</code>	FINITE PART INTEGRAL
U+02A0E		<code>\intBar</code>	INTEGRAL WITH DOUBLE STROKE
U+02A0F		<code>\fint</code>	INTEGRAL AVERAGE WITH SLASH
U+02A10		<code>\cirfnint</code>	CIRCULATION FUNCTION
U+02A11		<code>\awint</code>	ANTICLOCKWISE INTEGRATION LINE INTEGRATION WITH RECTANGULAR
U+02A12		<code>\rppoint</code>	PATH AROUND POLE LINE INTEGRATION WITH SEMICIRCULAR
U+02A13		<code>\scpoint</code>	PATH AROUND POLE LINE INTEGRATION NOT INCLUDING THE
U+02A14		<code>\npoint</code>	POLE

U+02A15		<code>\pointint</code>	INTEGRAL AROUND A POINT OPERATOR
U+02A16		<code>\sqint</code>	QUATERNION INTEGRAL OPERATOR
U+02A17		<code>\intlarhk</code>	HOOK
U+02A18		<code>\intx</code>	INTEGRAL WITH TIMES SIGN
U+02A19		<code>\intcap</code>	INTEGRAL WITH INTERSECTION
U+02A1A		<code>\intcup</code>	INTEGRAL WITH UNION
U+02A1B		<code>\upint</code>	INTEGRAL WITH OVERBAR
U+02A1C		<code>\lowint</code>	INTEGRAL WITH UNDERBAR
U+02A1D		<code>\Join</code>	JOIN
U+02A1E		<code>\bigtriangleleft</code>	LARGE LEFT TRIANGLE OPERATOR
U+02A1F		<code>\zcmp</code>	Z NOTATION SCHEMA COMPOSITION
U+02A20		<code>\zpipe</code>	Z NOTATION SCHEMA PIPING
U+02A21		<code>\zproject</code>	Z NOTATION SCHEMA PROJECTION
U+02AFC		<code>\biginterleave</code>	LARGE TRIPLE VERTICAL BAR OPERATOR
U+02AFF		<code>\bigtalloblong</code>	N-ARY WHITE VERTICAL BAR

`\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 , but that might be a matter of preference.

```

696 \tl_new:Nn \l_um_nolimits_tl {
697   \int\iint\iiint\iiiiint\oint\oiint\oiiint
698   \intclockwise\varointclockwise\ointctr-clockwise\sumint
699   \intbar\intBar\oint\circfnint\awint\rppoint
700   \scpoint\ncpoint\pointint\sqint\intlarhk\intx
701   \intcap\intcup\upint\lowint
702 }

```

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

```

703 \DeclareDocumentCommand \addnolimits {m} {
704   \tl_put_right:Nn \l_um_nolimits_tl {#1}
705 }

```

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

```
706 \DeclareDocumentCommand \removenolimits {m} {
707   \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
708 }
```

7.5 Radicals

The radical for square root is organised in `\um_set_mathsymbol:nNNn` on page ?? . 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?

`\um@radicals` We organise radicals in the same way as `nolimits`-operators; that is, in a comma-list.

```
709 \tl_new:Nn \l_um_radicals_tl {\sqrt}
```

$$\sqrt[2]{1 + \sqrt[3]{1+x}}$$

```
\setmathfont{Cambria Math}
\[\sqrt[2]{1+\sqrt[3]{1+x}}\]
```

7.6 Delimiters

























`\left` We redefine the primitive to be preceded by `\mathopen`; this gives much better spacing in cases such as `\sin\left....` Courtesy of Frank Mittelbach:

```
3754 \let\left@primitive\left
```



```
710 \def\left{\mathopen{}\left@primitive}
711
```

No re-definition is made for `\right` because it's not necessary.
Here are all `\mathopen` characters:

USV	Ex.	Macro	Description
U+00028	(\lparen	LEFT PARENTHESIS
U+0005B	[\lbrack	LEFT SQUARE BRACKET
U+0007B	{	\lbrace	LEFT CURLY BRACKET
U+0221A	√	\sqrt	RADICAL
U+0221B	∛	\cuberoot	CUBE ROOT
U+0221C	∜	\fourthroot	FOURTH ROOT
U+02308	⌈	\lceil	LEFT CEILING
U+0230A	⌊	\lfloor	LEFT FLOOR
U+0231C	⌜	\ulcorner	UPPER LEFT CORNER

U+0231E		<code>\llcorner</code>	LOWER LEFT CORNER LIGHT LEFT TORTOISE SHELL BRACKET
U+02772		<code>\lbrbrak</code>	ORNAMENT
U+027C5		<code>\lbag</code>	LEFT S-SHAPED BAG DELIMITER
U+027CC		<code>\longdivision</code>	LONG DIVISION MATHEMATICAL LEFT WHITE SQUARE
U+027E6		<code>\lBrack</code>	BRACKET
U+027E8		<code>\langle</code>	MATHEMATICAL LEFT ANGLE BRACKET MATHEMATICAL LEFT DOUBLE ANGLE
U+027EA		<code>\lAngle</code>	BRACKET MATHEMATICAL LEFT WHITE TORTOISE
U+027EC		<code>\Lbrbrak</code>	SHELL BRACKET
U+02983		<code>\lBrace</code>	LEFT WHITE CURLY BRACKET
U+02985		<code>\lParen</code>	LEFT WHITE PARENTHESIS
U+02987		<code>\llparenthesis</code>	Z NOTATION LEFT IMAGE BRACKET
U+02989		<code>\llangle</code>	Z NOTATION LEFT BINDING BRACKET
U+0298B		<code>\lbrackubar</code>	LEFT SQUARE BRACKET WITH UNDERBAR LEFT SQUARE BRACKET WITH TICK IN TOP
U+0298D		<code>\lbrackultick</code>	CORNER LEFT SQUARE BRACKET WITH TICK IN
U+0298F		<code>\lbracklltick</code>	BOTTOM CORNER
U+02991		<code>\langedot</code>	LEFT ANGLE BRACKET WITH DOT
U+02993		<code>\lparenless</code>	LEFT ARC LESS-THAN BRACKET
U+02995		<code>\Lparengtr</code>	DOUBLE LEFT ARC GREATER-THAN BRACKET
U+02997		<code>\lblkbrbrak</code>	LEFT BLACK TORTOISE SHELL BRACKET
U+029D8		<code>\lvzigzag</code>	LEFT WIGGLY FENCE
U+029DA		<code>\Lvzigzag</code>	LEFT DOUBLE WIGGLY FENCE
U+029FC		<code>\lcurvyangle</code>	LEFT POINTING CURVED ANGLE BRACKET
U+03014		<code>\lbrbrak</code>	LEFT BROKEN BRACKET
U+03018		<code>\Lbrbrak</code>	LEFT WHITE TORTOISE SHELL BRACKET

And `\mathclose`:

USV	Ex.	Macro	Description
U+00029)	<code>\rparen</code>	RIGHT PARENTHESIS
U+0005D]	<code>\rbrack</code>	RIGHT SQUARE BRACKET
U+0007D	}	<code>\rbrace</code>	RIGHT CURLY BRACKET
U+02309	⌋	<code>\rceil</code>	RIGHT CEILING
U+0230B	⌋	<code>\rfloor</code>	RIGHT FLOOR
U+0231D	⌋	<code>\urcorner</code>	UPPER RIGHT CORNER
U+0231F	⌋	<code>\lrcorner</code>	LOWER RIGHT CORNER LIGHT RIGHT TORTOISE SHELL BRACKET
U+02773		<code>\rbrbrak</code>	ORNAMENT
U+027C6		<code>\rbag</code>	RIGHT S-SHAPED BAG DELIMITER

U+027E7	⌋	\rBrack	MATHEMATICAL RIGHT WHITE SQUARE BRACKET
U+027E9	⌋	\rangle	MATHEMATICAL RIGHT ANGLE BRACKET MATHEMATICAL RIGHT DOUBLE ANGLE
U+027EB	⌋	\rAngle	BRACKET MATHEMATICAL RIGHT WHITE TORTOISE
U+027ED		\Rbrbrak	SHELL BRACKET
U+02984	⌋	\rBrace	RIGHT WHITE CURLY BRACKET
U+02986	⌋	\rParen	RIGHT WHITE PARENTHESIS
U+02988	⌋	\rrparenthesis	Z NOTATION RIGHT IMAGE BRACKET
U+0298A	⌋	\rrangle	Z NOTATION RIGHT BINDING BRACKET
U+0298C	⌋	\rbrackubar	RIGHT SQUARE BRACKET WITH UNDERBAR RIGHT SQUARE BRACKET WITH TICK IN
U+0298E	⌋	\rbracklrtick	BOTTOM CORNER RIGHT SQUARE BRACKET WITH TICK IN TOP
U+02990	⌋	\rbrackurtick	CORNER
U+02992	⌋	\rangledot	RIGHT ANGLE BRACKET WITH DOT
U+02994	⌋	\rpangtr	RIGHT ARC GREATER-THAN BRACKET
U+02996	⌋	\Rparenless	DOUBLE RIGHT ARC LESS-THAN BRACKET
U+02998	⌋	\rblkrbrak	RIGHT BLACK TORTOISE SHELL BRACKET
U+029D9	⌋	\rvzigzag	RIGHT WIGGLY FENCE
U+029DB	⌋	\Rvzigzag	RIGHT DOUBLE WIGGLY FENCE
U+029FD	⌋	\rcurvyangle	RIGHT POINTING CURVED ANGLE BRACKET
U+03015		\rbrbrak	RIGHT BROKEN BRACKET
U+03019		\Rbrbrak	RIGHT WHITE TORTOISE SHELL BRACKET

7.7 Maths accents

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

USV	Ex.	Macro	Description
U+00300	˘	\grave	GRAVE ACCENT
U+00301	ˆ	\acute	ACUTE ACCENT
U+00302	ˆ	\hat	CIRCUMFLEX ACCENT
U+00303	˜	\tilde	TILDE
U+00304	ˉ	\bar	MACRON
U+00305	ˆ	\overbar	OVERBAR EMBELLISHMENT
U+00306	˘	\breve	BREVE
U+00307	˙	\dot	DOT ABOVE
U+00308	¨	\ddot	DIERESIS
U+00309	ˆ	\ovhook	COMBINING HOOK ABOVE
U+0030A	ˆ	\ocirc	RING
U+0030C	ˇ	\check	CARON
U+00310	ˆ	\candra	CANDRABINDU (NON-SPACING)

U+00312	$\acute{\text{,}}$	<code>\oturnedcomma</code>	COMBINING TURNED COMMA ABOVE GREEK PSILI (SMOOTH BREATHING)
U+00313	$\acute{\text{,}}$	<code>\osmooth</code>	(NON-SPACING) GREEK DASIA (ROUGH BREATHING)
U+00314	$\acute{\text{,}}$	<code>\orough</code>	(NON-SPACING)
U+00315	$\acute{\text{,}}$	<code>\ocommatopright</code>	COMBINING COMMA ABOVE RIGHT
U+0031A	$\acute{\text{,}}$	<code>\droang</code>	LEFT ANGLE ABOVE (NON-SPACING) UNDER TILDE ACCENT (MULTIPLE
U+00330	x	<code>\wideutilde</code>	CHARACTERS AND NON-SPACING)
U+00331	x	<code>\underbar</code>	COMBINING MACRON BELOW
U+00338	x	<code>\not</code>	COMBINING LONG SOLIDUS OVERLAY
U+020D0	x	<code>\leftharpoonaccent</code>	COMBINING LEFT HARPOON ABOVE
U+020D1	x	<code>\rightharpoonaccent</code>	COMBINING RIGHT HARPOON ABOVE
U+020D2	x	<code>\vertoverlay</code>	COMBINING LONG VERTICAL LINE OVERLAY
U+020D6	x	<code>\overleftarrow</code>	COMBINING LEFT ARROW ABOVE
U+020D7	x	<code>\vec</code>	COMBINING RIGHT ARROW ABOVE
U+020DB	x	<code>\dddot</code>	COMBINING THREE DOTS ABOVE
U+020DC	x	<code>\ddddot</code>	COMBINING FOUR DOTS ABOVE
U+020E1	x	<code>\overleftrightharpoon</code>	COMBINING LEFT RIGHT ARROW ABOVE
U+020E7	x	<code>\annuity</code>	COMBINING ANNUITY SYMBOL
U+020E8	x	<code>\threeunderdot</code>	COMBINING TRIPLE UNDERDOT
U+020E9	x	<code>\widebridgeabove</code>	COMBINING WIDE BRIDGE ABOVE COMBINING RIGHTWARDS HARPOON WITH
U+020EC	x	<code>\underrightharpoondown</code>	BARB DOWNWARDS COMBINING LEFTWARDS HARPOON WITH
U+020ED	x	<code>\underleftharpoondown</code>	BARB DOWNWARDS
U+020EE	x	<code>\underleftarrow</code>	COMBINING LEFT ARROW BELOW
U+020EF	x	<code>\underrightarrow</code>	COMBINING RIGHT ARROW BELOW
U+020F0	x	<code>\asteraccent</code>	COMBINING ASTERISK ABOVE

8 Font features

`\um@zf@feature` Use the same method as `fontspec` for feature definition (*i.e.*, using `xkeyval`) but with a conditional to restrict the scope of these features to unicode-math commands.

```

712 \newcommand\um@zf@feature[2]{
713   \define@key[zf]{options}{#1}[] {
714     \bool_if:NTF \l_um_fontspec_feature_bool {
715       #2
716     } {
717       \PackageError{fontspec/unicode-math}
718       {The '#1' font feature can only be used for maths fonts}
719       {The feature you tried to use can only be in commands

```

```

720         like \protect\setmathfont}
721     }
722 }
723 }

```

8.1 OpenType maths font features

```

724 \um@zf@feature{ScriptStyle}{
725     \zf@update@ff{+ssty=0}
726 }
727 \um@zf@feature{ScriptScriptStyle}{
728     \zf@update@ff{+ssty=1}
729 }

```

8.2 Script and scriptscript font options

```

730 \define@cmdkey[um]{options}[um@]{script-features}{}
731 \define@cmdkey[um]{options}[um@]{sscript-features}{}
732 \define@cmdkey[um]{options}[um@]{script-font}{}
733 \define@cmdkey[um]{options}[um@]{sscript-font}{}

```

8.3 Range processing

The ‘ALL’ branch here is deprecated and happens automatically.

```

734 \seq_new:N \g_um_mathalph_seq
735 \seq_new:N \l_um_mathalph_seq
736 \seq_new:N \l_um_char_range_seq
737 \define@choicekey+[um]{options}{range}[\@tempa\@tempb]{ALL}{
738     \ifcase\@tempb\relax
739         \bool_set_true:N \l_um_init_bool
740     \fi
741 }{
742     \bool_set_false:N \l_um_init_bool
743     \seq_clear:N \l_um_char_range_seq
744     \seq_clear:N \l_um_mathalph_seq
745     \clist_map_inline:nn {#1} {
746         \um_if_mathalph_decl:nTF {##1} {
747             \seq_put_right:Nx \l_um_mathalph_seq { {\exp_not:V\l_um_tmpa_tl} {\exp_not:V\l_um_tmpb_tl} }
748         }{
749             \seq_put_right:Nn \l_um_char_range_seq {##1}
750         }
751     }
752 }
753 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
754     \tl_set:Nn \l_um_tmpa_tl {#1}
755     \tl_set:Nn \l_um_tmpb_tl {}
756     \tl_set:Nn \l_um_tmpc_tl {}
757     \tl_if_in:NnT \l_um_tmpa_tl {->} {

```

```

758     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
759   }
760   \tl_if_in:NnT \l_um_tmpa_tl {/} {
761     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
762   }
763   \seq_if_in:NVTF \g_um_mathalph_seq \l_um_tmpa_tl {
764     \prg_return_true:
765   }{
766     \prg_return_false:
767   }
768 }
769 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
770   \tl_set:Nn \l_um_tmpa_tl {#1}
771   \tl_set:Nn \l_um_tmppc_tl {#2}
772 }
773 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
774   \tl_set:Nn \l_um_tmpa_tl {#1}
775   \tl_set:Nn \l_um_tmppb_tl {#2}
776 }

```

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

`\um@parse@term` #1 : unicode character slot
 #2 : control sequence (character macro)
 #3 : control sequence (math type)
 #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 15.

