

Experimental unicode mathematical typesetting: The unicode-math package

Will Robertson

2010/05/30 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	7.3	The main <code>\setmathfont</code> macro	33
2	Acknowledgements	3	7.4	(Big) operators	39
3	Getting started	3	7.5	Radicals	42
	3.1 Package options	3	7.6	Delimiters	42
			7.7	Maths accents	44
4	Unicode maths font setup	4	8	Font features	46
	4.1 Using multiple fonts	5	8.1	OpenType maths font features	46
	4.2 Script and scriptscript fonts/features	6	8.2	Script and scriptscript font options	46
5	Maths input	6	8.3	Range processing	46
	5.1 Math ‘style’	6	8.4	Resolving Greek symbol name control sequences	52
	5.2 Bold style	7	9	Maths alphabets mapping definitions	53
	5.3 Sans serif style	8		9.1 Alphabets	57
	5.4 All (the rest) of the mathematical alphabets	9	10	Definitions of the math symbols	71
	5.5 Miscellaneous	10	11	Epilogue	73
I	The unicode-math package	16	12	Error messages	86
6	Things we need	16	13	stix table data extraction	87
	6.1 Options	24	A	Documenting maths support in the NFSS	87
	6.2 Overcoming <code>\@on-lypreamble</code>	30	B	X_YTeX math font dimensions	89
7	Fundamentals	30			
	7.1 Enlarging the number of maths families	30			
	7.2 Setting math chars, math codes, etc.	30			

1 Introduction

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

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

2 Acknowledgements

Many thanks to Microsoft for developing OpenType math as part of Office 2007; Jonathan Kew for implementing unicode math support in $\text{Xe}\TeX$; Barbara Beeton for her prodigious effort compiling the definitive list of unicode math glyphs and their $\text{L}\text{A}\text{T}\text{E}\text{X}$ names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use TEX in the future. Apostolos Syropoulos, Joel Salomon, and Khaled Hosny have been fantastic beta testers.

3 Getting started

Load unicode-math as a regular $\text{L}\text{A}\text{T}\text{E}\text{X}$ package. It should be loaded after any other maths or font-related package in case it needs to overwrite their definitions. Here's an example:

```
\usepackage{amsmath} % if desired
\usepackage{unicode-math}
\setmathfont{Cambria Math}
```

3.1 Package options

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

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

Table 1: Package options.

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

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

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

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

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

4 Unicode maths font setup

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

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

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

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

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

Table 2: Maths font options.

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

4.1 Using multiple fonts

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

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

where *<unicode range>* is a comma-separated list of unicode slots and ranges such as `{"27D0-"27EB", "27FF", "295B-"297F"}`. You may also use the macro for accessing the glyph, such as `\int`, or whole collection of symbols with the same math type, such as `\mathopen`, or complete math 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`.

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

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

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

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

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

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

5.2 Bold style

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

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

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

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

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

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

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

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

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

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical alphabet characters. These may be explicitly selected with the `\mathsfup`, `\mathsfit`, `\mathbfsup`, and `\mathbfsfit` commands discussed in section §5.4.

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

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

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

5.3.1 What about bold sans serif?

While you might want your bold upright and your sans serif italic, I don’t believe you’d also want your bold sans serif upright (or all vice versa, if that’s even conceivable). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

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

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

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

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

5.4 All (the rest) of the mathematical alphabets

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

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

5.4.1 Double-struck

The double-struck alphabet (also known as ‘blackboard bold’) consists of upright Latin letters $\{a-z, A-Z\}$, numerals $\mathbb{0}-\mathbb{9}$, 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, and `nabla=literal` respects the shape of the input character. 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`. `nabla=literal` is activated by default after `math-style=literal`.

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, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests and argues otherwise) `partial=italic`.¹ `partial=literal` is activated following `math-style=literal`.

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

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

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

Description		Glyph
Regular	Upright	∂
	Italic	∂
Bold	Upright	∂
	Italic	∂
Sans bold	Upright	∂
	Italic	∂

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

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

We have a package option to control this behaviour. With `\vargreek-shape=TeX`, `\phi` and `\epsilon` produce ϕ and ε and `\varphi` and `\varepsilon` produce ϕ and ϵ . With `\vargreek-shape=unicode`, these symbols are swapped. Note, however, that unicode characters are not affected by this option. That is, no remapping occurs of the characters/glyphs, only the control sequences.

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

5.5.4 Primes

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

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

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

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

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

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

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

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

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

5.5.5 Unicode subscripts and superscripts

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

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

5.5.6 Colon

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

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

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

To preserve input compatibility, we remap the `ASCII` input character `':'` to `U+2236`. Typing a literal `U+2236` char will result in the same output. If `amsmath` is loaded, then the definition of `\colon` is inherited from there (it looks like a

Table 8: Slashes and backslashes.

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

punctuation colon with additional space around it). Otherwise, `\colon` is made to output a colon with `\mathpunct` spacing.

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

5.5.7 Slashes and backslashes

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

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

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

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

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

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

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

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

²§4.4.5.11 <http://www.w3.org/TR/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

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

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 Pre-drawn fraction characters

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

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

For example, instead of writing ‘`\tfrac{12}{x}`’, it’s more readable to have ‘ $\frac{12}{x}$ ’ in the source instead.

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

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdotp</code>	.			
U+22C5	<code>\cdot</code>	.			
U+2219	<code>\vysmbkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smbkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbkcircle</code>	●	◉	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdbkcircle</code>	●	◯	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgbkcircle</code>	●	◯	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgbkcircle</code>	●	◯	<code>\lgwhtcircle</code>	U+25EF

Table 9: Filled and hollow unicode circles.

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

5.5.9 Circles

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

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

5.5.10 Triangles

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

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

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

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

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

Table 10: Different upwards pointing triangles.

5.5.11 Normalising some input characters

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

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

(Not yet implemented.)

File I

The unicode-math package

This is the package.

```

1 \ProvidesPackage{unicode-math}
2 [2010/05/30 v0.4 Unicode maths in XeLaTeX]
```

6 Things we need


```

3 \usepackage{ifxetex,ifluatex}
4 \ifxetex\else\ifluatex\else
5   \PackageError{unicode-math}{%
6     Cannot be run with pdfLaTeX!\MessageBreak
7     Use XeLaTeX or LuaLaTeX instead.%
8   }\@ehd
9 \fi\fi

```

Packages

```

10 \RequirePackage{expl3}[2009/08/12]
11 \RequirePackage{xparse}[2009/08/31]
12 \RequirePackage{l3keys2e}
13 \RequirePackage{fontspec}[2010/05/18]
    Start using LATEX3 — finally!
14 \ExplSyntaxOn
15 \@ifclassloaded{memoir}{
16   \cs_set_eq:NN \um_after_pkg:nn \AtEndPackage
17 }{
18   \RequirePackage{scrfile}
19   \cs_set_eq:NN \um_after_pkg:nn \AfterPackage
20 }

```

Extra expl3 variants

```

21 \cs_generate_variant:Nn \tl_put_right:Nn {cx}
22 \cs_generate_variant:Nn \seq_if_in:NnTF {NV}
23 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
24 \cs_generate_variant:Nn \prop_get:NnN {cxN}
25 \cs_generate_variant:Nn \prop_if_in:NnTF {cx}
26 \cs_new:Npn \exp_args:NNcc #1#2#3#4 {
27   \exp_after:wN #1 \exp_after:wN #2
28   \cs:w #3 \exp_after:wN \cs_end:
29   \cs:w #4 \cs_end:
30 }

```

Conditionals

```

31 \bool_new:N \l_um_fontspec_feature_bool
32 \bool_new:N \l_um_ot_math_bool
33 \bool_new:N \l_um_init_bool
34 \bool_new:N \l_um_implicit_alph_bool

```

For math-style:

```

35 \bool_new:N \g_um_literal_bool
36 \bool_new:N \g_um_upLatin_bool
37 \bool_new:N \g_um_uplatin_bool
38 \bool_new:N \g_um_upGreek_bool
39 \bool_new:N \g_um_upgreek_bool

```

For bold-style:

```
40 \bool_new:N \g_um_bfliteral_bool
41 \bool_new:N \g_um_bfupLatin_bool
42 \bool_new:N \g_um_bfuplatin_bool
43 \bool_new:N \g_um_bfupGreek_bool
44 \bool_new:N \g_um_bfupgreek_bool
```

For sans-style:

```
45 \bool_new:N \g_um_upsans_bool
46 \bool_new:N \g_um_sfliteral_bool
```

For assorted package options:

```
47 \bool_new:N \g_um_upNabla_bool
48 \bool_new:N \g_um_uppartial_bool
49 \bool_new:N \g_um_literal_Nabla_bool
50 \bool_new:N \g_um_literal_partial_bool
51 \bool_new:N \g_um_texgreek_bool
52 \bool_new:N \l_um_smallfrac_bool
53 \bool_new:N \g_um_literal_colon_bool
```

Variables

```
54 \int_new:N \g_um_fam_int

55 \tl_set:Nn \g_um_math_alphabet_name_latin_tl {Latin,~lowercase}
56 \tl_set:Nn \g_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
57 \tl_set:Nn \g_um_math_alphabet_name_greek_tl {Greek,~lowercase}
58 \tl_set:Nn \g_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
59 \tl_set:Nn \g_um_math_alphabet_name_num_tl {Numerals}
60 \tl_set:Nn \g_um_math_alphabet_name_misc_tl {Misc.}
```

6.0.12 Compatibility with LuaT_EX

```
61 \xetex_or luatex:nnn { \cs_new:Npn \um_cs_compat:n #1 }
62 { \cs_set_eq:cc {U#1} {XeTeX#1} }
63 { \cs_set_eq:cc {U#1} {luatexU#1} }
64 \um_cs_compat:n {mathcode}
65 \um_cs_compat:n {delcode}
66 \um_cs_compat:n {mathcodenum}
67 \um_cs_compat:n {mathcharnum}
68 \um_cs_compat:n {mathchardef}
69 \um_cs_compat:n {radical}
70 \um_cs_compat:n {mathaccent}
71 \um_cs_compat:n {delimiter}
```

6.0.13 Function variants

```
72 \cs_generate_variant:Nn \fontspec_select:nn {x}
```

6.0.14 Alphabet unicode positions

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

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

```
73 \cs_new:Npn \usv_set:nnn #1#2#3 {  
74   \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}  
75 }  
76 \cs_new:Npn \um_to_usv:nn #1#2 { g_um_#1_#2_usv }
```

Alphabets

```
77 \usv_set:nnn {up}{num}{48}  
78 \usv_set:nnn {up}{Latin}{65}  
79 \usv_set:nnn {up}{latin}{97}  
80 \usv_set:nnn {up}{Greek}{391}  
81 \usv_set:nnn {up}{greek}{3B1}  
82 \usv_set:nnn {it}{Latin}{1D434}  
83 \usv_set:nnn {it}{latin}{1D44E}  
84 \usv_set:nnn {it}{Greek}{1D6E2}  
85 \usv_set:nnn {it}{greek}{1D6FC}  
86 \usv_set:nnn {bb}{num}{1D7D8}  
87 \usv_set:nnn {bb}{Latin}{1D538}  
88 \usv_set:nnn {bb}{latin}{1D552}  
89 \usv_set:nnn {scr}{Latin}{1D49C}  
90 \usv_set:nnn {scr}{latin}{1D4B6}  
91 \usv_set:nnn {frak}{Latin}{1D504}  
92 \usv_set:nnn {frak}{latin}{1D51E}  
93 \usv_set:nnn {sf}{num}{1D7E2}  
94 \usv_set:nnn {sfup}{num}{1D7E2}  
95 \usv_set:nnn {sfit}{num}{1D7E2}  
96 \usv_set:nnn {sfup}{Latin}{1D5A0}  
97 \usv_set:nnn {sf}{Latin}{1D5A0}  
98 \usv_set:nnn {sfup}{latin}{1D5BA}  
99 \usv_set:nnn {sf}{latin}{1D5BA}  
100 \usv_set:nnn {sfit}{Latin}{1D608}  
101 \usv_set:nnn {sfit}{latin}{1D622}  
102 \usv_set:nnn {tt}{num}{1D7F6}  
103 \usv_set:nnn {tt}{Latin}{1D670}  
104 \usv_set:nnn {tt}{latin}{1D68A}
```

Bold:

```
105 \usv_set:nnn {bf}{num}{1D7CE}  
106 \usv_set:nnn {bfup}{num}{1D7CE}  
107 \usv_set:nnn {bfit}{num}{1D7CE}  
108 \usv_set:nnn {bfup}{Latin}{1D400}
```

