

Experimental unicode mathematical typesetting: The unicode-math package

Will Robertson

2009/10/21 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	15			
6	Things we need	15			
6.1	Options	19			
6.2	Overcoming <code>\@on-</code> preamble	23			
6.3	Other things	24			
7	Fundamentals	25			
7.1	Enlarging the number of maths families	25			
			7.2	<code>\DeclareMathSymbol</code> for unicode ranges	25
			7.3	The main <code>\setmathfont</code> macro	27
			7.4	(Big) operators	33
			7.5	Radicals	36
			7.6	Delimiters	37
			7.7	Maths accents	39
8	Font features	40			
8.1	OpenType maths font features	40			
8.2	Script and scriptscript font options	41			
8.3	Range processing	41			
8.4	Resolving Greek symbol name control sequences	47			
9	Maths alphabets mapping definitions	48			
9.1	Alphabets	52			
10	Definitions of the math symbols	64			
11	Epilogue	65			
12	stix table data extraction	74			
A	Documenting maths support in the NFSS	75			
B	X_YTeX math font dimensions	76			

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 \LuaTeX .

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 \XeTeX ; Barbara Beeton for her prodigious effort compiling the definitive list of unicode math glyphs and their \LaTeX 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 \LaTeX 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=\mathbfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (with `latin`, `Latin`, `num` as well for Latin lower-/upper-case and numbers).
- `[range=\mathsf->\mathbfsfit]` to map to different output alphabet(s) (which is rather useless right now but will become less useless in the future).

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

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

and because the math plane fractur glyphs will be missing, unicode-math will know to use the ASCII ones instead. If necessary (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>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\alpha, \beta, \Gamma, \Xi)$
<code>bold-style=TeX</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\alpha, \beta, \Gamma, \Xi)$
<code>bold-style=upright</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\alpha, \beta, \Gamma, \Xi)$

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 $\mathbb{0}-\mathbb{9}$, summation symbol $\mathbb{\Sigma}$, and four Greek letters only: $\{\mathbb{V}, \mathbb{W}, \mathbb{F}, \mathbb{M}\}$.

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: $\mathbb{D}, \mathbb{d}, \mathbb{E}, \mathbb{e}, \mathbb{J}$. 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`.

U+3B5: GREEK SMALL LETTER EPSILON

U+3F5: GREEK LUNATE EPSILON SYMBOL

U+3C6: GREEK SMALL LETTER PHI

U+3D5: GREEK SMALL LETTER SCRIPT PHI

5.5.4 Primes

Primes (x') may be input in several ways. You may use any combination of ascii straight quote (‘), unicode prime (’), 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 `\primedouble`, `\primetripel`, and `\primequadruple`.

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

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.

like this:

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

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: RATIO`. Typing a literal `U+2236: RATIO` 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`;
- `\fracslash`;
- `\slash`; and,
- `\backslash` (the only reverse slash).

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

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

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 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+2511: LATIN SMALL LETTER ALPHA

U+25B: LATIN SMALL LETTER EPSILON

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

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 [2009/10/21 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
```

6.0.9 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

```
29 \def\g_um_up_num_usv{48}
30 \def\g_um_up_Latin_usv{65}
31 \def\g_um_up_latin_usv{97}
32 \def\g_um_up_Greek_usv{"391}
33 \def\g_um_up_greek_usv{"3B1}
34 \def\g_um_it_Latin_usv{"1D434}
35 \def\g_um_it_latin_usv{"1D44E}
36 \def\g_um_it_Greek_usv{"1D6E2}
37 \def\g_um_it_greek_usv{"1D6FC}
38 \def\g_um_bb_num_usv{"1D7D8}
39 \def\g_um_bb_Latin_usv{"1D538}
40 \def\g_um_bb_latin_usv{"1D552}
41 \def\g_um_scr_Latin_usv{"1D49C}
42 \def\g_um_scr_latin_usv{"1D4B6}
43 \def\g_um_frac_Latin_usv{"1D504}
44 \def\g_um_frac_latin_usv{"1D51E}
45 \def\g_um_sf_num_usv{"1D7E2}
46 \def\g_um_sfup_num_usv{"1D7E2}
47 \def\g_um_sfit_num_usv{"1D7E2}
48 \def\g_um_sfup_Latin_usv{"1D5A0}
49 \def\g_um_sf_Latin_usv {"1D5A0}
50 \def\g_um_sfup_latin_usv{"1D5BA}
```

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


```

51 \def\g_um_sf_latin_usv{"1D5BA}
52 \def\g_um_sfit_Latin_usv{"1D608}
53 \def\g_um_sfit_latin_usv{"1D622}
54 \def\g_um_tt_num_usv{"1D7F6}
55 \def\g_um_tt_Latin_usv{"1D670}
56 \def\g_um_tt_latin_usv{"1D68A}

```

Bold:

```

57 \def\g_um_bf_num_usv {"1D7CE}
58 \def\g_um_bfup_num_usv{"1D7CE}
59 \def\g_um_bfit_num_usv{"1D7CE}
60 \def\g_um_bfup_Latin_usv{"1D400}
61 \def\g_um_bfup_latin_usv{"1D41A}
62 \def\g_um_bfup_Greek_usv{"1D6A8}
63 \def\g_um_bfup_greek_usv{"1D6C2}
64 \def\g_um_bfit_Latin_usv{"1D468}
65 \def\g_um_bfit_latin_usv{"1D482}
66 \def\g_um_bfit_Greek_usv{"1D71C}
67 \def\g_um_bfit_greek_usv{"1D736}
68 \def\g_um_bffrak_Latin_usv{"1D56C}
69 \def\g_um_bffrak_latin_usv{"1D586}
70 \def\g_um_bfscr_Latin_usv{"1D4D0}
71 \def\g_um_bfscr_latin_usv{"1D4EA}
72 \def\g_um_bfsf_num_usv {"1D7EC}
73 \def\g_um_bfsfup_num_usv{"1D7EC}
74 \def\g_um_bfsfit_num_usv{"1D7EC}
75 \def\g_um_bfsfup_Latin_usv{"1D5D4}
76 \def\g_um_bfsfup_latin_usv{"1D5EE}
77 \def\g_um_bfsfup_Greek_usv{"1D756}
78 \def\g_um_bfsfup_greek_usv{"1D770}
79 \def\g_um_bfsfit_Latin_usv{"1D63C}
80 \def\g_um_bfsfit_latin_usv{"1D656}
81 \def\g_um_bfsfit_Greek_usv{"1D790}
82 \def\g_um_bfsfit_greek_usv{"1D7AA}
83 \def\g_um_bfsf_Latin_usv { \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsf_Latin_usv }
84 \def\g_um_bfsf_latin_usv { \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsf_latin_usv }
85 \def\g_um_bfsf_Greek_usv { \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsf_Greek_usv }
86 \def\g_um_bfsf_greek_usv { \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsf_greek_usv }
87 \def\g_um_bf_Latin_usv { \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_Latin_usv }
88 \def\g_um_bf_latin_usv { \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_latin_usv }
89 \def\g_um_bf_Greek_usv { \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_Greek_usv }
90 \def\g_um_bf_greek_usv { \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_greek_usv }

```

Greek variants:

```

91 \def\g_um_up_varTheta_usv{"3F4}
92 \def\g_um_up_Digamma_usv{"3DC}
93 \def\g_um_up_varepsilon_usv{"3F5}
94 \def\g_um_up_vartheta_usv{"3D1}

```

```

95 \def\g_um_up_varkappa_usv{"3F0}
96 \def\g_um_up_varphi_usv{"3D5}
97 \def\g_um_up_varrho_usv{"3F1}
98 \def\g_um_up_varpi_usv{"3D6}
99 \def\g_um_up_digamma_usv{"3DD}

```

Bold:

```

100 \def\g_um_bfup_varTheta_usv{"1D6B9}
101 \def\g_um_bfup_Digamma_usv{"1D7CA}
102 \def\g_um_bfup_varepsilon_usv{"1D6DC}
103 \def\g_um_bfup_vartheta_usv{"1D6DD}
104 \def\g_um_bfup_varkappa_usv{"1D6DE}
105 \def\g_um_bfup_varphi_usv{"1D6DF}
106 \def\g_um_bfup_varrho_usv{"1D6E0}
107 \def\g_um_bfup_varpi_usv{"1D6E1}
108 \def\g_um_bfup_digamma_usv{"1D7CB}

```

Italic Greek variants:

```

109 \def\g_um_it_varTheta_usv{"1D6F3}
110 \def\g_um_it_varepsilon_usv{"1D716}
111 \def\g_um_it_vartheta_usv{"1D717}
112 \def\g_um_it_varkappa_usv{"1D718}
113 \def\g_um_it_varphi_usv{"1D719}
114 \def\g_um_it_varrho_usv{"1D71A}
115 \def\g_um_it_varpi_usv{"1D71B}

```

Bold italic:

```

116 \def\g_um_bfit_varTheta_usv{"1D72D}
117 \def\g_um_bfit_varepsilon_usv{"1D750}
118 \def\g_um_bfit_vartheta_usv{"1D751}
119 \def\g_um_bfit_varkappa_usv{"1D752}
120 \def\g_um_bfit_varphi_usv{"1D753}
121 \def\g_um_bfit_varrho_usv{"1D754}
122 \def\g_um_bfit_varpi_usv{"1D755}

```

Bold sans:

```

123 \def\g_um_bfsfup_varTheta_usv{"1D767}
124 \def\g_um_bfsfup_varepsilon_usv{"1D78A}
125 \def\g_um_bfsfup_vartheta_usv{"1D78B}
126 \def\g_um_bfsfup_varkappa_usv{"1D78C}
127 \def\g_um_bfsfup_varphi_usv{"1D78D}
128 \def\g_um_bfsfup_varrho_usv{"1D78E}
129 \def\g_um_bfsfup_varpi_usv{"1D78F}

```

Bold sans italic:

```

130 \def\g_um_bfsfit_varTheta_usv{"1D7A1}
131 \def\g_um_bfsfit_varepsilon_usv{"1D7C4}
132 \def\g_um_bfsfit_vartheta_usv{"1D7C5}
133 \def\g_um_bfsfit_varkappa_usv{"1D7C6}

```

```

134 \def\g_um_bfsfit_varphi_usv{"1D7C7}
135 \def\g_um_bfsfit_varrho_usv{"1D7C8}
136 \def\g_um_bfsfit_varpi_usv{"1D7C9}

```

Nabla:

```

137 \def\g_um_up_Nabla_usv{"2207}
138 \def\g_um_it_Nabla_usv{"1D6FB}
139 \def\g_um_bfup_Nabla_usv{"1D6C1}
140 \def\g_um_bfit_Nabla_usv{"1D735}
141 \def\g_um_bfsfup_Nabla_usv{"1D76F}
142 \def\g_um_bfsfit_Nabla_usv{"1D7A9}

```

Partial:

```

143 \def\g_um_up_partial_usv{"2202}
144 \def\g_um_it_partial_usv{"1D715}
145 \def\g_um_bfup_partial_usv{"1D6DB}
146 \def\g_um_bfit_partial_usv{"1D74F}
147 \def\g_um_bfsfup_partial_usv{"1D789}
148 \def\g_um_bfsfit_partial_usv{"1D7C3}

```

Latin ‘h’:

```

149 \def\g_um_up_h_usv {"0068}
150 \def\g_um_it_h_usv {"210E}
151 \def\g_um_bb_h_usv {"1D559}
152 \def\g_um_tt_h_usv {"1D691}
153 \def\g_um_scr_h_usv {"1D4BD}
154 \def\g_um_frac_h_usv{"1D525}
155 \def\g_um_bfup_h_usv{"1D421}
156 \def\g_um_bfit_h_usv{"1D489}
157 \def\g_um_sfup_h_usv{"1D5C1}
158 \def\g_um_sfit_h_usv{"1D629}
159 \def\g_um_bffrak_h_usv{"1D58D}
160 \def\g_um_bfscr_h_usv {"1D4F1}
161 \def\g_um_bfsfup_h_usv {"1D5F5}
162 \def\g_um_bfsfit_h_usv {"1D65D}

```

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.

```

163 \DeclareDocumentCommand \unimathsetup {m} {
164   \setkeys{unicode-math.sty}{#1}
165 }

```

math-style

```

166 \define@choicekey*{unicode-math.sty}
167   {math-style}[\@tempa\@tempb]{iso,tex,french,upright,literal}{
168   \ifcase\@tempb\relax
169     \bool_set_false:N \g_um_upGreek_bool
170     \bool_set_false:N \g_um_upgreek_bool
171     \bool_set_false:N \g_um_upLatin_bool
172     \bool_set_false:N \g_um_uplatin_bool
173     \bool_set_false:N \g_um_bfupGreek_bool
174     \bool_set_false:N \g_um_bfupgreek_bool
175     \bool_set_false:N \g_um_bfupLatin_bool
176     \bool_set_false:N \g_um_bfuplatin_bool
177     \bool_set_false:N \g_um_upNabla_bool
178     \bool_set_false:N \g_um_uppartial_bool
179     \bool_set_false:N \g_um_upsans_bool
180     \bool_set_false:N \g_um_texgreek_bool
181     \bool_set_false:N \g_um_literal_bool
182   \or
183     \bool_set_true:N \g_um_upGreek_bool
184     \bool_set_false:N \g_um_upgreek_bool
185     \bool_set_false:N \g_um_upLatin_bool
186     \bool_set_false:N \g_um_uplatin_bool
187     \bool_set_true:N \g_um_bfupGreek_bool
188     \bool_set_false:N \g_um_bfupgreek_bool
189     \bool_set_true:N \g_um_bfupLatin_bool
190     \bool_set_true:N \g_um_bfuplatin_bool
191     \bool_set_true:N \g_um_upNabla_bool
192     \bool_set_false:N \g_um_uppartial_bool
193     \bool_set_true:N \g_um_upsans_bool
194     \bool_set_false:N \g_um_texgreek_bool
195     \bool_set_false:N \g_um_literal_bool
196   \or
197     \bool_set_true:N \g_um_upGreek_bool
198     \bool_set_true:N \g_um_upgreek_bool
199     \bool_set_true:N \g_um_upLatin_bool
200     \bool_set_false:N \g_um_uplatin_bool
201     \bool_set_true:N \g_um_bfupGreek_bool
202     \bool_set_true:N \g_um_bfupgreek_bool
203     \bool_set_true:N \g_um_bfupLatin_bool
204     \bool_set_true:N \g_um_bfuplatin_bool
205     \bool_set_true:N \g_um_upNabla_bool
206     \bool_set_true:N \g_um_uppartial_bool
207     \bool_set_true:N \g_um_upsans_bool
208     \bool_set_false:N \g_um_texgreek_bool
209     \bool_set_false:N \g_um_literal_bool
210   \or

```

```

211 \bool_set_true:N \g_um_upGreek_bool
212 \bool_set_true:N \g_um_upgreek_bool
213 \bool_set_true:N \g_um_upLatin_bool
214 \bool_set_true:N \g_um_uplatin_bool
215 \bool_set_true:N \g_um_bfupGreek_bool
216 \bool_set_true:N \g_um_bfupgreek_bool
217 \bool_set_true:N \g_um_bfupLatin_bool
218 \bool_set_true:N \g_um_bfuplatin_bool
219 \bool_set_true:N \g_um_upNabla_bool
220 \bool_set_true:N \g_um_uppartial_bool
221 \bool_set_true:N \g_um_upsans_bool
222 \bool_set_false:N \g_um_texgreek_bool
223 \bool_set_false:N \g_um_literal_bool
224 \or
225 \bool_set_true:N \g_um_literal_bool
226 \bool_set_true:N \g_um_bfliteral_bool
227 \bool_set_true:N \g_um_sfliteral_bool
228 \bool_set_false:N \g_um_texgreek_bool
229 \fi
230 }

```

bold-style

```

231 \define@choicekey*{unicode-math.sty}{bold-style}[\@tempa\@tempb]{iso,tex,upright,literal}{
232 \ifcase\@tempb\relax
233 \bool_set_false:N \g_um_bfliteral_bool
234 \bool_set_false:N \g_um_bfupGreek_bool
235 \bool_set_false:N \g_um_bfupgreek_bool
236 \bool_set_false:N \g_um_bfupLatin_bool
237 \bool_set_false:N \g_um_bfuplatin_bool
238 \or
239 \bool_set_false:N \g_um_bfliteral_bool
240 \bool_set_true:N \g_um_bfupGreek_bool
241 \bool_set_false:N \g_um_bfupgreek_bool
242 \bool_set_true:N \g_um_bfupLatin_bool
243 \bool_set_true:N \g_um_bfuplatin_bool
244 \or
245 \bool_set_false:N \g_um_bfliteral_bool
246 \bool_set_true:N \g_um_bfupGreek_bool
247 \bool_set_true:N \g_um_bfupgreek_bool
248 \bool_set_true:N \g_um_bfupLatin_bool
249 \bool_set_true:N \g_um_bfuplatin_bool
250 \or
251 \bool_set_true:N \g_um_bfliteral_bool
252 \fi
253 }

```

sans-style

```
254 \bool_new:N \g_um_upsans_bool
255 \bool_new:N \g_um_sfliteral_bool
256 \define@choicekey*{unicode-math.sty}
257   {sans-style}[\@tempa\@tempb]{italic,upright,literal}{
258   \ifcase\@tempb\relax
259     \bool_set_false:N \g_um_upsans_bool
260   \or
261     \bool_set_true:N \g_um_upsans_bool
262   \or
263     \bool_set_true:N \g_um_sfliteral_bool
264   \fi
265 }
```

Symbol obliqueness

```
266 \define@choicekey*{unicode-math.sty}{nabla}[\@tempa\@tempb]{upright,italic}{
267   \ifcase\@tempb
268     \bool_set_true:N \g_um_upNabla_bool
269   \or
270     \bool_set_false:N \g_um_upNabla_bool
271   \fi
272 }
273 \cs_set:Nn \um_setup_nabla: {
274   \bool_if:NTF \g_um_upNabla_bool {
275     \tl_set:Nn \g_um_Nabla_up_or_it_usv { \g_um_up_Nabla_usv }
276     \tl_set:Nn \g_um_bfNabla_up_or_it_usv { \g_um_bfup_Nabla_usv }
277     \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfup_Nabla_usv }
278   }{
279     \tl_set:Nn \g_um_Nabla_up_or_it_usv { \g_um_it_Nabla_usv }
280     \tl_set:Nn \g_um_bfNabla_up_or_it_usv { \g_um_bfit_Nabla_usv }
281     \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfit_Nabla_usv }
282   }
283 }
284 \define@choicekey*{unicode-math.sty}{partial}[\@tempa\@tempb]{upright,italic}{
285   \ifcase\@tempb
286     \bool_set_true:N \g_um_uppartial_bool
287   \or
288     \bool_set_false:N \g_um_uppartial_bool
289   \fi
290 }
291 \cs_set:Nn \um_setup_partial: {
292   \bool_if:NTF \g_um_uppartial_bool {
293     \tl_set:Nn \g_um_partial_up_or_it_usv { \g_um_up_partial_usv }
294     \tl_set:Nn \g_um_bfpartial_up_or_it_usv { \g_um_bfup_partial_usv }
295     \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfup_partial_usv }

```

```

296   }{
297     \tl_set:Nn \g_um_partial_up_or_it_usv { \g_um_it_partial_usv }
298     \tl_set:Nn \g_um_bfpartial_up_or_it_usv { \g_um_bfit_partial_usv }
299     \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfit_partial_usv }
300   }
301 }

```

Epsilon and phi shapes

```

302 \define@choicekey*{unicode-math.sty}{vargreek-shape}[\@tempa\@tempb]{unicode,TeX}{
303   \ifcase\@tempb
304     \bool_set_false:N \g_um_texgreek_bool
305   \or
306     \bool_set_true:N \g_um_texgreek_bool
307   \fi
308 }

```

Colon style

```

309 \bool_new:N \g_um_literal_colon_bool
310 \define@choicekey*{unicode-math.sty}{colon}[\@tempa\@tempb]{literal,TeX}{
311   \ifcase\@tempb
312     \bool_set_true:N \g_um_literal_colon_bool
313   \or
314     \bool_set_false:N \g_um_literal_colon_bool
315   \fi
316 }

```

Slash delimiter style

```

317 \define@choicekey*{unicode-math.sty}{slash-delimiter}[\@tempa\@tempb]{ascii,frac,div}{
318   \ifcase\@tempb
319     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
320   \or
321     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
322   \or
323     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
324   \fi
325 }

326 \ExecuteOptionsX{math-style=TeX,slash-delimiter=ascii}
327 \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.

```

328 \tl_map_inline:nn {

```

```

329 \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
330 \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
331 \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
332 \version@list\version@elt\alpha@list\alpha@elt
333 \restore@mathversion\init@restore@version\dorestore@version\process@table
334 \new@mathversion\DeclareSymbolFont\group@list\group@elt
335 \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
336 \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
337 \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
338 \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter\@DeclareMathDelimiter
339 \@xDeclareMathDelimiter\set@mathdelimiter\set@mathdelimiter\DeclareMathRadical
340 \mathchar@type\DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
341 }{
342 \tl_remove_in:Nn \@preamblecmds {\do#1}
343 }

```

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.

```

344 \def\um_fontdimen_to_percent:nn#1#2{
345   0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
346 }

```

`\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.

```

347 \def\um@scaled@apply#1#2#3{
348   \ifx#1\scriptstyle
349     #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
350   \else
351     \ifx#1\scriptscriptstyle
352       #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
353     \else
354       #2#3%
355     \fi
356   \fi
357 }

```


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 `ltxssbas.dtx`) we want to redefine

```
358 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}  
359 \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_YL^AT_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.