Table 15: Ranges accepted by `\um@parse@range`.

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

Start by iterating over the commalist, ignoring empties, and initialising the scratch conditional:

```

777 \newcommand\um@parse@term[4]{
778   \seq_map_variable:NNn \l_um_char_range_seq \@ii {

```

```

779 \unless\ifx\@ii\@empty
780 \@tempswafalse

```

Match to either the character macro (`\alpha`) or the math type (`\mathbin`):

```

781 \expandafter\um@firstchar\expandafter{\@ii}
782 \ifx\@tempa\um@backslash
783 \expandafter\ifx\@ii#2\relax
784 \@tempswatrue
785 \else
786 \expandafter\ifx\@ii#3\relax
787 \@tempswatrue
788 \fi
789 \fi

```

Otherwise, we have a number range, which is passed to another macro:

```

790 \else
791 \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
792 \fi

```

If we have a match, execute the code! It also populates the `\um@char@num@range` macro, which is used when defining `\mathbf` (*etc.*) `\mathchar` remappings.

```

793 \if@tempswa
794 \ifx\um@char@num@range\@empty
795 \g@addto@macro\um@char@num@range{#1}
796 \else
797 \g@addto@macro\um@char@num@range{,#1}
798 \fi
799 #4%
800 \fi
801 \fi
802 }
803 }
804 \def\um@firstof#1#2\@nil{#1}
805 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
806 \def\um@firstchar#1{\edef\@tempa{\expandafter\um@firstof\string#1\@nil}}

```

`\um@parse@range` Weird syntax. As shown previously in table 15, this macro can be passed four different input types via `\um@parse@term`.

```

807 \def\um@parse@range#1-#2-#3\@nil#4\@nil{
808 \def\@tempa{#1}
809 \def\@tempb{#2}

```

Range	$r = x$
C-list input	<code>\@ii=X</code>
Macro input	<code>\um@parse@range X-\@marker-\@nil#1\@nil</code>
Arguments	$\textcolor{red}{\#1}-\textcolor{blue}{\#2}-\textcolor{green}{\#3} = \textcolor{red}{X}-\textcolor{blue}{\@marker}-\{\}$

```

810 \expandafter\ifx\expandafter\@marker\@tempb\relax
811 \ifnum#4=#1\relax

```

```

812     \@tempwattrue
813     \fi
814     \else

```

```

Range           $r \geq x$ 
C-list input    \@ii=X-
Macro input     \um@parse@range X--\@marker-\@nil#1\@nil
Arguments       #1-#2-#3 = X-{}-\@marker-

```

```

815     \ifx\@empty\@tempb
816         \ifnum#4>\numexpr#1-1\relax
817             \@tempwattrue
818         \fi
819     \else

```

```

Range           $r \leq y$ 
C-list input    \@ii=-Y
Macro input     \um@parse@range -Y-\@marker-\@nil#1\@nil
Arguments       #1-#2-#3 = {}-Y-\@marker-

```

```

820     \ifx\@empty\@tempa
821         \ifnum#4<\numexpr#2+1\relax
822             \@tempwattrue
823         \fi

```

```

Range           $x \leq r \leq y$ 
C-list input    \@ii=X-Y
Macro input     \um@parse@range X-Y-\@marker-\@nil#1\@nil
Arguments       #1-#2-#3 = X-Y-\@marker-

```

```

824     \else
825         \ifnum#4>\numexpr#1-1\relax
826         \ifnum#4<\numexpr#2+1\relax
827             \@tempwattrue
828         \fi
829     \fi
830 \fi
831 \fi
832 \fi
833 }

```

\um_map_char:nn #1 : Number of iterations
 \um_map_chars_xxvi:nn #2 : Starting input char(s)
 \um_map_chars_xxi:nn #3 : Starting output char
 Loops through character ranges setting \mathcode.

```

834 \cs_set:Nn \um_map_chars_range:nnn {
835   \clist_map_inline:nn {#2} {
836     \prg_stepwise_inline:nnnn {0}{1}{#1} {
837       \um_map_char_internal:nn {##1+####1}{#3+####1}
838     }

```

```

839 }
840 }
841 \cs_new:Nn \um_map_char_noparse:nn {
842   \um_set_mathcode:nnnn
843   {\numexpr #1 \relax}{\mathalpha}{\um_symfont_t1}{\numexpr #2 \relax}
844 }
845 \cs_new:Nn \um_map_char_parse:nn {
846   \um@parse@term {#1} {\@nil} {\mathalpha} {
847     \um_map_char_noparse:nn {#1}{#2}
848   }
849 }
850 \cs_set:Nn \um_map_chars_xxvi:nn {
851   \um_map_chars_range:nnn {25}{#1}{#2}
852 }
853 \cs_set:Nn \um_map_chars_xxiii:nn {
854   \um_map_chars_range:nnn {24}{#1}{#2}
855 }
856 \cs_set:Nn \um_map_chars_x:nn {
857   \um_map_chars_range:nnn {9}{#1}{#2}
858 }
859 \cs_set:Nn \um_map_chars_Latin:nn {
860   \clist_map_inline:nn {#1} {
861     \um_map_chars_xxvi:cc { \um_to_usv:nn{##1}{Latin} }{ \um_to_usv:nn{#2}{Latin} }
862   }
863 }
864 \cs_set:Nn \um_map_chars_latin:nn {
865   \clist_map_inline:nn {#1} {
866     \um_map_chars_xxvi:cc {g_um_ ##1 _latin_usv}{g_um_ #2 _latin_usv}
867   }
868 }
869 \cs_set:Nn \um_map_chars_greek:nn {
870   \clist_map_inline:nn {#1} {
871     \um_map_chars_xxiii:cc {g_um_ ##1 _greek_usv}{g_um_ #2 _greek_usv}
872     \um_map_char:cc {g_um_ ##1 _varepsilon_usv}{g_um_ #2 _varepsilon_usv}
873     \um_map_char:cc {g_um_ ##1 _vartheta_usv }{g_um_ #2 _vartheta_usv }
874     \um_map_char:cc {g_um_ ##1 _varkappa_usv }{g_um_ #2 _varkappa_usv }
875     \um_map_char:cc {g_um_ ##1 _varphi_usv }{g_um_ #2 _varphi_usv }
876     \um_map_char:cc {g_um_ ##1 _varrho_usv }{g_um_ #2 _varrho_usv }
877     \um_map_char:cc {g_um_ ##1 _varpi_usv }{g_um_ #2 _varpi_usv }
878   }
879 }
880 \cs_set:Nn \um_map_chars_Greek:nn {
881   \clist_map_inline:nn {#1} {
882     \um_map_chars_xxiii:cc {g_um_ ##1 _Greek_usv}{g_um_ #2 _Greek_usv}
883     \um_map_char:cc {g_um_ ##1 _varTheta_usv}{g_um_ #2 _varTheta_usv}
884   }

```



```

885 }
886 \cs_set:Nn \um_map_chars_numbers:nn {
887   \um_map_chars_x:cc {g_um_#1_num_usv}{g_um_#2_num_usv}
888 }
889 \cs_set:Nn \um_map_char:nn {
890   \um_map_chars_range:nnn {0}{#1}{#2}
891 }
892 \cs_set:Nn \um_map_single:nnn {
893   \clist_map_inline:nn {#2} {
894     \um_map_char:cc {g_um_##1_#1_usv}{g_um_#3_#1_usv}
895   }
896 }
897 \cs_generate_variant:Nn \um_map_char:nn {cc}
898 \cs_generate_variant:Nn \um_map_chars_xxiii:nn {cc}
899 \cs_generate_variant:Nn \um_map_chars_xxvi:nn {cc}
900 \cs_generate_variant:Nn \um_map_chars_x:nn {cc}

```

`\um_set_mathalphabet_char:Nnn` **#1** : Maths alphabet
#2 : Input char(s)
#3 : Output char
 Loops through character ranges setting `\mathcode`.

```

901 \cs_set:Npn \exp_args:Nnff {\::n::f::f::f::f::}
902 \cs_new:Nn \um_set_mathalphabet_char:Nnn {
903   \clist_map_variable:nNn {#2} \l_um_input_num {
904     \exp_args:Nnff \um_mathmap:Nnn {#1}
905     {\number\numexpr\l_um_input_num\relax} {\number\numexpr#3\relax}
906   }
907 }

```

`\um_set_mathalph_range:Nnn` [*(Number of iterations)*] **#1** : Maths alphabet
#2 : Starting input char(s)
#3 : Starting output char
 Loops through character ranges setting `\mathcode`.

```

908 \cs_new:Nn \um_set_mathalph_range:nNnn {
909   \clist_map_variable:nNn {#3} \l_um_input_num {
910     \errorcontextlines=999
911     \prg_stepwise_variable:nnnNn {0}{1}{#1} \l_um_inc_num {
912       \exp_args:Nnff \um_mathmap:Nnn {#2}
913       {\number\numexpr \l_um_inc_num + \l_um_input_num \relax}
914       {\number\numexpr \l_um_inc_num + #4 \relax}
915     }
916   }
917 }
918 \cs_new:Nn \um_set_mathalphabet_x:Nnn {
919   \um_set_mathalph_range:nNnn {9}{#1}{#2}{#3}
920 }

```

```

921 \cs_new:Nn \um_set_mathalphabet_xxvi:Nnn {
922   \um_set_mathalph_range:nNnn {25}{#1}{#2}{#3}
923 }
924 \cs_new:Nn \um_set_mathalphabet_xxiii:Nnn {
925   \um_set_mathalph_range:nNnn {24}{#1}{#2}{#3}
926 }
927
928 \cs_new:Nn \um_set_mathalphabet_pos:Nnnn {
929   \cs_if_exist:cT {g_um_#4_#2_usv} {
930     \clist_map_inline:nn {#3} {
931       \um_set_mathalphabet_char:Ncc #1 {g_um_##1_#2_usv}{g_um_#4_#2_usv}
932     }
933   }
934 }
935 \cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
936   \clist_map_inline:nn {#2} {
937     \um_set_mathalphabet_x:Ncc #1 {g_um_##1_num_usv}{g_um_#3_num_usv}
938   }
939 }
940 \cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
941   \clist_map_inline:nn {#2} {
942     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_Latin_usv}{g_um_#3_Latin_usv}
943   }
944 }
945 \cs_new:Nn \um_set_mathalphabet_latin:Nnn {
946   \clist_map_inline:nn {#2} {
947     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_latin_usv}{g_um_#3_latin_usv}
948     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_h_usv} {g_um_#3_h_usv}
949   }
950 }
951 \cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
952   \clist_map_inline:nn {#2} {
953     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_Greek_usv} {g_um_#3_Greek_usv}
954     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varTheta_usv}{g_um_#3_varTheta_usv}
955   }
956 }
957 \cs_new:Nn \um_set_mathalphabet_greek:Nnn {
958   \clist_map_inline:nn {#2} {
959     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_greek_usv} {g_um_#3_greek_usv}
960     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varepsilon_usv}{g_um_#3_varepsilon_usv}
961     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_vartheta_usv} {g_um_#3_vartheta_usv}
962     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varkappa_usv} {g_um_#3_varkappa_usv}
963     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varphi_usv} {g_um_#3_varphi_usv}
964     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varrho_usv} {g_um_#3_varrho_usv}
965     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varpi_usv} {g_um_#3_varpi_usv}
966   }

```

```

967 }
968 \cs_generate_variant:Nn \um_set_mathalphabet_char:Nnn {Ncc}
969 \cs_generate_variant:Nn \um_set_mathalphabet_xxiii:Nnn {Ncc}
970 \cs_generate_variant:Nn \um_set_mathalphabet_xxvi:Nnn {Ncc}
971 \cs_generate_variant:Nn \um_set_mathalphabet_x:Nnn {Ncc}

```

8.4 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.

```

972 \AtBeginDocument{\um_resolve_greek:}
973 \cs_new:Nn \um_resolve_greek: {
974   \clist_map_inline:nn {
975     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
976     alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
977     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
978     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,  chi,psi,omega,
979     varTheta,
980     varsigma,vartheta,varkappa,varrho,varpi
981   }{
982     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
983   }
984   \tl_set:Nn \epsilon {
985     \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
986   }
987   \tl_set:Nn \phi {
988     \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
989   }
990   \tl_set:Nn \varepsilon {
991     \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
992   }
993   \tl_set:Nn \varphi {
994     \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
995   }
996 }

```

9 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.

9.0.1 Macros

This is every math alphabet known to unicode-math:

`\g_um_mathalph_seq`

```

997 \seq_clear:Nn \g_um_mathalph_seq
998 \tl_map_inline:nn {
999   \mathup\mathit
1000   \mathbb\mathbbi
1001   \mathscr\mathfrak\mathtt
1002   \mathsf\mathsfup\mathsfit
1003   \mathbf\mathbfup\mathbfit
1004   \mathbfscr\mathbffrak
1005   \mathbfsf\mathbfsfup\mathbfsfit
1006 }{
1007   \seq_put_right:Nn \g_um_mathalph_seq {#1}
1008 }
```