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

```

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

131 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsf
132 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_latin_usv \g_um_bfsf
133 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsf
134 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsf
135 \usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_L
136 \usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_l
137 \usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_G
138 \usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_g

```

Greek variants:

```

139 \usv_set:nnn {up}{varTheta}{ "3F4}
140 \usv_set:nnn {up}{Digamma}{ "3DC}
141 \usv_set:nnn {up}{varepsilon}{ "3F5}
142 \usv_set:nnn {up}{vartheta}{ "3D1}
143 \usv_set:nnn {up}{varkappa}{ "3F0}
144 \usv_set:nnn {up}{varphi}{ "3D5}
145 \usv_set:nnn {up}{varrho}{ "3F1}
146 \usv_set:nnn {up}{varpi}{ "3D6}
147 \usv_set:nnn {up}{digamma}{ "3DD}

```

Bold:

```

148 \usv_set:nnn {bfup}{varTheta}{ "1D6B9}
149 \usv_set:nnn {bfup}{Digamma}{ "1D7CA}
150 \usv_set:nnn {bfup}{varepsilon}{ "1D6DC}
151 \usv_set:nnn {bfup}{vartheta}{ "1D6DD}

```

$\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{varkappa}\}\{"1\text{D6DE}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{varphi}\}\{"1\text{D6DF}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{varrho}\}\{"1\text{D6E0}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{varpi}\}\{"1\text{D6E1}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{digamma}\}\{"1\text{D7CB}\}$

Italic Greek variants:

$\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varTheta}\}\{"1\text{D6F3}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varepsilonpsilon}\}\{"1\text{D716}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varthetaeta}\}\{"1\text{D717}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varkappa}\}\{"1\text{D718}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varphi}\}\{"1\text{D719}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varrho}\}\{"1\text{D71A}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{varpi}\}\{"1\text{D71B}\}$

Bold italic:

$\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varTheta}\}\{"1\text{D72D}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varepsilonpsilon}\}\{"1\text{D750}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varthetaeta}\}\{"1\text{D751}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varkappa}\}\{"1\text{D752}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varphi}\}\{"1\text{D753}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varrho}\}\{"1\text{D754}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{varpi}\}\{"1\text{D755}\}$

Bold sans:

$\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varTheta}\}\{"1\text{D767}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varepsilonpsilon}\}\{"1\text{D78A}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varthetaeta}\}\{"1\text{D78B}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varkappa}\}\{"1\text{D78C}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varphi}\}\{"1\text{D78D}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varrho}\}\{"1\text{D78E}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{varpi}\}\{"1\text{D78F}\}$

Bold sans italic:

$\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varTheta}\}\{"1\text{D7A1}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varepsilonpsilon}\}\{"1\text{D7C4}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varthetaeta}\}\{"1\text{D7C5}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varkappa}\}\{"1\text{D7C6}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varphi}\}\{"1\text{D7C7}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varrho}\}\{"1\text{D7C8}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{varpi}\}\{"1\text{D7C9}\}$

Nabla:

$\backslash\text{usv_set:nnn}\{\text{up}\}\{\text{Nabla}\}\{"0\text{2207}\}$
 $\backslash\text{usv_set:nnn}\{\text{it}\}\{\text{Nabla}\}\{"1\text{D6FB}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfup}\}\{\text{Nabla}\}\{"1\text{D6C1}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfit}\}\{\text{Nabla}\}\{"1\text{D735}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfup}\}\{\text{Nabla}\}\{"1\text{D76F}\}$
 $\backslash\text{usv_set:nnn}\{\text{bfsfit}\}\{\text{Nabla}\}\{"1\text{D7A9}\}$

Partial:

```
191 \usv_set:nnn {up}    {partial}{"02202}  
192 \usv_set:nnn {it}    {partial}{"1D715}  
193 \usv_set:nnn {bfup}  {partial}{"1D6DB}  
194 \usv_set:nnn {bfit}  {partial}{"1D74F}  
195 \usv_set:nnn {bfsfup}{partial}{"1D789}  
196 \usv_set:nnn {bfsfit}{partial}{"1D7C3}
```

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

```
197 \usv_set:nnn {up}{B}{`\B}  
198 \usv_set:nnn {up}{C}{`\C}  
199 \usv_set:nnn {up}{D}{`\D}  
200 \usv_set:nnn {up}{E}{`\E}  
201 \usv_set:nnn {up}{F}{`\F}  
202 \usv_set:nnn {up}{H}{`\H}  
203 \usv_set:nnn {up}{I}{`\I}  
204 \usv_set:nnn {up}{L}{`\L}  
205 \usv_set:nnn {up}{M}{`\M}  
206 \usv_set:nnn {up}{N}{`\N}  
207 \usv_set:nnn {up}{P}{`\P}  
208 \usv_set:nnn {up}{Q}{`\Q}  
209 \usv_set:nnn {up}{R}{`\R}  
210 \usv_set:nnn {up}{Z}{`\Z}  
  
211 \usv_set:nnn {it}{B}{"1D435}  
212 \usv_set:nnn {it}{C}{"1D436}  
213 \usv_set:nnn {it}{D}{"1D437}  
214 \usv_set:nnn {it}{E}{"1D438}  
215 \usv_set:nnn {it}{F}{"1D439}  
216 \usv_set:nnn {it}{H}{"1D43B}  
217 \usv_set:nnn {it}{I}{"1D43C}  
218 \usv_set:nnn {it}{L}{"1D43F}  
219 \usv_set:nnn {it}{M}{"1D440}  
220 \usv_set:nnn {it}{N}{"1D441}  
221 \usv_set:nnn {it}{P}{"1D443}  
222 \usv_set:nnn {it}{Q}{"1D444}  
223 \usv_set:nnn {it}{R}{"1D445}  
224 \usv_set:nnn {it}{Z}{"1D44D}  
  
225 \usv_set:nnn {up}{d}{`\d}  
226 \usv_set:nnn {up}{e}{`\e}  
227 \usv_set:nnn {up}{g}{`\g}  
228 \usv_set:nnn {up}{h}{`\h}  
229 \usv_set:nnn {up}{i}{`\i}  
230 \usv_set:nnn {up}{j}{`\j}  
231 \usv_set:nnn {up}{o}{`\o}
```

²³² \usv_set:nnn {it}{d}{1D451}
²³³ \usv_set:nnn {it}{e}{1D452}
²³⁴ \usv_set:nnn {it}{g}{1D454}
²³⁵ \usv_set:nnn {it}{h}{0210E}
²³⁶ \usv_set:nnn {it}{i}{1D456}
²³⁷ \usv_set:nnn {it}{j}{1D457}
²³⁸ \usv_set:nnn {it}{o}{1D45C}

Latin ‘h’:

²³⁹ \usv_set:nnn {bb} {h}{1D559}
²⁴⁰ \usv_set:nnn {tt} {h}{1D691}
²⁴¹ \usv_set:nnn {scr} {h}{1D4BD}
²⁴² \usv_set:nnn {frak} {h}{1D525}
²⁴³ \usv_set:nnn {bfup} {h}{1D421}
²⁴⁴ \usv_set:nnn {bfit} {h}{1D489}
²⁴⁵ \usv_set:nnn {sfup} {h}{1D5C1}
²⁴⁶ \usv_set:nnn {sfit} {h}{1D629}
²⁴⁷ \usv_set:nnn {bffrak}{h}{1D58D}
²⁴⁸ \usv_set:nnn {bfscr} {h}{1D4F1}
²⁴⁹ \usv_set:nnn {bfsfup}{h}{1D5F5}
²⁵⁰ \usv_set:nnn {bfsfit}{h}{1D65D}

Dotless ‘i’ and ‘j’:

²⁵¹ \usv_set:nnn {up}{dotlessi}{00131}
²⁵² \usv_set:nnn {up}{dotlessj}{00237}
²⁵³ \usv_set:nnn {it}{dotlessi}{1D6A4}
²⁵⁴ \usv_set:nnn {it}{dotlessj}{1D6A5}

Blackboard:

²⁵⁵ \usv_set:nnn {bb}{C}{2102}
²⁵⁶ \usv_set:nnn {bb}{H}{210D}
²⁵⁷ \usv_set:nnn {bb}{N}{2115}
²⁵⁸ \usv_set:nnn {bb}{P}{2119}
²⁵⁹ \usv_set:nnn {bb}{Q}{211A}
²⁶⁰ \usv_set:nnn {bb}{R}{211D}
²⁶¹ \usv_set:nnn {bb}{Z}{2124}
²⁶² \usv_set:nnn {up}{Pi} {003A0}
²⁶³ \usv_set:nnn {up}{pi} {003C0}
²⁶⁴ \usv_set:nnn {up}{Gamma} {00393}
²⁶⁵ \usv_set:nnn {up}{gamma} {003B3}
²⁶⁶ \usv_set:nnn {up}{summation}{02211}
²⁶⁷ \usv_set:nnn {it}{Pi} {1D6F1}
²⁶⁸ \usv_set:nnn {it}{pi} {1D70B}
²⁶⁹ \usv_set:nnn {it}{Gamma} {1D6E4}
²⁷⁰ \usv_set:nnn {it}{gamma} {1D6FE}
²⁷¹ \usv_set:nnn {bb}{Pi} {0213F}
²⁷² \usv_set:nnn {bb}{pi} {0213C}
²⁷³ \usv_set:nnn {bb}{Gamma} {0213E}

```

274 \usv_set:nnn {bb}{gamma} {"0213D}
275 \usv_set:nnn {bb}{summation}{"02140}

```

Italic blackboard:

```

276 \usv_set:nnn {bbit}{D}{"2145}
277 \usv_set:nnn {bbit}{d}{"2146}
278 \usv_set:nnn {bbit}{e}{"2147}
279 \usv_set:nnn {bbit}{i}{"2148}
280 \usv_set:nnn {bbit}{j}{"2149}

```

Script exceptions:

```

281 \usv_set:nnn {scr}{B}{"212C}
282 \usv_set:nnn {scr}{E}{"2130}
283 \usv_set:nnn {scr}{F}{"2131}
284 \usv_set:nnn {scr}{H}{"210B}
285 \usv_set:nnn {scr}{I}{"2110}
286 \usv_set:nnn {scr}{L}{"2112}
287 \usv_set:nnn {scr}{M}{"2133}
288 \usv_set:nnn {scr}{R}{"211B}
289 \usv_set:nnn {scr}{e}{"212F}
290 \usv_set:nnn {scr}{g}{"210A}
291 \usv_set:nnn {scr}{o}{"2134}

```

Fraktur exceptions:

```

292 \usv_set:nnn {frak}{C}{"212D}
293 \usv_set:nnn {frak}{H}{"210C}
294 \usv_set:nnn {frak}{I}{"2111}
295 \usv_set:nnn {frak}{R}{"211C}
296 \usv_set:nnn {frak}{Z}{"2128}

```

Complete u.s.v. ranges These might be needed (with a whole bunch more) later:

```

297 \tl_new:Nn \g_um_mathup_latin_usv_range_tl {\a-\z}
298 \tl_new:Nn \g_um_mathup_Latin_usv_range_tl {\A-\Z}
299 \tl_new:Nn \g_um_mathup_greek_usv_range_tl {"3B1-"3C9,"3F5,"3D1,"3F0,"3D5,"3F1,"3D6,"3DD}
300 \tl_new:Nn \g_um_mathup_Greek_usv_range_tl {"391-"3A9,"3F4,"3DC}
301 \tl_new:Nn \g_um_mathup_num_usv_range_tl {\0-\9}
302 \tl_new:Nn \g_um_mathit_latin_usv_range_tl {"1D44E-"1D467,\g_um_it_h_usv}
303 \tl_new:Nn \g_um_mathit_Latin_usv_range_tl {"1D434-"1D44C}
304 \tl_new:Nn \g_um_mathit_greek_usv_range_tl {"1D6FC-"1D714,"1D716-1D71B}
305 \tl_new:Nn \g_um_mathit_Greek_usv_range_tl {"1D6E2-"1D6FA}

```

6.1 Options

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

```

306 \DeclareDocumentCommand \unimathsetup {m} {
307   \clist_clear:N \l_um_unknown_keys_clist