```
360 \cs_set:Nn \um_set_mathsymbol:nNNn {
```

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

```
361 \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`.

```
362 \begingroup  
363 \char_make_active:n {#4}  
364 \global\mathcode#4="8000\relax  
365 \um@scanactivedef #4 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }  
366 \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`.

```
367 \expandafter\global\expandafter\XeTeXmathchardef  
368 \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.

```

369 \cs_gset:cpx { \cs_to_str:N #2 _sym } {
370   \exp_not:c { \cs_to_str:N #2 op }
371   \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
372 }

```

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

```

373 \else

```

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

```

374 \ifx\mathopen#3\relax
375   \tl_if_in:NnTF \l_um_radicals_tl #2 {
376     \cs_gset:cpn { \cs_to_str:N #2 sign } { \XeTeXradical #1 #4 \relax }
377   }{
378     \cs_gset:Npn #2 { \XeTeXdelimiter "\mathchar@type#3 #1 #4 \relax }
379     \global\XeTeXdelcode#4=#1 #4 \relax
380     \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
381   }
382 \else
383   \ifx\mathclose#3\relax
384     \cs_gset:Npn #2 { \XeTeXdelimiter "\mathchar@type#3 #1 #4 \relax }
385     \global\XeTeXdelcode#4=#1 #4 \relax
386     \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
387   \else

```

Fences

```

388 \ifx\mathfence#3
389   \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
390   \global\XeTeXdelcode#4=#1 #4 \relax
391   \cs_gset:cpn {l \cs_to_str:N #2} { \XeTeXdelimiter "\math-
char@type\mathopen #1 #4 \relax }
392   \cs_gset:cpn {r \cs_to_str:N #2} { \XeTeXdelimiter "\math-
char@type\mathclose #1 #4 \relax }
393 \else

```

Accents

```

394 \ifx\mathaccent#3\relax
395   \cs_gset:Npx #2 { \XeTeXmathaccent "\mathchar@type#3 #1 #4 \relax }
396 \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.

```

397   \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4 \relax
398 \fi

```

```

399         \fi
400     \fi
401     \fi
402 \fi
403 }

```

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

```

404 \cs_set:Nn \um_set_mathcode:nnnn {
405     \XeTeXmathcode#1="\mathchar@type#2 \csname sym#3\endcsname #4\relax
406 }

```

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

```

407 \DeclareDocumentCommand \setmathfont { O{ } m } {

```

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

```

408     \let\glb@currsizel\relax

```

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

```

409     \bool_set_true:N \l_um_init_bool
410     \seq_clear:N \l_um_char_range_seq
411     \let\um@char@num@range\@empty

```

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

```

412     \csname S@\f@size\endcsname

```

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

```

413     \tl_set:Nn \l_um_mversion_tf {normal}
414     \DeclareMathVersion{\l_um_mversion_tf}

```

Define default font features for the script and scriptscript font.

```

415 \tl_set:Nn \l_um_script_features_tl {ScriptStyle}
416 \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
417 \tl_set:Nn \l_um_script_font_tl      {#2}
418 \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.

```

419 \setkeys*{unicode-math.sty}{#1}
420 \cs_set:Npx \um_tmp: {
421   \exp_not:N \setkeys*[um]{options}{\exp_not:V \XKV@rm}
422 }
423 \um_tmp:
424 \cs_set:Npx \um_tmp: {
425   \exp_not:N \zf@fontspec {
426     BoldFont = {}, ItalicFont = {},
427     Script = Math,
428     SizeFeatures = {
429       {Size = \tf@size-} ,
430       {Size = \sf@size-\tf@size ,
431         Font = \l_um_script_font_tl ,
432         \l_um_script_features_tl
433       } ,
434       {Size = -\sf@size ,
435         Font = \l_um_sscript_font_tl ,
436         \l_um_sscript_features_tl
437       }
438     },
439     \XKV@rm
440   }{#2}
441 }
442 \bool_set_true:N \l_um_fontspec_feature_bool
443 \um_tmp:
444 \bool_set_false:N \l_um_fontspec_feature_bool

```

Check for the correct number of `\fontdimens`:

```

445 \font\l_um_font="#2"\relax
446 %% \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
447 %% \bool_set_true:N \l_um_ot_math_bool
448 %% \else
449 %% \bool_set_false:N \l_um_ot_math_bool
450 %% \PackageWarningNoLine{unicode-math}{
451 %%   The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
452 %%   Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
453 %%   in~ a~ substandard~ manner
454 %% }

```

455 %% \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

```

456 \bool_if:NTF \l_um_init_bool {
457   \tl_set:Nn \um_symfont_tl {um_allsym}
458   \PackageInfo{unicode-math}{Defining~ the~ default~ maths~ font~ as~ '#2'}
459   \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_noparse:nnnn
460   \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
461   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
462   \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
463   \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_noparse:nn
464 }{
465   \int_incr:N \g_um_fam_int
466   \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
467   \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_parse:nnnn
468   \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_parse:Nnn
469   \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
470   \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
471   \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_parse:nn
472 }

```

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

```

473 \DeclareSymbolFont{\um_symfont_tl}
474   {\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`.

```

475 \@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` etc.)

```

476 \um_setup_nabla:
477 \um_setup_partial:
478 \um_remap_symbols:

```

```

479 \um_setup_mathactives:
480 \um_setup_delcodes:
481 \um_setup_alphabets:
482 }

```

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.

```

483 \cs_set:Nn \um_process_symbol_noparse:nnnn {
484   \exp_args:Nc \um_set_mathsymbol:nNNn {sym\um_symfont_tl}#2#3{#1}
485 }

486 \cs_set:Nn \um_process_symbol_parse:nnnn {
487   \um@parse@term{#1}{#2}{#3}{
488     \um_process_symbol_noparse:nnnn{#1}{#2}{#3}{#4}
489   }
490 }

```

`\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:nnnn`

```

491 \cs_new:Nn \um_remap_symbols: {
492   \um_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
493   \um_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "cen-
      tred asterisk"
494   \bool_if:NF \g_um_literal_colon_bool {
495     \um_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
496   }
497   \bool_if:NTF \g_um_literal_bool {
498     \um_remap_symbol:nnn {\g_um_up_Nabla_usv}{\mathord}{\g_um_up_Nabla_usv}
499     \um_remap_symbol:nnn {\g_um_it_Nabla_usv}{\mathord}{\g_um_it_Nabla_usv}
500     \um_remap_symbol:nnn {\g_um_up_partial_usv}{\mathord}{\g_um_up_partial_usv}
501     \um_remap_symbol:nnn {\g_um_it_partial_usv}{\mathord}{\g_um_it_partial_usv}
502   }{
503     \um_remap_symbol:nnn {\g_um_up_Nabla_usv,\g_um_it_Nabla_usv}{\mathord}{\g_um_Nabla_up_or_
504     \um_remap_symbol:nnn {\g_um_up_partial_usv,\g_um_it_partial_usv}{\mathord}{\g_um_partial_
505   }

```

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

```

506 \bool_if:NTF \g_um_bfliteral_bool {
507   \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv }{\mathord}{\g_um_bfup_Nabla_usv}
508   \um_remap_symbol:nnn {\g_um_bfit_Nabla_usv }{\mathord}{\g_um_bfit_Nabla_usv}
509   \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv }{\mathord}{\g_um_bfsfup_Nabla_usv}
510   \um_remap_symbol:nnn {\g_um_bfsfit_Nabla_usv }{\mathord}{\g_um_bfsfit_Nabla_usv}
511   \um_remap_symbol:nnn {\g_um_bfup_partial_usv }{\mathord}{\g_um_bfup_partial_usv}
512   \um_remap_symbol:nnn {\g_um_bfit_partial_usv }{\mathord}{\g_um_bfit_partial_usv}
513   \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv}{\mathord}{\g_um_bfsfup_partial_usv}

```

```

514 \um_remap_symbol:nnn {\g_um_bfsfit_partial_usv}{\mathord}{\g_um_bfsfit_partial_usv}
515 }{
516 \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv,\g_um_bfit_Nabla_usv}{\mathord}{\g_um_bfNabla_
517 \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv,\g_um_bfsfit_Nabla_usv}{\mathord}{\g_um_bfsf
518 \um_remap_symbol:nnn {\g_um_bfup_partial_usv,\g_um_bfit_partial_usv}{\mathord}{\g_um_bfpa
519 \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv,\g_um_bfsfit_partial_usv}{\mathord}{\g_um_
520 }
521 }

```

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

```

522 \cs_new:Nn \um_remap_symbol_parse:nnn {
523 \um@parse@term {#3} {\@nil} {#2} {
524 \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
525 }
526 }
527 \cs_new:Nn \um_remap_symbol_noparse:nnn {
528 \clist_map_inline:nn {#1} {
529 \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
530 }
531 }

```

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:`

```

532 \cs_new:Nn \um_setup_mathactives: {
533 \um_make_mathactive:nnn {"2032} \sprime \mathord
534 }

```

`\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!

```

535 \cs_new:Nn \um_make_mathactive:nnn {
536 \XeTeXmathchardef #2 = "\mathchar@type #3
537 \csname sym\um_symfont_tl\endcsname
538 #1 \scan_stop:
539 \XeTeXmathcodenum #1 = "1FFFFF \scan_stop:
540 }

```

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:`

```

541 \cs_new:Nn \um_setup_delcodes: {
542   \um_set_delcode:nn {\ /} {\g_um_slash_delimiter_usv}
543   \um_set_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
544   \um_set_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
545   \um_set_delcode:n {"005C} % backslash
546   \um_set_delcode:nn {\<} {"27E8} % angle brackets with ascii notation
547   \um_set_delcode:nn {\>} {"27E9} % angle brackets with ascii notation
548   \um_set_delcode:n {"2191} % up arrow
549   \um_set_delcode:n {"2193} % down arrow
550   \um_set_delcode:n {"2195} % updown arrow
551   \um_set_delcode:n {"219F} % up arrow twohead
552   \um_set_delcode:n {"21A1} % down arrow twohead
553   \um_set_delcode:n {"21A5} % up arrow from bar
554   \um_set_delcode:n {"21A7} % down arrow from bar
555   \um_set_delcode:n {"21A8} % updown arrow from bar
556   \um_set_delcode:n {"21BE} % up harpoon right
557   \um_set_delcode:n {"21BF} % up harpoon left
558   \um_set_delcode:n {"21C2} % down harpoon right
559   \um_set_delcode:n {"21C3} % down harpoon left
560   \um_set_delcode:n {"21C5} % arrows up down
561   \um_set_delcode:n {"21F5} % arrows down up
562   \um_set_delcode:n {"21C8} % arrows up up
563   \um_set_delcode:n {"21CA} % arrows down down
564   \um_set_delcode:n {"21D1} % double up arrow
565   \um_set_delcode:n {"21D3} % double down arrow
566   \um_set_delcode:n {"21D5} % double updown arrow
567   \um_set_delcode:n {"21DE} % up arrow double stroke
568   \um_set_delcode:n {"21DF} % down arrow double stroke
569   \um_set_delcode:n {"21E1} % up arrow dashed
570   \um_set_delcode:n {"21E3} % down arrow dashed
571   \um_set_delcode:n {"21E7} % up white arrow
572   \um_set_delcode:n {"21E9} % down white arrow
573   \um_set_delcode:n {"21EA} % up white arrow from bar
574   \um_set_delcode:n {"21F3} % updown white arrow
575 }

```

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

```

\um_set_delcode:n
576 \cs_new:Nn \um_set_delcode:nn {
577   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #2
578 }
579 \cs_new:Nn \um_set_delcode:n {
580   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #1
581 }

```


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.

```
582 \cs_set:Nn \um_mathmap_noparse:Nnn {  
583   \clist_map_inline:nn {#2} {  
584     \tl_put_right:cx {um_setup_\cs_to_str:N #1:} {  
585       \exp_not:N\um_set_mathcode:nnnn{##1}{\exp_not:N\mathalpha}{\um_symfont_tl}{#3}  
586     }  
587   }  
588 }
```

`\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.

```
589 \cs_set:Nn \um_mathmap_parse:Nnn {  
590   \clist_map_inline:Nn \um@char@num@range {  
591     \ifnum##1=#3\relax  
592       \um_mathmap_noparse:Nnn {#1}{#2}{#3}  
593     \fi  
594   }  
595 }
```

























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_{01}^1	<code>\Bbbsum</code>	DOUBLE-STRUCK N-ARY SUMMATION
U+0220F	\prod_{01}^1	<code>\prod</code>	PRODUCT OPERATOR
U+02210	\coprod_{01}^1	<code>\coprod</code>	COPRODUCT OPERATOR
U+02211	\sum_{01}^1	<code>\sum</code>	SUMMATION OPERATOR
U+0222B	\int_{01}^1	<code>\int</code>	INTEGRAL OPERATOR
U+0222C	\iint_{01}^1	<code>\iint</code>	DOUBLE INTEGRAL OPERATOR
U+0222D	\iiint_{01}^1	<code>\iiint</code>	TRIPLE INTEGRAL OPERATOR
U+0222E	\oint_{01}^1	<code>\oint</code>	CONTOUR INTEGRAL OPERATOR
U+0222F	\oiint_{01}^1	<code>\oiint</code>	DOUBLE CONTOUR INTEGRAL OPERATOR
U+02230	\oiiint_{01}^1	<code>\oiiint</code>	TRIPLE CONTOUR INTEGRAL OPERATOR
U+02231	\int_{01}^1	<code>\intclockwise</code>	CLOCKWISE INTEGRAL
U+02232	\oint_{01}^1	<code>\varointclockwise</code>	CONTOUR INTEGRAL, CLOCKWISE
U+02233	\oint_{01}^1	<code>\ointctrackwise</code>	CONTOUR INTEGRAL, ANTICLOCKWISE
U+022C0	\bigwedge_{01}^1	<code>\bigwedge</code>	LOGICAL OR OPERATOR
U+022C1	\bigvee_{01}^1	<code>\bigvee</code>	LOGICAL AND OPERATOR
U+022C2	\bigcap_{01}^1	<code>\bigcap</code>	INTERSECTION OPERATOR
U+022C3	\bigcup_{01}^1	<code>\bigcup</code>	UNION OPERATOR
U+027D5	\Join_{01}^1	<code>\leftouterjoin</code>	LEFT OUTER JOIN
U+027D6	\Join_{01}^1	<code>\rightouterjoin</code>	RIGHT OUTER JOIN
U+027D7	\Join_{01}^1	<code>\fullouterjoin</code>	FULL OUTER JOIN
U+027D8	\bot_{01}^1	<code>\bigbot</code>	LARGE UP TACK
U+027D9	\top_{01}^1	<code>\bigtop</code>	LARGE DOWN TACK
U+029F8	$/_{01}^1$	<code>\xsol</code>	BIG SOLIDUS
U+029F9	\backslash_{01}^1	<code>\xbsol</code>	BIG REVERSE SOLIDUS
U+02A00	\odot_{01}^1	<code>\bigodot</code>	N-ARY CIRCLED DOT OPERATOR

U+02A01		\bigoplus	N-ARY CIRCLED PLUS OPERATOR
U+02A02		\bigotimes	N-ARY CIRCLED TIMES OPERATOR
U+02A03		\bigcupdot	N-ARY UNION OPERATOR WITH DOT
U+02A04		\biguplus	N-ARY UNION OPERATOR WITH PLUS
U+02A05		\bigsqcap	N-ARY SQUARE INTERSECTION OPERATOR
U+02A06		\bigsqcup	N-ARY SQUARE UNION OPERATOR
U+02A07		\conjquant	TWO LOGICAL AND OPERATOR
U+02A08		\disjquant	TWO LOGICAL OR OPERATOR
U+02A09		\bigtimes	N-ARY TIMES OPERATOR
U+02A0B		\sumint	SUMMATION WITH INTEGRAL
U+02A0C		\iiint	QUADRUPLE INTEGRAL OPERATOR
U+02A0D		\intbar	FINITE PART INTEGRAL
U+02A0E		\intBar	INTEGRAL WITH DOUBLE STROKE
U+02A0F		\fint	INTEGRAL AVERAGE WITH SLASH
U+02A10		\cirfnint	CIRCULATION FUNCTION
U+02A11		\awint	ANTICLOCKWISE INTEGRATION LINE INTEGRATION WITH RECTANGULAR
U+02A12		\rppointint	PATH AROUND POLE LINE INTEGRATION WITH SEMICIRCULAR
U+02A13		\scpointint	PATH AROUND POLE LINE INTEGRATION NOT INCLUDING THE
U+02A14		\npointint	POLE INTEGRAL AROUND A POINT OPERATOR
U+02A15		\pointint	INTEGRAL AROUND A POINT OPERATOR
U+02A16		\sqint	QUATERNION INTEGRAL OPERATOR INTEGRAL WITH LEFTWARDS ARROW WITH
U+02A17		\intlarhk	HOOK
U+02A18		\intx	INTEGRAL WITH TIMES SIGN
U+02A19		\intcap	INTEGRAL WITH INTERSECTION
U+02A1A		\intcup	INTEGRAL WITH UNION
U+02A1B		\upint	INTEGRAL WITH OVERBAR
U+02A1C		\lowint	INTEGRAL WITH UNDERBAR
U+02A1D		\Join	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.

```

596 \tl_new:Nn \l_um_nolimits_tl {
597   \int\iint\iiint\iiiint\oint\oiint\oiint
598   \intclockwise\varointclockwise\ointctrlockwise\sumint
599   \intbar\intBar\oint\circfnint\awint\rppoint
600   \scpolint\ncpolint\pointint\sqint\intlarhk\intx
601   \intcap\intcup\upoint\lowint
602 }

```

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

```

603 \DeclareDocumentCommand \addnolimits {m} {
604   \tl_put_right:Nn \l_um_nolimits_tl {#1}
605 }