`\um_setup_alphabets:`

```

1009
1010 \tl_new:Nn \g_um_mathup_alph_clist {latin, Latin, greek, Greek, num}
1011 \tl_new:Nn \g_um_mathit_alph_clist {latin, Latin, greek, Greek}
1012 \tl_new:Nn \g_um_mathscr_alph_clist {latin, Latin}
1013 \tl_new:Nn \g_um_mathfrak_alph_clist {latin, Latin}
1014 \tl_new:Nn \g_um_mathbfscr_alph_clist {latin, Latin}
1015 \tl_new:Nn \g_um_mathbffrak_alph_clist {latin, Latin}
1016 \tl_new:Nn \g_um_mathbb_alph_clist {latin, Latin, num}
```

```

1017 \tl_new:Nn \g_um_mathbbit_alph_clist {}
1018 \tl_new:Nn \g_um_mathtt_alph_clist {latin,Latin,num}
1019 \tl_new:Nn \g_um_mathsf_alph_clist {}
1020 \tl_new:Nn \g_um_mathsfup_alph_clist {latin,Latin,num}
1021 \tl_new:Nn \g_um_mathsfitt_alph_clist {latin,Latin}
1022 \tl_new:Nn \g_um_mathbf_alph_clist {}
1023 \tl_new:Nn \g_um_mathbfup_alph_clist {latin,Latin,greek,Greek,num}
1024 \tl_new:Nn \g_um_mathbfitt_alph_clist {latin,Latin,greek,Greek}
1025 \tl_new:Nn \g_um_mathbfssf_alph_clist {}
1026 \tl_new:Nn \g_um_mathbfsfup_alph_clist {latin,Latin,greek,Greek,num}
1027 \tl_new:Nn \g_um_mathbfssfit_alph_clist {latin,Latin,greek,Greek}
1028
1029 \tl_new:Nn \g_um_mathup_latin_usv {\a-\z}
1030 \tl_new:Nn \g_um_mathup_Latin_usv {\A-\Z}
1031 \tl_new:Nn \g_um_mathup_greek_usv {"3B1-"3C9,"3F5,"3D1,"3F0,"3D5,"3F1,"3D6,"3DD}
1032 \tl_new:Nn \g_um_mathup_Greek_usv {"391-"3A9,"3F4,"3DC}
1033 \tl_new:Nn \g_um_mathup_num_usv {\0-\9}
1034
1035 \tl_new:Nn \g_um_mathit_latin_usv {"1D44E-"1D467,\g_um_it_h_usv}
1036 \tl_new:Nn \g_um_mathit_Latin_usv {"1D434-"1D44C}
1037 \tl_new:Nn \g_um_mathit_greek_usv {"1D6FC-"1D714,"1D716-1D71B}
1038 \tl_new:Nn \g_um_mathit_Greek_usv {"1D6E2-"1D6FA}
1039
1040 \seq_new:N \l_um_missing_alph_seq
1041 \cs_new:Nn \um_setup_alphabets: {
1042   \seq_clear:N \l_um_missing_alph_seq
1043   \seq_if_empty:NTF \l_um_mathalph_seq {
1044     \um_maybe_init_alphabet:n {sf}
1045     \um_maybe_init_alphabet:n {bf}
1046     \um_maybe_init_alphabet:n {bfsf}
1047     \um_setup_math_alphabet:Nvn \mathup \g_um_mathup_alph_clist {up }
1048     \um_setup_math_alphabet:Nvn \mathit \g_um_mathit_alph_clist {it }
1049     \um_setup_math_alphabet:Nvn \mathbb \g_um_mathbb_alph_clist {bb }
1050     \um_setup_math_alphabet:Nvn \mathscr \g_um_mathscr_alph_clist {scr }
1051     \um_setup_math_alphabet:Nvn \mathfrak \g_um_mathfrak_alph_clist {frak }
1052     \um_setup_math_alphabet:Nvn \mathsfup \g_um_mathsfup_alph_clist {sfup }
1053     \um_setup_math_alphabet:Nvn \mathsfitt \g_um_mathsfitt_alph_clist {sfitt }
1054     \um_setup_math_alphabet:Nvn \mathbfup \g_um_mathbfup_alph_clist {bfup }
1055     \um_setup_math_alphabet:Nvn \mathbfitt \g_um_mathbfitt_alph_clist {bfitt }
1056     \um_setup_math_alphabet:Nvn \mathbfssf \g_um_mathbfssf_alph_clist {bf-
1057       scr }
1058     \um_setup_math_alphabet:Nvn \mathbffrak \g_um_mathbffrak_alph_clist {bf-
1059       frak}
1059     \um_setup_math_alphabet:Nvn \mathbfsf \g_um_mathbfsf_alph_clist {bfsf }
1060     \um_setup_math_alphabet:Nvn \mathbfsfup \g_um_mathbfsfup_alph_clist {bfs-

```

```

fup}
1061 \um_setup_math_alphabet:Nv \mathbfsfit \g_um_mathbfsfit_alph_clist {bfs-
fit}
1062 \um_setup_math_mapping:n {up }
1063 \um_setup_math_mapping:n {it }
1064 \um_setup_math_mapping:n {bb }
1065 \um_maybe_init_alphabet:n {bbit }
1066 \um_setup_math_mapping:n {bbit }
1067 \um_setup_math_mapping:n {bfup }
1068 \um_setup_math_mapping:n {bfit }
1069 \um_setup_math_mapping:n {bfsfup}
1070 \um_setup_math_mapping:n {bfsfit}
1071 \seq_if_empty:NF \l_um_missing_alph_seq {
1072 \typeout{
1073 Package~unicode-math~Warning:~
1074 missing~math~alphabets~in~font~ \fontname\l_um_font
1075 }
1076 \seq_map_inline:Nn \l_um_missing_alph_seq {
1077 \typeout{\space\space\space\space##1}
1078 }
1079 }
1080 }{
1081 \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
1082 \seq_map_inline:Nn \l_um_mathalph_seq {
1083 \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
1084 \tl_set:No \l_um_tmpp_tl { \use_ii:nnn ##1 }
1085 \tl_set:No \l_um_remap_alphabet_tl { \use_iii:nnn ##1 }
1086 \tl_if_empty:NTF \l_um_remap_alphabet_tl {
1087 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \to-
ken_to_str:N \l_um_tmpa_tl}
1088 }{
1089 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \to-
ken_to_str:N \l_um_remap_alphabet_tl}
1090 }
1091 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \use_none:nnnnn \l_um_remap_alphabet_tl}
1092 \tl_if_empty:NT \l_um_tmpp_tl {
1093 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
1094 \tl_set:Nv \l_um_tmpp_tl { g_um_ \exp_after:wN \cs_to_str:N \l_um_tmpa_tl _alph_clist }
1095 }
1096 \um_setup_math_alphabet:VVV \l_um_tmpa_tl \l_um_tmpp_tl \l_um_remap_alphabet_tl
1097 }
1098 }
1099 }

```

`\um_setup_math_alphabet:Nn` #1 : Math font family name (e.g., `\mathbb`)
#2 : Math alphabets, comma separated of {latin, Latin, greek, Greek, num}
#3 : Math alphabets output string (usually same as input `bb`)

First check that at least one of the alphabets for the font shape is defined, and then loop through them defining the individual ranges.

```

1100 \cs_new:Nn \um_setup_math_alphabet:Nnn {
1101   \tl_set:Nx \l_um_tmpa_tl {\cs_to_str:N #1}
1102   \tl_set:Nx \l_um_tmpb_tl {\exp_after:wN \use_none:n \l_um_tmpa_tl}
1103   \clist_map_inline:nn {#2} {
1104     \um_glyph_if_exist:cT {g_um_ \l_um_tmpb_tl _##1_usv}{
1105       \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
1106       \clist_map_break:
1107     }
1108   }
1109   \clist_map_inline:nn {#2} {
1110     \um_glyph_if_exist:cTF {g_um_ \l_um_tmpb_tl _##1_usv}{
1111       \use:c {um_config_ \l_um_tmpa_tl _##1:n} {#3}
1112     }{
1113       \seq_put_right:Nx \l_um_missing_alph_seq {
1114         \@backslashchar
1115         \l_um_tmpa_tl\space(\tl_use:c{g_um_math_alphabet_name_##1_tl})
1116       }
1117     }
1118   }
1119 }
1120 \cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {NV,VVV}
1121
1122 \tl_set:Nn \g_um_math_alphabet_name_latin_tl {Latin,~lowercase}
1123 \tl_set:Nn \g_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
1124 \tl_set:Nn \g_um_math_alphabet_name_greek_tl {Greek,~lowercase}
1125 \tl_set:Nn \g_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
1126 \tl_set:Nn \g_um_math_alphabet_name_num_tl {Numerals}
1127 \cs_new:Nn \um_setup_math_mapping:n {
1128   \cs_if_exist:cT {um_setup_math#1:} {
1129     \use:c {um_config_math#1_misc:n} {#1}
1130   }
1131 }
1132 \cs_set:Nn \um_init_alphabet:n {
1133   \wlog{unicode-math:~Initialiasing~\@backslashchar math#1}
1134   \um_prepare_alph:n {#1}
1135   \cs_set_eq:cN {um_setup_math#1:} \prg_do_nothing:
1136 }

```

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

```

1137 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
1138   \etex_iffontchar:D \l_um_font #1 \scan_stop: \prg_return_true: \else: \prg_return_false: \fi:
1139 }
1140 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}

```

```

1141 \cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
1142 \cs_generate_variant:Nn \um_glyph_if_exist:nT {c}
1143 \cs_generate_variant:Nn \um_glyph_if_exist:nF {c}

```

`\um_prepare_alph:n` If `\mathXY` hasn't been (re-)declared yet, then define it in terms of unicode-math definitions. Use `\bgroup/\egroup` so s'scripts scan the whole thing.

```

1144 \cs_new:Nn \um_prepare_alph:n {
1145   \cs_if_exist:cF {um_math#1:n} {
1146     \cs_set:cpn {um_math#1:n} ##1 {
1147       \use:c {um_setup_math#1:} ##1 \egroup
1148     }
1149     \cs_set_protected:cpn {math#1} {
1150       \bgroup
1151       \mode_if_math:F {
1152         \egroup\expandafter
1153         \non@alpherr\expandafter{\csname math#1\endcsname\space}
1154       }
1155       \use:c {um_math#1:n}
1156     }
1157   }
1158 }

```

9.1 Alphabets

9.1.1 Upright: `\mathup`

```

1159 \cs_new:Nn \um_config_mathup_num:n {
1160   \um_map_chars_numbers:nn {up}{#1}
1161   \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}
1162 }
1163 \cs_new:Nn \um_config_mathup_Latin:n {
1164   \bool_if:NTF \g_um_literal_bool {
1165     \um_map_chars_Latin:nn {up} {#1}
1166   }{
1167     \bool_if:NT \g_um_upLatin_bool {
1168       \um_map_chars_Latin:nn {up,it} {#1}
1169     }
1170   }
1171   \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
1172 }
1173 \cs_new:Nn \um_config_mathup_latin:n {
1174   \bool_if:NTF \g_um_literal_bool {
1175     \um_map_chars_latin:nn {up} {#1}
1176   }{
1177     \bool_if:NT \g_um_uplatin_bool {
1178       \um_map_chars_latin:nn {up,it} {#1}
1179       \um_map_single:nnn {h}{up,it}{#1}

```



```

1180     }
1181   }
1182   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
1183 }
1184 \cs_new:Nn \um_config_mathup_Greek:n {
1185   \bool_if:NTF \g_um_literal_bool {
1186     \um_map_chars_Greek:nn {up}{#1}
1187   }{
1188     \bool_if:NT \g_um_upGreek_bool {
1189       \um_map_chars_Greek:nn {up,it}{#1}
1190     }
1191   }
1192   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
1193 }
1194 \cs_new:Nn \um_config_mathup_greek:n {
1195   \bool_if:NTF \g_um_literal_bool {
1196     \um_map_chars_greek:nn {up} {#1}
1197   }{
1198     \bool_if:NT \g_um_upgreek_bool {
1199       \um_map_chars_greek:nn {up,it} {#1}
1200     }
1201   }
1202   \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
1203 }
1204 \cs_new:Nn \um_config_mathup_misc:n {
1205   \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it}{#1}
1206   \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it}{#1}
1207 }

```

9.1.2 Italic: \mathit

```

1208 \cs_new:Nn \um_config_mathit_Latin:n {
1209   \bool_if:NTF \g_um_literal_bool {
1210     \um_map_chars_Latin:nn {it} {#1}
1211   }{
1212     \bool_if:NF \g_um_upLatin_bool {
1213       \um_map_chars_Latin:nn {up,it} {#1}
1214     }
1215   }
1216   \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
1217 }
1218 \cs_new:Nn \um_config_mathit_latin:n {
1219   \bool_if:NTF \g_um_literal_bool {
1220     \um_map_chars_latin:nn {it} {#1}
1221     \um_map_single:nnn {h}{it}{#1}
1222   }{
1223     \bool_if:NF \g_um_uplatin_bool {

```

```

1224     \um_map_chars_latin:nn {up,it} {#1}
1225     \um_map_single:nnn {h}{up,it}{#1}
1226   }
1227 }
1228 \um_set_mathalphabet_latin:Nnn \mathit {up,it}{#1}
1229 }
1230 \cs_new:Nn \um_config_mathit_Greek:n {
1231   \bool_if:NTF \g_um_literal_bool {
1232     \um_map_chars_Greek:nn {it}{#1}
1233   }{
1234     \bool_if:NF \g_um_upgreek_bool {
1235       \um_map_chars_Greek:nn {up,it}{#1}
1236     }
1237   }
1238   \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
1239 }
1240 \cs_new:Nn \um_config_mathit_greek:n {
1241   \bool_if:NTF \g_um_literal_bool {
1242     \um_map_chars_greek:nn {it} {#1}
1243   }{
1244     \bool_if:NF \g_um_upgreek_bool {
1245       \um_map_chars_greek:nn {it,up} {#1}
1246     }
1247   }
1248   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
1249 }
1250 \cs_new:Nn \um_config_mathit_misc:n {
1251   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
1252   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{#1}
1253 }

```

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

```

1254 \cs_new:Nn \um_config_mathbb_latin:n {
1255   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
1256 }
1257 \cs_new:Nn \um_config_mathbb_Latin:n {
1258   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
1259   \um_set_mathalphabet_pos:Nnnn \mathbb {C} {up,it} {#1}
1260   \um_set_mathalphabet_pos:Nnnn \mathbb {H} {up,it} {#1}
1261   \um_set_mathalphabet_pos:Nnnn \mathbb {N} {up,it} {#1}
1262   \um_set_mathalphabet_pos:Nnnn \mathbb {P} {up,it} {#1}
1263   \um_set_mathalphabet_pos:Nnnn \mathbb {Q} {up,it} {#1}
1264   \um_set_mathalphabet_pos:Nnnn \mathbb {R} {up,it} {#1}
1265   \um_set_mathalphabet_pos:Nnnn \mathbb {Z} {up,it} {#1}
1266 }
1267 \cs_new:Nn \um_config_mathbb_num:n {

```

```

1268 \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
1269 }
1270 \cs_new:Nn \um_config_mathbb_misc:n {
1271 \um_set_mathalphabet_pos:Nnnn \mathbb {Pi} {up,it} {#1}
1272 \um_set_mathalphabet_pos:Nnnn \mathbb {pi} {up,it} {#1}
1273 \um_set_mathalphabet_pos:Nnnn \mathbb {Gamma} {up,it} {#1}
1274 \um_set_mathalphabet_pos:Nnnn \mathbb {gamma} {up,it} {#1}
1275 \um_set_mathalphabet_pos:Nnnn \mathbb {summation} {up} {#1}
1276 }
1277 \cs_new:Nn \um_config_mathbbbit_misc:n {
1278 \um_set_mathalphabet_pos:Nnnn \mathbbbit {D} {up,it} {#1}
1279 \um_set_mathalphabet_pos:Nnnn \mathbbbit {d} {up,it} {#1}
1280 \um_set_mathalphabet_pos:Nnnn \mathbbbit {e} {up,it} {#1}
1281 \um_set_mathalphabet_pos:Nnnn \mathbbbit {i} {up,it} {#1}
1282 \um_set_mathalphabet_pos:Nnnn \mathbbbit {j} {up,it} {#1}
1283 }

```

9.1.4 Script or caligraphic: `\mathscr` and `\mathcal`

```

1284 \cs_new:Nn \um_config_mathscr_Latin:n {
1285 \um_set_mathalphabet_Latin:Nnn \mathscr {up,it}{#1}
1286 \um_set_mathalphabet_pos:Nnnn \mathscr {B}{up,it}{#1}
1287 \um_set_mathalphabet_pos:Nnnn \mathscr {E}{up,it}{#1}
1288 \um_set_mathalphabet_pos:Nnnn \mathscr {F}{up,it}{#1}
1289 \um_set_mathalphabet_pos:Nnnn \mathscr {H}{up,it}{#1}
1290 \um_set_mathalphabet_pos:Nnnn \mathscr {I}{up,it}{#1}
1291 \um_set_mathalphabet_pos:Nnnn \mathscr {L}{up,it}{#1}
1292 \um_set_mathalphabet_pos:Nnnn \mathscr {M}{up,it}{#1}
1293 \um_set_mathalphabet_pos:Nnnn \mathscr {R}{up,it}{#1}
1294 }
1295 \cs_new:Nn \um_config_mathscr_latin:n {
1296 \um_set_mathalphabet_latin:Nnn \mathscr {up,it}{#1}
1297 \um_set_mathalphabet_pos:Nnnn \mathscr {e}{up,it}{#1}
1298 \um_set_mathalphabet_pos:Nnnn \mathscr {g}{up,it}{#1}
1299 \um_set_mathalphabet_pos:Nnnn \mathscr {o}{up,it}{#1}
1300 }