```



```

308 \keys_set:nn {unicode-math} {#1}
309 }

```

math-style

```

310 \keys_define:nn {unicode-math} {
311   normal-style .choice_code:n =
312   {
313     \bool_set_false:N \g_um_literal_bool
314     \ifcase \l_keys_choice_int
315       \bool_set_false:N \g_um_upGreek_bool
316       \bool_set_false:N \g_um_upgreek_bool
317       \bool_set_false:N \g_um_upLatin_bool
318       \bool_set_false:N \g_um_uplatin_bool
319     \or
320       \bool_set_true:N \g_um_upGreek_bool
321       \bool_set_false:N \g_um_upgreek_bool
322       \bool_set_false:N \g_um_upLatin_bool
323       \bool_set_false:N \g_um_uplatin_bool
324     \or
325       \bool_set_true:N \g_um_upGreek_bool
326       \bool_set_true:N \g_um_upgreek_bool
327       \bool_set_true:N \g_um_upLatin_bool
328       \bool_set_false:N \g_um_uplatin_bool
329     \or
330       \bool_set_true:N \g_um_upGreek_bool
331       \bool_set_true:N \g_um_upgreek_bool
332       \bool_set_true:N \g_um_upLatin_bool
333       \bool_set_true:N \g_um_uplatin_bool
334     \or
335       \bool_set_true:N \g_um_literal_bool
336     \fi
337   } ,
338   normal-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
339 }
340 \keys_define:nn {unicode-math} {
341   math-style .choice_code:n =
342   {
343     \ifcase \l_keys_choice_int
344       \unimathsetup {
345         normal-style=ISO,
346         bold-style=ISO,
347         sans-style=italic,
348         nabla=italic,
349         partial=italic,
350       }
351     \or

```

```

352     \unimathsetup {
353         normal-style=TeX,
354         bold-style=TeX,
355         sans-style=upright,
356         nabla=upright,
357         partial=italic,
358     }
359 \or
360     \unimathsetup {
361         normal-style=french,
362         bold-style=upright,
363         sans-style=upright,
364         nabla=upright,
365         partial=upright,
366     }
367 \or
368     \unimathsetup {
369         normal-style=upright,
370         bold-style=upright,
371         sans-style=upright,
372         nabla=upright,
373         partial=upright,
374     }
375 \or
376     \unimathsetup {
377         normal-style=literal,
378         bold-style=literal,
379         sans-style=literal,
380         colon=literal,
381         nabla=literal,
382         partial=literal,
383     }
384 \fi
385 } ,
386 math-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
387 }

```

bold-style

```

388 \keys_define:nn {unicode-math} {
389     bold-style .choice_code:n = {
390         \bool_set_false:N \g_um_bfliteral_bool
391         \ifcase \l_keys_choice_int
392             \bool_set_false:N \g_um_bfupGreek_bool
393             \bool_set_false:N \g_um_bfupgreek_bool
394             \bool_set_false:N \g_um_bfupLatin_bool
395             \bool_set_false:N \g_um_bfuplatin_bool

```

```

396 \or
397 \bool_set_true:N \g_um_bfupGreek_bool
398 \bool_set_false:N \g_um_bfupgreek_bool
399 \bool_set_true:N \g_um_bfupLatin_bool
400 \bool_set_true:N \g_um_bfuplatin_bool
401 \or
402 \bool_set_true:N \g_um_bfupGreek_bool
403 \bool_set_true:N \g_um_bfupgreek_bool
404 \bool_set_true:N \g_um_bfupLatin_bool
405 \bool_set_true:N \g_um_bfuplatin_bool
406 \or
407 \bool_set_true:N \g_um_bfliteral_bool
408 \fi
409 } ,
410 bold-style .generate_choices:n = {ISO,TeX,upright,literal} ,
411 }

```

sans-style

```

412 \keys_define:nn {unicode-math} {
413   sans-style .choice_code:n = {
414     \ifcase \l_keys_choice_int
415       \bool_set_false:N \g_um_upsans_bool
416     \or
417       \bool_set_true:N \g_um_upsans_bool
418     \or
419       \bool_set_true:N \g_um_sfliteral_bool
420     \fi
421   } ,
422   sans-style .generate_choices:n = {italic,upright,literal} ,
423 }

```

Nabla and partial

```

424 \keys_define:nn {unicode-math} {
425   nabla .choice_code:n = {
426     \bool_set_false:N \g_um_literal_Nabla_bool
427     \ifcase \l_keys_choice_int
428       \bool_set_true:N \g_um_upNabla_bool
429     \or
430       \bool_set_false:N \g_um_upNabla_bool
431     \or
432       \bool_set_true:N \g_um_literal_Nabla_bool
433     \fi
434   } ,
435   nabla .generate_choices:n = {upright,italic,literal} ,
436 }

```

```

437 \keys_define:nn {unicode-math} {
438   partial .choice_code:n = {
439     \bool_set_false:N \g_um_literal_partial_bool
440     \ifcase \l_keys_choice_int
441       \bool_set_true:N \g_um_uppartial_bool
442     \or
443       \bool_set_false:N \g_um_uppartial_bool
444     \or
445       \bool_set_true:N \g_um_literal_partial_bool
446     \fi
447   } ,
448   partial .generate_choices:n = {upright,italic,literal} ,
449 }

```

Epsilon and phi shapes

```

450 \keys_define:nn {unicode-math} {
451   vargreek-shape .choice: ,
452   vargreek-shape / unicode .code:n = {
453     \bool_set_false:N \g_um_texgreek_bool
454   } ,
455   vargreek-shape / TeX .code:n = {
456     \bool_set_true:N \g_um_texgreek_bool
457   }
458 }

```

Colon style

```

459 \keys_define:nn {unicode-math} {
460   colon .choice: ,
461   colon / literal .code:n = {
462     \bool_set_true:N \g_um_literal_colon_bool
463   } ,
464   colon / TeX .code:n = {
465     \bool_set_false:N \g_um_literal_colon_bool
466   }
467 }

```

Slash delimiter style

```

468 \keys_define:nn {unicode-math} {
469   slash-delimiter .choice: ,
470   slash-delimiter / ascii .code:n = {
471     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
472   } ,
473   slash-delimiter / frac .code:n = {
474     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
475   } ,

```

```

476 slash-delimiter / div .code:n = {
477   \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
478 }
479 }

```

Active fraction style

```

480 \keys_define:nn {unicode-math} {
481   active-frac .choice: ,
482   active-frac / small .code:n = {
483     \cs_if_exist:NTF \tfrac {
484       \bool_set_true:N \l_um_smallfrac_bool
485     }{
486       \um_warning:n {no-tfrac}
487       \bool_set_false:N \l_um_smallfrac_bool
488     }
489     \use:c{um_setup_active_frac:}
490   } ,
491   active-frac / normalsize .code:n = {
492     \bool_set_false:N \l_um_smallfrac_bool
493     \use:c{um_setup_active_frac:}
494   }
495 }

```

Debug/tracing

```

496 \keys_define:nn {unicode-math} {
497   trace .choice: ,
498   trace / debug .code:n = {
499     \msg_redirect_module:nnn { unicode-math } { trace } { warning }
500   } ,
501   trace / on .code:n = {
502     \msg_redirect_module:nnn { unicode-math } { trace } { trace }
503   } ,
504   trace / off .code:n = {
505     \msg_redirect_module:nnn { unicode-math } { trace } { none }
506   } ,
507 }

508 \clist_new:N \l_um_unknown_keys_clist
509 \keys_define:nn {unicode-math} {
510   unknown .code:n = {
511     \clist_put_right:No \l_um_unknown_keys_clist {
512       \l_keys_key_tl = {#1}
513     }
514   }
515 }

```

```

516 \unimathsetup {math-style=TeX}
517 \unimathsetup {slash-delimiter=ascii}
518 \unimathsetup {trace=off}
519 \cs_if_exist:NT \tfrac {
520   \unimathsetup {active-frac=small}
521 }
522 \ProcessKeysOptions {unicode-math}

```

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.

```

523 \tl_map_inline:nn {
524   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
525   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
526   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
527   \version@list\version@elt\alpha@list\alpha@elt
528   \restore@mathversion\init@restore@version\dorestore@version\process@table
529   \new@mathversion\DeclareSymbolFont\group@list\group@elt
530   \new@symbolfont\SetSymbolFont\SetSymbolFont@get@cdp
531   \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
532   \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
533   \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
534   \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
535   \set@@mathdelimiter\DeclareMathRadical\mathchar@type
536   \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
537 }{
538   \tl_remove_in:Nn \@preamblecmds {\do#1}
539 }

```

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

```

540 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
541 \let\newfam\new@mathgroup

```

This is sufficient for L^AT_EX's `\DeclareSymbolFont`-type commands to be able to define 256 named maths fonts.

7.2 Setting math chars, math codes, etc.

`\um_set_mathsymbol:nNNn #1` : A L^AT_EX symbol font, e.g., operators

#2 : Symbol macro, *e.g.*, `\alpha`

#3 : Type, *e.g.*, `\mathalpha`

#4 : Slot, *e.g.*, "221E

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

```
542 \cs_set:Npn \um_set_mathsymbol:nNNn #1#2#3#4 {
543   \prg_case_tl:Nnn #3 {
544     \mathop {
545       \um_set_big_operator:nnn {#1} {#2} {#4}
546     }
547     \mathopen {
548       \tl_if_in:NnTF \l_um_radicals_tl {#2} {
549         \cs_gset:cpx {\cs_to_str:N #2 sign} { \um_radical:nn {#1} {#4} }
550       }{
551         \um_set_delcode:n {#4}
552         \um_set_mathcode:nnn {#4} \mathopen {#1}
553         \cs_gset:Npx #2 { \um_delimiter:Nnn \mathopen {#1} {#4} }
554       }
555     }
556     \mathclose {
557       \um_set_delcode:n {#4}
558       \um_set_mathcode:nnn {#4} \mathclose {#1}
559       \cs_gset:Npx #2 { \um_delimiter:Nnn \mathclose {#1} {#4} }
560     }
561     \mathfence {
562       \um_set_mathcode:nnn {#4} {#3} {#1}
563       \um_set_delcode:n {#4}
564       \cs_gset:cpx {l \cs_to_str:N #2} { \um_delimiter:Nnn \math-
565         open {#1} {#4} }
566       \cs_gset:cpx {r \cs_to_str:N #2} { \um_delimiter:Nnn \math-
567         close {#1} {#4} }
568     }
569     \mathaccent {
570       \cs_gset:Npx #2 { \um_accent:Nnn #3 {#1} {#4} }
571     }{
572       \um_set_mathcode:nnn {#4} {#3} {#1}
573     }
```

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

#2 : Macro to assign

#3 : Glyph slot

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

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

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

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

$$(\sum \rightarrow) \Sigma \rightarrow \sum_sym \rightarrow \sumop\nolimits$$

$$(\int \rightarrow) \int \rightarrow \int_sym \rightarrow \intop$$

```

574 \cs_new:Npn \um_set_big_operator:nnn #1#2#3 {
575   \group_begin:
576     \char_make_active:n {#3}
577     \char_gmake_mathactive:n {#3}
578     \um@scanactivedef #3 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }
579   \group_end:
580   \um_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
581   \cs_gset:cpx { \cs_to_str:N #2 _sym } {
582     \exp_not:c { \cs_to_str:N #2 op }
583     \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
584   }
585 }

\um_set_mathcode:nnnn
\um_set_mathcode:nnn
586 \cs_set:Npn \um_set_mathcode:nnnn #1#2#3#4 {
\um_set_mathchar:NNnn
587   \Umathcode \intexpr_eval:n {#1} =
\um_set_mathchar:cNnn
588   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
\um_radical:nn
589 }
\um_delimiter:Nnn
590 \cs_set:Npn \um_set_mathcode:nnn #1#2#3 {
\um_accent:Nnn
591   \Umathcode \intexpr_eval:n {#1} =
592   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#1} \scan_stop:
593 }
594 \cs_set:Npn \um_set_mathchar:NNnn #1#2#3#4 {
595   \Umathchardef #1 =
596   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
597 }
598 \cs_new:Npn \um_radical:nn #1#2 {
599   \Uradical \csname sym#1\endcsname #2 \scan_stop:
600 }
601 \cs_new:Npn \um_delimiter:Nnn #1#2#3 {
602   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:

```



```

603 }
604 \cs_new:Npn \um_accent:Nnn #1#2#3 {
605   \Umathaccent \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
606 }
607 \cs_generate_variant:Nn \um_set_mathchar:NNnn {c}

\char_gmake_mathactive:N
\char_gmake_mathactive:n
608 \cs_new:Npn \char_gmake_mathactive:N #1 {
609   \global\mathcode `#1 = "8000 \scan_stop:
610 }
611 \cs_new:Npn \char_gmake_mathactive:n #1 {
612   \global\mathcode #1 = "8000 \scan_stop:
613 }

```

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

```

614 \cs_new:Npn \um_init: {

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

615   \let\glb@currsizel\relax

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

616   \bool_set_true:N \l_um_init_bool
617   \seq_clear:N \l_um_char_range_seq
618   \clist_clear:N \l_um_char_num_range_clist
619   \seq_clear:N \l_um_mathalpha_seq
620   \clist_clear:N \l_um_unknown_keys_clist

621 }
622 \DeclareDocumentCommand \setmathfont { 0{ } m } {
623   \um_init:

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

624   \csname S@\f@size\endcsname

```

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

```
625 \tl_set:Nn \l_um_mversion_tf {normal}
626 \DeclareMathVersion{\l_um_mversion_tf}
```

Define default font features for the script and scriptscript font.

```
627 \tl_set:Nn \l_um_script_features_tl {ScriptStyle}
628 \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
629 \tl_set:Nn \l_um_script_font_tl {#2}
630 \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.