```

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

```

606 \DeclareDocumentCommand \removenolimits {m} {
607   \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
608 }

```

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.

```
609 \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:















```
%%: // . - . / - / 2 ? = / 3853 & /
3754
```

```
610 \let\left@primitive\left
611 \def\left{\mathopen{}\left@primitive}
```

No re-definition is made for `\right` because it's not necessary.












Here are all `\mathopen` characters:

USV	Ex.	Macro	Description
U+00028	(<code>\lparen</code>	LEFT PARENTHESIS
U+0005B	[<code>\lbrack</code>	LEFT SQUARE BRACKET
U+0007B	{	<code>\lbrace</code>	LEFT CURLY BRACKET
U+0221A	✓	<code>\sqrt</code>	RADICAL
U+0221B	∛	<code>\cuberoot</code>	CUBE ROOT
U+0221C	∜	<code>\fourthroot</code>	FOURTH ROOT
U+02308	⌈	<code>\lceil</code>	LEFT CEILING
U+0230A	⌋	<code>\lfloor</code>	LEFT FLOOR
U+0231C	⌞	<code>\ulcorner</code>	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
			MATHEMATICAL RIGHT WHITE SQUARE
U+027E7	⌋	<code>\rBrack</code>	BRACKET
U+027E9	⌋	<code>\rangle</code>	MATHEMATICAL RIGHT ANGLE BRACKET
			MATHEMATICAL RIGHT DOUBLE ANGLE
U+027EB	⌋	<code>\rAngle</code>	BRACKET
			MATHEMATICAL RIGHT WHITE TORTOISE
U+027ED		<code>\Rbrbrak</code>	SHELL BRACKET
U+02984	⌋	<code>\rBrace</code>	RIGHT WHITE CURLY BRACKET
U+02986	⌋	<code>\rParen</code>	RIGHT WHITE PARENTHESIS
U+02988	⌋	<code>\rrparenthesis</code>	Z NOTATION RIGHT IMAGE BRACKET
U+0298A	⌋	<code>\rrangle</code>	Z NOTATION RIGHT BINDING BRACKET
U+0298C	⌋	<code>\rbrackubar</code>	RIGHT SQUARE BRACKET WITH UNDERBAR

U+0298E		<code>\rbracklrtick</code>	RIGHT SQUARE BRACKET WITH TICK IN BOTTOM CORNER
U+02990		<code>\rbrackurtick</code>	RIGHT SQUARE BRACKET WITH TICK IN TOP CORNER
U+02992		<code>\rangledot</code>	RIGHT ANGLE BRACKET WITH DOT
U+02994		<code>\rpargtr</code>	RIGHT ARC GREATER-THAN BRACKET
U+02996		<code>\Rparenless</code>	DOUBLE RIGHT ARC LESS-THAN BRACKET
U+02998		<code>\rblbrbrak</code>	RIGHT BLACK TORTOISE SHELL BRACKET
U+029D9		<code>\rvzigzag</code>	RIGHT WIGGLY FENCE
U+029DB		<code>\Rvzigzag</code>	RIGHT DOUBLE WIGGLY FENCE
U+029FD		<code>\rcurvyangle</code>	RIGHT POINTING CURVED ANGLE BRACKET
U+03015		<code>\rbrbrak</code>	RIGHT BROKEN BRACKET
U+03019		<code>\Rbrbrak</code>	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		<code>\grave</code>	GRAVE ACCENT
U+00301		<code>\acute</code>	ACUTE ACCENT
U+00302		<code>\hat</code>	CIRCUMFLEX ACCENT
U+00303		<code>\tilde</code>	TILDE
U+00304		<code>\bar</code>	MACRON
U+00305		<code>\overbar</code>	OVERBAR EMBELLISHMENT
U+00306		<code>\breve</code>	BREVE
U+00307		<code>\dot</code>	DOT ABOVE
U+00308		<code>\ddot</code>	DIERESIS
U+00309		<code>\ovhook</code>	COMBINING HOOK ABOVE
U+0030A		<code>\ocirc</code>	RING
U+0030C		<code>\check</code>	CARON
U+00310		<code>\candra</code>	CANDRABINDU (NON-SPACING)
U+00312		<code>\oturnedcomma</code>	COMBINING TURNED COMMA ABOVE GREEK PSILI (SMOOTH BREATHING)
U+00313		<code>\osmooth</code>	(NON-SPACING) GREEK DASIA (ROUGH BREATHING)
U+00314		<code>\orough</code>	(NON-SPACING)
U+00315		<code>\ocommatopright</code>	COMBINING COMMA ABOVE RIGHT
U+0031A		<code>\droang</code>	LEFT ANGLE ABOVE (NON-SPACING)
U+00330		<code>\wideutilde</code>	UNDER TILDE ACCENT (MULTIPLE CHARACTERS AND NON-SPACING)
U+00331		<code>\underbar</code>	COMBINING MACRON BELOW
U+00338		<code>\not</code>	COMBINING LONG SOLIDUS OVERLAY

U+020D0		<code>\leftharpoonaccent</code>	COMBINING LEFT HARPOON ABOVE
U+020D1		<code>\rightharpoonaccent</code>	COMBINING RIGHT HARPOON ABOVE
U+020D2		<code>\vertoverlay</code>	COMBINING LONG VERTICAL LINE OVERLAY
U+020D6		<code>\overleftarrow</code>	COMBINING LEFT ARROW ABOVE
U+020D7		<code>\vec</code>	COMBINING RIGHT ARROW ABOVE
U+020DB		<code>\dddot</code>	COMBINING THREE DOTS ABOVE
U+020DC		<code>\ddddot</code>	COMBINING FOUR DOTS ABOVE
U+020E1		<code>\overleftrightharpoon</code>	COMBINING LEFT RIGHT ARROW ABOVE
U+020E7		<code>\annuity</code>	COMBINING ANNUITY SYMBOL
U+020E8		<code>\threeunderdot</code>	COMBINING TRIPLE UNDERDOT
U+020E9		<code>\widebridgeabove</code>	COMBINING WIDE BRIDGE ABOVE
U+020EC		<code>\underrightharpoondown</code>	BARB DOWNWARDS COMBINING LEFTWARDS HARPOON WITH
U+020ED		<code>\underleftharpoondown</code>	BARB DOWNWARDS
U+020EE		<code>\underleftarrow</code>	COMBINING LEFT ARROW BELOW
U+020EF		<code>\underrightarrow</code>	COMBINING RIGHT ARROW BELOW
U+020F0		<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.

```

612 \newcommand\um@zf@feature[2]{
613   \define@key[zf]{options}{#1}[] {
614     \bool_if:NTF \l_um_fontspec_feature_bool {
615       #2
616     } {
617       \PackageError{fontspec/unicode-math}
618         {The ‘#1’ font feature can only be used for maths fonts}
619         {The feature you tried to use can only be in commands
620          like \protect\setmathfont}
621     }
622   }
623 }
```

8.1 OpenType maths font features

```

624 \um@zf@feature{ScriptStyle}{
625   \zf@update@ff{+ssty=0}
626 }
627 \um@zf@feature{ScriptScriptStyle}{
```



```

628 \zf@update@ff{+ssty=1}
629 }

```

8.2 Script and scriptscript font options

```

630 \define@cmdkey[um]{options}[um@]{script-features}{}
631 \define@cmdkey[um]{options}[um@]{sscript-features}{}
632 \define@cmdkey[um]{options}[um@]{script-font}{}
633 \define@cmdkey[um]{options}[um@]{sscript-font}{}

```

8.3 Range processing

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

```

634 \seq_new:N \g_um_mathalph_seq
635 \seq_new:N \l_um_mathalph_seq
636 \seq_new:N \l_um_char_range_seq
637 \define@choicekey+[um]{options}{range}[\@tempa\@tempb]{ALL}{
638   \ifcase\@tempb\relax
639     \bool_set_true:N \l_um_init_bool
640   \fi
641 }{
642   \bool_set_false:N \l_um_init_bool
643   \seq_clear:N \l_um_char_range_seq
644   \seq_clear:N \l_um_mathalph_seq
645   \clist_map_inline:nn {#1} {
646     \um_if_mathalph_decl:nTF {##1} {
647       \seq_put_right:Nx \l_um_mathalph_seq { {\exp_not:V\l_um_tmpa_tl} {\exp_not:V\l_um_tmpb_tl} }
648     }{
649       \seq_put_right:Nn \l_um_char_range_seq {##1}
650     }
651   }
652 }
653 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
654   \tl_set:Nn \l_um_tmpa_tl {#1}
655   \tl_set:Nn \l_um_tmpb_tl {}
656   \tl_set:Nn \l_um_tmpc_tl {}
657   \tl_if_in:NnT \l_um_tmpa_tl {->} {
658     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
659   }
660   \tl_if_in:NnT \l_um_tmpa_tl {/} {
661     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
662   }
663   \seq_if_in:NVTF \g_um_mathalph_seq \l_um_tmpa_tl {
664     \prg_return_true:
665   }{
666     \prg_return_false:
667   }
668 }

```

```

669 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
670   \tl_set:Nn \l_um_tmpa_tl {#1}
671   \tl_set:Nn \l_um_tmpc_tl {#2}
672 }
673 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
674   \tl_set:Nn \l_um_tmpa_tl {#1}
675   \tl_set:Nn \l_um_tmpb_tl {#2}
676 }

```

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 13.

Table 13: 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:

```

677 \newcommand\um@parse@term[4]{
678   \seq_map_variable:NNn \l_um_char_range_seq \@ii {
679     \unless\ifx\@ii\@empty
680       \@tempswafalse

```

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

```

681     \expandafter\um@firstchar\expandafter{\@ii}
682     \ifx\@tempa\um@backslash
683       \expandafter\ifx\@ii#2\relax
684         \@tempswatrue
685     \else
686       \expandafter\ifx\@ii#3\relax
687         \@tempswatrue

```

```

688         \fi
689     \fi

```

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

```

690     \else
691         \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
692     \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.

```

693     \if@tempswa
694         \ifx\um@char@num@range\@empty
695             \g@addto@macro\um@char@num@range{#1}
696         \else
697             \g@addto@macro\um@char@num@range{, #1}
698         \fi
699         #4%
700     \fi
701 \fi
702 }
703 }
704 \def\um@firstof#1#2\@nil{#1}
705 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
706 \def\um@firstchar#1{\edef\@tempa{\expandafter\um@firstof\string#1\@nil}}

```

`\um@parse@range` Weird syntax. As shown previously in table 13, this macro can be passed four different input types via `\um@parse@term`.

```

707 \def\um@parse@range#1-#2-#3\@nil#4\@nil{
708     \def\@tempa{#1}
709     \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	$\textcolor{red}{\#1}-\textcolor{blue}{\#2}-\textcolor{blue}{\#3} = \textcolor{red}{X}-\textcolor{blue}{\@marker}-\textcolor{blue}{\{}}$

```

710     \expandafter\ifx\expandafter\@marker\@tempb\relax
711         \ifnum#4=#1\relax
712             \@tempwattrue
713         \fi
714     \else

```

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

```

715     \ifx\@empty\@tempb
716         \ifnum#4>\numexpr#1-1\relax
717             \@tempwattrue

```

```

718     \fi
719     \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-


---


720     \ifx\@empty\@tempa
721         \ifnum#4<\numexpr#2+1\relax
722             \@tempwattrue
723         \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-


---


724     \else
725         \ifnum#4>\numexpr#1-1\relax
726         \ifnum#4<\numexpr#2+1\relax
727             \@tempwattrue
728         \fi
729     \fi
730 \fi
731 \fi
732 \fi
733 }

\um_map_char:nn #1 : Number of iterations
\um_map_chars_xxvi:nn #2 : Starting input char(s)
\um_map_chars_xxiii:nn #3 : Starting output char
Loops through character ranges setting \mathcode.
734 \cs_set:Nn \um_map_chars_range:nnn {
735     \clist_map_inline:nn {#2} {
736         \prg_stepwise_inline:nnnn {0}{1}{#1} {
737             \um_map_char_internal:nn {##1+###1}{#3+###1}
738         }
739     }
740 }
741 \cs_new:Nn \um_map_char_noparse:nn {
742     \um_set_mathcode:nnnn
743     {\numexpr #1 \relax}{\mathalpha}{\um_symfont_tl}{\numexpr #2 \relax}
744 }
745 \cs_new:Nn \um_map_char_parse:nn {
746     \um@parse@term {#1} {\@nil} {\mathalpha} {
747         \um_map_char_noparse:nn {#1}{#2}
748     }
749 }

```

```

750 \cs_set:Nn \um_map_chars_xxvi:nn {
751   \um_map_chars_range:nnn {25}{#1}{#2}
752 }
753 \cs_set:Nn \um_map_chars_xxiii:nn {
754   \um_map_chars_range:nnn {24}{#1}{#2}
755 }
756 \cs_set:Nn \um_map_chars_Latin:nn {
757   \clist_map_inline:nn {#1} {
758     \um_map_chars_xxvi:cc {g_um_ ##1 _Latin_usv}{g_um_ #2 _Latin_usv}
759   }
760 }
761 \cs_set:Nn \um_map_chars_latin:nn {
762   \clist_map_inline:nn {#1} {
763     \um_map_chars_xxvi:cc {g_um_ ##1 _latin_usv}{g_um_ #2 _latin_usv}
764   }
765 }
766 \cs_set:Nn \um_map_chars_greek:nn {
767   \clist_map_inline:nn {#1} {
768     \um_map_chars_xxiii:cc {g_um_ ##1 _greek_usv}{g_um_ #2 _greek_usv}
769     \um_map_char:cc {g_um_ ##1 _varepsilon_usv}{g_um_ #2 _varepsilon_usv}
770     \um_map_char:cc {g_um_ ##1 _vartheta_usv }{g_um_ #2 _vartheta_usv }
771     \um_map_char:cc {g_um_ ##1 _varkappa_usv }{g_um_ #2 _varkappa_usv }
772     \um_map_char:cc {g_um_ ##1 _varphi_usv }{g_um_ #2 _varphi_usv }
773     \um_map_char:cc {g_um_ ##1 _varrho_usv }{g_um_ #2 _varrho_usv }
774     \um_map_char:cc {g_um_ ##1 _varpi_usv }{g_um_ #2 _varpi_usv }
775   }
776 }
777 \cs_set:Nn \um_map_chars_Greek:nn {
778   \clist_map_inline:nn {#1} {
779     \um_map_chars_xxiii:cc {g_um_ ##1 _Greek_usv}{g_um_ #2 _Greek_usv}
780     \um_map_char:cc {g_um_ ##1 _varTheta_usv}{g_um_ #2 _varTheta_usv}
781   }
782 }
783 \cs_set:Nn \um_map_chars_numbers:nn {
784   \um_map_chars_range:nnn {9}{#1}{#2}
785 }
786 \cs_set:Nn \um_map_char:nn {
787   \um_map_chars_range:nnn {0}{#1}{#2}
788 }
789 \cs_generate_variant:Nn \um_map_char:nn {cc}
790 \cs_generate_variant:Nn \um_map_chars_xxiii:nn {cc}
791 \cs_generate_variant:Nn \um_map_chars_xxvi:nn {cc}

```

\um_set_mathalphabet_char:Nnn #1 : Maths alphabet
 #2 : Input char(s)
 #3 : Output char

Loops through character ranges setting \mathcode.

```

792 \cs_set:Npn \exp_args:Nnff {\::n\::f\::f\:::}
793 \cs_new:Nn \um_set_mathalphabet_char:Nnn {
794   \clist_map_variable:nNn {#2} \l_um_input_num {
795     \exp_args:Nnff \um_mathmap:Nnn {#1}
796     {\number\numexpr\l_um_input_num\relax} {\number\numexpr#3\relax}
797   }
798 }

```

\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.

```

799 \cs_new:Nn \um_set_mathalph_range:nNnn {
800   \clist_map_variable:nNn {#3} \l_um_input_num {
801     \errorcontextlines=999
802     \prg_stepwise_variable:nnnNn {0}{1}{#1} \l_um_inc_num {
803       \exp_args:Nnff \um_mathmap:Nnn {#2}
804       {\number\numexpr \l_um_inc_num + \l_um_input_num \relax}
805       {\number\numexpr \l_um_inc_num + #4 \relax}
806     }
807   }
808 }
809 \cs_new:Nn \um_set_mathalphabet_x:Nnn {
810   \um_set_mathalph_range:nNnn {9}{#1}{#2}{#3}
811 }
812 \cs_new:Nn \um_set_mathalphabet_xxvi:Nnn {
813   \um_set_mathalph_range:nNnn {25}{#1}{#2}{#3}
814 }
815 \cs_new:Nn \um_set_mathalphabet_xxiii:Nnn {
816   \um_set_mathalph_range:nNnn {24}{#1}{#2}{#3}
817 }
818
819 \cs_new:Nn \um_set_mathalphabet_pos:Nnnn {
820   \clist_map_inline:nn {#3} {
821     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_#2_usv}{g_um_#4_#2_usv}
822   }
823 }
824 \cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
825   \clist_map_inline:nn {#2} {
826     \um_set_mathalphabet_x:Ncc #1 {g_um_##1_num_usv}{g_um_#3_num_usv}
827   }
828 }
829 \cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
830   \clist_map_inline:nn {#2} {
831     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_Latin_usv}{g_um_#3_Latin_usv}

```

```

832 }
833 }
834 \cs_new:Nn \um_set_mathalphabet_latin:Nnn {
835   \clist_map_inline:nn {#2} {
836     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_latin_usv}{g_um_#3_latin_usv}
837     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_h_usv} {g_um_#3_h_usv}
838   }
839 }
840 \cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
841   \clist_map_inline:nn {#2} {
842     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_Greek_usv} {g_um_#3_Greek_usv}
843     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varTheta_usv}{g_um_#3_varTheta_usv}
844   }
845 }
846 \cs_new:Nn \um_set_mathalphabet_greek:Nnn {
847   \clist_map_inline:nn {#2} {
848     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_greek_usv} {g_um_#3_greek_usv}
849     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varepsilon_usv}{g_um_#3_varepsilon_usv}
850     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_vartheta_usv} {g_um_#3_vartheta_usv}
851     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varkappa_usv} {g_um_#3_varkappa_usv}
852     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varphi_usv} {g_um_#3_varphi_usv}
853     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varrho_usv} {g_um_#3_varrho_usv}
854     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varpi_usv} {g_um_#3_varpi_usv}
855   }
856 }
857 \cs_generate_variant:Nn \um_set_mathalphabet_char:Nnn {Ncc}
858 \cs_generate_variant:Nn \um_set_mathalphabet_xxiii:Nnn {Ncc}
859 \cs_generate_variant:Nn \um_set_mathalphabet_xxvi:Nnn {Ncc}
860 \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.

```

861 \AtBeginDocument{\um_resolve_greek:}
862 \cs_new:Nn \um_resolve_greek: {
863   \clist_map_inline:nn {
864     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
865     alpha,beta,gamma,delta, zeta,eta,theta,iota,kappa,lambda,
866     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
867     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon, chi,psi,omega,
868     varTheta,
869     varsigma,vartheta,varkappa,varrho,varpi
870   }{

```

```

871 \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
872 }
873 \tl_set:Nn \epsilon {
874   \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
875 }
876 \tl_set:Nn \phi {
877   \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
878 }
879 \tl_set:Nn \varepsilon {
880   \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
881 }
882 \tl_set:Nn \varphi {
883   \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
884 }
885 }

```

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`

```
886 \seq_clear:Nn \g_um_mathalph_seq
887 \tl_map_inline:nn {
888   \mathup\mathit
889   \mathbb\mathscr\mathfrak\mathtt
890   \mathsf\mathsfup\mathsfit
891   \mathbf\mathbfup\mathbfit
892   \mathbfscr\mathbffrac
893   \mathbfsf\mathbfsfup\mathbfsfit
894 }{
895   \seq_put_right:Nn \g_um_mathalph_seq {#1}
896 }
```