```

9.1.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1301 \cs_new:Nn \um_config_mathfrak_Latin:n {
1302 \um_set_mathalphabet_Latin:Nnn \mathfrak {up,it}{#1}
1303 \um_set_mathalphabet_pos:Nnnn \mathfrak {C}{up,it}{#1}
1304 \um_set_mathalphabet_pos:Nnnn \mathfrak {H}{up,it}{#1}
1305 \um_set_mathalphabet_pos:Nnnn \mathfrak {I}{up,it}{#1}
1306 \um_set_mathalphabet_pos:Nnnn \mathfrak {R}{up,it}{#1}
1307 \um_set_mathalphabet_pos:Nnnn \mathfrak {Z}{up,it}{#1}
1308 }
1309 \cs_new:Nn \um_config_mathfrak_latin:n {
1310 \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{#1}

```

1311 }

9.1.6 Sans serif upright: `\mathsfup`

```
1312 \cs_new:Nn \um_config_mathsfup_num:n {
1313   \um_set_mathalphabet_numbers:Nnn \mathsf {up}{#1}
1314   \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
1315 }
1316 \cs_new:Nn \um_config_mathsfup_Latin:n {
1317   \bool_if:NTF \g_um_sfliteral_bool {
1318     \um_map_chars_Latin:nn {sfup} {#1}
1319     \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
1320   }{
1321     \bool_if:NT \g_um_upsans_bool {
1322       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1323       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1324     }
1325   }
1326   \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}
1327 }
1328 \cs_new:Nn \um_config_mathsfup_latin:n {
1329   \bool_if:NTF \g_um_sfliteral_bool {
1330     \um_map_chars_latin:nn {sfup} {#1}
1331     \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
1332   }{
1333     \bool_if:NT \g_um_upsans_bool {
1334       \um_map_chars_latin:nn {sfup,sfit} {#1}
1335       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1336     }
1337   }
1338   \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
1339 }
```

9.1.7 Sans serif italic: `\mathsfit`

```
1340 \cs_new:Nn \um_config_mathsfital_Latin:n {
1341   \bool_if:NTF \g_um_sfliteral_bool {
1342     \um_map_chars_Latin:nn {sfit} {#1}
1343     \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
1344   }{
1345     \bool_if:NF \g_um_upsans_bool {
1346       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1347       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1348     }
1349   }
1350   \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{#1}
1351 }
1352 \cs_new:Nn \um_config_mathsfital_latin:n {
1353   \bool_if:NTF \g_um_sfliteral_bool {
```

```

1354 \um_map_chars_latin:nn {sfit} {#1}
1355 \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
1356 }{
1357 \bool_if:NF \g_um_upsans_bool {
1358 \um_map_chars_latin:nn {sfup,sfit} {#1}
1359 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1360 }
1361 }
1362 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1363 }

```

9.1.8 Typewriter or monospaced: `\mathtt`

```

1364 \cs_new:Nn \um_config_mathtt_num:n {
1365 \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
1366 }
1367 \cs_new:Nn \um_config_mathtt_Latin:n {
1368 \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
1369 }
1370 \cs_new:Nn \um_config_mathtt_latin:n {
1371 \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
1372 }

```

9.1.9 Bold Italic: `\mathbfit`

```

1373 \cs_new:Nn \um_config_mathbfit_Latin:n {
1374 \bool_if:NF \g_um_bfupLatin_bool {
1375 \um_map_chars_Latin:nn {bfup,bfit} {#1}
1376 }
1377 \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
1378 \bool_if:NTF \g_um_bfliteral_bool {
1379 \um_map_chars_Latin:nn {bfit} {#1}
1380 \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
1381 }{
1382 \bool_if:NF \g_um_bfupLatin_bool {
1383 \um_map_chars_Latin:nn {bfup,bfit} {#1}
1384 \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1385 }
1386 }
1387 }
1388 \cs_new:Nn \um_config_mathbfit_latin:n {
1389 \bool_if:NF \g_um_bfuplatin_bool {
1390 \um_map_chars_latin:nn {bfup,bfit} {#1}
1391 }
1392 \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
1393 \bool_if:NTF \g_um_bfliteral_bool {
1394 \um_map_chars_latin:nn {bfit} {#1}
1395 \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
1396 }{

```

```

1397     \bool_if:NF \g_um_bfuplatin_bool {
1398       \um_map_chars_latin:nn {bfup,bfit} {#1}
1399       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1400     }
1401   }
1402 }
1403 \cs_new:Nn \um_config_mathbfit_Greek:n {
1404   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
1405   \bool_if:NTF \g_um_bfliteral_bool {
1406     \um_map_chars_Greek:nn {bfit}{#1}
1407     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
1408   }{
1409     \bool_if:NF \g_um_bfupGreek_bool {
1410       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1411       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1412     }
1413   }
1414 }
1415 \cs_new:Nn \um_config_mathbfit_greek:n {
1416   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
1417   \bool_if:NTF \g_um_bfliteral_bool {
1418     \um_map_chars_greek:nn {bfit} {#1}
1419     \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
1420   }{
1421     \bool_if:NF \g_um_bfupgreek_bool {
1422       \um_map_chars_greek:nn {bfit,bfup} {#1}
1423       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1424     }
1425   }
1426 }
1427 \cs_new:Nn \um_config_mathbfit_misc:n {
1428   \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{#1}
1429   \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{#1}
1430   \bool_if:NTF \g_um_bfliteral_bool {
1431     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{#1}
1432     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {it}{#1}
1433   }{
1434     \bool_if:NF \g_um_upNabla_bool {
1435       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1436     }
1437     \bool_if:NF \g_um_uppartial_bool {
1438       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1439     }
1440   }
1441 }

```

9.1.10 Bold Upright: `\mathbfup`

```

1442 \cs_new:Nn \um_config_mathbfup_num:n {
1443   \um_set_mathalphabet_numbers:Nnn \mathbf {up}{#1}
1444   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
1445 }
1446 \cs_new:Nn \um_config_mathbfup_Latin:n {
1447   \bool_if:NT \g_um_bfupLatin_bool {
1448     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1449   }
1450   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
1451   \bool_if:NTF \g_um_bfliteral_bool {
1452     \um_map_chars_Latin:nn {bfup} {#1}
1453     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
1454   }{
1455     \bool_if:NT \g_um_bfupLatin_bool {
1456       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1457       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1458     }
1459   }
1460 }
1461 \cs_new:Nn \um_config_mathbfup_latin:n {
1462   \bool_if:NT \g_um_bfuplatin_bool {
1463     \um_map_chars_latin:nn {bfup,bfit} {#1}
1464   }
1465   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
1466   \bool_if:NTF \g_um_bfliteral_bool {
1467     \um_map_chars_latin:nn {bfup} {#1}
1468     \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
1469   }{
1470     \bool_if:NT \g_um_bfuplatin_bool {
1471       \um_map_chars_latin:nn {bfup,bfit} {#1}
1472       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1473     }
1474   }
1475 }
1476 \cs_new:Nn \um_config_mathbfup_Greek:n {
1477   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
1478   \bool_if:NTF \g_um_bfliteral_bool {
1479     \um_map_chars_Greek:nn {bfup}{#1}
1480     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}
1481   }{
1482     \bool_if:NT \g_um_bfupGreek_bool {
1483       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1484       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1485     }
1486   }

```

```

1487 }
1488 \cs_new:Nn \um_config_mathbfup_greek:n {
1489   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
1490   \bool_if:NTF \g_um_bfliteral_bool {
1491     \um_map_chars_greek:nn {bfup} {#1}
1492     \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
1493   }{
1494     \bool_if:NT \g_um_bfupgreek_bool {
1495       \um_map_chars_greek:nn {bfup,bfit} {#1}
1496       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1497     }
1498   }
1499 }
1500 \cs_new:Nn \um_config_mathbfup_misc:n {
1501   \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{#1}
1502   \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{#1}
1503   \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{#1}
1504   \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{#1}
1505   \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{#1}
1506   \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{#1}
1507   \bool_if:NTF \g_um_bfliteral_bool {
1508     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{#1}
1509     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{#1}
1510   }{
1511     \bool_if:NT \g_um_upNabla_bool {
1512       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1513     }
1514     \bool_if:NT \g_um_uppartial_bool {
1515       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1516     }
1517   }
1518 }

```

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

```

1519 \cs_new:Nn \um_config_mathbffrak_Latin:n {
1520   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
1521 }
1522 \cs_new:Nn \um_config_mathbffrak_latin:n {
1523   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
1524 }

```

9.1.12 Bold script or calligraphic: `\mathbfscr`

```

1525 \cs_new:Nn \um_config_mathbfscr_Latin:n {
1526   \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{#1}
1527 }
1528 \cs_new:Nn \um_config_mathbfscr_latin:n {
1529   \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{#1}

```


1530 }

9.1.13 Bold upright sans serif: `\mathbfsfup`

```
1531 \cs_new:Nn \um_config_mathbfsfup_num:n {
1532   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{#1}
1533   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
1534 }
1535 \cs_new:Nn \um_config_mathbfsfup_Latin:n {
1536   \bool_if:NTF \g_um_sfliteral_bool {
1537     \um_map_chars_Latin:nn {bfsfup} {#1}
1538     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
1539   }{
1540     \bool_if:NT \g_um_upsans_bool {
1541       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1542       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1543     }
1544   }
1545   \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
1546 }
1547 \cs_new:Nn \um_config_mathbfsfup_latin:n {
1548   \bool_if:NTF \g_um_sfliteral_bool {
1549     \um_map_chars_latin:nn {bfsfup} {#1}
1550     \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
1551   }{
1552     \bool_if:NT \g_um_upsans_bool {
1553       \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1554       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1555     }
1556   }
1557   \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
1558 }
1559 \cs_new:Nn \um_config_mathbfsfup_Greek:n {
1560   \bool_if:NTF \g_um_sfliteral_bool {
1561     \um_map_chars_Greek:nn {bfsfup}{#1}
1562     \um_set_mathalphabet_Greek:Nnn \mathbfsf {up}{#1}
1563   }{
1564     \bool_if:NT \g_um_upsans_bool {
1565       \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
1566       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
1567     }
1568   }
1569   \um_set_mathalphabet_Greek:Nnn \mathbfsfup {up,it}{#1}
1570 }
1571 \cs_new:Nn \um_config_mathbfsfup_greek:n {
1572   \bool_if:NTF \g_um_sfliteral_bool {
1573     \um_map_chars_greek:nn {bfsfup} {#1}
```

```

1574 \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
1575 }{
1576 \bool_if:NT \g_um_upsans_bool {
1577 \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1578 \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1579 }
1580 }
1581 \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
1582 }
1583 \cs_new:Nn \um_config_mathbfsfup_misc:n {
1584 \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{#1}
1585 \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{#1}
1586 \bool_if:NTF \g_um_sfliteral_bool {
1587 \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{#1}
1588 \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{#1}
1589 }{
1590 \bool_if:NT \g_um_upNabla_bool {
1591 \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1592 }
1593 \bool_if:NT \g_um_uppartial_bool {
1594 \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1595 }
1596 }
1597 }

```

9.1.14 Bold italic sans serif: `\mathbfsfit`

```

1598 \cs_new:Nn \um_config_mathbfsfit_Latin:n {
1599 \bool_if:NTF \g_um_sfliteral_bool {
1600 \um_map_chars_Latin:nn {bfsfit} {#1}
1601 \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
1602 }{
1603 \bool_if:NF \g_um_upsans_bool {
1604 \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1605 \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1606 }
1607 }
1608 \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
1609 }
1610 \cs_new:Nn \um_config_mathbfsfit_latin:n {
1611 \bool_if:NTF \g_um_sfliteral_bool {
1612 \um_map_chars_latin:nn {bfsfit} {#1}
1613 \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
1614 }{
1615 \bool_if:NF \g_um_upsans_bool {
1616 \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1617 \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}

```

```

1618     }
1619   }
1620   \um_set_mathalphabet_latinnnn \mathbfsfit {up,it}{#1}
1621 }
1622 \cs_new:Nn \um_config_mathbfsfit_greek:n {
1623   \bool_if:NTF \g_um_sfliteral_bool {
1624     \um_map_chars_greek:nn {bfsfit}{#1}
1625     \um_set_mathalphabet_greek:Nnn \mathbfsf {it}{#1}
1626   }{
1627     \bool_if:NF \g_um_upsans_bool {
1628       \um_map_chars_greek:nn {bfsfup,bfsfit}{#1}
1629       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it}{#1}
1630     }
1631   }
1632   \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it}{#1}
1633 }
1634 \cs_new:Nn \um_config_mathbfsfit_greek:n {
1635   \bool_if:NTF \g_um_sfliteral_bool {
1636     \um_map_chars_greek:nn {bfsfit} {#1}
1637     \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
1638   }{
1639     \bool_if:NF \g_um_upsans_bool {
1640       \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1641       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1642     }
1643   }
1644   \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
1645 }
1646 \cs_new:Nn \um_config_mathbfsfit_misc:n {
1647   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{#1}
1648   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{#1}
1649   \bool_if:NTF \g_um_sfliteral_bool {
1650     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{#1}
1651     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{#1}
1652   }{
1653     \bool_if:NF \g_um_upNabla_bool {
1654       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1655     }
1656     \bool_if:NF \g_um_uppartial_bool {
1657       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1658     }
1659   }
1660 }

```

10 Definitions of the math symbols

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

`\um@scancharlet` We need to do some trickery to transform the `\UnicodeMathSymbol` argument `\um@scanactivedef` "ABCDEF into the X_YTeX ‘caret input’ form `^^^^^abcdef`. It is *very important* that the argument has five characters. Otherwise we need to change the number of ^ chars.

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.

```

1661 \begingroup
1662   \char_make_other:N \^
1663   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
1664     \lowercase{
1665       \tl_rescan:nn {
1666         \char_make_other:N \{
1667         \char_make_other:N \}
1668         \char_make_other:N \&
1669         \char_make_other:N \%
1670         \char_make_other:N \$
1671       }{
1672         \global\let#1=^^^^^#2
1673     }
1674   }
1675 }
```

Making ^ the right catcode isn’t strictly necessary right now but it helps to future proof us with, e.g., `breqn`.

```

1676 \gdef\um@scanactivedef"#1\@nil#2{
1677   \lowercase{
1678     \tl_rescan:nn{
1679       \ExplSyntaxOn
1680       \char_make_math_superscript:N\^
1681     }{
1682       \global\def^^^^^#1{#2}
1683     }
1684   }
1685 }
1686 \endgroup
```

Now give `\UnicodeMathSymbol` a definition in terms of `\um@scancharlet` and we’re good to go. Make sure # is an ‘other’ so that we don’t get confused with `\mathoctothorpe`.

```

1687 \begingroup
```

```

1688 \char_make_math_superscript:N\^
1689 \def\UnicodeMathSymbol#1#2#3#4{
1690   \um@scancharlet#2=#1\@nil\ignorespaces
1691 }
1692 \char_make_other:N \#
1693 \@input{unicode-math-table.tex}
1694 \endgroup
Fix \backslash:
1695 \group_begin:
1696   \lccode`*=`\\
1697   \char_make_escape:N \|
1698   \char_make_other:N \\
1699   |lowercase{
1700 |group_end:|let|backslash=*}

```

11 Epilogue

Lots of little things to tidy up.

11.0.15 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 behaviour OpenType font with the MATH table, we also see different behaviour after the `ssty` feature is applied:

```

u+2032 prime in the ‘scriptstyle’ font:  $x'$ 

```

The shrinking and offsetting is done as it is turned into a superscript. This means, luckily, that by default things work nicely for single primes.

However, it would be nice to use the pre-composed primes above if they exist in the font; consider x''' vs. x''' . 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.