```
631 \keys_set:nn {unicode-math} {#1}
632 \um_fontspec_select_font:n {#2}
```

Check for the correct number of `\fontdimens`:

```
633 %% \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
634 %% \bool_set_true:N \l_um_ot_math_bool
635 %% \else
636 %% \bool_set_false:N \l_um_ot_math_bool
637 %% \PackageWarningNoLine{unicode-math}{
638 %% The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
639 %% Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
640 %% in~ a~ substandard~ manner
641 %% }
642 %% \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

```
643 \bool_if:NTF \l_um_init_bool {
644 \tl_set:Nn \um_symfont_tl {um_allsym}
645 \msg_trace:nxx {unicode-math} {default-math-font} {#2}
646 \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_noparse:nnnn
647 \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
648 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
649 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
650 \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
651 }{
652 \int_incr:N \g_um_fam_int
653 \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
654 \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_parse:nnnn
655 \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_parse:Nnn
```

```

656 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
657 \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
658 \cs_set_eq:NN \um_map_char_single:nn \um_map_char_parse:nn
659 }

```

Now defined `\um_symfont_t1` as the \LaTeX math font to access everything:

```

660 \DeclareSymbolFont{\um_symfont_t1}
661 {\encodingdefault}{\zf@family}{\mddefault}{\updefault}

```

And now we input every single maths char. See File 13 for the source to `unicode-math.tex` which is used to create `unicode-math-table.tex`.

```

662 \@input{unicode-math-table.tex}
663 \cs_set_eq:NN \UnicodeMathSymbol \use_none:nnnn

```

Finally,

- 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.)

```

664 \um_remap_symbols:
665 \um_setup_mathactives:
666 \um_setup_delcodes:
667 \um_setup_alphabets:

```

Prevent spaces:

```

668 \ignorespaces
669 }

```

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

```

670 \cs_new:Npn \um_fontspec_select_font:n #1 {
671 \bool_set_true:N \l_um_fontspec_feature_bool
672 \fontspec_select:xn
673 {
674 BoldFont = {}, ItalicFont = {},
675 Script = Math,
676 SizeFeatures = {
677 {Size = \tf@size-} ,
678 {Size = \sf@size-\tf@size ,
679 Font = \l_um_script_font_t1 ,
680 \l_um_script_features_t1
681 } ,
682 {Size = -\sf@size ,
683 Font = \l_um_sscript_font_t1 ,
684 \l_um_sscript_features_t1

```

```

685     }
686   },
687   \l_um_unknown_keys_clist
688 }
689 {#1}
690 \tl_set_eq:NN \l_um_font \zf@basefont
691 \bool_set_false:N \l_um_fontspec_feature_bool
692 }

```

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
`\um_process_symbol_parse:nnnn` are to be defined. See section §8.3 for the code that enables this.

```

693 \cs_set:Npn \um_process_symbol_noparse:nnnn #1#2#3#4 {
694   \um_set_mathsymbol:nNNn {\um_symfont_tl} #2#3{#1}
695 }
696 \cs_set:Npn \um_process_symbol_parse:nnnn #1#2#3#4 {
697   \um@parse@term{#1}{#2}{#3}{
698     \um_process_symbol_noparse:nnnn{#1}{#2}{#3}{#4}
699   }
700 }

```

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

```

701 \cs_new:Npn \um_remap_symbols: {
702   \um_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
703   \um_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "cen-
       tred asterisk"
704   \bool_if:NF \g_um_literal_colon_bool {
705     \um_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
706   }
707 }

```

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

```

708 \cs_new:Npn \um_remap_symbol_parse:nnn #1#2#3 {
709   \um@parse@term {#3} {\@nil} {#2} {
710     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
711   }
712 }
713 \cs_new:Npn \um_remap_symbol_noparse:nnn #1#2#3 {
714   \clist_map_inline:nn {#1} {
715     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
716   }
717 }

```

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:`

```
718 \cs_new:Npn \um_setup_mathactives: {  
719   \um_make_mathactive:nnn {"2032} \um_prime_single_mchar \mathord  
720   \um_make_mathactive:nnn {"2033} \um_prime_double_mchar \mathord  
721   \um_make_mathactive:nnn {"2034} \um_prime_triple_mchar \mathord  
722   \um_make_mathactive:nnn {"2057} \um_prime_quad_mchar \mathord  
723   \um_make_mathactive:nnn {"2035} \um_backprime_single_mchar \mathord  
724   \um_make_mathactive:nnn {"2036} \um_backprime_double_mchar \mathord  
725   \um_make_mathactive:nnn {"2037} \um_backprime_triple_mchar \mathord  
726   \um_make_mathactive:nnn {"`'} \mathstraitquote \mathord  
727   \um_make_mathactive:nnn {"`\` } \mathbacktick \mathord  
728 }
```

`\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!

```
729 \cs_new:Npn \um_make_mathactive:nnn #1#2#3 {  
730   \um_set_mathchar:NNnn #2 #3 {\um_symfont_t1} {#1}  
731   \char_gmake_mathactive:n {#1}  
732 }
```

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:`

```
733 \cs_new:Npn \um_setup_delcodes: {  
734   \um_set_delcode:nn {"\ /} {\g_um_slash_delimiter_usv}  
735   \um_set_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash  
736   \um_set_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash  
737   \um_set_delcode:n {"005C} % backslash  
738   \um_set_delcode:nn {"\<} {"27E8} % angle brackets with ascii notation  
739   \um_set_delcode:nn {"\>} {"27E9} % angle brackets with ascii notation  
740   \um_set_delcode:n {"2191} % up arrow  
741   \um_set_delcode:n {"2193} % down arrow  
742   \um_set_delcode:n {"2195} % updown arrow  
743   \um_set_delcode:n {"219F} % up arrow twohead  
744   \um_set_delcode:n {"21A1} % down arrow twohead
```

```

745 \um_set_delcode:n {"21A5} % up arrow from bar
746 \um_set_delcode:n {"21A7} % down arrow from bar
747 \um_set_delcode:n {"21A8} % updown arrow from bar
748 \um_set_delcode:n {"21BE} % up harpoon right
749 \um_set_delcode:n {"21BF} % up harpoon left
750 \um_set_delcode:n {"21C2} % down harpoon right
751 \um_set_delcode:n {"21C3} % down harpoon left
752 \um_set_delcode:n {"21C5} % arrows up down
753 \um_set_delcode:n {"21F5} % arrows down up
754 \um_set_delcode:n {"21C8} % arrows up up
755 \um_set_delcode:n {"21CA} % arrows down down
756 \um_set_delcode:n {"21D1} % double up arrow
757 \um_set_delcode:n {"21D3} % double down arrow
758 \um_set_delcode:n {"21D5} % double updown arrow
759 \um_set_delcode:n {"21DE} % up arrow double stroke
760 \um_set_delcode:n {"21DF} % down arrow double stroke
761 \um_set_delcode:n {"21E1} % up arrow dashed
762 \um_set_delcode:n {"21E3} % down arrow dashed
763 \um_set_delcode:n {"21E7} % up white arrow
764 \um_set_delcode:n {"21E9} % down white arrow
765 \um_set_delcode:n {"21EA} % up white arrow from bar
766 \um_set_delcode:n {"21F3} % updown white arrow
767 }

```

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

```

\um_set_delcode:n 768 \cs_new:Npn \um_set_delcode:nn #1#2 {
769   \Udelcode#1 = \csname sym\um_symfont_t1\endcsname #2
770 }
771 \cs_new:Npn \um_set_delcode:n #1 {
772   \Udelcode#1 = \csname sym\um_symfont_t1\endcsname #1
773 }

```

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.

```

774 \cs_set:Npn \um_mathmap_noparse:Nnn #1#2#3 {
775   \clist_map_inline:nn {#2} {
776     \tl_put_right:cx {um_switchto_\cs_to_str:N #1:} {
777       \um_set_mathcode:nnnn{##1}{\mathalpha}{\um_symfont_t1}{#3}
778     }

```

```

779 }
780 }

```

`\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 `\l_um_char_num_range_clist` macro with slot numbers corresponding to the specified range. This range is used to conditionally add `\um_set_mathcode:nnnn` declarations to the maths alphabet definition.

```

781 \cs_set:Npn \um_mathmap_parse:Nnn #1#2#3 {
782   \clist_if_in:NnT \l_um_char_num_range_clist {#3} {
783     \um_mathmap_noparse:Nnn {#1}{#2}{#3}
784   }
785 }

```

























7.4 (Big) operators

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

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

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

usv	Ex.	Macro	Description
U+02140	$\sum\limits_0^1$	<code>\Bbbsum</code>	DOUBLE-STRUCK N-ARY SUMMATION
U+0220F	$\prod\limits_0^1$	<code>\prod</code>	PRODUCT OPERATOR
U+02210	$\coprod\limits_0^1$	<code>\coprod</code>	COPRODUCT OPERATOR
U+02211	$\sum\limits_0^1$	<code>\sum</code>	SUMMATION OPERATOR
U+0222B	$\int\limits_0^1$	<code>\int</code>	INTEGRAL OPERATOR
U+0222C	$\iint\limits_0^1$	<code>\iint</code>	DOUBLE INTEGRAL OPERATOR
U+0222D	$\iiint\limits_0^1$	<code>\iiint</code>	TRIPLE INTEGRAL OPERATOR
U+0222E	$\oint\limits_0^1$	<code>\oint</code>	CONTOUR INTEGRAL OPERATOR
U+0222F	$\oiint\limits_0^1$	<code>\oiint</code>	DOUBLE CONTOUR INTEGRAL OPERATOR

U+02230		<code>\oiint</code>	TRIPLE CONTOUR INTEGRAL OPERATOR
U+02231		<code>\intclockwise</code>	CLOCKWISE INTEGRAL
U+02232		<code>\varointclockwise</code>	CONTOUR INTEGRAL, CLOCKWISE
U+02233		<code>\ointctrackwise</code>	CONTOUR INTEGRAL, ANTICLOCKWISE
U+022C0		<code>\bigwedge</code>	LOGICAL OR OPERATOR
U+022C1		<code>\bigvee</code>	LOGICAL AND OPERATOR
U+022C2		<code>\bigcap</code>	INTERSECTION OPERATOR
U+022C3		<code>\bigcup</code>	UNION OPERATOR
U+027D5		<code>\leftouterjoin</code>	LEFT OUTER JOIN
U+027D6		<code>\rightouterjoin</code>	RIGHT OUTER JOIN
U+027D7		<code>\fullouterjoin</code>	FULL OUTER JOIN
U+027D8		<code>\bigup tack</code>	LARGE UP TACK
U+027D9		<code>\bigdown tack</code>	LARGE DOWN TACK
U+029F8		<code>\xsolidus</code>	BIG SOLIDUS
U+029F9		<code>\xbsolidus</code>	BIG REVERSE SOLIDUS
U+02A00		<code>\bigodot</code>	N-ARY CIRCLED DOT OPERATOR
U+02A01		<code>\bigoplus</code>	N-ARY CIRCLED PLUS OPERATOR
U+02A02		<code>\bigotimes</code>	N-ARY CIRCLED TIMES OPERATOR
U+02A03		<code>\bigcupdot</code>	N-ARY UNION OPERATOR WITH DOT
U+02A04		<code>\bigupplus</code>	N-ARY UNION OPERATOR WITH PLUS
U+02A05		<code>\bigsqcap</code>	N-ARY SQUARE INTERSECTION OPERATOR
U+02A06		<code>\bigsqcup</code>	N-ARY SQUARE UNION OPERATOR
U+02A07		<code>\conjquant</code>	TWO LOGICAL AND OPERATOR
U+02A08		<code>\disjquant</code>	TWO LOGICAL OR OPERATOR

clude the multiple integrals such as \iiint , but that might be a matter of preference.

```

786 \tl_new:Nn \l_um_nolimits_tl {
787   \int\iint\iiint\iiiint\oint\oiint\oiiint
788   \intclockwise\varointclockwise\ointctrclockwise\sumint
789   \intbar\intBar\oint\circfint\awint\rppoint
790   \scpoint\ntpoint\pointint\sqint\intlarhk\intx
791   \intcap\intcup\upoint\lowint
792 }