`\um_setup_alphabets:`

```
897
898 \tl_new:Nn \g_um_mathup_alph_clist {latin, Latin, greek, Greek, num}
899 \tl_new:Nn \g_um_mathit_alph_clist {latin, Latin, greek, Greek}
900 \tl_new:Nn \g_um_mathscr_alph_clist {latin, Latin}
901 \tl_new:Nn \g_um_mathfrak_alph_clist {latin, Latin}
902 \tl_new:Nn \g_um_mathbfscr_alph_clist {latin, Latin}
903 \tl_new:Nn \g_um_mathbffrac_alph_clist {latin, Latin}
904 \tl_new:Nn \g_um_mathbb_alph_clist {latin, Latin, num}
905 \tl_new:Nn \g_um_mathtt_alph_clist {latin, Latin, num}
906 \tl_new:Nn \g_um_mathsf_alph_clist {latin, Latin, num}
907 \tl_new:Nn \g_um_mathsfup_alph_clist {latin, Latin, num}
908 \tl_new:Nn \g_um_mathsfit_alph_clist {latin, Latin}
909 \tl_new:Nn \g_um_mathbf_alph_clist {latin, Latin, greek, Greek, num}
910 \tl_new:Nn \g_um_mathbfup_alph_clist {latin, Latin, greek, Greek, num}
911 \tl_new:Nn \g_um_mathbfit_alph_clist {latin, Latin, greek, Greek, num}
912 \tl_new:Nn \g_um_mathbfsf_alph_clist {latin, Latin, greek, Greek, num}
913 \tl_new:Nn \g_um_mathbfsfup_alph_clist {latin, Latin, greek, Greek, num}
914 \tl_new:Nn \g_um_mathbfsfit_alph_clist {latin, Latin, greek, Greek}
915
916 \tl_new:Nn \g_um_mathup_latin_usv {\a-`z}
917 \tl_new:Nn \g_um_mathup_Latin_usv {\A-`Z}
918 \tl_new:Nn \g_um_mathup_greek_usv {"3B1-"3C9,"3F5,"3D1,"3F0,"3D5,"3F1,"3D6,"3DD}
919 \tl_new:Nn \g_um_mathup_Greek_usv {"391-"3A9,"3F4,"3DC}
920 \tl_new:Nn \g_um_mathup_num_usv {\0-`9}
921
922 \tl_new:Nn \g_um_mathit_latin_usv {"1D44E-"1D467,\g_um_it_h_usv}
923 \tl_new:Nn \g_um_mathit_Latin_usv {"1D434-"1D44C}
924 \tl_new:Nn \g_um_mathit_greek_usv {"1D6FC-"1D714,"1D716-1D71B}
925 \tl_new:Nn \g_um_mathit_Greek_usv {"1D6E2-"1D6FA}
```

```

926
927 \seq_new:N \l_um_missing_alph_seq
928 \cs_new:Nn \um_setup_alphabets: {
929   \seq_clear:N \l_um_missing_alph_seq
930   \seq_if_empty:NTF \l_um_mathalph_seq {
931     \um_setup_math_alphabet:NV \mathup      \g_um_mathup_alph_clist
932     \um_setup_math_alphabet:NV \mathit      \g_um_mathit_alph_clist
933     \um_setup_math_alphabet:NV \mathbb      \g_um_mathbb_alph_clist
934     \um_setup_math_alphabet:NV \mathscr     \g_um_mathscr_alph_clist
935     \um_setup_math_alphabet:NV \mathfrak    \g_um_mathfrak_alph_clist
936     \um_setup_math_alphabet:NV \mathsf      \g_um_mathsf_alph_clist
937     \um_setup_math_alphabet:NV \mathsfup    \g_um_mathsfup_alph_clist
938     \um_setup_math_alphabet:NV \mathsfit    \g_um_mathsfit_alph_clist
939     \um_setup_math_alphabet:NV \mathtt      \g_um_mathtt_alph_clist
940     \um_setup_math_alphabet:NV \mathbf      \g_um_mathbf_alph_clist
941     \um_setup_math_alphabet:NV \mathbfup    \g_um_mathbfup_alph_clist
942     \um_setup_math_alphabet:NV \mathbfit    \g_um_mathbfit_alph_clist
943     \um_setup_math_alphabet:NV \mathbfscr    \g_um_mathbfscr_alph_clist
944     \um_setup_math_alphabet:NV \mathbffrak  \g_um_mathbffrak_alph_clist
945     \um_setup_math_alphabet:NV \mathbfsf    \g_um_mathbfsf_alph_clist
946     \um_setup_math_alphabet:NV \mathbfsfup  \g_um_mathbfsfup_alph_clist
947     \um_setup_math_alphabet:NV \mathbfsfit  \g_um_mathbfsfit_alph_clist
948     \um_setup_math_mapping:n {up }
949     \um_setup_math_mapping:n {it }
950     \um_setup_math_mapping:n {bb }
951     \um_maybe_init_alphabet:n {bbit }
952     \um_setup_math_mapping:n {bbit }
953     \um_setup_math_mapping:n {bfup }
954     \um_setup_math_mapping:n {bfit }
955     \um_setup_math_mapping:n {bfsfup}
956     \um_setup_math_mapping:n {bfsfit}
957   \seq_if_empty:NF \l_um_missing_alph_seq {
958     \typeout{
959       Package~unicode-math~Warning:~
960       missing~math~alphabets~in~font~ \fontname\l_um_font
961     }
962     \seq_map_inline:Nn \l_um_missing_alph_seq {
963       \typeout{\space\space\space\space##1}
964     }
965   }
966 }{
967   \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
968   \seq_map_inline:Nn \l_um_mathalph_seq {
969     \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
970     \tl_set:No \l_um_tmppb_tl { \use_ii:nnn ##1 }
971     \tl_set:No \l_um_tmppc_tl { \use_iii:nnn ##1 }

```

```

972 \tl_if_empty:NF \l_um_tmpc_tl {
973 \PackageWarning{unicode-math}{alphabet~remapping~not~yet~implemented}
974 }
975 \tl_if_empty:NT \l_um_tmpb_tl {
976 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
977 \tl_set:Nv \l_um_tmpb_tl { g_um_ \exp_after:wN \cs_to_str:N \l_um_tmpa_tl _alph_clist }
978 }
979 \um_setup_math_alphabet:VV \l_um_tmpa_tl \l_um_tmpb_tl
980 }
981 }
982 }

```

`\um_setup_math_alphabet:Nn` #1 : Math font family name (e.g., `\mathbb`)
 #2 : Math alphabets, comma separated of {latin, Latin, greek, Greek, num}
 First check that at least one of the alphabets for the font shape is defined, and then loop through them defining the individual ranges.

```

983 \cs_new:Nn \um_setup_math_alphabet:Nn {
984 \tl_set:Nx \l_um_tmpa_tl {\cs_to_str:N #1}
985 \tl_set:Nx \l_um_tmpb_tl {\exp_after:wN \use_none:nnnn \l_um_tmpa_tl}
986 \clist_map_inline:nn {#2} {
987 \um_glyph_if_exist:cT {g_um_ \l_um_tmpb_tl _##1_usv}{
988 \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
989 \clist_map_break:
990 }
991 }
992 \clist_map_inline:nn {#2} {
993 \um_glyph_if_exist:cTF {g_um_ \l_um_tmpb_tl _##1_usv}{
994 \use:c {um_config_ \l_um_tmpa_tl _##1:}
995 }{
996 \seq_put_right:Nx \l_um_missing_alph_seq {
997 \@backslashchar
998 \l_um_tmpa_tl\space(\tl_use:c{g_um_math_alphabet_name_##1_tl})
999 }
1000 }
1001 }
1002 }
1003 \cs_generate_variant:Nn \um_setup_math_alphabet:Nn {NV,VV}
1004
1005 \tl_set:Nn \g_um_math_alphabet_name_latin_tl {Latin,~lowercase}
1006 \tl_set:Nn \g_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
1007 \tl_set:Nn \g_um_math_alphabet_name_greek_tl {Greek,~lowercase}
1008 \tl_set:Nn \g_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
1009 \tl_set:Nn \g_um_math_alphabet_name_num_tl {Numerals}
1010 \cs_new:Nn \um_setup_math_mapping:n {
1011 \cs_if_exist:cT {um_setup_math#1:} {
1012 \use:c {um_config_math#1_misc:}

```

```

1013 }
1014 }

1015 \cs_set:Nn \um_init_alphabet:n {
1016   \wlog{unicode-math:~Initialiasing~\@backslashchar math#1}
1017   \um_prepare_alph:n {#1}
1018   \cs_set_eq:cN {um_setup_math#1:} \prg_do_nothing:
1019 }

```

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

```

1020 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
1021   \etex_iffontchar:D \l_um_font #1 \scan_stop: \prg_return_true: \else: \prg_return_false: \fi:
1022 }
1023 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
1024 \cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
1025 \cs_generate_variant:Nn \um_glyph_if_exist:nT {c}
1026 \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.

```

1027 \cs_new:Nn \um_prepare_alph:n {
1028   \cs_if_exist:cF {um_math#1:n} {
1029     \cs_set:cpn {um_math#1:n} ##1 {
1030       \use:c {um_setup_math#1:} ##1 \egroup
1031     }
1032     \cs_set_protected:cpn {math#1} {
1033       \bgroup
1034       \mode_if_math:F {
1035         \egroup\expandafter
1036         \non@alpherr\expandafter{\csname math#1\endcsname\space}
1037       }
1038       \use:c {um_math#1:n}
1039     }
1040   }
1041 }

```

9.1 Alphabets

9.1.1 Upright: `\mathup`

```

1042 \cs_new:Npn \um_config_mathup_num: {
1043   \um_map_chars_numbers:nn {\0}{\0}
1044   \um_set_mathalphabet_numbers:Nnn \mathup {up}{up}
1045 }
1046 \cs_new:Npn \um_config_mathup_Latin: {
1047   \bool_if:NTF \g_um_literal_bool {

```

```

1048     \um_map_chars_Latin:nn {up} {up}
1049   }{
1050     \bool_if:NT \g_um_upLatin_bool {
1051       \um_map_chars_Latin:nn {up,it} {up}
1052     }
1053   }
1054   \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{up}
1055 }
1056 \cs_new:Npn \um_config_mathup_latin: {
1057   \bool_if:NTF \g_um_literal_bool {
1058     \um_map_chars_latin:nn {up} {up}
1059   }{
1060     \bool_if:NT \g_um_uplatin_bool {
1061       \um_map_chars_latin:nn {up,it} {up}
1062       \um_map_char:nn {\g_um_up_h_usv,\g_um_it_h_usv}{\g_um_up_h_usv}% KEEP
1063     }
1064   }
1065   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{up}
1066 }
1067 \cs_new:Npn \um_config_mathup_Greek: {
1068   \bool_if:NTF \g_um_literal_bool {
1069     \um_map_chars_Greek:nn {up}{up}
1070   }{
1071     \bool_if:NT \g_um_upGreek_bool {
1072       \um_map_chars_Greek:nn {up,it}{up}
1073     }
1074   }
1075   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{up}
1076 }
1077 \cs_new:Npn \um_config_mathup_greek: {
1078   \bool_if:NTF \g_um_literal_bool {
1079     \um_map_chars_greek:nn {up} {up}
1080   }{
1081     \bool_if:NT \g_um_upgreek_bool {
1082       \um_map_chars_greek:nn {up,it} {up}
1083     }
1084   }
1085   \um_set_mathalphabet_greek:Nnn \mathup {up,it} {up}
1086 }
1087 \cs_new:Npn \um_config_mathup_misc: {
1088   \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it}{up}
1089   \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it}{up}
1090 }

```

9.1.2 Italic: \mathit

```

1091 \cs_new:Npn \um_config_mathit_Latin: {

```

```

1092 \bool_if:NTF \g_um_literal_bool {
1093   \um_map_chars_Latin:nn {it} {it}
1094 }{
1095   \bool_if:NF \g_um_upLatin_bool {
1096     \um_map_chars_Latin:nn {up,it} {it}
1097   }
1098 }
1099 \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{it}
1100 }
1101 \cs_new:Npn \um_config_mathit_latin: {
1102   \bool_if:NTF \g_um_literal_bool {
1103     \um_map_chars_latin:nn {it} {it}
1104     \um_map_char:nn {\g_um_it_h_usv}{\g_um_it_h_usv}% KEEP
1105   }{
1106     \bool_if:NF \g_um_uplatin_bool {
1107       \um_map_chars_latin:nn {up,it} {it}
1108       \um_map_char:nn {\g_um_up_h_usv,\g_um_it_h_usv}{\g_um_it_h_usv}% KEEP
1109     }
1110   }
1111   \um_set_mathalphabet_latin:Nnn \mathit {up,it}{it}
1112 }
1113 \cs_new:Npn \um_config_mathit_greek: {
1114   \bool_if:NTF \g_um_literal_bool {
1115     \um_map_chars_greek:nn {it}{it}
1116   }{
1117     \bool_if:NF \g_um_upgreek_bool {
1118       \um_map_chars_greek:nn {up,it}{it}
1119     }
1120   }
1121   \um_set_mathalphabet_greek:Nnn \mathit {up,it}{it}
1122 }
1123 \cs_new:Npn \um_config_mathit_greek: {
1124   \bool_if:NTF \g_um_literal_bool {
1125     \um_map_chars_greek:nn {it} {it}
1126   }{
1127     \bool_if:NF \g_um_upgreek_bool {
1128       \um_map_chars_greek:nn {it,up} {it}
1129     }
1130   }
1131   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {it}
1132 }
1133 \cs_new:Npn \um_config_mathit_misc: {
1134   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{it}
1135   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{it}
1136 }

```

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

```

1137 \cs_new:Npn \um_config_mathbb_latin: {
1138   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{bb}
1139 }
1140 \cs_new:Npn \um_config_mathbb_Latin: {
1141   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{bb}
1142   \um_set_mathalphabet_char:Nnn{\mathbb}{`C,"1D436}{`"2102}
1143   \um_set_mathalphabet_char:Nnn{\mathbb}{`H,"1D43B}{`"210D}
1144   \um_set_mathalphabet_char:Nnn{\mathbb}{`N,"1D441}{`"2115}
1145   \um_set_mathalphabet_char:Nnn{\mathbb}{`P,"1D443}{`"2119}
1146   \um_set_mathalphabet_char:Nnn{\mathbb}{`Q,"1D444}{`"211A}
1147   \um_set_mathalphabet_char:Nnn{\mathbb}{`R,"1D445}{`"211D}
1148   \um_set_mathalphabet_char:Nnn{\mathbb}{`Z,"1D44D} {"2124}
1149 }
1150 \cs_new:Npn \um_config_mathbb_num: {
1151   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{bb}
1152 }
1153 \cs_new:Npn \um_config_mathbb_misc: {
1154   \um_set_mathalphabet_char:Nnn \mathbb {"03A0,"1D6F1}{`"213F} % Pi
1155   \um_set_mathalphabet_char:Nnn \mathbb {"03C0,"1D70B}{`"213C} % pi
1156   \um_set_mathalphabet_char:Nnn \mathbb {"0393,"1D6E4}{`"213E} % Gamma
1157   \um_set_mathalphabet_char:Nnn \mathbb {"03B3,"1D6FE}{`"213D} % gamma
1158   \um_set_mathalphabet_char:Nnn \mathbb {"2211}{`"2140} % summation
1159 }
1160 \cs_new:Npn \um_config_mathbbi_misc: {
1161   \um_set_mathalphabet_char:Nnn \mathbbi {"D,"1D437}{`"2145}
1162   \um_set_mathalphabet_char:Nnn \mathbbi {"d,"1D451}{`"2146}
1163   \um_set_mathalphabet_char:Nnn \mathbbi {"e,"1D452}{`"2147}
1164   \um_set_mathalphabet_char:Nnn \mathbbi {"i,"1D456}{`"2148}
1165   \um_set_mathalphabet_char:Nnn \mathbbi {"j,"1D457}{`"2149}
1166 }

```

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

```

1167 \cs_new:Npn \um_config_mathscr_Latin: {
1168   \um_set_mathalphabet_Latin:Nnn \mathscr {up,it}{scr}
1169   \um_set_mathalphabet_char:Nnn \mathscr {"B,"1D435}{`"212C}
1170   \um_set_mathalphabet_char:Nnn \mathscr {"E,"1D438}{`"2130}
1171   \um_set_mathalphabet_char:Nnn \mathscr {"F,"1D439}{`"2131}
1172   \um_set_mathalphabet_char:Nnn \mathscr {"H,"1D43B}{`"210B}
1173   \um_set_mathalphabet_char:Nnn \mathscr {"I,"1D43C}{`"2110}
1174   \um_set_mathalphabet_char:Nnn \mathscr {"L,"1D43F}{`"2112}
1175   \um_set_mathalphabet_char:Nnn \mathscr {"M,"1D440}{`"2133}
1176   \um_set_mathalphabet_char:Nnn \mathscr {"R,"1D445}{`"211B}
1177 }
1178 \cs_new:Npn \um_config_mathscr_latin: {
1179   \um_set_mathalphabet_latin:Nnn \mathscr {up,it}{scr}

```

```

1180 \um_set_mathalphabet_char:Nnn \mathscr {\`e,"1D452}{ "212F}
1181 \um_set_mathalphabet_char:Nnn \mathscr {\`g,"1D454}{ "210A}
1182 \um_set_mathalphabet_char:Nnn \mathscr {\`o,"1D45C}{ "2134}
1183 }

```

9.1.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1184 \cs_new:Npn \um_config_mathfrak_Latin: {
1185   \um_set_mathalphabet_Latin:Nnn \mathfrak {up,it}{frak}
1186   \um_set_mathalphabet_char:Nnn \mathfrak {\`C,"1D436}{ "212D}
1187   \um_set_mathalphabet_char:Nnn \mathfrak {\`H,"1D43B}{ "210C}
1188   \um_set_mathalphabet_char:Nnn \mathfrak {\`I,"1D43C}{ "2111}
1189   \um_set_mathalphabet_char:Nnn \mathfrak {\`R,"1D445}{ "211C}
1190   \um_set_mathalphabet_char:Nnn \mathfrak {\`Z,"1D44D}{ "2128}
1191 }
1192 \cs_new:Npn \um_config_mathfrak_latin: {
1193   \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{frak}
1194 }

```

9.1.6 Sans serif upright: `\mathsfup`

```

1195 \cs_new:Npn \um_config_mathsfup_num: {
1196   \um_set_mathalphabet_numbers:Nnn \mathsf {up}{sf}
1197   \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{sf}
1198 }
1199 \cs_new:Npn \um_config_mathsfup_Latin: {
1200   \bool_if:NTF \g_um_sfliteral_bool {
1201     \um_map_chars_Latin:nn {sfup} {sfup}
1202     \um_set_mathalphabet_Latin:Nnn \mathsf {up}{sfup}
1203   }{
1204     \bool_if:NT \g_um_upsans_bool {
1205       \um_map_chars_Latin:nn {sfup,sfit} {sfup}
1206       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{sfup}
1207     }
1208   }
1209   \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{sfup}
1210 }
1211 \cs_new:Npn \um_config_mathsfup_latin: {
1212   \bool_if:NTF \g_um_sfliteral_bool {
1213     \um_map_chars_latin:nn {sfup} {sfup}
1214     \um_set_mathalphabet_latin:Nnn \mathsf {up}{sfup}
1215   }{
1216     \bool_if:NT \g_um_upsans_bool {
1217       \um_map_chars_latin:nn {sfup,sfit} {sfup}
1218       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{sfup}
1219     }
1220   }
1221   \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{sfup}
1222 }