```

1701 \muskip_new:N \g_um_primekern_muskip
1702 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
1703 \num_new:N \l_um_primecount_num

1704 \cs_new:Nn \um_nprimes:Nn {
1705   ^{
1706     #1
1707     \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
1708   }
1709 }
1710 \cs_new:Nn \um_nprimes_select:nn {
1711   \prg_case_int:nnn {#2}{
1712     {1} { ^{#1} }
1713     {2} {
1714       \um_glyph_if_exist:nTF {"2033} { ^{\um_prime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1715     }
1716     {3} {
1717       \um_glyph_if_exist:nTF {"2034} { ^{\um_prime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1718     }
1719     {4} {
1720       \um_glyph_if_exist:nTF {"2057} { ^{\um_prime_quad_mchar} } {\um_nprimes:Nn #1 {#2}}
1721     }
1722   }{
1723     \um_nprimes:Nn #1 {#2}
1724   }
1725 }
1726 \cs_new:Nn \um_nbackprimes_select:nn {
1727   \prg_case_int:nnn {#2}{
1728     {1} { ^{#1} }
1729     {2} {
1730       \um_glyph_if_exist:nTF {"2033} { ^{\um_backprime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1731     }
1732     {3} {
1733       \um_glyph_if_exist:nTF {"2034} { ^{\um_backprime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1734     }
1735   }{
1736     \um_nprimes:Nn #1 {#2}
1737   }

```

1738 }

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

```
1739 \cs_new:Nn \um_scan_prime: {
1740   \num_zero:N \l_um_primecount_num
1741   \um_scanprime_collect:N \um_prime_single_mchar
1742 }
1743 \cs_new:Nn \um_scan_dprime: {
1744   \num_set:Nn \l_um_primecount_num {1}
1745   \um_scanprime_collect:N \um_prime_single_mchar
1746 }
1747 \cs_new:Nn \um_scan_trprime: {
1748   \num_set:Nn \l_um_primecount_num {2}
1749   \um_scanprime_collect:N \um_prime_single_mchar
1750 }
1751 \cs_new:Nn \um_scan_qprime: {
1752   \num_set:Nn \l_um_primecount_num {3}
1753   \um_scanprime_collect:N \um_prime_single_mchar
1754 }
1755 \cs_new:Nn \um_scanprime_collect:N {
1756   \num_incr:N \l_um_primecount_num
1757   \peek_meaning_remove:NTF ' {
1758     \um_scanprime_collect:N #1
1759   }{
1760     \peek_meaning_remove:NTF \um_scan_prime: {
1761       \um_scanprime_collect:N #1
1762     }{
1763       \peek_meaning_remove:NTF ^^^^2032 {
1764         \um_scanprime_collect:N #1
1765       }{
1766         \peek_meaning_remove:NTF \um_scan_dprime: {
1767           \num_incr:N \l_um_primecount_num
1768           \um_scanprime_collect:N #1
1769         }{
1770           \peek_meaning_remove:NTF ^^^^2033 {
1771             \num_incr:N \l_um_primecount_num
1772             \um_scanprime_collect:N #1
1773           }{
1774             \peek_meaning_remove:NTF \um_scan_trprime: {
1775               \num_add:Nn \l_um_primecount_num {2}
1776               \um_scanprime_collect:N #1
1777             }{
1778               \peek_meaning_remove:NTF ^^^^2034 {
1779                 \num_add:Nn \l_um_primecount_num {2}
1780                 \um_scanprime_collect:N #1
1781               }{
1782                 \peek_meaning_remove:NTF \um_scan_qprime: {
```

```

1783         \num_add:Nn \l_um_primecount_num {3}
1784         \um_scanprime_collect:N #1
1785     }{
1786         \peek_meaning_remove:NTF ^^^^2057 {
1787             \num_add:Nn \l_um_primecount_num {3}
1788             \um_scanprime_collect:N #1
1789         }{
1790             \um_nprimes_select:nn {#1} {\l_um_primecount_num}
1791         }
1792     }
1793 }
1794 }
1795 }
1796 }
1797 }
1798 }
1799 }
1800 }
1801 \cs_new:Nn \um_scan_backprime: {
1802     \num_zero:N \l_um_primecount_num
1803     \um_scanbackprime_collect:N \um_backprime_single_mchar
1804 }
1805 \cs_new:Nn \um_scan_backdprime: {
1806     \num_set:Nn \l_um_primecount_num {1}
1807     \um_scanbackprime_collect:N \um_backprime_single_mchar
1808 }
1809 \cs_new:Nn \um_scan_backtrprime: {
1810     \num_set:Nn \l_um_primecount_num {2}
1811     \um_scanbackprime_collect:N \um_backprime_single_mchar
1812 }
1813 \cs_new:Nn \um_scanbackprime_collect:N {
1814     \num_incr:N \l_um_primecount_num
1815     \peek_meaning_remove:NTF ` {
1816         \um_scanbackprime_collect:N #1
1817     }{
1818         \peek_meaning_remove:NTF \um_scan_backprime: {
1819             \um_scanbackprime_collect:N #1
1820         }{
1821             \peek_meaning_remove:NTF ^^^^2035 {
1822                 \um_scanbackprime_collect:N #1
1823             }{
1824                 \peek_meaning_remove:NTF \um_scan_backdprime: {
1825                     \num_incr:N \l_um_primecount_num
1826                     \um_scanbackprime_collect:N #1
1827                 }{
1828                     \peek_meaning_remove:NTF ^^^^2036 {

```



```

1829         \num_incr:N \l_um_primecount_num
1830         \um_scanbackprime_collect:N #1
1831     }{
1832         \peek_meaning_remove:NTF \um_scan_backtrprime: {
1833             \num_add:Nn \l_um_primecount_num {2}
1834             \um_scanbackprime_collect:N #1
1835         }{
1836             \peek_meaning_remove:NTF ^^^^2037 {
1837                 \num_add:Nn \l_um_primecount_num {2}
1838                 \um_scanbackprime_collect:N #1
1839             }{
1840                 \um_nbackprimes_select:nn {#1} {\l_um_primecount_num}
1841             }
1842         }
1843     }
1844 }
1845 }
1846 }
1847 }
1848 }

1849 \cs_set_eq:NN \prime \um_scan_prime:
1850 \cs_set_eq:NN \drime \um_scan_dprime:
1851 \cs_set_eq:NN \trprime \um_scan_trprime:
1852 \cs_set_eq:NN \qprime \um_scan_qprime:
1853 \cs_set_eq:NN \backprime \um_scan_backprime:
1854 \cs_set_eq:NN \backdprime \um_scan_backdprime:
1855 \cs_set_eq:NN \backtrprime \um_scan_backtrprime:
1856 \group_begin:
1857   \char_make_active:N \'
1858   \char_make_active:N `
1859   \char_make_active:n {"2032}
1860   \char_make_active:n {"2033}
1861   \char_make_active:n {"2034}
1862   \char_make_active:n {"2057}
1863   \char_make_active:n {"2035}
1864   \char_make_active:n {"2036}
1865   \char_make_active:n {"2037}
1866   \cs_gset_eq:NN ' \um_scan_prime:
1867   \cs_gset_eq:NN ^^^^2032 \um_scan_prime:
1868   \cs_gset_eq:NN ^^^^2033 \um_scan_dprime:
1869   \cs_gset_eq:NN ^^^^2034 \um_scan_trprime:
1870   \cs_gset_eq:NN ^^^^2057 \um_scan_qprime:
1871   \cs_gset_eq:NN ` \um_scan_backprime:
1872   \cs_gset_eq:NN ^^^^2035 \um_scan_backprime:
1873   \cs_gset_eq:NN ^^^^2036 \um_scan_backdprime:
1874   \cs_gset_eq:NN ^^^^2037 \um_scan_backtrprime:

```

```
1875 \group_end:
```

11.0.16 Unicode radicals

Undo the damage made to `\sqrt`:

```
1876 \DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
```

`\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`.

```
1877 \def\r@@t#1#2{
1878   \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
1879   \um@scaled@apply{#1}{\kern}{\fontdimen63\l_um_font}
1880   \raise \dimexpr(
1881     \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-
1882     \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
1883   )\relax
1884   \copy \rootbox
1885   \um@scaled@apply{#1}{\kern}{\fontdimen64\l_um_font}
1886   \box \z@
1887 }
```

11.0.17 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_YL^AT_EX 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?

First, the setup of each mathactive char:

```
1888 \prop_new:N \g_um_supers_prop
1889 \prop_new:N \g_um_subs_prop
1890
1891 \group_begin:
1892
1893 % Populate a property list with superscript characters; their mean-
1894 % ing as their key,
1895 % for reasons that will become apparent soon, and their replace-
1896 % ment as each key's value.
1897 % Then make the superscript active and bind it to the scanning function.
1898 %
```

```

1897 % \cs{scantokens} makes this process much simpler since we can acti-
1898 vate the char
1899 % and assign its meaning in one step.
1899 \cs_set:Nn \um_setup_active_superscript:nn {
1900   \prop_gput:Nxn \g_um_supers_prop   {\meaning #1} {#2}
1901   \char_make_active:n {\#1}
1902   \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1903   \scantokens{
1904     \cs_gset:Npn #1 {
1905       \tl_set:Nn \l_um_ss_chain_tl {#2}
1906       \cs_set_eq:NN \um_sub_or_super:n \sp
1907       \tl_set:Nn \l_um_tmpa_tl {supers}
1908       \um_scan_sscript:
1909     }
1910   }
1911 }
1912
1913 \um_setup_active_superscript:nn {^^^2070} {0}
1914 \um_setup_active_superscript:nn {^^^00b9} {1}
1915 \um_setup_active_superscript:nn {^^^00b2} {2}
1916 \um_setup_active_superscript:nn {^^^00b3} {3}
1917 \um_setup_active_superscript:nn {^^^2074} {4}
1918 \um_setup_active_superscript:nn {^^^2075} {5}
1919 \um_setup_active_superscript:nn {^^^2076} {6}
1920 \um_setup_active_superscript:nn {^^^2077} {7}
1921 \um_setup_active_superscript:nn {^^^2078} {8}
1922 \um_setup_active_superscript:nn {^^^2079} {9}
1923 \um_setup_active_superscript:nn {^^^207a} {+}
1924 \um_setup_active_superscript:nn {^^^207b} {-}
1925 \um_setup_active_superscript:nn {^^^207c} {=}
1926 \um_setup_active_superscript:nn {^^^207d} {(}
1927 \um_setup_active_superscript:nn {^^^207e} {)}
1928 \um_setup_active_superscript:nn {^^^2071} {i}
1929 \um_setup_active_superscript:nn {^^^207f} {n}
1930
1931 % Ditto above.
1932 \cs_set:Nn \um_setup_active_subscript:nn {
1933   \prop_gput:Nxn \g_um_subs_prop   {\meaning #1} {#2}
1934   \char_make_active:n {\#1}
1935   \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1936   \scantokens{
1937     \cs_gset:Npn #1 {
1938       \tl_set:Nn \l_um_ss_chain_tl {#2}
1939       \cs_set_eq:NN \um_sub_or_super:n \sb
1940       \tl_set:Nn \l_um_tmpa_tl {subs}
1941       \um_scan_sscript:

```

```

1942     }
1943   }
1944 }
1945
1946 \um_setup_active_subscript:nn {^^^2080} {0}
1947 \um_setup_active_subscript:nn {^^^2081} {1}
1948 \um_setup_active_subscript:nn {^^^2082} {2}
1949 \um_setup_active_subscript:nn {^^^2083} {3}
1950 \um_setup_active_subscript:nn {^^^2084} {4}
1951 \um_setup_active_subscript:nn {^^^2085} {5}
1952 \um_setup_active_subscript:nn {^^^2086} {6}
1953 \um_setup_active_subscript:nn {^^^2087} {7}
1954 \um_setup_active_subscript:nn {^^^2088} {8}
1955 \um_setup_active_subscript:nn {^^^2089} {9}
1956 \um_setup_active_subscript:nn {^^^208a} {+}
1957 \um_setup_active_subscript:nn {^^^208b} {-}
1958 \um_setup_active_subscript:nn {^^^208c} {=}
1959 \um_setup_active_subscript:nn {^^^208d} {(}
1960 \um_setup_active_subscript:nn {^^^208e} {)}
1961 \um_setup_active_subscript:nn {^^^2090} {a}
1962 \um_setup_active_subscript:nn {^^^2091} {e}
1963 \um_setup_active_subscript:nn {^^^1d62} {i}
1964 \um_setup_active_subscript:nn {^^^2092} {o}
1965 \um_setup_active_subscript:nn {^^^1d63} {r}
1966 \um_setup_active_subscript:nn {^^^1d64} {u}
1967 \um_setup_active_subscript:nn {^^^1d65} {v}
1968 \um_setup_active_subscript:nn {^^^2093} {x}
1969 \um_setup_active_subscript:nn {^^^1d66} {\beta}
1970 \um_setup_active_subscript:nn {^^^1d67} {\gamma}
1971 \um_setup_active_subscript:nn {^^^1d68} {\rho}
1972 \um_setup_active_subscript:nn {^^^1d69} {\phi}
1973 \um_setup_active_subscript:nn {^^^1d6a} {\chi}
1974
1975 \group_end:
1976
1977 % The scanning command, evident in its purpose:
1978 \cs_new:Nn \um_scan_sscript: {
1979   \um_scan_sscript:TF {
1980     \um_scan_sscript:
1981   }{
1982     \um_sub_or_super:n {\l_um_ss_chain_tl}
1983   }
1984 }
1985
1986 % The main theme here is stolen from the source to the vari-
    ous \cs{peek_} functions.

```

```

1987 % Consider this function as simply boilerplate:
1988 \cs_new:Nn \um_scan_sscript:TF {
1989   \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
1990   \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
1991   \tl_set:Nx \l_peek_false_tl { \exp_not:n{ \group_align_safe_end: #2 } }
1992   \group_align_safe_begin:
1993     \peek_after:NN \um_peek_execute_branches_ss:
1994 }
1995
1996 % We do not skip spaces when scanning ahead, and we explicitly wish to
1997 % bail out on encountering a space or a brace.
1998 \cs_new:Npn \um_peek_execute_branches_ss: {
1999   \bool_if:nTF {
2000     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2001     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2002     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2003   }
2004   { \l_peek_false_tl }
2005   { \um_peek_execute_branches_ss_aux: }
2006 }
2007
2008 % This is the actual comparison code.
2009 % Because the peeking has already tokenised the next token,
2010 % it's too late to extract its charcode directly. Instead,
2011 % we look at its meaning, which remains a `character' even
2012 % though it is itself math-active. If the character is ever
2013 % made fully active, this will break our assumptions!
2014 %
2015 % If the char's meaning exists as a property list key, we
2016 % build up a chain of sub-/superscripts and iterate. (If not, exit and
2017 % typeset what we've already collected.)
2018 \cs_new:Nn \um_peek_execute_branches_ss_aux: {
2019   \prop_if_in:cxTF
2020     {g_um_\l_um_tmpa_tl _prop}
2021     {\meaning\l_peek_token}
2022   {
2023     \prop_get:cxN
2024       {g_um_\l_um_tmpa_tl _prop}
2025       {\meaning\l_peek_token}
2026     \l_um_tmpb_tl
2027     \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
2028     \l_peek_true_tl
2029   }
2030   {\l_peek_false_tl}
2031 }

```

11.0.18 Synonyms and all the rest

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

```
2032 \def\operator@font{\um_setup_mathup:}
2033 \def\to{\rightarrow}
2034 \def\overrightarrow{\vec}
2035 \def\le{\leq}
2036 \def\ge{\geq}
2037 \def\neq{\neq}
2038 \def\triangle{\mathord{\bigtriangleup}}
2039 \def\bigcirc{\mdlgwhtcircle}
2040 \def\circ{\vysmwhtcircle}
2041 \def\bullet{\smbkcircle}
2042 \def\mathyen{\yen}
2043 \def\mathsterling{\sterling}
```

Define `\colon` as a `mathpunct` `':'`. This is wrong: it should be U+003A colon instead!