```

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

```

793 \DeclareDocumentCommand \addnolimits {m} {
794   \tl_put_right:Nn \l_um_nolimits_tl {#1}
795 }

```

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

```

796 \DeclareDocumentCommand \removenolimits {m} {
797   \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
798 }

```

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.

```

799 \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:

<http://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3853&prlatex/3754>

```

800 \let\left@primitive\left
801 \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
U+02772		<code>\lbrbrak</code>	ORNAMENT
U+027C5	⌵	<code>\lbag</code>	LEFT S-SHAPED BAG DELIMITER
U+027CC	⌵	<code>\longdivision</code>	LONG DIVISION
U+027E6	⌵	<code>\lBrack</code>	MATHEMATICAL LEFT WHITE SQUARE BRACKET
U+027E8	⌵	<code>\langle</code>	BRACKET
U+027EA	⌵	<code>\lAngle</code>	MATHEMATICAL LEFT ANGLE BRACKET
U+027EC		<code>\lbrbrak</code>	MATHEMATICAL LEFT DOUBLE ANGLE BRACKET
U+02983	⌵	<code>\lBrace</code>	BRACKET
U+02985	(<code>\lParen</code>	MATHEMATICAL LEFT WHITE TORTOISE SHELL BRACKET
U+02987	⌵	<code>\llparenthesis</code>	SHELL BRACKET
U+02989	⌵	<code>\llangle</code>	LEFT WHITE CURLY BRACKET
U+0298B	⌵	<code>\lbrackubar</code>	LEFT WHITE PARENTHESIS
U+0298D	⌵	<code>\lbrackultick</code>	Z NOTATION LEFT IMAGE BRACKET
U+0298F	⌵	<code>\lbracklltick</code>	Z NOTATION LEFT BINDING BRACKET
U+02991	⌵	<code>\lbrackdot</code>	LEFT SQUARE BRACKET WITH UNDERBAR
U+02993	⌵	<code>\lparenless</code>	LEFT SQUARE BRACKET WITH TICK IN TOP
U+02995	⌵	<code>\lparengtr</code>	CORNER
U+02997	⌵	<code>\lblkbrbrak</code>	LEFT SQUARE BRACKET WITH TICK IN
U+029D8	⌵	<code>\lvzigzag</code>	BOTTOM CORNER
U+029DA	⌵	<code>\lvzigzag</code>	LEFT ANGLE BRACKET WITH DOT
U+029FC	⌵	<code>\lcurvyangle</code>	LEFT ARC LESS-THAN BRACKET
U+03014		<code>\lbrbrak</code>	DOUBLE LEFT ARC GREATER-THAN BRACKET
U+03018		<code>\lbrbrak</code>	LEFT BLACK TORTOISE SHELL BRACKET
			LEFT WIGGLY FENCE
			LEFT DOUBLE WIGGLY FENCE
			LEFT POINTING CURVED ANGLE BRACKET
			LEFT BROKEN BRACKET
			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
			RIGHT SQUARE BRACKET WITH TICK IN
U+0298E	⌋	<code>\rbracklrtick</code>	BOTTOM CORNER
			RIGHT SQUARE BRACKET WITH TICK IN TOP
U+02990	⌋	<code>\rbrackurtick</code>	CORNER
U+02992	⌋	<code>\rangledot</code>	RIGHT ANGLE BRACKET WITH DOT
U+02994	⌋	<code>\rparengtr</code>	RIGHT ARC GREATER-THAN BRACKET
U+02996	⌋	<code>\Rparenless</code>	DOUBLE RIGHT ARC LESS-THAN BRACKET
U+02998	⌋	<code>\rblkrbrak</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		\grave	GRAVE ACCENT
U+00301		\acute	ACUTE ACCENT
U+00302		\hat	CIRCUMFLEX ACCENT
U+00303		\tilde	TILDE
U+00304		\bar	MACRON
U+00305		\overbar	OVERBAR EMBELLISHMENT
U+00306		\breve	BREVE
U+00307		\dot	DOT ABOVE
U+00308		\ddot	DIERESIS
U+00309		\ovhook	COMBINING HOOK ABOVE
U+0030A		\ocirc	RING
U+0030C		\check	CARON
U+00310		\candra	CANDRABINDU (NON-SPACING)
U+00312		\turnedcomma	COMBINING TURNED COMMA ABOVE GREEK PSILI (SMOOTH BREATHING)
U+00313		\smooth	(NON-SPACING) GREEK DASIA (ROUGH BREATHING)
U+00314		\orough	(NON-SPACING)
U+00315		\ocommatopright	COMBINING COMMA ABOVE RIGHT
U+0031A	¡	\droang	LEFT ANGLE ABOVE (NON-SPACING) UNDER TILDE ACCENT (MULTIPLE
U+00330	¢	\wideutilde	CHARACTERS AND NON-SPACING)
U+00331	£	\underbar	COMBINING MACRON BELOW
U+00338	¤	\not	COMBINING LONG SOLIDUS OVERLAY
U+020D0	¥	\leftharpoonaccent	COMBINING LEFT HARPOON ABOVE
U+020D1	¦	\rightharpoonaccent	COMBINING RIGHT HARPOON ABOVE
U+020D2	§	\vertoverlay	COMBINING LONG VERTICAL LINE OVERLAY
U+020D6	¨	\overleftarrow	COMBINING LEFT ARROW ABOVE
U+020D7	©	\vec	COMBINING RIGHT ARROW ABOVE
U+020DB	ª	\dddot	COMBINING THREE DOTS ABOVE
U+020DC	«	\ddddot	COMBINING FOUR DOTS ABOVE
U+020E1	¬	\overleftrighthararrow	COMBINING LEFT RIGHT ARROW ABOVE
U+020E7	­	\annuity	COMBINING ANNUITY SYMBOL
U+020E8	®	\threeunderdot	COMBINING TRIPLE UNDERDOT
U+020E9	¯	\widebridgeabove	COMBINING WIDE BRIDGE ABOVE COMBINING RIGHTWARDS HARPOON WITH
U+020EC	°	\underrightharpoondown	BARB DOWNWARDS COMBINING LEFTWARDS HARPOON WITH
U+020ED	±	\underleftharpoondown	BARB DOWNWARDS
U+020EE	²	\underleftarrow	COMBINING LEFT ARROW BELOW
U+020EF	³	\underrightarrow	COMBINING RIGHT ARROW BELOW
U+020F0	´	\asteraccent	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.

```
802 \newcommand\um@zf@feature[2]{
803   \define@key[zf]{options}{#1}[]{
804     \bool_if:NTF \l_um_fontspec_feature_bool {
805       #2
806     }{
807       \um_warning:n {maths-feature-only}
808     }
809   }
810 }
```

8.1 OpenType maths font features

```
811 \um@zf@feature{ScriptStyle}{
812   \zf@update@ff{+ssty=0}
813 }
814 \um@zf@feature{ScriptScriptStyle}{
815   \zf@update@ff{+ssty=1}
816 }
```

8.2 Script and scriptscript font options

```
817 \keys_define:nn {unicode-math}
818 {
819   script-features .tl_set:N = \l_um_script_features_tl ,
820   sscript-features .tl_set:N = \l_um_sscript_features_tl ,
821   script-font .tl_set:N = \l_um_script_font_tl ,
822   sscript-font .tl_set:N = \l_um_sscript_font_tl ,
823 }
```

8.3 Range processing

```
824 \seq_new:N \l_um_mathalph_seq
825 \seq_new:N \l_um_char_range_seq
826 \keys_define:nn {unicode-math} {
827   range .code:n = {
828     \bool_set_false:N \l_um_init_bool
829     \seq_clear:N \l_um_char_range_seq
830     \seq_clear:N \l_um_mathalph_seq
831     \clist_map_inline:nn {#1} {
832       \um_if_mathalph_decl:nTF {##1} {
833         \seq_put_right:Nx \l_um_mathalph_seq {
834           { \exp_not:V \l_um_tmpa_tl }

```

```

835         { \exp_not:V \l_um_tmpb_tl }
836         { \exp_not:V \l_um_tmpc_tl }
837     }
838   }{
839     \seq_put_right:Nn \l_um_char_range_seq {##1}
840   }
841 }
842 }
843 }
844 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
845   \KV_remove_surrounding_spaces:nw {\tl_set:Nf\l_um_tmpa_tl} #1 \q_nil
846   \tl_set:Nn \l_um_tmpb_tl {}
847   \tl_set:Nn \l_um_tmpc_tl {}
848   \tl_if_in:NnT \l_um_tmpa_tl {->} {
849     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
850   }
851   \tl_if_in:NnT \l_um_tmpa_tl {/#} {
852     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
853   }
854   \seq_if_in:NVTF \g_um_mathalph_seq \l_um_tmpa_tl {
855     \prg_return_true:
856   }{
857     \prg_return_false:
858   }
859 }
860 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
861   \tl_set:Nn \l_um_tmpa_tl {#1}
862   \tl_set:Nn \l_um_tmpc_tl {#2}
863 }
864 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
865   \tl_set:Nn \l_um_tmpa_tl {#1}
866   \tl_set:Nn \l_um_tmpb_tl {#2}
867 }

```

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

`\um@parse@term`

- #1** : unicode character slot
- #2** : control sequence (character macro)
- #3** : control sequence (math type)
- #4** : code to execute

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

Character ranges are passed to `\um@parse@range`, which accepts input in the form shown in table 15.

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

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

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

```

868 \newcommand\um@parse@term[4]{
869   \seq_map_variable:NNn \l_um_char_range_seq \@ii {
870     \unless\ifx\@ii\@empty
871       \@tempswafalse

```

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

```

872   \expandafter\um@firstchar\expandafter{\@ii}
873   \ifx\@tempa\um@backslash
874     \expandafter\ifx\@ii#2\relax
875       \@tempswatrue
876   \else
877     \expandafter\ifx\@ii#3\relax
878       \@tempswatrue
879   \fi
880   \fi

```

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

```

881   \else
882     \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
883   \fi

```

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

```

884   \if@tempswa
885     \clist_put_right:Nx \l_um_char_num_range_clist { \int-
886       expr_eval:n {#1} }
887     #4
888   \fi
889   \fi
890 }
891 \def\um@firstof#1#2\@nil{#1}
892 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
893 \def\um@firstchar#1{\edef\@tempa{\expandafter\um@firstof\string#1\@nil}}

```


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

```

894 \def\um@parse@range#1-#2-#3\@nil#4\@nil{
895   \def\@tempa{#1}
896   \def\@tempb{#2}

```

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

```

897   \expandafter\ifx\expandafter\@marker\@tempb\relax
898     \intexpr_compare:nT {#4=#1} \@tempswattrue
899   \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	$\textcolor{red}{\#1}-\textcolor{blue}{\#2}-\textcolor{green}{\#3} = X-\{\}-\textcolor{blue}{\@marker}-$

```

900   \ifx\@empty\@tempb
901     \intexpr_compare:nT {#4>#1-1} \@tempswattrue
902   \else

```

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

```

903   \ifx\@empty\@tempa
904     \intexpr_compare:nT {#4<#2+1} \@tempswattrue

```

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

```

905   \else
906     \intexpr_compare:nT {#4>#1-1} {
907       \intexpr_compare:nT {#4<#2+1} \@tempswattrue
908     }
909   \fi
910 \fi
911 \fi
912 }

```

#1 : Starting input char (single)

#2 : Starting output char

Loops through character ranges setting `\mathcode`.