```


9.1.7 Sans serif italic: `\mathsf{it}`

```
1223 \cs_new:Npn \um_config_mathsf_Latin: {  
1224   \bool_if:NTF \g_um_sfliteral_bool {  
1225     \um_map_chars_Latin:nn {sf} {sf}  
1226     \um_set_mathalphabet_Latin:Nnn \mathsf {it}{sf}  
1227   }{  
1228     \bool_if:NF \g_um_upsans_bool {  
1229       \um_map_chars_Latin:nn {sfup,sf} {sf}  
1230       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{sf}  
1231     }  
1232   }  
1233   \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{sf}  
1234 }  
1235 \cs_new:Npn \um_config_mathsf_latin: {  
1236   \bool_if:NTF \g_um_sfliteral_bool {  
1237     \um_map_chars_latin:nn {sf} {sf}  
1238     \um_set_mathalphabet_latin:Nnn \mathsf {it}{sf}  
1239   }{  
1240     \bool_if:NF \g_um_upsans_bool {  
1241       \um_map_chars_latin:nn {sfup,sf} {sf}  
1242       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{sf}  
1243     }  
1244   }  
1245   \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{sf}  
1246 }
```

9.1.8 Typewriter or monospaced: `\mathtt`

```
1247 \cs_new:Npn \um_config_mathtt_num: {  
1248   \um_set_mathalphabet_numbers:Nnn \mathtt {up}{tt}  
1249 }  
1250 \cs_new:Npn \um_config_mathtt_Latin: {  
1251   \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{tt}  
1252 }  
1253 \cs_new:Npn \um_config_mathtt_latin: {  
1254   \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{tt}  
1255 }
```

9.1.9 Bold Italic: `\mathbf{it}`

```
1256 \cs_new:Npn \um_config_mathbf_Latin: {  
1257   \bool_if:NF \g_um_bfup_Latin_bool {  
1258     \um_map_chars_Latin:nn {bfup,bf} {bfup}  
1259   }  
1260   \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{bf}  
1261   \bool_if:NTF \g_um_bfliteral_bool {  
1262     \um_map_chars_Latin:nn {bf} {bf}  
1263     \um_set_mathalphabet_Latin:Nnn \mathbf {it}{bf}
```

```

1264 }{
1265   \bool_if:NF \g_um_bfupLatin_bool {
1266     \um_map_chars_Latin:nn {bfup,bfit} {bfit}
1267     \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{bfit}
1268   }
1269 }
1270 }
1271 \cs_new:Npn \um_config_mathbfit_latin: {
1272   \bool_if:NF \g_um_bfuplatin_bool {
1273     \um_map_chars_latin:nn {bfup,bfit} {bfit}
1274   }
1275   \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{bfit}
1276   \bool_if:NTF \g_um_bfliteral_bool {
1277     \um_map_chars_latin:nn {bfit} {bfit}
1278     \um_set_mathalphabet_latin:Nnn \mathbf {it}{bfit}
1279   }{
1280     \bool_if:NF \g_um_bfuplatin_bool {
1281       \um_map_chars_latin:nn {bfup,bfit} {bfit}
1282       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{bfit}
1283     }
1284   }
1285 }
1286 \cs_new:Npn \um_config_mathbfit_Greek: {
1287   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{bfit}
1288   \bool_if:NTF \g_um_bfliteral_bool {
1289     \um_map_chars_Greek:nn {bfit}{bfit}
1290     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{bfit}
1291   }{
1292     \bool_if:NF \g_um_bfupgreek_bool {
1293       \um_map_chars_Greek:nn {bfup,bfit}{bfit}
1294       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{bfit}
1295     }
1296   }
1297 }
1298 \cs_new:Npn \um_config_mathbfit_greek: {
1299   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {bfit}
1300   \bool_if:NTF \g_um_bfliteral_bool {
1301     \um_map_chars_greek:nn {bfit} {bfit}
1302     \um_set_mathalphabet_greek:Nnn \mathbfit {it} {bfit}
1303   }{
1304     \bool_if:NF \g_um_bfupgreek_bool {
1305       \um_map_chars_greek:nn {bfit,bfup} {bfit}
1306     }
1307     \bool_if:NF \g_um_bfupgreek_bool {
1308       \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {bfit}
1309     }

```

```

1310 }
1311 }
1312 \cs_new:Npn \um_config_mathbfit_misc: {
1313   \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{bfit}
1314   \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{bfit}
1315   \bool_if:NTF \g_um_bfliteral_bool {
1316     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{bfit}
1317     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {it}{bfit}
1318   }{
1319     \bool_if:NF \g_um_upNabla_bool {
1320       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{bfit}
1321     }
1322     \bool_if:NF \g_um_uppartial_bool {
1323       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{bfit}
1324     }
1325   }
1326 }

```

9.1.10 Bold Upright: `\mathbfup`

```

1327 \cs_new:Npn \um_config_mathbfup_num: {
1328   \um_set_mathalphabet_numbers:Nnn \mathbf {up}{bfup}
1329   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{bfup}
1330 }
1331 \cs_new:Npn \um_config_mathbfup_Latin: {
1332   \bool_if:NT \g_um_bfupLatin_bool {
1333     \um_map_chars_Latin:nn {bfup,bfit} {bfit}
1334   }
1335   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{bfup}
1336   \bool_if:NTF \g_um_bfliteral_bool {
1337     \um_map_chars_Latin:nn {bfup} {bfup}
1338     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{bfup}
1339   }{
1340     \bool_if:NT \g_um_bfupLatin_bool {
1341       \um_map_chars_Latin:nn {bfup,bfit} {bfup}
1342       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{bfup}
1343     }
1344   }
1345 }
1346 \cs_new:Npn \um_config_mathbfup_latin: {
1347   \bool_if:NT \g_um_bfuplatin_bool {
1348     \um_map_chars_latin:nn {bfup,bfit} {bfup}
1349   }
1350   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{bfup}
1351   \bool_if:NTF \g_um_bfliteral_bool {
1352     \um_map_chars_latin:nn {bfup} {bfup}
1353     \um_set_mathalphabet_latin:Nnn \mathbf {up}{bfup}

```

```

1354 }{
1355   \bool_if:NT \g_um_bfuplatin_bool {
1356     \um_map_chars_latin:nn {bfup,bfit} {bfup}
1357     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{bfup}
1358   }
1359 }
1360 }
1361 \cs_new:Npn \um_config_mathbfup_Greek: {
1362   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{bfup}
1363   \bool_if:NTF \g_um_bfliteral_bool {
1364     \um_map_chars_Greek:nn {bfup}{bfup}
1365     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{bfup}
1366   }{
1367     \bool_if:NF \g_um_bfupGreek_bool {
1368       \um_map_chars_Greek:nn {bfup,bfit}{bfup}
1369       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{bfup}
1370     }
1371   }
1372 }
1373 \cs_new:Npn \um_config_mathbfup_greek: {
1374   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {bfup}
1375   \bool_if:NTF \g_um_bfliteral_bool {
1376     \um_map_chars_greek:nn {bfup} {bfup}
1377     \um_set_mathalphabet_greek:Nnn \mathbf {up} {bfup}
1378   }{
1379     \bool_if:NT \g_um_bfupgreek_bool {
1380       \um_map_chars_greek:nn {bfup,bfit} {bfup}
1381     }
1382     \bool_if:NT \g_um_bfupgreek_bool {
1383       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {bfup}
1384     }
1385   }
1386 }
1387 \cs_new:Npn \um_config_mathbfup_misc: {
1388   \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{bfup}
1389   \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{bfup}
1390   \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{bfup}
1391   \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{bfup}
1392   \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{bfup}
1393   \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{bfup}
1394   \bool_if:NTF \g_um_bfliteral_bool {
1395     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{bfup}
1396     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{bfup}
1397   }{
1398     \bool_if:NT \g_um_upNabla_bool {
1399       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{bfup}

```

```

1400     }
1401     \bool_if:NT \g_um_uppartial_bool {
1402       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{bfup}
1403     }
1404   }
1405 }

```

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

```

1406 \cs_new:Npn \um_config_mathbffrak_Latin: {
1407   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{bffrak}
1408 }
1409 \cs_new:Npn \um_config_mathbffrak_latin: {
1410   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{bffrak}
1411 }

```

9.1.12 Bold script or calligraphic: `\mathbfscr`

```

1412 \cs_new:Npn \um_config_mathbfscr_Latin: {
1413   \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{bfscr}
1414 }
1415 \cs_new:Npn \um_config_mathbfscr_latin: {
1416   \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{bfscr}
1417 }

```

9.1.13 Bold upright sans serif: `\mathbfsfup`

```

1418 \cs_new:Npn \um_config_mathbfsfup_num: {
1419   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{bfsfup}
1420   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{bfsfup}
1421 }
1422 \cs_new:Npn \um_config_mathbfsfup_Latin: {
1423   \bool_if:NTF \g_um_sfliteral_bool {
1424     \um_map_chars_Latin:nn {bfsfup} {bfsfup}
1425     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{bfsfup}
1426   }{
1427     \bool_if:NT \g_um_upsans_bool {
1428       \um_map_chars_Latin:nn {bfsfup,bfsfit} {bfsfup}
1429       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{bfsfup}
1430     }
1431   }
1432   \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{bfsfup}
1433 }
1434 \cs_new:Npn \um_config_mathbfsfup_latin: {
1435   \bool_if:NTF \g_um_sfliteral_bool {
1436     \um_map_chars_latin:nn {bfsfup} {bfsfup}
1437     \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{bfsfup}
1438   }{
1439     \bool_if:NT \g_um_upsans_bool {

```

```

1440     \um_map_chars_latin:nn {bfsfup,bfsfit} {bfsfup}
1441     \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{bfsfup}
1442   }
1443 }
1444 \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{bfsfup}
1445 }
1446 \cs_new:Npn \um_config_mathbfsfup_Greek: {
1447   \bool_if:NTF \g_um_sfliteral_bool {
1448     \um_map_chars_Greek:nn {bfsfup}{bfsfup}
1449     \um_set_mathalphabet_Greek:Nnn \mathbfsf {up}{bfsfup}
1450   }{
1451     \bool_if:NT \g_um_upsans_bool {
1452       \um_map_chars_Greek:nn {bfsfup,bfsfit}{bfsfup}
1453       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{bfsfup}
1454     }
1455   }
1456   \um_set_mathalphabet_Greek:Nnn \mathbfsfup {up,it}{bfsfup}
1457 }
1458 \cs_new:Npn \um_config_mathbfsfup_greek: {
1459   \bool_if:NTF \g_um_sfliteral_bool {
1460     \um_map_chars_greek:nn {bfsfup} {bfsfup}
1461     \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {bfsfup}
1462   }{
1463     \bool_if:NT \g_um_upsans_bool {
1464       \um_map_chars_greek:nn {bfsfup,bfsfit} {bfsfup}
1465       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {bfsfup}
1466     }
1467   }
1468   \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {bfsfup}
1469 }
1470 \cs_new:Npn \um_config_mathbfsfup_misc: {
1471   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{bfsfup}
1472   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{bfsfup}
1473   \bool_if:NTF \g_um_sfliteral_bool {
1474     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{bfsfup}
1475     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{bfsfup}
1476   }{
1477     \bool_if:NT \g_um_upNabla_bool {
1478       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{bfsfup}
1479     }
1480     \bool_if:NT \g_um_uppartial_bool {
1481       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{bfsfup}
1482     }
1483   }
1484 }

```

9.1.14 Bold italic sans serif: `\mathbfsfit`

```
1485 \cs_new:Npn \um_config_mathbfsfit_Latin: {
1486   \bool_if:NTF \g_um_sfliteral_bool {
1487     \um_map_chars_Latin:nn {bfsfit} {bfsfit}
1488     \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{bfsfit}
1489   }{
1490     \bool_if:NF \g_um_upsans_bool {
1491       \um_map_chars_Latin:nn {bfsfup,bfsfit} {bfsfit}
1492       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{bfsfit}
1493     }
1494   }
1495   \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{bfsfit}
1496 }
1497 \cs_new:Npn \um_config_mathbfsfit_latin: {
1498   \bool_if:NTF \g_um_sfliteral_bool {
1499     \um_map_chars_latin:nn {bfsfit} {bfsfit}
1500     \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{bfsfit}
1501   }{
1502     \bool_if:NF \g_um_upsans_bool {
1503       \um_map_chars_latin:nn {bfsfup,bfsfit} {bfsfit}
1504       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{bfsfit}
1505     }
1506   }
1507   \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{bfsfit}
1508 }
1509 \cs_new:Npn \um_config_mathbfsfit_Greek: {
1510   \bool_if:NTF \g_um_sfliteral_bool {
1511     \um_map_chars_Greek:nn {bfsfit}{bfsfit}
1512     \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{bfsfit}
1513   }{
1514     \bool_if:NF \g_um_upsans_bool {
1515       \um_map_chars_Greek:nn {bfsfup,bfsfit}{bfsfit}
1516       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{bfsfit}
1517     }
1518   }
1519   \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{bfsfit}
1520 }
1521 \cs_new:Npn \um_config_mathbfsfit_greek: {
1522   \bool_if:NTF \g_um_sfliteral_bool {
1523     \um_map_chars_greek:nn {bfsfit} {bfsfit}
1524     \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {bfsfit}
1525   }{
1526     \bool_if:NF \g_um_upsans_bool {
1527       \um_map_chars_greek:nn {bfsfup,bfsfit} {bfsfit}
1528       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {bfsfit}
1529     }
1530   }
```

```

1530 }
1531 \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {bfsfit}
1532 }
1533 \cs_new:Npn \um_config_mathbfsfit_misc: {
1534   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{bfsfit}
1535   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{bfsfit}
1536   \bool_if:NTF \g_um_sfliteral_bool {
1537     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{bfsfit}
1538     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{bfsfit}
1539   }{
1540     \bool_if:NF \g_um_upNabla_bool {
1541       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{bfsfit}
1542     }
1543     \bool_if:NF \g_um_uppartial_bool {
1544       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{bfsfit}
1545     }
1546   }
1547 }

```

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 \TeX ‘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.

```

1548 \begingroup
1549   \char_make_other:N \^
1550   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
1551     \lowercase{
1552       \scantokens{\global\let#1=^^^#2}
1553     }
1554   }

```

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

```

1555 \gdef\um@scanactivedef"#1\@nil#2{
1556   \lowercase{
1557     \tl_rescan:nn{
1558       \ExplSyntaxOn
1559       \char_make_math_superscript:N\^

```



```

1560     }{
1561     \global\def^^^^^#1{#2}
1562     }
1563   }
1564 }
1565 \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`.

```

1566 \begingroup
1567 \def\UnicodeMathSymbol#1#2#3#4{
1568   \um@scancharlet#2=#1\@nil
1569 }
1570 \char_make_other:N \#
1571 \@input{unicode-math-table.tex}
1572 \endgroup

```

Fix `\backslash`:

```

1573 \group_begin:
1574 \lccode`*=`\\
1575 \char_make_escape:N \\
1576 \char_make_other:N \\
1577 |lowercase{
1578 |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 (\sprime): 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 super-scripted. 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, \sprime, 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.

```

1579 \muskip_new:N \g_um_primekern_muskip
1580 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
1581 \num_new:N \l_um_primecount_num

1582 \cs_new:Nn \um_nprimes:n {
1583   ^{
1584     \sprime
1585     \prg_replicate:nn {#1-1} { \mskip \g_um_primekern_muskip \sprime }
1586   }
1587 }
1588 \cs_new:Nn \um_nprimes_select:n {
1589   \prg_case_int:nnn {#1}{
1590     {1} { ^{\sprime} }
1591     {2} {
1592       \um_glyph_if_exist:nTF {"2033} { ^{\dprime} } {\um_nprimes:n {#1}}
1593     }
1594     {3} {
1595       \um_glyph_if_exist:nTF {"2034} { ^{\trprime} } {\um_nprimes:n {#1}}
1596     }
1597     {4} {
1598       \um_glyph_if_exist:nTF {"2057} { ^{\qprime} } {\um_nprimes:n {#1}}
1599     }
1600   }{
1601     \um_nprimes:n {#1}
1602   }
1603 }
```

Scanning is more annoying than you'd think because we want to support all three of `\prime`, `'`, and the unicode prime. And `\ifx` doesn't work with mathactive chars.

```

1604 \cs_new:Nn \um_scanprime: {
1605   \num_zero:N \l_um_primecount_num
1606   \um_scanprime_collect:
1607 }
1608 \cs_new:Nn \um_scanprime_collect: {
1609   \num_incr:N \l_um_primecount_num
1610   \peek_meaning_remove:NTF ' {
1611     \um_scanprime_collect:
1612   }{
1613     \peek_meaning_remove:NTF \um_scanprime: {
1614       \um_scanprime_collect:
1615     }{
1616       \peek_meaning_remove:NTF ^^^2032 {
1617         \um_scanprime_collect:
1618       }{
1619         \um_nprimes_select:n {\l_um_primecount_num}
1620       }
1621     }
1622   }
1623 }

1624 \cs_set_eq:NN \prime \um_scanprime:
1625 \group_begin:
1626   \char_make_active:N \'
1627   \char_make_active:n {"2032}
1628   \cs_gset_eq:NN ' \um_scanprime:
1629   \cs_gset_eq:NN ^^^2032 \um_scanprime:
1630 \group_end:

```

11.0.16 Unicode radicals

Undo the damage made to `\sqrt`:

```

1631 \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`.