```
2044 \@ifpackageloaded{amsmath}{
2045   % define their own colon, perhaps I should just steal it.
2046 }{
2047   \cs_set_protected:Npn \colon {
2048     \bool_if:NTF \g_um_literal_colon_bool {:] { \mathpunct{:} }
2049   }
2050 }
```

`\mathcal`

```
2051 \def\mathcal{\mathscr}
```

`\mathrm`

```
2052 \def\mathrm{\mathup}
2053 \let\mathfence\mathord
```

11.0.19 Compatibility

Note that `amsmath` will always be loaded before `unicode-math`. (Conflicts occur if you try it the other way around.)

- 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:

```
2054 \bool_new:N \g_um_amsmath_bool
2055 \@ifpackageloaded{amsmath}{
2056   \bool_set_true:N \g_um_amsmath_bool
2057 }
```

```

2058     \bool_set_false:N \g_um_amsmath_bool
2059   }
2060   \bool_if:NT \g_um_amsmath_bool {
2061     \tl_remove_in:Nn \@begindocumenthook {
2062       \mathchardef\std@minus\mathcode`\-\relax
2063       \mathchardef\std@equal\mathcode`\=\relax
2064     }
2065     \AtBeginDocument {
2066       \def\std@minus{\XeTeXmathcharnum\XeTeXmathcodenum`\-\relax}
2067       \def\std@equal{\XeTeXmathcharnum\XeTeXmathcodenum`\=\relax}
2068     }
2069   }

```

- 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:

```

2070   \@ifpackageloaded{amsopn}{
2071     \cs_set:Npn \newmcodes@ {
2072       \mathcode`\'39
2073       \mathcode`\*42
2074       \mathcode`\."613A%
2075     % \ifnum\mathcode`\- =45 \else
2076     %   \mathchardef\std@minus\mathcode`\-\relax
2077     % \fi
2078       \mathcode`\-45
2079       \mathcode`\ /47
2080       \mathcode`\."603A\relax
2081     }
2082   }{}

```

- \mathinner items:

```

2083     \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2084     \cs_set:Npn \cdots {\mathinner{\unicodcdots}}
2085     \bool_if:NT \g_um_amsmath_bool {
2086       \cs_set_eq:NN \@cdots \cdots
2087       \cs_set_eq:NN \dotso@ \cdots
2088     }

```

Octothorpe is an odd one:

```

2089 \AtBeginDocument{
2090   \def\widehat{\hat}
2091   \def\widetilde{\tilde}
2092 }

```

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

```

\Digamma
2093 \def\digamma{\updigamma}
2094 \def\Digamma{\upDigamma}

```

Overriding amsmath definitions:

```
2095 \AtBeginDocument{
2096   \def\@cdots{\mathinner{\cdots}}
2097 }
```

Interaction with beamer:

```
2098 \@ifclassloaded{beamer}{
2099   \ifbeamer@suppressreplacements\else
2100     \PackageWarningNoLine{unicode-math}{
2101       Disabling~ beamer's~ math~ setup.^{^}
2102       Please~ load~ beamer~ with~ the~ [professionalfonts]~ class~ option
2103     }
2104     \beamer@suppressreplacementstrue
2105   \fi
2106 }{}
```

The end.

```
2107 \ExplSyntaxOff
```

12 STIX table data extraction

The source for the \TeX 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 \XeTeX , and then hand-edited by the author; the result is `unicode-math-table.tex`.

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.

```
2108 #!/bin/sh
2109
2110 cat stix-tbl.txt |
2111 awk '

```

If the USV isn't repeated (TODO: check this is valid!) and the entry isn't one of the weird ones in the big block at the end of the STIX table (TODO: check that out!)

```
2112 {if (usv != substr($0,2,5) && substr($0,2,1) != " ")
2113   {usv = substr($0,2,5);
2114     texname = substr($0,84,25);
2115     class = substr($0,57,1);
2116     description = tolower(substr($0,233,350));

```



```

2117     if (texname ~ /[\\]/ &&
2118         substr(texname,0,5) != "\\text" &&
2119         substr(texname,0,4) != "\\ipa" &&
2120         substr(texname,0,5) != "\\tone" &&
2121         substr(texname,3,1) != " " &&
2122         class != " " &&
2123         description !~ /<reserved>/ )

```

```

2124     print "\\UnicodeMathSymbol{\\\" \" \\
2125         usv \"{\" \\
2126         texname \"{\" \\
2127         class \"{\" \\
2128         description \"%\";
2129 }{' - |
```

```
2130 sed -e ' s/{N}/{\\mathord}/ ' \
```

```

2131 -e ' s/{F}/{\\mathord}/ ' \
2132 -e ' s/{A}/{\\mathalpha}/ ' \
2133 -e ' s/{D}/{\\mathaccent}/ ' \
2134 -e ' s/{P}/{\\mathpunct}/ ' \
2135 -e ' s/{B}/{\\mathbin}/ ' \
2136 -e ' s/{R}/{\\mathrel}/ ' \
2137 -e ' s/{L}/{\\mathop}/ ' \
2138 -e ' s/{O}/{\\mathopen}/ ' \
2139 -e ' s/{C}/{\\mathclose}/ ' \

```

```
2140 -e ' s/\^/\string^/ ' > unicode-math.tex
```

`\DeclareSymbolFont{⟨name⟩}⟨NFSS decl.⟩`
 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{<cmd>}{NFSS decl.}
```

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{<cmd>}{<name>}
```

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{<maths version>}` command.

```
\SetSymbolFont{<name>}{<maths version>}{NFSS decl.}
```

```
\SetMathAlphabet{<cmd>}{<maths version>}{NFSS decl.}
```

Maths symbols Symbol definitions in maths for both characters (=) and macros (`\eqdef`): `\DeclareMathSymbol{<symbol>}{<type>}{<named font>}{<slot>}` 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 \XTeX . The syntax used in \LaTeX ’s NFSS is therefore not so relevant here.

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