```

913 \cs_set:Npn \um_map_chars_range:nnn #1#2#3 {
914   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {

```

```

915     \um_map_char_single:nn {#2+##1}{#3+##1}
916   }
917 }
918 \cs_generate_variant:Nn \um_map_chars_range:nnn {ncc}

\um_map_chars_range:nnnn #3 : Number of chars (26)
                        #4 : From style, one or more (it)
                        #5 : To style (up)
                        #6 : Alphabet name (Latin)

919 \cs_new:Npn \um_map_chars_range:nnnn #1#2#3#4 {
920   \um_map_chars_range:ncc {#1} { \um_to_usv:nn {#2}{#4} }
921                               { \um_to_usv:nn {#3}{#4} }
922 }

923 \cs_new:Npn \um_map_char_noparse:nn #1#2 {
924   \um_set_mathcode:nnnn {#1}{\mathalpha}{\um_symfont_t1}{#2}
925 }
926 \cs_new:Npn \um_map_char_parse:nn #1#2 {
927   \um@parse@term {#1} {\@nil} {\mathalpha} {
928     \um_map_char_noparse:nn {#1}{#2}
929   }
930 }
931 \cs_set:Npn \um_map_chars_Latin:nn #1#2 {
932   \clist_map_inline:nn {#1} {
933     \um_map_chars_range:nnnn {26} {##1} {#2} {Latin}
934   }
935 }
936 \cs_set:Npn \um_map_chars_latin:nn #1#2 {
937   \clist_map_inline:nn {#1} {
938     \um_map_chars_range:nnnn {26} {##1} {#2} {latin}
939   }
940 }
941 \cs_set:Npn \um_map_chars_greek:nn #1#2 {
942   \clist_map_inline:nn {#1} {
943     \um_map_chars_range:nnnn {25} {##1} {#2} {greek}
944     \um_map_char_single:nnn {##1} {#2} {\varepsilon}
945     \um_map_char_single:nnn {##1} {#2} {\vartheta}
946     \um_map_char_single:nnn {##1} {#2} {\varkappa}
947     \um_map_char_single:nnn {##1} {#2} {\varphi}
948     \um_map_char_single:nnn {##1} {#2} {\varrho}
949     \um_map_char_single:nnn {##1} {#2} {\varpi}
950   }
951 }
952 \cs_set:Npn \um_map_chars_Greek:nn #1#2 {
953   \clist_map_inline:nn {#1} {
954     \um_map_chars_range:nnnn {25} {##1} {#2} {Greek}
955     \um_map_char_single:nnn {##1} {#2} {\varTheta}

```

```

956   }
957 }
958 \cs_set:Npn \um_map_chars_numbers:nn #1#2 {
959   \um_map_chars_range:nnnn {10} {#1} {#2} {num}
960 }

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

961 \cs_new:Npn \um_map_char_single:cc { \exp_args:Ncc \um_map_char_single:nn }
962 \cs_new:Npn \um_map_char_single:nnn #1#2#3 {
963   \um_map_char_single:cc { \um_to_usv:nn {#1}{#3} }
964   { \um_to_usv:nn {#2}{#3} }
965 }
966 \cs_set:Npn \um_map_single:nnn #1#2#3 {
967   \cs_if_exist:cT { \um_to_usv:nn {#3} {#1} }
968   {
969     \clist_map_inline:nn {#2} {
970       \um_map_char_single:nnn {##1} {#3} {#1}
971     }
972   }
973 }

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

974 \cs_new:Npn \um_set_mathalph_range:nNnn #1#2#3#4 {
975   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {
976     \um_set_mathalphabet_char:Nnn {#2} { ##1 + #3 } { ##1 + #4 }
977   }
978 }
979 \cs_generate_variant:Nn \um_set_mathalph_range:nNnn {nNcc}

980 \cs_new:Npn \um_set_mathalphabet_pos:Nnnn #1#2#3#4 {
981   \cs_if_exist:cT { \um_to_usv:nn {#4}{#2} } {
982     \clist_map_inline:nn {#3} {
983       \um_set_mathalphabet_char:Nnnn #1 {##1} {#4} {#2}
984     }
985   }
986 }
987 \cs_new:Npn \um_set_mathalphabet_numbers:Nnn #1#2#3 {
988   \clist_map_inline:nn {#2} {
989     \um_set_mathalph_range:nNnn {10} #1 {##1} {#3} {num}
990   }
991 }
992 \cs_new:Npn \um_set_mathalphabet_Latin:Nnn #1#2#3 {

```

```

993 \clist_map_inline:nn {#2} {
994     \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {Latin}
995 }
996 }
997 \cs_new:Npn \um_set_mathalphabet_latin:Nnn #1#2#3 {
998     \clist_map_inline:nn {#2} {
999         \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {latin}
1000         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {h}
1001     }
1002 }
1003 \cs_new:Npn \um_set_mathalphabet_greek:Nnn #1#2#3 {
1004     \clist_map_inline:nn {#2} {
1005         \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {Greek}
1006         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varTheta}
1007     }
1008 }
1009 \cs_new:Npn \um_set_mathalphabet_greek:Nnn #1#2#3 {
1010     \clist_map_inline:nn {#2} {
1011         \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {greek}
1012         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varepsilon}
1013         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {vartheta}
1014         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varkappa}
1015         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varphi}
1016         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varrho}
1017         \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varpi}
1018     }
1019 }
1020 \cs_new:Npn \um_set_mathalphabet_char:Ncc {
1021     \exp_args:NNcc \um_set_mathalphabet_char:Nnn
1022 }
1023 \cs_new:Npn \um_set_mathalphabet_char:Nnnn #1#2#3#4 {
1024     \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {#2} {#4} }
1025     { \um_to_usv:nn {#3} {#4} }
1026 }
1027 \cs_new:Npn \um_set_mathalph_range:nNnnn #1#2#3#4#5 {
1028     \um_set_mathalph_range:nNcc {#1} #2 { \um_to_usv:nn {#3} {#5} }
1029     { \um_to_usv:nn {#4} {#5} }
1030 }

```

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.

```

1031 \AtBeginDocument{\um_resolve_greek:}

```

```

1032 \cs_new:Npn \um_resolve_greek: {
1033   \clist_map_inline:nn {
1034     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1035     alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
1036     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1037     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,    chi,psi,omega,
1038     varTheta,
1039     varsigma,vartheta,varkappa,varrho,varpi
1040   }{
1041     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
1042   }
1043   \tl_set:Nn \epsilon {
1044     \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
1045   }
1046   \tl_set:Nn \phi {
1047     \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
1048   }
1049   \tl_set:Nn \varepsilon {
1050     \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
1051   }
1052   \tl_set:Nn \varphi {
1053     \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
1054   }
1055 }

```

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

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

```

1056 \cs_new:Npn \um_prepare_alph:n #1 {
1057   \um_init_alphabet:x {\use_none:nnnn #1}
1058   \cs_set:cpn {um_#1:n} ##1 {
1059     \use:c {um_switchto_#1:} ##1 \egroup
1060   }
1061   \cs_set_protected:cpn {#1} {
1062     \exp_not:n{
1063       \bgroup
1064         \mode_if_math:F {
1065           \egroup\expandafter
1066             \non@alpherr\expandafter{\csname #1\endcsname\space}
1067         }
1068       }
1069     \exp_not:c {um_#1:n}
1070   }
1071 }
1072 \cs_generate_variant:Nn \um_prepare_alph:n {f}

```

This is every math alphabet known to unicode-math:

`\g_um_mathalph_seq`

```

1073 \seq_new:N \g_um_mathalph_seq
1074 \AtEndOfPackage{
1075   \tl_map_inline:nn {
1076     \mathup\mathit\mathbb\mathbbit
1077     \mathscr\mathfrak\mathtt
1078     \mathsf\mathsfup\mathsfit
1079     \mathbf\mathbfup\mathbfit
1080     \mathbfscr\mathbffrac
1081     \mathbfssf\mathbfsfup\mathbfsfif
1082   }{
1083     \seq_put_right:Nn \g_um_mathalph_seq {#1}
1084     \um_prepare_alph:f {\cs_to_str:N #1}
1085   }
1086 }

1087 \seq_new:N \g_um_default_mathalph_seq
1088 \clist_map_inline:nn {
1089   {\mathup} {\latin,Latin,greek,Greek,num,misc} {\mathup} ,

```

```

1090 {\mathit } {latin,Latin,greek,Greek,misc} {\mathit } ,
1091 {\mathbb } {latin,Latin,num,misc} {\mathbb } ,
1092 {\mathbbi } {misc} {\mathbbi } ,
1093 {\mathscr } {latin,Latin} {\mathscr } ,
1094 {\mathfrak } {latin,Latin} {\mathfrak } ,
1095 {\mathtt } {latin,Latin,num} {\mathtt } ,
1096 {\mathsfup } {latin,Latin,num} {\mathsfup } ,
1097 {\mathsfit } {latin,Latin} {\mathsfit } ,
1098 {\mathbfup } {latin,Latin,greek,Greek,num,misc} {\mathbfup } ,
1099 {\mathbfit } {latin,Latin,greek,Greek,misc} {\mathbfit } ,
1100 {\mathbfscr } {latin,Latin} {\mathbfscr } ,
1101 {\mathbffrak } {latin,Latin} {\mathbffrak } ,
1102 {\mathbfsfup } {latin,Latin,greek,Greek,num,misc} {\mathbfsfup } ,
1103 {\mathbfsfit } {latin,Latin,greek,Greek,misc} {\mathbfsfit }
1104 }{
1105 \seq_put_right:Nn \g_um_default_mathalph_seq {#1}
1106 }

```

\um_setup_alphabets: Variables:

```

1107 \seq_new:N \l_um_missing_alph_seq
1108 \cs_new:Npn \um_setup_alphabets: {
1109 \seq_clear:N \l_um_missing_alph_seq
1110 \seq_if_empty:NTF \l_um_mathalph_seq {
1111 \um_trace:n {setup-implicit}
1112 \seq_set_eq:NN \l_um_mathalph_seq \g_um_default_mathalph_seq
1113 \bool_set_true:N \l_um_implicit_alph_bool
1114 \um_maybe_init_alphabet:n {sf}
1115 \um_maybe_init_alphabet:n {bf}
1116 \um_maybe_init_alphabet:n {bfsf}
1117 }{
1118 \um_trace:n {setup-explicit}
1119 \bool_set_false:N \l_um_implicit_alph_bool
1120 \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
1121 \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
1122 }
1123 \seq_map_inline:Nn \l_um_mathalph_seq {
1124 \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
1125 \tl_set:No \l_um_tmpb_tl { \use_ii:nnn ##1 }
1126 \tl_set:No \l_um_remap_alphabet_tl { \use_iii:nnn ##1 }
1127 \tl_if_empty:NTF \l_um_remap_alphabet_tl {
1128 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \token_to_str:N \l_um_tmpa_tl}
1129 }{
1130 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \token_to_str:N \l_um_remap_alphabet_tl}
1131 }
1132 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \use_none:nnnnn \l_um_remap_alphabet_tl}
1133 \tl_if_empty:NT \l_um_tmpb_tl {

```

```

1134     \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
1135     \tl_set:Nn \l_um_tmpb_tl { latin, Latin, greek, Greek, num, misc }
1136   }
1137   \um_setup_math_alphabet:VVV \l_um_tmpa_tl \l_um_tmpb_tl \l_um_remap_alphabet_tl
1138 }
1139 \um_warn_missing_alphabets:
1140 }

1141 \cs_new:Npn \um_warn_missing_alphabets: {
1142   \seq_if_empty:NF \l_um_missing_alph_seq {
1143     \typeout{
1144       Package~unicode-math~Warning:~
1145       missing~math~alphabets~in~font~ \fontname\l_um_font
1146     }
1147     \seq_map_inline:Nn \l_um_missing_alph_seq {
1148       \typeout{\space\space\space\space##1}
1149     }
1150   }
1151 }

```

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

```

1152 \cs_new:Npn \um_setup_math_alphabet:Nnn #1#2#3 {
1153   \tl_set:Nx \l_um_tmpa_tl {\cs_to_str:N #1}
1154   \tl_set:Nx \l_um_tmpb_tl {\exp_after:wN \use_none:n \l_um_tmpa_tl}

```

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

```

1155   \clist_map_inline:nn {#2} {
1156     \cs_if_exist:cT {um_config_ \l_um_tmpa_tl _##1:n} {
1157       \tl_if_eq:nnTF {##1}{misc} {
1158         \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
1159         \clist_map_break:
1160       }{
1161         \um_glyph_if_exist:cT { \um_to_usv:nn {#3}{##1} }{
1162           \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
1163           \clist_map_break:
1164         }
1165       }
1166     }
1167   }

```

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

```

1168   \clist_map_inline:nn {#2} {
1169     \cs_if_exist:cT {um_config_ \l_um_tmpa_tl _##1:n} {
1170       \tl_if_eq:nnTF {##1}{misc} {
1171         \um_trace:nx {setup-alph} {\l_um_tmpa_tl~(##1)}
1172         \use:c {um_config_ \l_um_tmpa_tl _##1:n} {#3}

```



```

1173     }{
1174       \um_glyph_if_exist:cTF { \um_to_usv:nn {#3}{##1} } {
1175         \um_trace:nx {setup-alph} {\l_um_tmpa_tl~(##1)}
1176         \use:c {um_config_ \l_um_tmpa_tl _##1:n} {#3}
1177       }{
1178         \bool_if:NTF \l_um_implicit_alph_bool {
1179           \seq_put_right:Nx \l_um_missing_alph_seq {
1180             \@backslashchar
1181             \l_um_tmpa_tl\space(\tl_use:c{g_um_math_alphabet_name_##1_tl})
1182           }
1183         }{
1184           \use:c {um_config_ \l_um_tmpa_tl _##1:n} {up}
1185         }
1186       }
1187     }
1188   }
1189 }
1190 }
1191 \cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {NV,VVV}