```

1632 \def\r@@t#1#2{
1633   \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
1634   \um@scaled@apply{#1}{\kern}{\fontdimen63\l_um_font}
1635   \raise \dimexpr(
1636     \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-

```

```

1637     \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
1638   )\relax
1639   \copy \rootbox
1640   \um@scaled@apply{#1}{\kern}{\fontdimen64\l_um_font}
1641   \box \z@
1642 }

```

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 Xe_{La}TeX 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:

```

1643 \prop_new:N \g_um_supers_prop
1644 \prop_new:N \g_um_subs_prop
1645
1646 \group_begin:
1647
1648 % Populate a property list with superscript characters; their mean-
1649 % ing as their key,
1650 % for reasons that will become apparent soon, and their replace-
1651 % ment as each key's value.
1652 % Then make the superscript active and bind it to the scanning function.
1653 %
1654 % \cs{scantokens} makes this process much simpler since we can acti-
1655 % vate the char
1656 % and assign its meaning in one step.
1657 \cs_set:Nn \um_setup_active_superscript:nn {
1658   \prop_gput:Nxn \g_um_supers_prop {\meaning #1} {#2}
1659   \char_make_active:n {`#1}
1660   \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1661   \scantokens{
1662     \cs_gset:Npn #1 {
1663       \tl_set:Nn \l_um_ss_chain_tl {#2}
1664       \cs_set_eq:NN \um_sub_or_super:n \sp
1665       \tl_set:Nn \l_um_tmpa_tl {supers}
1666       \um_scan_sscript:
1667     }
1668   }
1669 }

```

```

1668 \um_setup_active_superscript:nn {^^^^2070} {0}
1669 \um_setup_active_superscript:nn {^^^^00b9} {1}
1670 \um_setup_active_superscript:nn {^^^^00b2} {2}
1671 \um_setup_active_superscript:nn {^^^^00b3} {3}
1672 \um_setup_active_superscript:nn {^^^^2074} {4}
1673 \um_setup_active_superscript:nn {^^^^2075} {5}
1674 \um_setup_active_superscript:nn {^^^^2076} {6}
1675 \um_setup_active_superscript:nn {^^^^2077} {7}
1676 \um_setup_active_superscript:nn {^^^^2078} {8}
1677 \um_setup_active_superscript:nn {^^^^2079} {9}
1678 \um_setup_active_superscript:nn {^^^^207a} {+}
1679 \um_setup_active_superscript:nn {^^^^207b} {-}
1680 \um_setup_active_superscript:nn {^^^^207c} {=}
1681 \um_setup_active_superscript:nn {^^^^207d} {(}
1682 \um_setup_active_superscript:nn {^^^^207e} {)}
1683 \um_setup_active_superscript:nn {^^^^2071} {i}
1684 \um_setup_active_superscript:nn {^^^^207f} {n}
1685
1686 % Ditto above.
1687 \cs_set:Nn \um_setup_active_subscript:nn {
1688   \prop_gput:Nxn \g_um_subs_prop {\meaning #1} {#2}
1689   \char_make_active:n {\#1}
1690   \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1691   \scantokens{
1692     \cs_gset:Npn #1 {
1693       \tl_set:Nn \l_um_ss_chain_tl {#2}
1694       \cs_set_eq:NN \um_sub_or_super:n \sb
1695       \tl_set:Nn \l_um_tmpa_tl {subs}
1696       \um_scan_sscript:
1697     }
1698   }
1699 }
1700
1701 \um_setup_active_subscript:nn {^^^^2080} {0}
1702 \um_setup_active_subscript:nn {^^^^2081} {1}
1703 \um_setup_active_subscript:nn {^^^^2082} {2}
1704 \um_setup_active_subscript:nn {^^^^2083} {3}
1705 \um_setup_active_subscript:nn {^^^^2084} {4}
1706 \um_setup_active_subscript:nn {^^^^2085} {5}
1707 \um_setup_active_subscript:nn {^^^^2086} {6}
1708 \um_setup_active_subscript:nn {^^^^2087} {7}
1709 \um_setup_active_subscript:nn {^^^^2088} {8}
1710 \um_setup_active_subscript:nn {^^^^2089} {9}
1711 \um_setup_active_subscript:nn {^^^^208a} {+}
1712 \um_setup_active_subscript:nn {^^^^208b} {-}
1713 \um_setup_active_subscript:nn {^^^^208c} {=}

```

```

1714 \um_setup_active_subscript:nn {^^^208d} {(}
1715 \um_setup_active_subscript:nn {^^^208e} {)}
1716 \um_setup_active_subscript:nn {^^^2090} {a}
1717 \um_setup_active_subscript:nn {^^^2091} {e}
1718 \um_setup_active_subscript:nn {^^^1d62} {i}
1719 \um_setup_active_subscript:nn {^^^2092} {o}
1720 \um_setup_active_subscript:nn {^^^1d63} {r}
1721 \um_setup_active_subscript:nn {^^^1d64} {u}
1722 \um_setup_active_subscript:nn {^^^1d65} {v}
1723 \um_setup_active_subscript:nn {^^^2093} {x}
1724 \um_setup_active_subscript:nn {^^^1d66} {\beta}
1725 \um_setup_active_subscript:nn {^^^1d67} {\gamma}
1726 \um_setup_active_subscript:nn {^^^1d68} {\rho}
1727 \um_setup_active_subscript:nn {^^^1d69} {\phi}
1728 \um_setup_active_subscript:nn {^^^1d6a} {\chi}
1729
1730 \group_end:
1731
1732 % The scanning command, evident in its purpose:
1733 \cs_new:Nn \um_scan_sscript: {
1734   \um_scan_sscript:TF {
1735     \um_scan_sscript:
1736   }{
1737     \um_sub_or_super:n {\l_um_ss_chain_tl}
1738   }
1739 }
1740
1741 % The main theme here is stolen from the source to the vari-
1742   ous \cs{peek_} functions.
1743 % Consider this function as simply boilerplate:
1744 \cs_new:Nn \um_scan_sscript:TF {
1745   \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
1746   \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
1747   \tl_set:Nx \l_peek_false_tl { \exp_not:n{ \group_align_safe_end: #2 } }
1748   \group_align_safe_begin:
1749   \peek_after:NN \um_peek_execute_branches_ss:
1750 }
1751
1752 % We do not skip spaces when scanning ahead, and we explicitly wish to
1753 % bail out on encountering a space or a brace.
1754 \cs_new:Npn \um_peek_execute_branches_ss: {
1755   \bool_if:nTF {
1756     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
1757     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
1758     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
1759   }

```

```

1759 { \l_peek_false_tl }
1760 { \um_peek_execute_branches_ss_aux: }
1761 }
1762
1763 % This is the actual comparison code.
1764 % Because the peeking has already tokenised the next token,
1765 % it's too late to extract its charcode directly. Instead,
1766 % we look at its meaning, which remains a `character' even
1767 % though it is itself math-active. If the character is ever
1768 % made fully active, this will break our assumptions!
1769 %
1770 % If the char's meaning exists as a property list key, we
1771 % build up a chain of sub-/superscripts and iterate. (If not, exit and
1772 % typeset what we've already collected.)
1773 \cs_new:Nn \um_peek_execute_branches_ss_aux: {
1774   \prop_if_in:cxTF
1775     {g_um_\l_um_tmpa_tl _prop}
1776     {\meaning\l_peek_token}
1777   {
1778     \prop_get:cxN
1779       {g_um_\l_um_tmpa_tl _prop}
1780       {\meaning\l_peek_token}
1781       \l_um_tmpb_tl
1782       \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
1783       \l_peek_true_tl
1784     }
1785     {\l_peek_false_tl}
1786   }

```

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`:

```

1787 \def\operator@font{\um_setup_mathup:}
1788 \def\to{\rightarrow}
1789 \def\overrightarrow{\vec}
1790 \def\le{\leq}
1791 \def\ge{\geq}
1792 \def\neq{\neq}
1793 \def\triangle{\mathord{\bigtriangleup}}
1794 \def\bigcirc{\mdlgwhtcircle}
1795 \def\circ{\vysmwhtcircle}
1796 \def\mathyen{\yen}
1797 \def\mathsterling{\sterling}

```

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

```
1798 \@ifpackageloaded{amsmath}{
1799   % define their own colon, perhaps I should just steal it.
1800 }{
1801   \cs_set_protected:Npn \colon {
1802     \bool_if:NTF \g_um_literal_colon_bool {:\} { \mathpunct{:} }
1803   }
1804 }
```

`\mathcal`

```
1805 \def\mathcal{\mathscr}
```

`\mathrm`

```
1806 \def\mathrm{\mathup}
1807 \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:

```
1808 \bool_new:N \g_um_amsmath_bool
1809 \@ifpackageloaded{amsmath}{
1810   \bool_set_true:N \g_um_amsmath_bool
1811 }{
1812   \bool_set_false:N \g_um_amsmath_bool
1813 }
1814 \bool_if:NT \g_um_amsmath_bool {
1815   \tl_remove_in:Nn \@begindocumenthook {
1816     \mathchardef\std@minus\mathcode`\-\relax
1817     \mathchardef\std@equal\mathcode`\=\relax
1818   }
1819 }
```

- 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:

```
1820 \@ifpackageloaded{amsopn}{
1821   \cs_set:Npn \newmcodes@ {
1822     \mathcode`\'39
1823     \mathcode`\*42
1824     \mathcode`\."613A%
```



```

1825 % \ifnum\mathcode`\-=45 \else
1826 % \mathchardef\std@minus\mathcode`\-\relax
1827 % \fi
1828 \mathcode`\-45
1829 \mathcode`\-/47
1830 \mathcode`\:"603A\relax
1831 }
1832 }{}

```

- `\mathinner` items:

```

1833 \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
1834 \cs_set:Npn \cdots {\mathinner{\unicodcdots}}
1835 \bool_if:NT \g_um_amsmath_bool {
1836   \cs_set_eq:NN \@cdots \cdots
1837   \cs_set_eq:NN \dotso@ \cdots
1838 }

```

Octothorpe is an odd one:

```

1839 \AtBeginDocument{
1840   \def\#{\mode_if_math:TF{\mathoctothorpe}{\char` \#}}
1841   \def\widehat{\hat}
1842   \def\widetilde{\tilde}
1843 }

```

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

```

\Digamma 1844 \def\digamma{\updigamma}
1845 \def\Digamma{\upDigamma}

```

Overriding amsmath definitions:

```

1846 \AtBeginDocument{
1847   \def\@cdots{\mathinner{\cdots}}
1848 }

```

Interaction with beamer:

```

1849 \@ifclassloaded{beamer}{
1850   \ifbeamer@suppressreplacements\else
1851     \PackageWarningNoLine{unicode-math}{
1852       Disabling~ beamer's~ math~ setup.^{}
1853       Please~ load~ beamer~ with~ the~ [professionalfonts]~ class~ option
1854     }
1855     \beamer@suppressreplacementstrue
1856   \fi
1857 }{}

```

The end.

```

1858 \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.

```
1859 #!/bin/sh
1860
1861 cat stix-tbl.txt |
1862 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!)...

```
1863 {if (usv != substr($0,2,5) && substr($0,2,1) != " ")
1864     {usv = substr($0,2,5);
1865     texname = substr($0,84,25);
1866     class = substr($0,57,1);
1867     description = tolower(substr($0,233,350));
```

If the USV has a macro name, which isn't `\text...`, and isn't a single character macro (e.g., `\#`, `\S`, ...), and has a class, and it isn't reserved (*i.e.*, doubled up with a previously assigned glyph):

```
1868     if (texname ~ /[\\]/ &&
1869         substr(texname,0,5) != "\\text" &&
1870         substr(texname,0,4) != "\\ipa" &&
1871         substr(texname,0,5) != "\\tone" &&
1872         substr(texname,3,1) != " " &&
1873         class != " " &&
1874         description !~ /<reserved>/ )
```

Print the actual entry corresponding to the unicode character:

```
1875     print "\\UnicodeMathSymbol{" "\" \" \
1876         usv "}" "\" \
1877         texname "}" "\" \
1878         class "}" "\" \
1879         description "%";
1880     }}" - |
```

Now replace the STIX class abbreviations with their \TeX macro names.

```
1881 sed -e ' s/{N}/{\\mathord}/ ' \
```

A ‘fence’ defined by the STIX table is something like `\vert`; in \XeTeX this is just a `\mathord` that will grow with the magic of `\XeTeXmathchardef`.

```

1882 -e ' s/{F}/{\mathord}/ ' \
1883 -e ' s/{A}/{\mathalpha}/ ' \
1884 -e ' s/{D}/{\mathaccent}/ ' \
1885 -e ' s/{P}/{\mathpunct}/ ' \
1886 -e ' s/{B}/{\mathbin}/ ' \
1887 -e ' s/{R}/{\mathrel}/ ' \
1888 -e ' s/{L}/{\mathop}/ ' \
1889 -e ' s/{O}/{\mathopen}/ ' \
1890 -e ' s/{C}/{\mathclose}/ ' \

```

Fixing up a couple of things in the STIX table.

```

1891 -e ' s/{\^}/{\string^}/ ' > unicode-math.tex

```

A Documenting maths support in the NFSS

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

Maths symbol fonts Fonts for symbols: α , \leq , \rightarrow

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

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

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

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

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

`\DeclareSymbolFontAlphabet{<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>}\langle NFSS decl. \rangle`

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

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 \XeTeX . 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 \XeTeX .

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 \XeTeX math font dimensions

These are the extended `\fontdimens` available for suitable fonts in \XeTeX . 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%.

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

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

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

\fontdimen	Dimension name	Description
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.
51	SKEWEDFRACTION-HORIZONTALGAP	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SKEWEDFRACTIONVERTICAL-GAP	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OVERBARVERTICALGAP	Distance between the overbar and the (ink) top of the base. Suggested: 3×default rule thickness.
54	OVERBARRULETHICKNESS	Thickness of overbar. Suggested: default rule thickness.
55	OVERBAREXTRAASCENDER	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UNDERBARVERTICALGAP	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UNDERBARRULETHICKNESS	Thickness of underbar. Suggested: default rule thickness.
58	UNDERBAREXTRADESCENDER	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.

\fontdimen	Dimension name	Description
59	RADICALVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RADICALDISPLAYSTYLE- VERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
61	RADICALRULETHICKNESS	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RADICALEXTRAASCENDER	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.
63	RADICALKERNBEFOREDEGREE	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RADICALKERNAFTERDEGREE	Negative kern after the degree of a radical, if such is present. Suggested: -10/18 of em.
65	RADICALDEGREEBOTTOM- RAISEPERCENT	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.

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	
\"	1875
\#	1570, 1840
\'	1626, 1822
*	493, 1574, 1823
\-	492, 1816, 1825, 1826, 1828
\.	1824
\/	542, 1829
\:	495, 1830
\::	792
\::f	792
\::n	792
\<	546
\=	1817
\>	547
\@DeclareMathDelimiter	338
\@DeclareMathSizes	330
\@backslashchar	997, 1016
\@begindocumenthook	1815
\@cclvi	358
\@cdots	1836, 1847
\@empty	411, 679, 694, 715, 720
\@ifclassloaded	1849
\@ifnextchar	1631
\@ifpackageloaded	1798, 1809, 1820
\@ii	678, 679, 681, 683, 686, 691
\@input	475, 1571
\@marker	691, 710
\@nil	365, 523, 691, 704–707, 746, 1550, 1555, 1568
\@preamblecmds	342
\@sqrt	1631
\@tempa	167, 231, 257, 266, 284, 302, 310, 317, 637, 682, 706, 708, 720
\@tempb	167, 168, 231, 232, 257, 258, 266, 267, 284, 285, 302, 303, 310, 311, 317, 318, 637, 638, 709, 710, 715
\@tempswafalse	680
\@tempswattrue	684, 687, 712, 717, 722, 727
\@xDeclareMathDelimiter	339
\@xxDeclareMathDelimiter	338
\	1574, 1576, 1868–1871, 1875, 1881–1891
\^	1549, 1559, 1891
\	1575
Numbers	
\0	920, 1043
\9	920
_	1875–1878, 1881–1890
A	
\A	917
\a	916
\addnolimits	<u>603</u>
\addtoversion	331
\alloc@	358
\alpha@elt	332
\alpha@list	332
\AtBeginDocument	861, 1839, 1846
\awint	599
B	
\B	1169
\beamer@suppressreplacementstrue	1855
\begingroup	362, 1548, 1566
\beta	1724
\bgroup	1033
\bigcirc	1794
\bigtriangleup	1793
\bool_if:NF	494, 1095, 1106, 1117, 1127, 1228, 1240, 1257, 1265, 1272, 1280, 1292, 1304, 1307, 1319, 1322, 1367, 1490, 1502, 1514, 1526, 1540, 1543
\bool_if:NT	1050, 1060, 1071, 1081, 1204, 1216, 1332, 1340, 1347, 1355, 1379, 1382, 1398, 1401, 1427, 1439, 1451, 1463, 1477, 1480, 1814, 1835

\bool_if:NTF	83–90, 274, 292, 456, 497, 506, 614, 874, 877, 880, 883, 1047, 1057, 1068, 1078, 1092, 1102, 1114, 1124, 1200, 1212, 1224, 1236, 1261, 1276, 1288, 1300, 1315, 1336, 1351, 1363, 1375, 1394, 1423, 1435, 1447, 1459, 1473, 1486, 1498, 1510, 1522, 1536, 1802
\bool_if:nTF	1754
\bool_new:N ..	13–28, 254, 255, 309, 1808
\bool_set_false:N
	169–181, 184–186, 188, 192, 194, 195, 200, 208, 209, 222, 223, 228, 233–237, 239, 241, 245, 259, 270, 288, 304, 314, 444, 449, 642, 1812
\bool_set_true:N	183, 187, 189–191, 193, 197–199, 201–207, 211–221, 225–227, 240, 242, 243, 246–249, 251, 261, 263, 268, 286, 306, 312, 409, 442, 447, 639, 1810
\box	1641
C	
\C	1142, 1186
\c_group_begin_token	1755
\c_group_end_token	1756
\c_peek_true_remove_next_tl ...	1745
\c_space_token	1757
\cdots	1834, 1836, 1837, 1847
\cdp@elt	329
\cdp@list	329
\char	1840
\char_make_active:N	1626
\char_make_active:n	363, 1627, 1656, 1689
\char_make_escape:N	1575
\char_make_math_superscript:N ..	1559
\char_make_other:N ..	1549, 1570, 1576
\chardef	358
\chi	1728
\circ	1795
\cirfnint	599
\clist_map_break:	989
\clist_map_inline:Nn	590
\clist_map_inline:nn	528, 583, 645, 735, 757, 762, 767, 778, 820, 825, 830, 835, 841, 847, 863, 986, 992
\clist_map_variable:nNn	794, 800
\colon	1801
\copy	1639
\cs	1652, 1741
\cs_generate_variant:Nn ...	7–11, 789–791, 857–860, 1003, 1023–1026
\cs_gset:cpn	376, 391, 392
\cs_gset:cpx	369
\cs_gset:Npn ..	378, 384, 1550, 1659, 1692
\cs_gset:Npx	395
\cs_gset_eq:NN	1628, 1629
\cs_if_exist:cF	1028
\cs_if_exist:cT	1011
\cs_new:Nn
	491, 522, 527, 532, 535, 541, 576, 579, 741, 745, 793, 799, 809, 812, 815, 819, 824, 829, 834, 840, 846, 862, 928, 983, 1010, 1027, 1582, 1588, 1604, 1608, 1733, 1743, 1773
\cs_new:Npn .	1042, 1046, 1056, 1067, 1077, 1087, 1091, 1101, 1113, 1123, 1133, 1137, 1140, 1150, 1153, 1160, 1167, 1178, 1184, 1192, 1195, 1199, 1211, 1223, 1235, 1247, 1250, 1253, 1256, 1271, 1286, 1298, 1312, 1327, 1331, 1346, 1361, 1373, 1387, 1406, 1409, 1412, 1415, 1418, 1422, 1434, 1446, 1458, 1470, 1485, 1497, 1509, 1521, 1533, 1753
\cs_set:cpn	1029
\cs_set:Nn	273, 291, 360, 404, 483, 486, 582, 589, 734, 750, 753, 756, 761, 766, 777, 783, 786, 1015, 1654, 1687
\cs_set:Npn	669, 673, 792, 1821, 1833, 1834
\cs_set:Npx	420, 424
\cs_set_eq:cN	1018
\cs_set_eq:NN ...	459–463, 467–471, 967, 976, 1624, 1661, 1694, 1836, 1837
\cs_set_protected:cpn	1032
\cs_set_protected:Npn	1801
\cs_to_str:N	365, 368–370, 376, 391, 392, 584, 977, 984
\csname
	365, 368, 405, 412, 537, 577, 580, 1036