```
\DeclareMathDelimiter{<symbol>}{<type>}{<sym. font>}{<slot>}{<sym. font>}{<slot>}
```

Radicals Similar to delimiters (`\DeclareMathRadical` takes the same syntax) but behave ‘weirdly’. `\sqrt` might very well be the only one.

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 \XTeX .

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 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).

<code>\fontdimen</code>	Dimension name	Description
10	<code>SCRIPTPERCENTSCALEDOWN</code>	Percentage of scaling down for script level 1. Suggested value: 80%.
11	<code>SCRIPTSCRIPTPERCENTSCALEDOWN</code>	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	<code>DELIMITEDSUBFORMULAMINHEIGHT</code>	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height \times 1.5.
13	<code>DISPLAYOPERATORMINHEIGHT</code>	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.
14	<code>MATHLEADING</code>	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 <code>os2.sTypoAscender</code> + <code>os2.sTypoLineGap</code> – <code>MathLeading</code> or with ink going below <code>os2.sTypoDescender</code> will result in increasing line height.
15	<code>AXISHEIGHT</code>	Axis height of the font.
16	<code>ACCENTBASEHEIGHT</code>	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (<code>os2.sxHeight</code>) plus any possible overshoots.

\fontdimen	Dimension name	Description
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: $\frac{1}{5}$ 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.
21	SUPERSCRIPSHIFTUP	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.
22	SUPERSCRIPSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPBOTTOMMIN	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×default rule thickness.
26	SUPERSCRIPBOTTOMMAX-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.

\fontdimen	Dimension name	Description
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	LOWERLIMITBASELINE DROP-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	STACKTOPDISPLAYSTYLESHIFTUP	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.
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	STRETCHSTACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.

\fontdimen	Dimension name	Description
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
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.

\fontdimen	Dimension name	Description
51	SKEWEDFRACTION-HORIZONTALGAP	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SKEWEDFRACTIONVERTICALGAP	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	RADICALDISPLAYSTYLEVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
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.

\fontdimen	Dimension name	Description
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%.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols			
<code>\"</code>	2124	<code>\@tempswafalse</code>	780
<code>\#</code>	1692	<code>\@tempwattrue</code>	784, 787, 812, 817, 822, 827
<code>\\$</code>	1670	<code>\@xDeclareMathDelimiter</code>	431
<code>\%</code>	1669	<code>\@xxDeclareMathDelimiter</code>	430
<code>\&</code>	1668	<code>\\</code>	1696, 1698, 2117–2120, 2124, 2130–2140
<code>\'</code>	1857, 2072	<code>\{</code>	1666
<code>*</code>	588, 1696, 2073	<code>\}</code>	1667
<code>\-</code>	587, 2062, 2066, 2075, 2076, 2078	<code>\^</code>	1662, 1680, 1688, 2140
<code>\.</code>	2074	<code>\`</code>	635, 1858
<code>\/</code>	642, 2079	<code>\ </code>	1697
<code>\:</code>	590, 2080	Numbers	
<code>\:::</code>	901	<code>\0</code>	1033
<code>\::f</code>	901	<code>\9</code>	1033
<code>\::n</code>	901		
<code>\<</code>	646		
<code>\=</code>	2063, 2067	<code>_</code>	2124–2127, 2130–2139
<code>\></code>	647	A	
<code>\@DeclareMathDelimiter</code>	430	<code>\A</code>	1030
<code>\@DeclareMathSizes</code>	422	<code>\a</code>	1029
<code>\@backslashchar</code>	1114, 1133	<code>\addnolimits</code>	703
<code>\@begindocumenthook</code>	2061	<code>\addtoversion</code>	423
<code>\@cclvi</code>	450	<code>\alloc@</code>	450
<code>\@cdots</code>	2086, 2096	<code>\alpha@elt</code>	424
<code>\@empty</code>	506, 779, 794, 815, 820	<code>\alpha@list</code>	424
<code>\@ifclassloaded</code>	2098	<code>\AtBeginDocument</code>	972, 2065, 2089, 2095
<code>\@ifnextchar</code>	1876	<code>\awint</code>	699
<code>\@ifpackageloaded</code>	2044, 2055, 2070	B	
<code>\@ii</code>	778, 779, 781, 783, 786, 791	<code>\B</code>	35
<code>\@input</code>	570, 1693	<code>\backdprime</code>	1854
<code>\@marker</code>	791, 810	<code>\backprime</code>	1853
<code>\@nil</code>	457, 618, 791, 804–807, 846, 1663, 1676, 1690	<code>\backtrprime</code>	1855
<code>\@preamblecmds</code>	434	<code>\beamer@suppressreplacementstrue</code>	2104
<code>\@sqrt</code>	1876	<code>\begingroup</code>	454, 1661, 1687
<code>\@tempa</code>	259, 323, 349, 358, 376, 394, 402, 409, 737, 782, 806, 808, 820	<code>\beta</code>	1969
<code>\@tempb</code>	259, 260, 323, 324, 349, 350, 358, 359, 376, 377, 394, 395, 402, 403, 409, 410, 737, 738, 809, 810, 815	<code>\bgroup</code>	1150
		<code>\bigcirc</code>	2039
		<code>\bigtriangleup</code>	2038

90

<code>\cs_set:cpn</code>	1146	<code>\do</code>	434
<code>\cs_set:Nn</code>	365, 383, 452, 496, 499, 578, 581, 682, 689, 834, 850, 853, 856, 859, 864, 869, 880, 886, 889, 892, 1132, 1899, 1932	<code>\dorestore@version</code>	425
<code>\cs_set:Npn</code> 769, 773, 901, 2071, 2083, 2084		<code>\dotsb@</code>	2087
<code>\cs_set:Npx</code>	515, 519	<code>\dp</code>	1882
<code>\cs_set_eq:cN</code>	1135	<code>\drime</code>	1850
<code>\cs_set_eq:NN</code>			
. 554–558, 562–566, 1081, 1093, 1849–1855, 1906, 1939, 2086, 2087		E	
<code>\cs_set_protected:cpn</code>	1149	<code>\E</code>	38
<code>\cs_set_protected:Npn</code>	2047	<code>\e</code>	66
<code>\cs_to_str:N</code>	457, 460–462, 468, 483, 484, 684, 1094, 1101	<code>\edef</code>	805, 806
<code>\csname</code>		<code>\egroup</code>	1147, 1152
457, 460, 497, 500, 507, 677, 680, 1153		<code>\else</code>	442, 445, 465, 474, 479, 485, 488, 543, 785, 790, 796, 814, 819, 824, 2075, 2099
D		<code>\else:</code>	1138
<code>\D</code>	37	<code>\encodingdefault</code>	569
<code>\d</code>	65	<code>\endcsname</code>	
<code>\DeclareDocumentCommand</code>		457, 460, 497, 500, 507, 677, 680, 1153	
..... 255, 502, 703, 706		<code>\endgroup</code>	458, 1686, 1694
<code>\DeclareMathAccent</code>	429	<code>\epsilon</code>	984
<code>\DeclareMathAlphabet</code>	428	<code>\errorcontextlines</code>	910
<code>\DeclareMathDelimiter</code>	430	<code>\etex_iffontchar:D</code>	1138
<code>\DeclareMathRadical</code>	431	<code>\ExecuteOptionsX</code>	418
<code>\DeclareMathSizes</code>	421	<code>\exp_after:wN</code>	
<code>\DeclareMathSymbol</code>	429	758, 761, 1087, 1089, 1091, 1094, 1102	
<code>\DeclareMathVersion</code>	423, 509	<code>\exp_args:Nc</code>	579
<code>\DeclareRobustCommand</code>	1876	<code>\exp_args:Nnff</code>	901, 904, 912
<code>\DeclareSymbolFont</code>	426, 568	<code>\exp_args:NV</code>	1105
<code>\DeclareSymbolFontAlphabet</code>	432	<code>\exp_not:c</code>	462, 982
<code>\DeclareSymbolFontAlphabet@</code>	432	<code>\exp_not:N</code>	516, 520, 685
<code>\def</code>	436, 439, 450, 711, 804, 806–809, 1682, 1689, 1877, 2032–2043, 2051, 2052, 2066, 2067, 2090, 2091, 2093, 2094, 2096	<code>\exp_not:n</code>	463, 1989, 1991
<code>\define@choicekey</code>	258, 323, 348, 358, 376, 394, 402, 409, 737	<code>\exp_not:V</code>	516, 747
<code>\define@cmdkey</code>	730–733	<code>\expandafter</code>	459, 781, 783, 786, 791, 805, 806, 810, 1152, 1153
<code>\define@key</code>	713	<code>\ExplSyntaxOff</code>	2107
<code>\define@mathalphabet</code>	423	<code>\ExplSyntaxOn</code>	6, 1679
<code>\define@mathgroup</code>	423		
<code>\Digamma</code>	2093	F	
<code>\digamma</code>	2093	<code>\F</code>	39
<code>\dimexpr</code>	437, 541, 1880	<code>\f@size</code>	507
		<code>\fi</code> . 321, 344, 356, 363, 381, 399, 407, 416, 447, 448, 490–494, 550, 693, 740, 788, 789, 792, 798, 800, 801, 813, 818, 823, 828–832, 2077, 2105	
		<code>\fi:</code>	1138
		<code>\fint</code>	699
		<code>\font</code>	540

\fontdimen	437, 541, 1879, 1885	\g_um_bfupgreek_bool	25, 142, 266, 280,
\fontname	1074		294, 308, 327, 333, 339, 1421, 1494
G			
\g	67	\g_um_bfupLatin_bool	22, 139, 267, 281, 295, 309,
\g@addto@macro	795, 797		328, 334, 340, 1374, 1382, 1447, 1455
\g_um_amsmath_bool	2054, 2056, 2058, 2060, 2085	\g_um_bfuplatin_bool	23, 140, 268, 282, 296, 310,
\g_um_bfit_Greek_usv	141		329, 335, 341, 1389, 1397, 1462, 1470
\g_um_bfit_greek_usv	142	\g_um_fam_int	12, 560, 561
\g_um_bfit_Latin_usv	139	\g_um_it_h_usv	1035
\g_um_bfit_latin_usv	140	\g_um_it_Nabla_usv	371, 594, 598
\g_um_bfit_Nabla_usv	372, 603, 611	\g_um_it_partial_usv	389, 596, 599
\g_um_bfit_partial_usv ..	390, 607, 613	\g_um_literal_bool ..	16, 273, 287,
\g_um_bfliteral_bool	21, 318, 325, 331, 337,		301, 315, 317, 592, 1164, 1174,
	343, 601, 1378, 1393, 1405, 1417,		1185, 1195, 1209, 1219, 1231, 1241
	1430, 1451, 1466, 1478, 1490, 1507	\g_um_literal_colon_bool	401, 404, 406, 589, 2048
\g_um_bfNabla_up_or_it_usv	368, 372, 611	\g_um_math_alphabet_name_Greek_tl	1125
\g_um_bfpartial_up_or_it_usv	386, 390, 613	\g_um_math_alphabet_name_greek_tl	1124
\g_um_bfsfit_Greek_usv	137	\g_um_math_alphabet_name_Latin_tl	1123
\g_um_bfsfit_greek_usv	138		1122
\g_um_bfsfit_Latin_usv	135	\g_um_math_alphabet_name_num_tl	1126
\g_um_bfsfit_latin_usv	136	\g_um_mathalph_seq	734, 763, 997
\g_um_bfsfit_Nabla_usv ..	373, 605, 612	\g_um_mathbb_alph_clist ...	1016, 1049
\g_um_bfsfit_partial_usv	391, 609, 614	\g_um_mathbbbit_alph_clist	1017
\g_um_bfsfNabla_up_or_it_usv	369, 373, 612	\g_um_mathbbf_alph_clist	1022
\g_um_bfsfpartial_up_or_it_usv ..	387, 391, 614	\g_um_mathbbfrak_alph_clist	1015, 1058
\g_um_bfsfup_Greek_usv	137	\g_um_mathbfit_alph_clist .	1024, 1056
\g_um_bfsfup_greek_usv	138	\g_um_mathbfscr_alph_clist	1014, 1057
\g_um_bfsfup_Latin_usv	135	\g_um_mathbfsf_alph_clist .	1025, 1059
\g_um_bfsfup_latin_usv	136	\g_um_mathbfsfit_alph_clist	1027, 1061
\g_um_bfsfup_Nabla_usv ..	369, 604, 612	\g_um_mathbfsfup_alph_clist	1026, 1060
\g_um_bfsfup_partial_usv	387, 608, 614	\g_um_mathbfup_alph_clist .	1023, 1055
\g_um_bfup_Greek_usv	141	\g_um_mathfrak_alph_clist .	1013, 1051
\g_um_bfup_greek_usv	142	\g_um_mathit_alph_clist ...	1011, 1048
\g_um_bfup_Latin_usv	139	\g_um_mathit_Greek_usv	1038
\g_um_bfup_latin_usv	140	\g_um_mathit_greek_usv	1037
\g_um_bfup_Nabla_usv	368, 602, 611	\g_um_mathit_Latin_usv	1036
\g_um_bfup_partial_usv ..	386, 606, 613	\g_um_mathit_latin_usv	1035
\g_um_bfupGreek_bool	24, 141, 265, 279,	\g_um_mathscr_alph_clist ..	1012, 1050
	293, 307, 326, 332, 338, 1409, 1482	\g_um_mathsf_alph_clist	1019

<code>\g_um_mathsfup_alph_clist</code>	1021, 1053	<code>\geq</code>	2036
<code>\g_um_mathsfup_alph_clist</code>	1020, 1052	<code>\get@cdp</code>	427
<code>\g_um_mathtt_alph_clist</code>	1018, 1054	<code>\glb@currsz</code>	503
<code>\g_um_mathup_alph_clist</code>	1010, 1047	<code>\global</code>	456, 459, 471, 472, 477, 478, 481, 482, 489, 1672, 1682, 1902, 1935
<code>\g_um_mathup_greek_usv</code>	1032	<code>\group@elt</code>	426
<code>\g_um_mathup_greek_usv</code>	1031	<code>\group@list</code>	426
<code>\g_um_mathup_Latin_usv</code>	1030	<code>\group_align_safe_begin:</code>	1992
<code>\g_um_mathup_Latin_usv</code>	1029	<code>\group_align_safe_end:</code>	1991
<code>\g_um_mathup_num_usv</code>	1033	<code>\group_begin:</code>	1695, 1856, 1891
<code>\g_um_Nabla_up_or_it_usv</code>	367, 371, 598	<code>\group_end:</code>	1875, 1975
<code>\g_um_partial_up_or_it_usv</code>	385, 389, 599	H	
<code>\g_um_primekern_muskip</code>	1701, 1702, 1707	<code>\H</code>	40
<code>\g_um_sfliteral_bool</code>	319, 347, 355, 1317, 1329, 1341, 1353, 1536, 1548, 1560, 1572, 1586, 1599, 1611, 1623, 1635, 1649	<code>\hat</code>	2090
<code>\g_um_slash_delimiter_usv</code>	411, 413, 415, 642–644	<code>\hbox</code>	1878
<code>\g_um_subs_prop</code>	1889, 1933	<code>\ht</code>	1881
<code>\g_um_supers_prop</code>	1888, 1900	I	
<code>\g_um_texp_greek_bool</code>	28, 272, 286, 300, 314, 320, 396, 398, 985, 988, 991, 994	<code>\I</code>	41
<code>\g_um_up_Nabla_usv</code>	367, 593, 598	<code>\i</code>	69
<code>\g_um_up_partial_usv</code>	385, 595, 599	<code>\if@tempwa</code>	793
<code>\g_um_upGreek_bool</code>	19, 137, 261, 275, 289, 303, 1188, 1234	<code>\ifbeamer@suppressreplacements</code>	2099
<code>\g_um_upgreek_bool</code>	20, 138, 262, 276, 290, 304, 1198, 1244	<code>\ifcase</code>	260, 324, 350, 359, 377, 395, 403, 410, 738
<code>\g_um_upLatin_bool</code>	17, 135, 263, 277, 291, 305, 1167, 1212	<code>\ifdim</code>	541
<code>\g_um_uplatin_bool</code>	18, 136, 264, 278, 292, 306, 1177, 1223	<code>\ifnum</code>	691, 811, 816, 821, 825, 826, 2075
<code>\g_um_upNabla_bool</code>	26, 269, 283, 297, 311, 360, 362, 366, 1434, 1511, 1590, 1653	<code>\ifx</code>	440, 443, 453, 466, 475, 480, 486, 779, 782, 783, 786, 794, 810, 815, 820
<code>\g_um_uppartial_bool</code>	27, 270, 284, 298, 312, 378, 380, 384, 1437, 1514, 1593, 1656	<code>\ignorespaces</code>	1690
<code>\g_um_upsans_bool</code>	271, 285, 299, 313, 346, 351, 353, 1321, 1333, 1345, 1357, 1540, 1552, 1564, 1576, 1603, 1615, 1627, 1639	<code>\iiiint</code>	697
<code>\gamma</code>	1970	<code>\iiint</code>	697
<code>\gdef</code>	1676	<code>\iint</code>	697
<code>\ge</code>	2036	<code>\init@restore@version</code>	425
		<code>\int</code>	697
		<code>\int_incr:N</code>	560
		<code>\int_new:N</code>	12
		<code>\int_use:N</code>	561
		<code>\intBar</code>	699
		<code>\intbar</code>	699
		<code>\intcap</code>	701
		<code>\intclockwise</code>	698
		<code>\intcup</code>	701
		<code>\intlarkh</code>	700
		<code>\intx</code>	700
		J	
		<code>\j</code>	70

K	
<code>\kern</code>	1879, 1885
L	
<code>\l</code>	42
<code>\l_peek_false_tl</code>	1991, 2004, 2030
<code>\l_peek_token</code> ...	2000–2002, 2021, 2025
<code>\l_peek_true_aux_tl</code>	1989
<code>\l_peek_true_tl</code>	1990, 2028
<code>\l_um_char_range_seq</code>	505, 736, 743, 749, 778
<code>\l_um_font</code>	441, 444, 540, 541, 1074, 1138, 1879, 1881, 1882, 1885
<code>\l_um_fontspec_feature_bool</code>	13, 537, 539, 714
<code>\l_um_inc_num</code>	911, 913, 914
<code>\l_um_init_bool</code> ..	15, 504, 551, 739, 742
<code>\l_um_input_num</code>	903, 905, 909, 913
<code>\l_um_mathalph_seq</code>	735, 744, 747, 1043, 1082
<code>\l_um_missing_alph_seq</code>	1040, 1042, 1071, 1076, 1113
<code>\l_um_mversion_tf</code>	508, 509
<code>\l_um_nolimits_tl</code> ...	463, <u>696</u> , 704, 707
<code>\l_um_ot_math_bool</code>	14, 542, 544
<code>\l_um_primecount_num</code>	1703, 1740, 1744, 1748, 1752, 1756, 1767, 1771, 1775, 1779, 1783, 1787, 1790, 1802, 1806, 1810, 1814, 1825, 1829, 1833, 1837, 1840