1192 \cs_set:Npn \um_init_alphabet:n #1 {
1193   \um_trace:nx {alph-initialise} {#1}
1194   \cs_set_eq:cN {um_switchto_math#1:} \prg_do_nothing:
1195 }
1196 \cs_generate_variant:Nn \um_init_alphabet:n {x}

```

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

```

1197 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
1198   \etex_iffontchar:D \l_um_font #1 \scan_stop:
1199   \prg_return_true:
1200   \else:
1201     \prg_return_false:
1202   \fi:
1203 }
1204 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
1205 \cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
1206 \cs_generate_variant:Nn \um_glyph_if_exist:nT {c}
1207 \cs_generate_variant:Nn \um_glyph_if_exist:nF {c}

```

9.1 Alphabets

9.1.1 Upright: `\mathup`

```

1208 \cs_new:Npn \um_config_mathup_num:n #1 {
1209   \um_map_chars_numbers:nn {up}{#1}
1210   \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}

```

```

1211 }
1212 \cs_new:Npn \um_config_mathup_Latin:n #1 {
1213   \bool_if:NTF \g_um_literal_bool {
1214     \um_map_chars_Latin:nn {up} {#1}
1215   }{
1216     \bool_if:NT \g_um_upLatin_bool {
1217       \um_map_chars_Latin:nn {up,it} {#1}
1218     }
1219   }
1220   \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
1221 }
1222 \cs_new:Npn \um_config_mathup_latin:n #1 {
1223   \bool_if:NTF \g_um_literal_bool {
1224     \um_map_chars_latin:nn {up} {#1}
1225   }{
1226     \bool_if:NT \g_um_uplatin_bool {
1227       \um_map_chars_latin:nn {up,it} {#1}
1228       \um_map_single:nnn {h} {up,it} {#1}
1229       \um_map_single:nnn {dotlessi} {up,it} {#1}
1230       \um_map_single:nnn {dotlessj} {up,it} {#1}
1231     }
1232   }
1233   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
1234 }
1235 \cs_new:Npn \um_config_mathup_Greek:n #1 {
1236   \bool_if:NTF \g_um_literal_bool {
1237     \um_map_chars_Greek:nn {up}{#1}
1238   }{
1239     \bool_if:NT \g_um_upGreek_bool {
1240       \um_map_chars_Greek:nn {up,it}{#1}
1241     }
1242   }
1243   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
1244 }
1245 \cs_new:Npn \um_config_mathup_greek:n #1 {
1246   \bool_if:NTF \g_um_literal_bool {
1247     \um_map_chars_greek:nn {up} {#1}
1248   }{
1249     \bool_if:NT \g_um_upgreek_bool {
1250       \um_map_chars_greek:nn {up,it} {#1}
1251     }
1252   }
1253   \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
1254 }
1255 \cs_new:Npn \um_config_mathup_misc:n #1 {
1256   \bool_if:NTF \g_um_literal_Nabla_bool {

```

```

1257 \um_map_single:nnn {Nabla}{up}{up}
1258 }{
1259 \bool_if:NT \g_um_upNabla_bool {
1260 \um_map_single:nnn {Nabla}{up,it}{up}
1261 }
1262 }
1263 \bool_if:NTF \g_um_literal_partial_bool {
1264 \um_map_single:nnn {partial}{up}{up}
1265 }{
1266 \bool_if:NT \g_um_uppartial_bool {
1267 \um_map_single:nnn {partial}{up,it}{up}
1268 }
1269 }
1270 \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it} {#1}
1271 \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it} {#1}
1272 \um_set_mathalphabet_pos:Nnnn \mathup {dotlessi} {up,it} {#1}
1273 \um_set_mathalphabet_pos:Nnnn \mathup {dotlessj} {up,it} {#1}
1274 }

```

9.1.2 Italic: `\mathit`

```

1275 \cs_new:Npn \um_config_mathit_Latin:n #1 {
1276 \bool_if:NTF \g_um_literal_bool {
1277 \um_map_chars_Latin:nn {it} {#1}
1278 }{
1279 \bool_if:NF \g_um_upLatin_bool {
1280 \um_map_chars_Latin:nn {up,it} {#1}
1281 }
1282 }
1283 \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
1284 }
1285 \cs_new:Npn \um_config_mathit_latin:n #1 {
1286 \bool_if:NTF \g_um_literal_bool {
1287 \um_map_chars_latin:nn {it} {#1}
1288 \um_map_single:nnn {h}{it}{#1}
1289 }{
1290 \bool_if:NF \g_um_uplatin_bool {
1291 \um_map_chars_latin:nn {up,it} {#1}
1292 \um_map_single:nnn {h}{up,it}{#1}
1293 \um_map_single:nnn {dotlessi}{up,it}{#1}
1294 \um_map_single:nnn {dotlessj}{up,it}{#1}
1295 }
1296 }
1297 \um_set_mathalphabet_latin:Nnn \mathit {up,it} {#1}
1298 \um_set_mathalphabet_pos:Nnnn \mathit {dotlessi} {up,it} {#1}
1299 \um_set_mathalphabet_pos:Nnnn \mathit {dotlessj} {up,it} {#1}
1300 }

```

```

1301 \cs_new:Npn \um_config_mathit_Greek:n #1 {
1302   \bool_if:NTF \g_um_literal_bool {
1303     \um_map_chars_Greek:nn {it}{#1}
1304   }{
1305     \bool_if:NF \g_um_upGreek_bool {
1306       \um_map_chars_Greek:nn {up,it}{#1}
1307     }
1308   }
1309   \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
1310 }
1311 \cs_new:Npn \um_config_mathit_greek:n #1 {
1312   \bool_if:NTF \g_um_literal_bool {
1313     \um_map_chars_greek:nn {it} {#1}
1314   }{
1315     \bool_if:NF \g_um_upgreek_bool {
1316       \um_map_chars_greek:nn {it,up} {#1}
1317     }
1318   }
1319   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
1320 }
1321 \cs_new:Npn \um_config_mathit_misc:n #1 {
1322   \bool_if:NTF \g_um_literal_Nabla_bool {
1323     \um_map_single:nnn {Nabla}{it}{it}
1324   }{
1325     \bool_if:NF \g_um_upNabla_bool {
1326       \um_map_single:nnn {Nabla}{up,it}{it}
1327     }
1328   }
1329   \bool_if:NTF \g_um_literal_partial_bool {
1330     \um_map_single:nnn {partial}{it}{it}
1331   }{
1332     \bool_if:NF \g_um_uppartial_bool {
1333       \um_map_single:nnn {partial}{up,it}{it}
1334     }
1335   }
1336   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
1337   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{#1}
1338 }

```

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

```

1339 \cs_new:Npn \um_config_mathbb_latin:n #1 {
1340   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
1341 }
1342 \cs_new:Npn \um_config_mathbb_Latin:n #1 {
1343   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
1344   \um_set_mathalphabet_pos:Nnnn \mathbb {C} {up,it} {#1}

```

```

1345 \um_set_mathalphabet_pos:Nnnn \mathbb {H} {up,it} {#1}
1346 \um_set_mathalphabet_pos:Nnnn \mathbb {N} {up,it} {#1}
1347 \um_set_mathalphabet_pos:Nnnn \mathbb {P} {up,it} {#1}
1348 \um_set_mathalphabet_pos:Nnnn \mathbb {Q} {up,it} {#1}
1349 \um_set_mathalphabet_pos:Nnnn \mathbb {R} {up,it} {#1}
1350 \um_set_mathalphabet_pos:Nnnn \mathbb {Z} {up,it} {#1}
1351 }
1352 \cs_new:Npn \um_config_mathbb_num:n #1 {
1353   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
1354 }
1355 \cs_new:Npn \um_config_mathbb_misc:n #1 {
1356   \um_set_mathalphabet_pos:Nnnn \mathbb {Pi} {up,it} {#1}
1357   \um_set_mathalphabet_pos:Nnnn \mathbb {pi} {up,it} {#1}
1358   \um_set_mathalphabet_pos:Nnnn \mathbb {Gamma} {up,it} {#1}
1359   \um_set_mathalphabet_pos:Nnnn \mathbb {gamma} {up,it} {#1}
1360   \um_set_mathalphabet_pos:Nnnn \mathbb {summation} {up} {#1}
1361 }
1362 \cs_new:Npn \um_config_mathbbbit_misc:n #1 {
1363   \um_set_mathalphabet_pos:Nnnn \mathbbbit {D} {up,it} {#1}
1364   \um_set_mathalphabet_pos:Nnnn \mathbbbit {d} {up,it} {#1}
1365   \um_set_mathalphabet_pos:Nnnn \mathbbbit {e} {up,it} {#1}
1366   \um_set_mathalphabet_pos:Nnnn \mathbbbit {i} {up,it} {#1}
1367   \um_set_mathalphabet_pos:Nnnn \mathbbbit {j} {up,it} {#1}
1368 }

```

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

```

1369 \cs_new:Npn \um_config_mathscr_Latin:n #1 {
1370   \um_set_mathalphabet_Latin:Nnn \mathscr {up,it}{#1}
1371   \um_set_mathalphabet_pos:Nnnn \mathscr {B}{up,it}{#1}
1372   \um_set_mathalphabet_pos:Nnnn \mathscr {E}{up,it}{#1}
1373   \um_set_mathalphabet_pos:Nnnn \mathscr {F}{up,it}{#1}
1374   \um_set_mathalphabet_pos:Nnnn \mathscr {H}{up,it}{#1}
1375   \um_set_mathalphabet_pos:Nnnn \mathscr {I}{up,it}{#1}
1376   \um_set_mathalphabet_pos:Nnnn \mathscr {L}{up,it}{#1}
1377   \um_set_mathalphabet_pos:Nnnn \mathscr {M}{up,it}{#1}
1378   \um_set_mathalphabet_pos:Nnnn \mathscr {R}{up,it}{#1}
1379 }
1380 \cs_new:Npn \um_config_mathscr_latin:n #1 {
1381   \um_set_mathalphabet_latin:Nnn \mathscr {up,it}{#1}
1382   \um_set_mathalphabet_pos:Nnnn \mathscr {e}{up,it}{#1}
1383   \um_set_mathalphabet_pos:Nnnn \mathscr {g}{up,it}{#1}
1384   \um_set_mathalphabet_pos:Nnnn \mathscr {o}{up,it}{#1}
1385 }

```

9.1.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1386 \cs_new:Npn \um_config_mathfrak_Latin:n #1 {
1387   \um_set_mathalphabet_Latin:Nnn \mathfrak {up,it}{#1}

```

```

1388 \um_set_mathalphabet_pos:Nnnn \mathfrak {C}{up,it}{#1}
1389 \um_set_mathalphabet_pos:Nnnn \mathfrak {H}{up,it}{#1}
1390 \um_set_mathalphabet_pos:Nnnn \mathfrak {I}{up,it}{#1}
1391 \um_set_mathalphabet_pos:Nnnn \mathfrak {R}{up,it}{#1}
1392 \um_set_mathalphabet_pos:Nnnn \mathfrak {Z}{up,it}{#1}
1393 }
1394 \cs_new:Npn \um_config_mathfrak_latin:n #1 {
1395 \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{#1}
1396 }

```

9.1.6 Sans serif upright: \mathsfup

```

1397 \cs_new:Npn \um_config_mathsfup_num:n #1 {
1398 \um_set_mathalphabet_numbers:Nnn \mathsf {up}{#1}
1399 \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
1400 }
1401 \cs_new:Npn \um_config_mathsfup_Latin:n #1 {
1402 \bool_if:NTF \g_um_sfliteral_bool {
1403 \um_map_chars_Latin:nn {sfup} {#1}
1404 \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
1405 }{
1406 \bool_if:NT \g_um_upsans_bool {
1407 \um_map_chars_Latin:nn {sfup,sfit} {#1}
1408 \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1409 }
1410 }
1411 \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}
1412 }
1413 \cs_new:Npn \um_config_mathsfup_latin:n #1 {
1414 \bool_if:NTF \g_um_sfliteral_bool {
1415 \um_map_chars_latin:nn {sfup} {#1}
1416 \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
1417 }{
1418 \bool_if:NT \g_um_upsans_bool {
1419 \um_map_chars_latin:nn {sfup,sfit} {#1}
1420 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1421 }
1422 }
1423 \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
1424 }

```

9.1.7 Sans serif italic: \mathsfit

```

1425 \cs_new:Npn \um_config_mathsfital_Latin:n #1 {
1426 \bool_if:NTF \g_um_sfliteral_bool {
1427 \um_map_chars_Latin:nn {sfit} {#1}
1428 \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
1429 }{
1430 \bool_if:NF \g_um_upsans_bool {