D	
<code>\D</code>	1161
<code>\d</code>	1162
<code>\DeclareDocumentCommand</code>	163, 407, 603, 606
<code>\DeclareMathAccent</code>	337
<code>\DeclareMathAlphabet</code>	336
<code>\DeclareMathDelimiter</code>	338
<code>\DeclareMathRadical</code>	339
<code>\DeclareMathSizes</code>	329
<code>\DeclareMathSymbol</code>	337
<code>\DeclareMathVersion</code>	331, 414
<code>\DeclareRobustCommand</code>	1631
<code>\DeclareSymbolFont</code>	334, 473
<code>\DeclareSymbolFontAlphabet</code>	340
<code>\DeclareSymbolFontAlphabet@</code>	340
<code>\def</code>	29–162, 344, 347, 358, 611, 704, 706–709, 1561, 1567, 1632, 1787–1797, 1805, 1806, 1840–1842, 1844, 1845, 1847
<code>\define@choicekey</code>	166, 231, 256, 266, 284, 302, 310, 317, 637
<code>\define@cmdkey</code>	630–633
<code>\define@key</code>	613
<code>\define@mathalphabet</code>	331
<code>\define@mathgroup</code>	331
<code>\Digamma</code>	<u>1844</u>
<code>\digamma</code>	<u>1844</u>
<code>\dimexpr</code>	345, 446, 1635
<code>\do</code>	342
<code>\dorestore@version</code>	333
<code>\dotsb@</code>	1837
<code>\dp</code>	1637
<code>\dprime</code>	1592
E	
<code>\E</code>	1170
<code>\e</code>	1163, 1180
<code>\edef</code>	705, 706
<code>\egroup</code>	1030, 1035
<code>\else</code>	350, 353, 373, 382, 387, 393, 396, 448, 685, 690, 696, 714, 719, 724, 1825, 1850
<code>\else:</code>	1021
<code>\encodingdefault</code>	474
<code>\endcsname</code>	365, 368, 405, 412, 537, 577, 580, 1036
<code>\endgroup</code>	366, 1565, 1572
<code>\epsilon</code>	873
<code>\errorcontextlines</code>	801
<code>\etex_iffontchar:D</code>	1021
<code>\ExecuteOptionsX</code>	326
<code>\exp_after:wN</code>	658, 661, 977, 985
<code>\exp_args:Nc</code>	484
<code>\exp_args:Nnff</code>	792, 795, 803
<code>\exp_args:NV</code>	988
<code>\exp_not:c</code>	370, 871
<code>\exp_not:N</code>	421, 425, 585
<code>\exp_not:n</code>	371, 1744, 1746
<code>\exp_not:V</code>	421, 647
<code>\expandafter</code>	367, 681, 683, 686, 691, 705, 706, 710, 1035, 1036
<code>\ExplSyntaxOff</code>	1858
<code>\ExplSyntaxOn</code>	6, 1558
F	
<code>\F</code>	1171
<code>\f@size</code>	412
<code>\fi</code>	229, 252, 264, 271, 289, 307, 315, 324, 355, 356, 398–402, 455, 593, 640, 688, 689, 692, 698, 700, 701, 713, 718, 723, 728–732, 1827, 1856
<code>\fi:</code>	1021
<code>\fint</code>	599
<code>\font</code>	445
<code>\fontdimen</code>	345, 446, 1634, 1640
<code>\fontname</code>	960
G	
<code>\g</code>	1181
<code>\g@addto@macro</code>	695, 697
<code>\g_um_amsmath_bool</code>	1808, 1810, 1812, 1814, 1835
<code>\g_um_bb_h_usv</code>	151
<code>\g_um_bb_Latin_usv</code>	39
<code>\g_um_bb_latin_usv</code>	40
<code>\g_um_bb_num_usv</code>	38
<code>\g_um_bf_Greek_usv</code>	89
<code>\g_um_bf_greek_usv</code>	90
<code>\g_um_bf_Latin_usv</code>	87
<code>\g_um_bf_latin_usv</code>	88
<code>\g_um_bf_num_usv</code>	57
<code>\g_um_bffrak_h_usv</code>	159
<code>\g_um_bffrak_Latin_usv</code>	68

\g_um_bffrak_latin_usv	69	\g_um_bfsfit_varTheta_usv	130
\g_um_bfit_Greek_usv	66, 89	\g_um_bfsfit_vartheta_usv	132
\g_um_bfit_greek_usv	67, 90	\g_um_bfsfNabla_up_or_it_usv	
\g_um_bfit_h_usv	156	277, 281, 517
\g_um_bfit_Latin_usv	64, 87	\g_um_bfsfpartial_up_or_it_usv ..	
\g_um_bfit_latin_usv	65, 88	295, 299, 519
\g_um_bfit_Nabla_usv .	140, 280, 508, 516	\g_um_bfsfup_Greek_usv	77, 85
\g_um_bfit_num_usv	59	\g_um_bfsfup_greek_usv	78, 86
\g_um_bfit_partial_usv	146, 298, 512, 518	\g_um_bfsfup_h_usv	161
\g_um_bfit_varepsilon_usv	117	\g_um_bfsfup_Latin_usv	75, 83
\g_um_bfit_varkappa_usv	119	\g_um_bfsfup_latin_usv	76, 84
\g_um_bfit_varphi_usv	120	\g_um_bfsfup_Nabla_usv	141, 277, 509, 517
\g_um_bfit_varpi_usv	122	\g_um_bfsfup_num_usv	73
\g_um_bfit_varrho_usv	121	\g_um_bfsfup_partial_usv	
\g_um_bfit_varTheta_usv	116	147, 295, 513, 519
\g_um_bfit_vartheta_usv	118	\g_um_bfsfup_varepsilon_usv	124
\g_um_bfliteral_bool		\g_um_bfsfup_varkappa_usv	126
.....	21, 226, 233, 239, 245,	\g_um_bfsfup_varphi_usv	127
	251, 506, 1261, 1276, 1288, 1300,	\g_um_bfsfup_varpi_usv	129
	1315, 1336, 1351, 1363, 1375, 1394	\g_um_bfsfup_varrho_usv	128
\g_um_bfNabla_up_or_it_usv		\g_um_bfsfup_varTheta_usv	123
.....	276, 280, 516	\g_um_bfsfup_vartheta_usv	125
\g_um_bfpartial_up_or_it_usv		\g_um_bfup_Digamma_usv	101
.....	294, 298, 518	\g_um_bfup_digamma_usv	108
\g_um_bfscr_h_usv	160	\g_um_bfup_Greek_usv	62, 89
\g_um_bfscr_Latin_usv	70	\g_um_bfup_greek_usv	63, 90
\g_um_bfscr_latin_usv	71	\g_um_bfup_h_usv	155
\g_um_bfsf_Greek_usv	85	\g_um_bfup_Latin_usv	60, 87
\g_um_bfsf_greek_usv	86	\g_um_bfup_latin_usv	61, 88
\g_um_bfsf_Latin_usv	83	\g_um_bfup_Nabla_usv .	139, 276, 507, 516
\g_um_bfsf_latin_usv	84	\g_um_bfup_num_usv	58
\g_um_bfsf_num_usv	72	\g_um_bfup_partial_usv	145, 294, 511, 518
\g_um_bfsfit_Greek_usv	81, 85	\g_um_bfup_varepsilon_usv	102
\g_um_bfsfit_greek_usv	82, 86	\g_um_bfup_varkappa_usv	104
\g_um_bfsfit_h_usv	162	\g_um_bfup_varphi_usv	105
\g_um_bfsfit_Latin_usv	79, 83	\g_um_bfup_varpi_usv	107
\g_um_bfsfit_latin_usv	80, 84	\g_um_bfup_varrho_usv	106
\g_um_bfsfit_Nabla_usv	142, 281, 510, 517	\g_um_bfup_varTheta_usv	100
\g_um_bfsfit_num_usv	74	\g_um_bfup_vartheta_usv	103
\g_um_bfsfit_partial_usv		\g_um_bfupGreek_bool	24, 89, 173, 187,
.....	148, 299, 514, 519		201, 215, 234, 240, 246, 1292, 1367
\g_um_bfsfit_varepsilon_usv	131	\g_um_bfupgreek_bool	
\g_um_bfsfit_varkappa_usv	133	25, 90, 174, 188, 202, 216,
\g_um_bfsfit_varphi_usv	134		235, 241, 247, 1304, 1307, 1379, 1382
\g_um_bfsfit_varpi_usv	136		
\g_um_bfsfit_varrho_usv	135		

<code>\g_um_bfupLatin_bool</code>		<code>\g_um_mathbfsfup_alph_clist</code>	913, 946
..... 22, 87, 175, 189, 203, 217,		<code>\g_um_mathbfup_alph_clist</code> ..	910, 941
236, 242, 248, 1257, 1265, 1332, 1340		<code>\g_um_mathfrak_alph_clist</code> ..	901, 935
<code>\g_um_bfuplatin_bool</code>		<code>\g_um_mathit_alph_clist</code>	899, 932
..... 23, 88, 176, 190, 204, 218,		<code>\g_um_mathit_Greek_usv</code>	925
237, 243, 249, 1272, 1280, 1347, 1355		<code>\g_um_mathit_greek_usv</code>	924
<code>\g_um_fam_int</code>	12, 465, 466	<code>\g_um_mathit_Latin_usv</code>	923
<code>\g_um_frac_h_usv</code>	154	<code>\g_um_mathit_latin_usv</code>	922
<code>\g_um_frac_Latin_usv</code>	43	<code>\g_um_mathscr_alph_clist</code> ...	900, 934
<code>\g_um_frac_latin_usv</code>	44	<code>\g_um_mathsf_alph_clist</code>	906, 936
<code>\g_um_it_Greek_usv</code>	36	<code>\g_um_mathsfitt_alph_clist</code> ..	908, 938
<code>\g_um_it_greek_usv</code>	37	<code>\g_um_mathsfup_alph_clist</code> ..	907, 937
<code>\g_um_it_h_usv</code>	150, 922, 1062, 1104, 1108	<code>\g_um_mathtt_alph_clist</code>	905, 939
<code>\g_um_it_Latin_usv</code>	34	<code>\g_um_mathup_alph_clist</code>	898, 931
<code>\g_um_it_latin_usv</code>	35	<code>\g_um_mathup_Greek_usv</code>	919
<code>\g_um_it_Nabla_usv</code> ..	138, 279, 499, 503	<code>\g_um_mathup_greek_usv</code>	918
<code>\g_um_it_partial_usv</code> ..	144, 297, 501, 504	<code>\g_um_mathup_Latin_usv</code>	917
<code>\g_um_it_varepsilon_usv</code>	110	<code>\g_um_mathup_latin_usv</code>	916
<code>\g_um_it_varkappa_usv</code>	112	<code>\g_um_mathup_num_usv</code>	920
<code>\g_um_it_varphi_usv</code>	113	<code>\g_um_Nabla_up_or_it_usv</code> ..	275, 279, 503
<code>\g_um_it_varpi_usv</code>	115	<code>\g_um_partial_up_or_it_usv</code>	
<code>\g_um_it_varrho_usv</code>	114 293, 297, 504	
<code>\g_um_it_varTheta_usv</code>	109	<code>\g_um_primekern_muskip</code> ..	1579, 1580, 1585
<code>\g_um_it_vartheta_usv</code>	111	<code>\g_um_scr_h_usv</code>	153
<code>\g_um_literal_bool</code> ..	16, 181, 195,	<code>\g_um_scr_Latin_usv</code>	41
209, 223, 225, 497, 1047, 1057,		<code>\g_um_scr_latin_usv</code>	42
1068, 1078, 1092, 1102, 1114, 1124		<code>\g_um_sf_Latin_usv</code>	49
<code>\g_um_literal_colon_bool</code>		<code>\g_um_sf_latin_usv</code>	51
..... 309, 312, 314, 494, 1802		<code>\g_um_sf_num_usv</code>	45
<code>\g_um_math_alphabet_name_Greek_tl</code>		<code>\g_um_sfit_h_usv</code>	158
..... 1008		<code>\g_um_sfit_Latin_usv</code>	52
<code>\g_um_math_alphabet_name_greek_tl</code>		<code>\g_um_sfit_latin_usv</code>	53
..... 1007		<code>\g_um_sfit_num_usv</code>	47
<code>\g_um_math_alphabet_name_Latin_tl</code>		<code>\g_um_sfliteral_bool</code>	
..... 1006		.. 227, 255, 263, 1200, 1212, 1224,	
<code>\g_um_math_alphabet_name_latin_tl</code>		1236, 1423, 1435, 1447, 1459,	
..... 1005		1473, 1486, 1498, 1510, 1522, 1536	
<code>\g_um_math_alphabet_name_num_tl</code> ..	1009	<code>\g_um_sfup_h_usv</code>	157
<code>\g_um_mathalph_seq</code>	634, 663, 886	<code>\g_um_sfup_Latin_usv</code>	48
<code>\g_um_mathbb_alph_clist</code>	904, 933	<code>\g_um_sfup_latin_usv</code>	50
<code>\g_um_mathbf_alph_clist</code>	909, 940	<code>\g_um_sfup_num_usv</code>	46
<code>\g_um_mathbffrac_alph_clist</code> ..	903, 944	<code>\g_um_slash_delimiter_usv</code>	
<code>\g_um_mathbfit_alph_clist</code> ..	911, 942 319, 321, 323, 542–544	
<code>\g_um_mathbfscr_alph_clist</code> ..	902, 943	<code>\g_um_subs_prop</code>	1644, 1688
<code>\g_um_mathbfsf_alph_clist</code> ..	912, 945	<code>\g_um_supers_prop</code>	1643, 1655
<code>\g_um_mathbfsfit_alph_clist</code> ..	914, 947		

\g_um_texgreek_bool	28, 180, 194, 208, 222, 228, 304, 306, 874, 877, 880, 883
\g_um_tt_h_usv	152
\g_um_tt_Latin_usv	55
\g_um_tt_latin_usv	56
\g_um_tt_num_usv	54
\g_um_up_Digamma_usv	92
\g_um_up_digamma_usv	99
\g_um_up_Greek_usv	32
\g_um_up_greek_usv	33
\g_um_up_h_usv	149, 1062, 1108
\g_um_up_Latin_usv	30
\g_um_up_latin_usv	31
\g_um_up_Nabla_usv	137, 275, 498, 503
\g_um_up_num_usv	29
\g_um_up_partial_usv	143, 293, 500, 504
\g_um_up_varepsilon_usv	93
\g_um_up_varkappa_usv	95
\g_um_up_varphi_usv	96
\g_um_up_varpi_usv	98
\g_um_up_varrho_usv	97
\g_um_up_varTheta_usv	91
\g_um_up_vartheta_usv	94
\g_um_upGreek_bool	19, 85, 169, 183, 197, 211, 1071, 1117
\g_um_upgreek_bool	20, 86, 170, 184, 198, 212, 1081, 1127
\g_um_upLatin_bool	17, 83, 171, 185, 199, 213, 1050, 1095
\g_um_uplatin_bool	18, 84, 172, 186, 200, 214, 1060, 1106
\g_um_upNabla_bool	26, 177, 191, 205, 219, 268, 270, 274, 1319, 1398, 1477, 1540
\g_um_uppartial_bool	27, 178, 192, 206, 220, 286, 288, 292, 1322, 1401, 1480, 1543
\g_um_upsans_bool	179, 193, 207, 221, 254, 259, 261, 1204, 1216, 1228, 1240, 1427, 1439, 1451, 1463, 1490, 1502, 1514, 1526
\gamma	1725
\gdef	1555
\ge	1791
\geq	1791
\get@cdp	335
\glb@currsiz	408
\global	364, 367, 379, 380, 385, 386, 389, 390, 397, 1552, 1561, 1657, 1690
\group@elt	334
\group@list	334
\group_align_safe_begin:	1747
\group_align_safe_end:	1746
\group_begin:	1573, 1625, 1646
\group_end:	1630, 1730
H	
\H	1143, 1172, 1187
\hat	1841
\hbox	1633
\ht	1636
I	
\I	1173, 1188
\i	1164
\if@tempswa	693
\ifbeamer@suppressreplacements	1850
\ifcase	168, 232, 258, 267, 285, 303, 311, 318, 638
\ifdim	446
\ifnum	591, 711, 716, 721, 725, 726, 1825
\ifx	348, 351, 361, 374, 383, 388, 394, 679, 682, 683, 686, 694, 710, 715, 720
\iiiint	597
\iiint	597
\iint	597
\init@restore@version	333
\int	597
\int_incr:N	465
\int_new:N	12
\int_use:N	466
\intBar	599
\intbar	599
\intcap	601
\intclockwise	598
\intcup	601
\intlarhk	600
\intx	600
J	
\j	1165
K	
\kern	1634, 1640

L	M
\L 1174	\M 1175
\l_peek_false_tl 1746, 1759, 1785	\m@th 1633
\l_peek_token ... 1755–1757, 1776, 1780	\mathaccent 394
\l_peek_true_aux_tl 1744	\mathalpha 585, 743, 746
\l_peek_true_tl 1745, 1783	\mathbb 889, 933, 1138, 1141–1148, 1151, 1154–1158
\l_um_char_range_seq 410, 636, 643, 649, 678	\mathbbbit 1161–1165
\l_um_font 349, 352, 445, 446, 960, 1021, 1634, 1636, 1637, 1640	\mathbf 891, 940, 1263, 1267, 1278, 1282, 1290, 1294, 1316, 1317, 1320, 1323, 1328, 1338, 1342, 1353, 1357, 1365, 1369, 1377, 1383, 1392, 1393, 1395, 1396, 1399, 1402
\l_um_fontspec_feature_bool 13, 442, 444, 614	\mathbffrak 892, 944, 1407, 1410
\l_um_inc_num 802, 804, 805	\mathbfit 891, 942, 1260, 1275, 1287, 1299, 1302, 1308, 1313, 1314
\l_um_init_bool .. 15, 409, 456, 639, 642	\mathbfscr 892, 943, 1413, 1416
\l_um_input_num 794, 796, 800, 804	\mathbfsf 893, 945, 1419, 1425, 1429, 1437, 1441, 1449, 1453, 1461, 1465, 1474, 1475, 1478, 1481, 1488, 1492, 1500, 1504, 1512, 1516, 1524, 1528, 1537, 1538, 1541, 1544
\l_um_mathalph_seq 635, 644, 647, 930, 968	\mathbfsfit 893, 947, 1495, 1507, 1519, 1531, 1534, 1535
\l_um_missing_alph_seq 927, 929, 957, 962, 996	\mathbfsfup 893, 946, 1420, 1432, 1444, 1456, 1468, 1471, 1472
\l_um_mversion_tf 413, 414	\mathbfup 891, 941, 1329, 1335, 1350, 1362, 1374, 1388–1391
\l_um_nolimits_tl ... 371, 596, 604, 607	\mathbin 492, 493
\l_um_ot_math_bool 14, 447, 449	\mathcal 1805
\l_um_primecount_num 1581, 1605, 1609, 1619	\mathchar@type 340, 368, 378, 380, 384, 386, 389, 391, 392, 395, 397, 405, 536
\l_um_radicals_tl 375, 609	\mathchardef 1816, 1817, 1826
\l_um_script_features_tl ... 415, 432	\mathclose 383, 392
\l_um_script_font_tl 417, 431	\mathcode 364, 1816, 1817, 1822–1826, 1828–1830
\l_um_ss_chain_tl 1660, 1693, 1737, 1782	\mathellipsis 1833
\l_um_sscript_features_tl .. 416, 436	\mathfence 388, 1807
\l_um_sscript_font_tl 418, 435	\mathfrak 889, 935, 1185–1190, 1193
\l_um_tmpa_tl 647, 654, 657, 658, 660, 661, 663, 670, 674, 969, 977, 979, 984, 985, 994, 998, 1662, 1695, 1775, 1779	\mathgroup 358
\l_um_tmpb_tl 647, 655, 675, 970, 975, 977, 979, 985, 987, 988, 993, 1781, 1782	\mathinner 1833, 1834, 1847
\l_um_tmpc_tl ... 647, 656, 671, 971, 972	\mathit 888, 932, 1099, 1111, 1121, 1131, 1134, 1135
\lccode 1574	\mathoctothorpe 1840
\le 1790	\mathop 361
\left 610	
\left@primitive 610, 611	
\leq 1790	
\let 359, 408, 411, 610, 1552, 1807	
\lowercase 1551, 1556	
\lowint 601	