<code>\l_um_radicals_tl</code>	467, 709
<code>\l_um_remap_alphabet_tl</code>	1085–1087, 1089, 1091, 1096
<code>\l_um_script_features_tl</code> ...	510, 527
<code>\l_um_script_font_tl</code>	512, 526
<code>\l_um_ss_chain_tl</code> 1905, 1938, 1982, 2027	
<code>\l_um_sscript_features_tl</code> ..	511, 531
<code>\l_um_sscript_font_tl</code>	513, 530
<code>\l_um_tmpa_tl</code>	747, 754, 757, 758, 760, 761, 763, 770, 774, 1083, 1087, 1094, 1096, 1101, 1102, 1111, 1115, 1907, 1940, 2020, 2024
<code>\l_um_tmpb_tl</code>	747, 755, 775, 1084, 1092, 1094, 1096, 1102, 1104, 1105, 1110, 2026, 2027
<code>\l_um_tmpc_tl</code>	747, 756, 771
<code>\lccode</code>	1696
<code>\le</code>	2035
<code>\left</code>	<u>710</u>
<code>\left@primitive</code>	710, 711
<code>\leq</code>	2035
<code>\let</code>	451, 503, 506, 710, 1672, 2053
<code>\lowercase</code>	1664, 1677
<code>\lowint</code>	701
M	
<code>\M</code>	43
<code>\m@th</code>	1878
<code>\mathaccent</code>	486
<code>\mathalpha</code>	685, 843, 846
<code>\mathbb</code>	1000, 1049, 1255, 1258–1265, 1268, 1271–1275
<code>\mathbbbit</code>	1000, 1278–1282
<code>\mathbf</code>	1003, 1380, 1384, 1395, 1399, 1407, 1411, 1419, 1423, 1431, 1432, 1435, 1438, 1443, 1453, 1457, 1468, 1472, 1480, 1484, 1492, 1496, 1505, 1506, 1508, 1509, 1512, 1515
<code>\mathbffrac</code>	1004, 1058, 1520, 1523
<code>\mathbfbit</code>	1003, 1056, 1377, 1392, 1404, 1416, 1428, 1429
<code>\mathbfscr</code>	1004, 1057, 1526, 1529
<code>\mathbfssf</code>	1005, 1059, 1532, 1538, 1542, 1550, 1554, 1562, 1566, 1574, 1578, 1587, 1588, 1591, 1594, 1601, 1605, 1613, 1617, 1625, 1629, 1637, 1641, 1650, 1651, 1654, 1657
<code>\mathbfsfrit</code>	1005, 1061, 1608, 1620, 1632, 1644, 1647, 1648
<code>\mathbfsfup</code>	1005, 1060, 1533, 1545, 1557, 1569, 1581, 1584, 1585
<code>\mathbfup</code>	1003, 1055, 1444, 1450, 1465, 1477, 1489, 1501–1504
<code>\mathbin</code>	587, 588
<code>\mathcal</code>	<u>2051</u>
<code>\mathchar@type</code> 432, 460, 470, 472, 476, 478, 481, 483, 484, 487, 489, 497, 500	
<code>\mathchardef</code>	2062, 2063, 2076
<code>\mathclose</code>	475, 484
<code>\mathcode</code>	456, 2062, 2063, 2072–2076, 2078–2080
<code>\mathellipsis</code>	2083

<code>\prg_stepwise_variable:nnnNn</code>	911	<code>\seq_if_empty:NTF</code>	1043
<code>\prime</code>	1849	<code>\seq_if_in:NnTF</code>	8
<code>\process@table</code>	425	<code>\seq_if_in:NVTF</code>	763
<code>\ProcessOptionsX</code>	419	<code>\seq_map_inline:Nn</code>	1076, 1082
<code>\prop_get:cxN</code>	2023	<code>\seq_map_variable:NNn</code>	778
<code>\prop_get:NnN</code>	10	<code>\seq_new:N</code>	734–736, 1040
<code>\prop_gput:Nnn</code>	9	<code>\seq_put_right:Nn</code>	749, 1007
<code>\prop_gput:Nxn</code>	1900, 1933	<code>\seq_put_right:Nx</code>	747, 1113
<code>\prop_if_in:cxTF</code>	2019	<code>\set@@mathdelimiter</code>	431
<code>\prop_if_in:NnTF</code>	11	<code>\set@mathaccent</code>	429
<code>\prop_new:N</code>	1888, 1889	<code>\set@mathchar</code>	429
<code>\protect</code>	720	<code>\set@mathdelimiter</code>	431
<code>\ProvidesPackage</code>	1	<code>\set@mathsymbol</code>	430
Q			
<code>\Q</code>	46	<code>\setbox</code>	1878
<code>\q_nil</code>	758, 761, 769, 773	<code>\setkeys</code>	256, 514, 516
<code>\qprime</code>	1852	<code>\SetMathAlphabet</code>	428
R			
<code>\R</code>	47	<code>\SetMathAlphabet@</code>	428
<code>\r@@t</code>	1877	<code>\setmathfont</code>	502, 720
<code>\raise</code>	1880	<code>\SetSymbolFont</code>	427
<code>\relax</code>	260, 324, 350, 437, 453, 456, 460, 466, 468, 470–472, 475–478, 481–484, 486, 487, 489, 497, 500, 503, 540, 541, 691, 738, 783, 786, 810, 811, 816, 821, 825, 826, 843, 905, 913, 914, 1883, 2062, 2063, 2066, 2067, 2076, 2080	<code>\setSymbolFont@</code>	427
<code>\removenolimits</code>	706	<code>\sf@size</code>	525, 529
<code>\RequirePackage</code>	3–5	<code>\smbkcircle</code>	2041
<code>\restore@mathversion</code>	425	<code>\sp</code>	1906
<code>\rho</code>	1971	<code>\space</code>	1077, 1115, 1153
<code>\rightarrow</code>	2033	<code>\sqint</code>	700
<code>\rootbox</code>	1884	<code>\sqrt</code>	709, 1876
<code>\rppolint</code>	699	<code>\sqrtsign</code>	1876, 1878
S			
<code>\sb</code>	1939	<code>\std@equal</code>	2063, 2067
<code>\scan_stop:</code>	635, 639, 1138, 1902, 1935	<code>\std@minus</code>	2062, 2066, 2076
<code>\scantokens</code>	1903, 1936	<code>\sterling</code>	2043
<code>\scpolint</code>	700	<code>\string</code>	805, 806
<code>\scriptscriptstyle</code>	443	<code>\strip@pt</code>	437
<code>\scriptstyle</code>	440	<code>\sumint</code>	698
<code>\seq_clear:N</code>	505, 743, 744, 997, 1042	T	
<code>\seq_if_empty:NF</code>	1071	<code>\tf@size</code>	524, 525
		<code>\thinmuskip</code>	1702
		<code>\tilde</code>	2091
		<code>\tl_if_empty:NT</code>	1092
		<code>\tl_if_empty:NTF</code>	1086
		<code>\tl_if_in:NnT</code>	463, 757, 760
		<code>\tl_if_in:NnTF</code>	467
		<code>\tl_map_inline:nn</code>	420, 998
		<code>\tl_new:Nn</code>	696, 709, 1010–1027, 1029–1033, 1035–1038
		<code>\tl_put_right:cx</code>	684
		<code>\tl_put_right:Nn</code>	7, 704

\tl_put_right:NV	2027	\um_config_mathbffrak_Latin:n ..	1522
\tl_remove_all_in:Nn	707	\um_config_mathbffit_Greek:n ...	1403
\tl_remove_in:Nn	434, 2061	\um_config_mathbffit_greek:n ...	1415
\tl_rescan:nn	1665, 1678	\um_config_mathbffit_Latin:n ...	1373
\tl_set:cn	30	\um_config_mathbffit_Latin:n ...	1388
\tl_set:cx	982	\um_config_mathbffit_misc:n	1427
\tl_set:Nn	367–369, 371–373, 385–387, 389–391, 411, 413, 415, 508, 510–513, 552, 754–756, 770, 771, 774, 775, 984, 987, 990, 993, 1122–1126, 1905, 1907, 1938, 1940	\um_config_mathbfscr_Latin:n ...	1525
\tl_set:No	1083–1085	\um_config_mathbfscr_Latin:n ...	1528
\tl_set:Nv	1094	\um_config_mathbfssfit_Greek:n ..	1622
\tl_set:Nx	561, 1087, 1089, 1091, 1101, 1102, 1989, 1991	\um_config_mathbfssfit_greek:n ..	1634
\tl_set_eq:NN	1990	\um_config_mathbfssfit_Latin:n ..	1598
\tl_use:c	1115	\um_config_mathbfssfit_Latin:n ..	1610
\to	2033	\um_config_mathbfssfit_misc:n ...	1646
\token_if_eq_catcode_p:NN ..	2000, 2001	\um_config_mathbfssup_Greek:n ..	1559
\token_if_eq_meaning_p:NN	2002	\um_config_mathbfssup_greek:n ..	1571
\token_to_str:N	1087, 1089	\um_config_mathbfssup_Latin:n ..	1535
\triangle	2038	\um_config_mathbfssup_Latin:n ..	1547
\trprime	1851	\um_config_mathbfssup_misc:n ...	1583
\typeout	1072, 1077	\um_config_mathbfssup_num:n ...	1531
U		\um_config_mathbfup_Greek:n ...	1476
\um@backslash	782, 805	\um_config_mathbfup_greek:n ...	1488
\um@char@num@range	506, 690, 794, 795, 797	\um_config_mathbfup_Latin:n ...	1446
\um@firstchar	781, 806	\um_config_mathbfup_Latin:n ...	1461
\um@firsttof	804–806	\um_config_mathbfup_misc:n	1500
\um@parse@range	791, 807	\um_config_mathbfup_num:n	1442
\um@parse@term	582, 618, 777, 846	\um_config_mathfrak_Latin:n ...	1301
\um@radicals	709	\um_config_mathfrak_Latin:n ...	1309
\um@scaled@apply	439, 1879, 1885	\um_config_mathit_Greek:n	1230
\um@scanactivedef	457, 1661	\um_config_mathit_greek:n	1240
\um@scancharlet	1661, 1690	\um_config_mathit_Latin:n	1208
\um@zf@feature	712, 724, 727	\um_config_mathit_Latin:n	1218
\um_backprime_double_mchar	633, 1730	\um_config_mathit_misc:n	1250
\um_backprime_single_mchar	632, 1803, 1807, 1811	\um_config_mathscr_Latin:n	1284
\um_backprime_triple_mchar	634, 1733	\um_config_mathscr_Latin:n	1295
\um_config_mathbb_Latin:n	1257	\um_config_mathsf_Latin:n ...	1340
\um_config_mathbb_Latin:n	1254	\um_config_mathsf_Latin:n ...	1352
\um_config_mathbb_misc:n	1270	\um_config_mathsfup_Latin:n ...	1316
\um_config_mathbb_num:n	1267	\um_config_mathsfup_Latin:n ...	1328
\um_config_mathbbit_misc:n	1277	\um_config_mathsfup_num:n	1312
\um_config_mathbffrak_Latin:n ..	1519	\um_config_mathhtt_Latin:n	1367
		\um_config_mathhtt_Latin:n	1370
		\um_config_mathhtt_num:n	1364
		\um_config_mathup_Greek:n	1184
		\um_config_mathup_greek:n	1194
		\um_config_mathup_Latin:n	1163
		\um_config_mathup_Latin:n	1173

\um_config_mathup_misc:n	1204	\um_map_chars_xxvi:cc	861, 866
\um_config_mathup_num:n	1159	\um_map_chars_xxvi:nn	850, 899
\um_fontdimen_to_percent:nn	436, 441, 444, 1881, 1882	\um_map_chars_xxvi:nn_	834
\um_glyph_if_exist:cT	1104	\um_map_single:nnn	892, 1179, 1221, 1225
\um_glyph_if_exist:cTF	1110	\um_mathmap:Nnn	555, 563, 904, 912, 1081
\um_glyph_if_exist:n	1137	\um_mathmap_noparse:Nnn	555, 682, 692, 1081
\um_glyph_if_exist:nF	1143	\um_mathmap_parse:Nnn	563, 689
\um_glyph_if_exist:nT	1142	\um_maybe_init_alphabet:n	557, 565, 1044–1046, 1065, 1093, 1105
\um_glyph_if_exist:nTF	1137, 1714, 1717, 1720, 1730, 1733	\um_nbackprimes_select:nn	1726, 1840
\um_glyph_if_exist_p:n	1140	\um_nprimes:Nn	1704, 1714, 1717, 1720, 1723, 1730, 1733, 1736
\um_if_mathalph_decl:n	753	\um_nprimes_select:nn	1710, 1790
\um_if_mathalph_decl:nTF	746	\um_peek_execute_branches_ss:	1993, 1998
\um_init_alphabet:n	557, 1093, 1132	\um_peek_execute_branches_ss_aux:	2005, 2018
\um_make_mathactive:nNN	628–634, 637	\um_prepare_alph:n	1134, 1144
\um_map_char:cc	872–877, 883, 894	\um_prime_double_mchar	629, 1714
\um_map_char:nn	889, 897	\um_prime_quad_mchar	631, 1720
\um_map_char:nn_	834	\um_prime_single_mchar	628, 1741, 1745, 1749, 1753
\um_map_char_internal:nn	558, 566, 837	\um_prime_triple_mchar	630, 1717
\um_map_char_noparse:nn	558, 841, 847	\um_process_symbol_noparse:nnnn	554, 578
\um_map_char_parse:nn	566, 845	\um_process_symbol_parse:nnnn	562, 578
\um_map_chars_Greek:nn	880, 1186, 1189, 1232, 1235, 1406, 1410, 1479, 1483, 1561, 1565, 1624, 1628	\um_remap_symbol:nnn	556, 564, 587, 588, 590, 593–596, 598, 599, 602–609, 611–614
\um_map_chars_greek:nn	869, 1196, 1199, 1242, 1245, 1418, 1422, 1491, 1495, 1573, 1577, 1636, 1640	\um_remap_symbol_noparse:nnn	556, 586
\um_map_chars_Latin:nn	859, 1165, 1168, 1210, 1213, 1318, 1322, 1342, 1346, 1375, 1379, 1383, 1448, 1452, 1456, 1537, 1541, 1600, 1604	\um_remap_symbol_parse:nnn	564, 586, 617
\um_map_chars_latin:nn	864, 1175, 1178, 1220, 1224, 1330, 1334, 1354, 1358, 1390, 1394, 1398, 1463, 1467, 1471, 1549, 1553, 1612, 1616	\um_remap_symbols:	573, 586
\um_map_chars_numbers:nn	886, 1160	\um_resolve_greek:	972
\um_map_chars_range:nnn	834, 851, 854, 857, 890	\um_scan_backdprime:	1805, 1824, 1854, 1873
\um_map_chars_x:cc	887	\um_scan_backprime:	1801, 1818, 1853, 1871, 1872
\um_map_chars_x:nn	856, 900	\um_scan_backtrprime:	1809, 1832, 1855, 1874
\um_map_chars_xxiii:cc	871, 882	\um_scan_dprime:	1743, 1766, 1850, 1868
\um_map_chars_xxiii:nn	853, 898	\um_scan_prime:	1739, 1760, 1849, 1866, 1867
\um_map_chars_xxiii:nn_	834	\um_scan_qprime:	1751, 1782, 1852, 1870
		\um_scan_sscript:	1908, 1941, 1978, 1980

\um_scan_sscript:TF 1979, 1988	1508, 1509, 1512, 1515, 1584,	
\um_scan_trprime:	1747, 1774, 1851, 1869	1585, 1587, 1588, 1591, 1594,	
\um_scanbackprime_collect:N	1647, 1648, 1650, 1651, 1654, 1657	
..	1803, 1807, 1811, 1813, 1816,	\um_set_mathalphabet_x:Ncc 937
1819, 1822, 1826, 1830, 1834, 1838		\um_set_mathalphabet_x:Nnn	. 918, 971
\um_scanprime_collect:N	\um_set_mathalphabet_xxiii:Ncc	..
.....	1741, 1745, 1749,	953, 959
1753, 1755, 1758, 1761, 1764,		\um_set_mathalphabet_xxiii:Nnn	..
1768, 1772, 1776, 1780, 1784, 1788		924, 969
\um_set_delcode:n	... 645, 648–674, <u>676</u>	\um_set_mathalphabet_xxvi:Ncc	942, 947
\um_set_delcode:nn	642–644, 646, 647, <u>676</u>	\um_set_mathalphabet_xxvi:Nnn	921, 970
\um_set_mathalph_range:Nnn <u>908</u>	\um_set_mathchar:Nnnn 496, 638
\um_set_mathalph_range:nNnn	\um_set_mathcode:nnnn	<u>496</u> , 624, 685, 842
.....	908, 919, 922, 925	\um_set_mathsymbol:nNnn <u>452</u> , 579
\um_set_mathalphabet_char:Ncc	...	\um_setup_active_subscript:nn	...
.....	931, 948, 954, 960–965	1932, 1946–1973
\um_set_mathalphabet_char:Nnn	<u>901</u> , 968	\um_setup_active_superscript:nn	.
\um_set_mathalphabet_Greek:Nnn	1899, 1913–1929
.....	951, 1192, 1238, 1404,	\um_setup_alphabets: 576, <u>1009</u>
1407, 1411, 1477, 1480, 1484,		\um_setup_delcodes: 575, <u>641</u>
1562, 1566, 1569, 1625, 1629, 1632		\um_setup_math_alphabet:Nn <u>1100</u>
\um_set_mathalphabet_greek:Nnn	..	\um_setup_math_alphabet:Nnn	1100, 1120
.....	957, 1202, 1248, 1416,	\um_setup_math_alphabet:NVn	1047–1061
1419, 1423, 1489, 1492, 1496,		\um_setup_math_alphabet:VVV	... 1096
1574, 1578, 1581, 1637, 1641, 1644		\um_setup_math_mapping:n
\um_set_mathalphabet_Latin:Nnn	1062–1064, 1066–1070, 1127
940, 1171, 1216, 1258, 1285, 1302,		\um_setup_mathactives: 574, <u>627</u>
1319, 1323, 1326, 1343, 1347,		\um_setup_mathup: 2032
1350, 1368, 1377, 1380, 1384,		\um_setup_nabla: 365, 571
1450, 1453, 1457, 1520, 1526,		\um_setup_partial: 383, 572
1538, 1542, 1545, 1601, 1605, 1608		\um_split_arrow:w 758, 769
\um_set_mathalphabet_latin:Nnn	..	\um_split_slash:w 761, 773
945, 1182, 1228, 1255, 1296, 1310,		\um_sub_or_super:n	.. 1906, 1939, 1982
1331, 1335, 1338, 1355, 1359,		\um_symfont_tl 552, 561,
1362, 1371, 1392, 1395, 1399,		568, 579, 624, 638, 677, 680, 685, 843	
1465, 1468, 1472, 1523, 1529,		\um_tmp: 515, 518, 519, 538
1550, 1554, 1557, 1613, 1617, 1620		\um_to_usv:nn 30, 32, 861
\um_set_mathalphabet_numbers:Nnn	\unicodecdots 2084
.....	935, 1161, 1268, 1313,	\unicodeellipsis 2083
1314, 1365, 1443, 1444, 1532, 1533		\UnicodeMathSymbol 554, 562, 1689
\um_set_mathalphabet_pos:Nnnn	...	\unimathsetup <u>255</u>
.....	928, 1205, 1206, 1251,	\unless 779
1252, 1259–1265, 1271–1275,		\updefault 569
1278–1282, 1286–1293, 1297–1299,		\upDigamma 2094
1303–1307, 1428, 1429, 1431,		\updigamma 2093
1432, 1435, 1438, 1501–1506,		\upint 701

<code>\use:c</code>	1111, 1129, 1147, 1155	X	
<code>\use_i:nnn</code>	1083	<code>\XeTeXdelcode</code> . . .	471, 477, 482, 677, 680
<code>\use_ii:nnn</code>	1084	<code>\XeTeXdelimiter</code>	470, 476, 483, 484
<code>\use_iii:nnn</code>	1085	<code>\XeTeXmathaccent</code>	487
<code>\use_none:n</code>	565	<code>\XeTeXmathchardef</code>	459, 500
<code>\use_none:nnnn</code>	1102	<code>\XeTeXmathcharnum</code>	2066, 2067
<code>\use_none:nnnnn</code>	1091	<code>\XeTeXmathcode</code> . .	472, 478, 481, 489, 497
<code>\usv_set:nnn</code>	29,	<code>\XeTeXmathcodenum</code>	
35–48, 50–63, 65–71, 73–79, 81–254		. . .	635, 639, 1902, 1935, 2066, 2067
V		<code>\XeTeXradical</code>	468
<code>\varepsilon</code>	990	<code>\XKV@rm</code>	516, 534
<code>\varointclockwise</code>	698	Y	
<code>\varphi</code>	993	<code>\yen</code>	2042
<code>\vec</code>	2034	Z	
<code>\version@elt</code>	424	<code>\Z</code>	48, 1030
<code>\version@list</code>	424	<code>\z</code>	1029
<code>\vysmwhtcircle</code>	2040	<code>\z@</code>	1878, 1881, 1882, 1886
W		<code>\zf@family</code>	569
<code>\widehat</code>	2090	<code>\zf@fontspec</code>	520
<code>\widetilde</code>	2091	<code>\zf@update@ff</code>	725, 728
<code>\wlog</code>	1133		