<code>\mathopen</code>	374, 391, 611	<code>\num_incr:N</code>	1609
<code>\mathord</code>	498–501, 503, 504, 507–514, 516–519, 533, 1793, 1807	<code>\num_new:N</code>	1581
<code>\mathpunct</code>	1802	<code>\num_zero:N</code>	1605
<code>\mathrel</code>	495	<code>\number</code>	796, 804, 805
<code>\mathrm</code>	1806	<code>\numexpr</code>	716, 721, 725, 726, 743, 796, 804, 805
<code>\mathscr</code>	889, 934, 1168–1176, 1179–1182, 1805	O	
<code>\mathsf</code> ..	890, 936, 1196, 1202, 1206, 1214, 1218, 1226, 1230, 1238, 1242	<code>\o</code>	1182
<code>\mathsf{fit}</code>	890, 938, 1233, 1245	<code>\oiint</code>	597
<code>\mathsf{fup}</code>	890, 937, 1197, 1209, 1221	<code>\oint</code>	597
<code>\mathsterling</code>	1797	<code>\ointctrackwise</code>	598
<code>\mathtt</code>	889, 939, 1248, 1251, 1254	<code>\operator@font</code>	1787
<code>\mathup</code>	888, 931, 1044, 1054, 1065, 1075, 1085, 1088, 1089, 1806	<code>\or</code> .	182, 196, 210, 224, 238, 244, 250, 260, 262, 269, 287, 305, 313, 320, 322
<code>\mathyen</code>	1796	<code>\overrightarrow</code>	1789
<code>\mddefault</code>	474	P	
<code>\mdlgwhtcircle</code>	1794	<code>\P</code>	1145
<code>\meaning</code>	1655, 1688, 1776, 1780	<code>\PackageError</code>	617
<code>\mitepsilon</code>	874, 880	<code>\PackageInfo</code>	458
<code>\mitphi</code>	877, 883	<code>\PackageWarning</code>	973
<code>\mitvarepsilon</code>	874, 880	<code>\PackageWarningNoLine</code>	450, 1851
<code>\mitvarphi</code>	877, 883	<code>\peek_after:NN</code>	1748
<code>\mode_if_math:F</code>	1034	<code>\peek_meaning_remove:NTF</code>	1610, 1613, 1616
<code>\mode_if_math:TF</code>	1840	<code>\phi</code>	876, 1727
<code>\mskip</code>	1585	<code>\pointint</code>	600
<code>\muskip_gset:Nn</code>	1580	<code>\prg_case_int:nnn</code>	1589
<code>\muskip_new:N</code>	1579	<code>\prg_do_nothing:</code>	1018
N		<code>\prg_new_conditional:Nnn</code> ..	653, 1020
<code>\N</code>	1144	<code>\prg_replicate:nn</code>	1585
<code>\ne</code>	1792	<code>\prg_return_false:</code>	666, 1021
<code>\neq</code>	1792	<code>\prg_return_true:</code>	664, 1021
<code>\new@mathalphabet</code>	336	<code>\prg_stepwise_inline:nnnn</code>	736
<code>\new@mathgroup</code>	329, 358, 359	<code>\prg_stepwise_variable:nnnNn</code>	802
<code>\new@mathversion</code>	334	<code>\prime</code>	1624
<code>\new@symbolfont</code>	335	<code>\process@table</code>	333
<code>\newcommand</code>	612, 677	<code>\ProcessOptionsX</code>	327
<code>\newfam</code>	359	<code>\prop_get:cxN</code>	1778
<code>\newmathalphabet</code>	330	<code>\prop_get:NnN</code>	10
<code>\newmathalphabet@@</code>	330	<code>\prop_gput:Nnn</code>	9
<code>\newmathalphabet@@@</code>	330	<code>\prop_gput:Nxn</code>	1655, 1688
<code>\newmcodes@</code>	1821	<code>\prop_if_in:cxTF</code>	1774
<code>\nolimits</code>	371	<code>\prop_if_in:NnTF</code>	11
<code>\non@alpherr</code>	1036	<code>\prop_new:N</code>	1643, 1644
<code>\npolint</code>	600		

<code>\protect</code>	620	<code>\set@mathsymbol</code>	338
<code>\ProvidesPackage</code>	1	<code>\setbox</code>	1633
Q			
<code>\Q</code>	1146	<code>\setkeys</code>	164, 419, 421
<code>\q_nil</code>	658, 661, 669, 673	<code>\SetMathAlphabet</code>	336
<code>\qprime</code>	1598	<code>\SetMathAlphabet@</code>	336
R			
<code>\R</code>	1147, 1176, 1189	<code>\setmathfont</code>	407, 620
<code>\r@@t</code>	1632	<code>\SetSymbolFont</code>	335
<code>\raise</code>	1635	<code>\SetSymbolFont@</code>	335
<code>\relax</code> ...	168, 232, 258, 345, 361, 364, 368, 374, 376, 378–380, 383–386, 389–392, 394, 395, 397, 405, 408, 445, 446, 591, 638, 683, 686, 710, 711, 716, 721, 725, 726, 743, 796, 804, 805, 1638, 1816, 1817, 1826, 1830	<code>\sf@size</code>	430, 434
<code>\removenolimits</code>	606	<code>\sp</code>	1661
<code>\RequirePackage</code>	3–5	<code>\space</code>	963, 998, 1036
<code>\restore@mathversion</code>	333	<code>\sprime</code>	533, 1584, 1585, 1590
<code>\rho</code>	1726	<code>\sqint</code>	600
<code>\rightarrow</code>	1788	<code>\sqrt</code>	609, 1631
<code>\rootbox</code>	1639	<code>\sqrtsign</code>	1631, 1633
<code>\rppolint</code>	599	<code>\std@equal</code>	1817
S			
<code>\sb</code>	1694	<code>\std@minus</code>	1816, 1826
<code>\scan_stop:</code> ..	538, 539, 1021, 1657, 1690	<code>\sterling</code>	1797
<code>\scantokens</code>	1552, 1658, 1691	<code>\string</code>	705, 706
<code>\scpolint</code>	600	<code>\strip@pt</code>	345
<code>\scriptscriptstyle</code>	351	<code>\sumint</code>	598
<code>\scriptstyle</code>	348	T	
<code>\seq_clear:N</code> ...	410, 643, 644, 886, 929	<code>\tf@size</code>	429, 430
<code>\seq_if_empty:N</code>	957	<code>\thinmuskip</code>	1580
<code>\seq_if_empty:N</code>	930	<code>\tilde</code>	1842
<code>\seq_if_in:N</code>	8	<code>\tl_if_empty:N</code>	972
<code>\seq_if_in:N</code>	663	<code>\tl_if_empty:NT</code>	975
<code>\seq_map_inline:N</code>	962, 968	<code>\tl_if_in:N</code>	371, 657, 660
<code>\seq_map_variable:NN</code>	678	<code>\tl_if_in:N</code>	375
<code>\seq_new:N</code>	634–636, 927	<code>\tl_map_inline:nn</code>	328, 887
<code>\seq_put_right:N</code>	649, 895	<code>\tl_new:N</code>	596, 609, 898–914, 916–920, 922–925
<code>\seq_put_right:N</code>	647, 996	<code>\tl_put_right:cx</code>	584
<code>\set@@mathdelimiter</code>	339	<code>\tl_put_right:N</code>	7, 604
<code>\set@mathaccent</code>	337	<code>\tl_put_right:N</code>	1782
<code>\set@mathchar</code>	337	<code>\tl_remove_all_in:N</code>	607
<code>\set@mathdelimiter</code>	339	<code>\tl_remove_in:N</code>	342, 1815
		<code>\tl_rescan:nn</code>	1557
		<code>\tl_set:cx</code>	871
		<code>\tl_set:N</code>	275–277, 279–281, 293–295, 297–299, 319, 321, 323, 413, 415–418, 457, 654–656, 670, 671, 674, 675, 873, 876, 879, 882, 1005–1009, 1660, 1662, 1693, 1695
		<code>\tl_set:No</code>	969–971
		<code>\tl_set:N</code>	977

\tl_set:Nx 466, 984, 985, 1744, 1746	\um_config_mathbfup_Greek: 1361
\tl_set_eq:NN 1745	\um_config_mathbfup_greek: 1373
\tl_use:c 998	\um_config_mathbfup_Latin: 1331
\to 1788	\um_config_mathbfup_latin: 1346
\token_if_eq_catcode_p:NN	. 1755, 1756	\um_config_mathbfup_misc: 1387
\token_if_eq_meaning_p:NN 1757	\um_config_mathbfup_num: 1327
\triangle 1793	\um_config_mathfrak_Latin: 1184
\trprime 1595	\um_config_mathfrak_latin: 1192
\typeout 958, 963	\um_config_mathit_Greek: 1113
U			
\um@backslash 682, 705	\um_config_mathit_greek: 1123
\um@char@num@range	411, 590, 694, 695, 697	\um_config_mathit_Latin: 1091
\um@firstchar 681, 706	\um_config_mathit_latin: 1101
\um@firsttof 704–706	\um_config_mathit_misc: 1133
\um@parse@range 691, 707	\um_config_mathscr_Latin: 1167
\um@parse@term 487, 523, 677, 746	\um_config_mathscr_latin: 1178
\um@radicals 609	\um_config_mathsf_Latin: 1223
\um@scaled@apply 347, 1634, 1640	\um_config_mathsf_latin: 1235
\um@scanactivedef 365, 1548	\um_config_mathsfup_Latin: 1199
\um@scancharlet 1548, 1568	\um_config_mathsfup_latin: 1211
\um@zf@feature 612, 624, 627	\um_config_mathsfup_num: 1195
\um_config_mathbb_Latin: 1140	\um_config_mathhtt_Latin: 1250
\um_config_mathbb_latin: 1137	\um_config_mathhtt_latin: 1253
\um_config_mathbb_misc: 1153	\um_config_mathhtt_num: 1247
\um_config_mathbb_num: 1150	\um_config_mathup_Greek: 1067
\um_config_mathbbit_misc: 1160	\um_config_mathup_greek: 1077
\um_config_mathbffrak_Latin:	... 1406	\um_config_mathup_Latin: 1046
\um_config_mathbffrak_latin:	... 1409	\um_config_mathup_latin: 1056
\um_config_mathbfit_Greek: 1286	\um_config_mathup_misc: 1087
\um_config_mathbfit_greek: 1298	\um_config_mathup_num: 1042
\um_config_mathbfit_Latin: 1256	\um_fontdimen_to_percent:nn
\um_config_mathbfit_latin: 1271 344, 349, 352, 1636, 1637	
\um_config_mathbfit_misc: 1312	\um_glyph_if_exist:cT 987
\um_config_mathbfscr_Latin:	... 1412	\um_glyph_if_exist:cTF 993
\um_config_mathbfscr_latin:	... 1415	\um_glyph_if_exist:n 1020
\um_config_mathbfsfit_Greek:	... 1509	\um_glyph_if_exist:nF 1026
\um_config_mathbfsfit_greek:	... 1521	\um_glyph_if_exist:nT 1025
\um_config_mathbfsfit_Latin:	... 1485	\um_glyph_if_exist:nTF
\um_config_mathbfsfit_latin:	... 1497 1020, 1592, 1595, 1598	
\um_config_mathbfsfit_misc:	... 1533	\um_glyph_if_exist_p:n 1023
\um_config_mathbfsfup_Greek:	... 1446	\um_if_mathalph_decl:n 653
\um_config_mathbfsfup_greek:	... 1458	\um_if_mathalph_decl:nTF 646
\um_config_mathbfsfup_Latin:	... 1422	\um_init_alphabet:n	... 462, 976, 1015
\um_config_mathbfsfup_latin:	... 1434	\um_make_mathactive:nNN 533, 535
\um_config_mathbfsfup_misc:	... 1470	\um_map_char:cc 769–774, 780
\um_config_mathbfsfup_num: 1418	\um_map_char:nn	786, 789, 1062, 1104, 1108
		\um_map_char:nn␣ 734

\um_map_char_internal:nn	463, 471, 737	498–501, 503, 504, 507–514, 516–519
\um_map_char_noparse:nn	. 463, 741, 747	\um_remap_symbol_noparse:nnn 461, <u>491</u>
\um_map_char_parse:nn 471, 745	\um_remap_symbol_parse:nnn
\um_map_chars_Greek:nn	. 777, 1069, 469, <u>491</u> , 522
1072, 1115, 1118, 1289, 1293,		\um_remap_symbols: 478, <u>491</u>
1364, 1368, 1448, 1452, 1511, 1515		\um_resolve_greek: <u>861</u>
\um_map_chars_greek:nn	. 766, 1079,	\um_scan_sscript: 1663, 1696, 1733, 1735
1082, 1125, 1128, 1301, 1305,		\um_scan_sscript:TF 1734, 1743
1376, 1380, 1460, 1464, 1523, 1527		\um_scanprime:
\um_map_chars_Latin:nn 1604, 1613, 1624, 1628, 1629
..... 756, 1048, 1051,		\um_scanprime_collect:
1093, 1096, 1201, 1205, 1225,	 1606, 1608, 1611, 1614, 1617
1229, 1258, 1262, 1266, 1333,		\um_set_delcode:n ... 545, 548–574, <u>576</u>
1337, 1341, 1424, 1428, 1487, 1491		\um_set_delcode:nn 542–544, 546, 547, <u>576</u>
\um_map_chars_latin:nn	\um_set_mathalph_range:Nnn <u>799</u>
..... 761, 1058, 1061,		\um_set_mathalph_range:Nnn
1103, 1107, 1213, 1217, 1237,	 799, 810, 813, 816
1241, 1273, 1277, 1281, 1348,		\um_set_mathalphabet_char:Ncc ...
1352, 1356, 1436, 1440, 1499, 1503	 821, 837, 843, 849–854
\um_map_chars_numbers:nn	.. 783, 1043	\um_set_mathalphabet_char:Nnn ...
\um_map_chars_range:nnn	<u>792</u> , 857, 1142–1148, 1154–1158,
..... 734, 751, 754, 784, 787		1161–1165, 1169–1176, 1180–1182,
\um_map_chars_xxiii:cc 768, 779	1186–1190
\um_map_chars_xxiii:nn 753, 790	\um_set_mathalphabet_Greek:Nnn ..
\um_map_chars_xxiii:nn <u>734</u> 840, 1075, 1121, 1287,
\um_map_chars_xxvi:cc 758, 763	1290, 1294, 1362, 1365, 1369,
\um_map_chars_xxvi:nn 750, 791	1449, 1453, 1456, 1512, 1516, 1519
\um_map_chars_xxvi:nn <u>734</u>	\um_set_mathalphabet_greek:Nnn ..
\um_mathmap:Nnn	. 460, 468, 795, 803, 967 846, 1085, 1131, 1299,
\um_mathmap_noparse:Nnn	1302, 1308, 1374, 1377, 1383,
..... 460, <u>582</u> , 592, 967		1461, 1465, 1468, 1524, 1528, 1531
\um_mathmap_parse:Nnn 468, <u>589</u>	\um_set_mathalphabet_Latin:Nnn ..
\um_maybe_init_alphabet:n	829, 1054, 1099, 1141, 1168, 1185,
..... 462, 470, 951, 976, 988		1202, 1206, 1209, 1226, 1230,
\um_nprimes:n	1582, 1592, 1595, 1598, 1601	1233, 1251, 1260, 1263, 1267,
\um_nprimes_select:n 1588, 1619	1335, 1338, 1342, 1407, 1413,
\um_peek_execute_branches_ss:	...	1425, 1429, 1432, 1488, 1492, 1495
..... 1748, 1753		\um_set_mathalphabet_latin:Nnn ..
\um_peek_execute_branches_ss_aux:		834, 1065, 1111, 1138, 1179, 1193,
..... 1760, 1773		1214, 1218, 1221, 1238, 1242,
\um_prepare_alph:n 1017, <u>1027</u>	1245, 1254, 1275, 1278, 1282,
\um_process_symbol_noparse:nnnn	.	1350, 1353, 1357, 1410, 1416,
..... 459, 483		1437, 1441, 1444, 1500, 1504, 1507
\um_process_symbol_parse:nnnn	467, <u>483</u>	\um_set_mathalphabet_numbers:Nnn
\um_remap_symbol:nnn 824, 1044, 1151, 1196,
..... 461, 469, 492, 493, 495,		1197, 1248, 1328, 1329, 1419, 1420

<code>\um_set_mathalphabet_pos:Nnnn</code> . . .	<code>\upDigamma</code> 1845
. 819, 1088, 1089,	<code>\updigamma</code> 1844
1134, 1135, 1313, 1314, 1316,	<code>\upint</code> 601
1317, 1320, 1323, 1388–1393,	<code>\use:c</code> 994, 1012, 1030, 1038
1395, 1396, 1399, 1402, 1471,	<code>\use_i:nnn</code> 969
1472, 1474, 1475, 1478, 1481,	<code>\use_ii:nnn</code> 970
1534, 1535, 1537, 1538, 1541, 1544	<code>\use_iii:nnn</code> 971
<code>\um_set_mathalphabet_x:Ncc</code> 826	<code>\use_none:n</code> 470
<code>\um_set_mathalphabet_x:Nnn</code> . 809, 860	<code>\use_none:nnnn</code> 985
<code>\um_set_mathalphabet_xxiii:Ncc</code> . .	
. 842, 848	V
<code>\um_set_mathalphabet_xxiii:Nnn</code> . .	<code>\varepsilon</code> 879
. 815, 858	<code>\varointclockwise</code> 598
<code>\um_set_mathalphabet_xxvi:Ncc</code> 831, 836	<code>\varphi</code> 882
<code>\um_set_mathalphabet_xxvi:Nnn</code> 812, 859	<code>\vec</code> 1789
<code>\um_set_mathcode:nnnn</code> 404, 529, 585, 742	<code>\version@elt</code> 332
<code>\um_set_mathsymbol:nNNn</code> . . . 360, 484	<code>\version@list</code> 332
<code>\um_setup_active_subscript:nn</code> . . .	<code>\vysmwhtcircle</code> 1795
. 1687, 1701–1728	
<code>\um_setup_active_superscript:nn</code> .	W
. 1654, 1668–1684	<code>\widehat</code> 1841
<code>\um_setup_alphabets:</code> 481, 897	<code>\widetilde</code> 1842
<code>\um_setup_delcodes:</code> 480, 541	<code>\wlog</code> 1016
<code>\um_setup_math_alphabet:Nn</code> 983	
<code>\um_setup_math_alphabet:NV</code> . 931–947	X
<code>\um_setup_math_alphabet:VV</code> 979	<code>\XeTeXdelcode</code> . . . 379, 385, 390, 577, 580
<code>\um_setup_math_mapping:n</code>	<code>\XeTeXdelimiter</code> 378, 384, 391, 392
. 948–950, 952–956, 1010	<code>\XeTeXmathaccent</code> 395
<code>\um_setup_mathactives:</code> 479, 532	<code>\XeTeXmathchardef</code> 367, 536
<code>\um_setup_mathup:</code> 1787	<code>\XeTeXmathcode</code> . . 380, 386, 389, 397, 405
<code>\um_setup_nabla:</code> 273, 476	<code>\XeTeXmathcodenum</code> 539, 1657, 1690
<code>\um_setup_partial:</code> 291, 477	<code>\XeTeXradical</code> 376
<code>\um_split_arrow:w</code> 658, 669	<code>\XKV@rm</code> 421, 439
<code>\um_split_slash:w</code> 661, 673	
<code>\um_sub_or_super:n</code> . . 1661, 1694, 1737	Y
<code>\um_symfont_tl</code> 457, 466,	<code>\yen</code> 1796
473, 484, 529, 537, 577, 580, 585, 743	
<code>\um_tmp:</code> 420, 423, 424, 443	Z
<code>\unicodecdots</code> 1834	<code>\Z</code> 917, 1148, 1190
<code>\unicodeellipsis</code> 1833	<code>\z</code> 916
<code>\UnicodeMathSymbol</code> 459, 467, 1567	<code>\z@</code> 1633, 1636, 1637, 1641
<code>\unimathsetup</code> 163	<code>\zf@family</code> 474
<code>\unless</code> 679	<code>\zf@fontspec</code> 425
<code>\updefault</code> 474	<code>\zf@update@ff</code> 625, 628