```

```

1431     \um_map_chars_Latin:nn {sfup,sfit} {#1}
1432     \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1433   }
1434 }
1435 \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1436 }
1437 \cs_new:Npn \um_config_mathsf_latin:n #1 {
1438   \bool_if:NTF \g_um_sfliteral_bool {
1439     \um_map_chars_latin:nn {sfup,sfit} {#1}
1440     \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
1441   }{
1442     \bool_if:NF \g_um_upsans_bool {
1443       \um_map_chars_latin:nn {sfup,sfit} {#1}
1444       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1445     }
1446   }
1447   \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1448 }

```

9.1.8 Typewriter or monospaced: `\mathtt`

```

1449 \cs_new:Npn \um_config_mathtt_num:n #1 {
1450   \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
1451 }
1452 \cs_new:Npn \um_config_mathtt_Latin:n #1 {
1453   \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
1454 }
1455 \cs_new:Npn \um_config_mathtt_latin:n #1 {
1456   \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
1457 }

```

9.1.9 Bold Italic: `\mathbfit`

```

1458 \cs_new:Npn \um_config_mathbfit_Latin:n #1 {
1459   \bool_if:NF \g_um_bfupLatin_bool {
1460     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1461   }
1462   \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
1463   \bool_if:NTF \g_um_bfliteral_bool {
1464     \um_map_chars_Latin:nn {bfit} {#1}
1465     \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
1466   }{
1467     \bool_if:NF \g_um_bfupLatin_bool {
1468       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1469       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1470     }
1471   }
1472 }
1473 \cs_new:Npn \um_config_mathbfit_latin:n #1 {

```

```

1474 \bool_if:NF \g_um_bfuplatin_bool {
1475   \um_map_chars_latin:nn {bfup,bfit} {#1}
1476 }
1477 \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
1478 \bool_if:NTF \g_um_bfliteral_bool {
1479   \um_map_chars_latin:nn {bfit} {#1}
1480   \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
1481 }{
1482   \bool_if:NF \g_um_bfuplatin_bool {
1483     \um_map_chars_latin:nn {bfup,bfit} {#1}
1484     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1485   }
1486 }
1487 }
1488 \cs_new:Npn \um_config_mathbfit_Greek:n #1 {
1489   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
1490   \bool_if:NTF \g_um_bfliteral_bool {
1491     \um_map_chars_Greek:nn {bfit}{#1}
1492     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
1493   }{
1494     \bool_if:NF \g_um_bfupGreek_bool {
1495       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1496       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1497     }
1498   }
1499 }
1500 \cs_new:Npn \um_config_mathbfit_greek:n #1 {
1501   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
1502   \bool_if:NTF \g_um_bfliteral_bool {
1503     \um_map_chars_greek:nn {bfit} {#1}
1504     \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
1505   }{
1506     \bool_if:NF \g_um_bfupgreek_bool {
1507       \um_map_chars_greek:nn {bfit,bfup} {#1}
1508       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1509     }
1510   }
1511 }
1512 \cs_new:Npn \um_config_mathbfit_misc:n #1 {
1513   \bool_if:NTF \g_um_literal_Nabla_bool {
1514     \um_map_single:nnn {Nabla}{bfit}{#1}
1515   }{
1516     \bool_if:NF \g_um_upNabla_bool {
1517       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1518     }
1519   }

```



```

1520 \bool_if:NTF \g_um_literal_partial_bool {
1521   \um_map_single:nnn {partial}{bfit}{#1}
1522 }{
1523   \bool_if:NF \g_um_uppartial_bool {
1524     \um_map_single:nnn {partial}{bfup,bfit}{#1}
1525   }
1526 }
1527 \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{#1}
1528 \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{#1}
1529 \bool_if:NTF \g_um_literal_partial_bool {
1530   \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{#1}
1531 }{
1532   \bool_if:NF \g_um_uppartial_bool {
1533     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1534   }
1535 }
1536 \bool_if:NTF \g_um_literal_Nabla_bool {
1537   \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {it}{#1}
1538 }{
1539   \bool_if:NF \g_um_upNabla_bool {
1540     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1541   }
1542 }
1543 }

```

9.1.10 Bold Upright: \mathbf{bfup}

```

1544 \cs_new:Npn \um_config_mathbfup_num:n #1 {
1545   \um_set_mathalphabet_numbers:Nnn \mathbf {up}{#1}
1546   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
1547 }
1548 \cs_new:Npn \um_config_mathbfup_Latin:n #1 {
1549   \bool_if:NT \g_um_bfupLatin_bool {
1550     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1551   }
1552   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
1553   \bool_if:NTF \g_um_bfliteral_bool {
1554     \um_map_chars_Latin:nn {bfup} {#1}
1555     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
1556   }{
1557     \bool_if:NT \g_um_bfupLatin_bool {
1558       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1559       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1560     }
1561   }
1562 }
1563 \cs_new:Npn \um_config_mathbfup_latin:n #1 {

```

```

1564 \bool_if:NT \g_um_bfuplatin_bool {
1565   \um_map_chars_latin:nn {bfup,bfit} {#1}
1566 }
1567 \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
1568 \bool_if:NTF \g_um_bfliteral_bool {
1569   \um_map_chars_latin:nn {bfup} {#1}
1570   \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
1571 }{
1572   \bool_if:NT \g_um_bfuplatin_bool {
1573     \um_map_chars_latin:nn {bfup,bfit} {#1}
1574     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1575   }
1576 }
1577 }
1578 \cs_new:Npn \um_config_mathbfup_Greek:n #1 {
1579   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
1580   \bool_if:NTF \g_um_bfliteral_bool {
1581     \um_map_chars_Greek:nn {bfup}{#1}
1582     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}
1583   }{
1584     \bool_if:NT \g_um_bfupGreek_bool {
1585       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1586       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1587     }
1588   }
1589 }
1590 \cs_new:Npn \um_config_mathbfup_greek:n #1 {
1591   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
1592   \bool_if:NTF \g_um_bfliteral_bool {
1593     \um_map_chars_greek:nn {bfup} {#1}
1594     \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
1595   }{
1596     \bool_if:NT \g_um_bfupgreek_bool {
1597       \um_map_chars_greek:nn {bfup,bfit} {#1}
1598       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1599     }
1600   }
1601 }
1602 \cs_new:Npn \um_config_mathbfup_misc:n #1 {
1603   \bool_if:NTF \g_um_literal_Nabla_bool {
1604     \um_map_single:nnn {Nabla}{bfup}{#1}
1605   }{
1606     \bool_if:NT \g_um_upNabla_bool {
1607       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1608     }
1609   }

```

```

1610 \bool_if:NTF \g_um_literal_partial_bool {
1611   \um_map_single:nnn {partial}{bfup}{#1}
1612 }{
1613   \bool_if:NT \g_um_uppartial_bool {
1614     \um_map_single:nnn {partial}{bfup,bfit}{#1}
1615   }
1616 }
1617 \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{#1}
1618 \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{#1}
1619 \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{#1}
1620 \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{#1}
1621 \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{#1}
1622 \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{#1}
1623 \bool_if:NTF \g_um_literal_partial_bool {
1624   \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{#1}
1625 }{
1626   \bool_if:NT \g_um_uppartial_bool {
1627     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1628   }
1629 }
1630 \bool_if:NTF \g_um_literal_Nabla_bool {
1631   \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{#1}
1632 }{
1633   \bool_if:NT \g_um_upNabla_bool {
1634     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1635   }
1636 }
1637 }

```

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

```

1638 \cs_new:Npn \um_config_mathbffrak_Latin:n #1 {
1639   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
1640 }
1641 \cs_new:Npn \um_config_mathbffrak_latin:n #1 {
1642   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
1643 }

```

9.1.12 Bold script or calligraphic: `\mathbfsc`

```

1644 \cs_new:Npn \um_config_mathbfsc_Latin:n #1 {
1645   \um_set_mathalphabet_Latin:Nnn \mathbfsc {up,it}{#1}
1646 }
1647 \cs_new:Npn \um_config_mathbfsc_latin:n #1 {
1648   \um_set_mathalphabet_latin:Nnn \mathbfsc {up,it}{#1}
1649 }

```

9.1.13 Bold upright sans serif: `\mathbfsfup`

```

1650 \cs_new:Npn \um_config_mathbfsfup_num:n #1 {
1651   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{#1}
1652   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
1653 }
1654 \cs_new:Npn \um_config_mathbfsfup_Latin:n #1 {
1655   \bool_if:NTF \g_um_sfliteral_bool {
1656     \um_map_chars_Latin:nn {bfsfup} {#1}
1657     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
1658   }{
1659     \bool_if:NT \g_um_upsans_bool {
1660       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1661       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1662     }
1663   }
1664   \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
1665 }
1666 \cs_new:Npn \um_config_mathbfsfup_latin:n #1 {
1667   \bool_if:NTF \g_um_sfliteral_bool {
1668     \um_map_chars_latin:nn {bfsfup} {#1}
1669     \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
1670   }{
1671     \bool_if:NT \g_um_upsans_bool {
1672       \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1673       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1674     }
1675   }
1676   \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
1677 }
1678 \cs_new:Npn \um_config_mathbfsfup_Greek:n #1 {
1679   \bool_if:NTF \g_um_sfliteral_bool {
1680     \um_map_chars_Greek:nn {bfsfup}{#1}
1681     \um_set_mathalphabet_Greek:Nnn \mathbfsf {up}{#1}
1682   }{
1683     \bool_if:NT \g_um_upsans_bool {
1684       \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
1685       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
1686     }
1687   }
1688   \um_set_mathalphabet_Greek:Nnn \mathbfsfup {up,it}{#1}
1689 }
1690 \cs_new:Npn \um_config_mathbfsfup_greek:n #1 {
1691   \bool_if:NTF \g_um_sfliteral_bool {
1692     \um_map_chars_greek:nn {bfsfup} {#1}
1693     \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
1694   }{
1695     \bool_if:NT \g_um_upsans_bool {

```

```

1696     \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1697     \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1698   }
1699 }
1700 \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
1701 }
1702 \cs_new:Npn \um_config_mathbfsfup_misc:n #1 {
1703   \bool_if:NTF \g_um_literal_Nabla_bool {
1704     \um_map_single:nnn {Nabla}{bfsfup}{#1}
1705   }{
1706     \bool_if:NT \g_um_upNabla_bool {
1707       \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
1708     }
1709   }
1710   \bool_if:NTF \g_um_literal_partial_bool {
1711     \um_map_single:nnn {partial}{bfsfup}{#1}
1712   }{
1713     \bool_if:NT \g_um_uppartial_bool {
1714       \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
1715     }
1716   }
1717   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{#1}
1718   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{#1}
1719   \bool_if:NTF \g_um_literal_partial_bool {
1720     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{#1}
1721   }{
1722     \bool_if:NT \g_um_uppartial_bool {
1723       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1724     }
1725   }
1726   \bool_if:NTF \g_um_literal_Nabla_bool {
1727     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{#1}
1728   }{
1729     \bool_if:NT \g_um_upNabla_bool {
1730       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1731     }
1732   }
1733 }

```

9.1.14 Bold italic sans serif: \mathbfsfit

```

1734 \cs_new:Npn \um_config_mathbfsfit_Latin:n #1 {
1735   \bool_if:NTF \g_um_sfliteral_bool {
1736     \um_map_chars_Latin:nn {bfsfit} {#1}
1737     \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
1738   }{
1739     \bool_if:NF \g_um_upsans_bool {

```

```

1740     \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1741     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1742   }
1743 }
1744 \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
1745 }
1746 \cs_new:Npn \um_config_mathbfsfit_latin:n #1 {
1747   \bool_if:NTF \g_um_sfliteral_bool {
1748     \um_map_chars_latin:nn {bfsfit} {#1}
1749     \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
1750   }{
1751     \bool_if:NF \g_um_upsans_bool {
1752       \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1753       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1754     }
1755   }
1756   \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{#1}
1757 }
1758 \cs_new:Npn \um_config_mathbfsfit_greek:n #1 {
1759   \bool_if:NTF \g_um_sfliteral_bool {
1760     \um_map_chars_greek:nn {bfsfit}{#1}
1761     \um_set_mathalphabet_greek:Nnn \mathbfsf {it}{#1}
1762   }{
1763     \bool_if:NF \g_um_upsans_bool {
1764       \um_map_chars_greek:nn {bfsfup,bfsfit}{#1}
1765       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it}{#1}
1766     }
1767   }
1768   \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it}{#1}
1769 }
1770 \cs_new:Npn \um_config_mathbfsfit_greek:n #1 {
1771   \bool_if:NTF \g_um_sfliteral_bool {
1772     \um_map_chars_greek:nn {bfsfit} {#1}
1773     \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
1774   }{
1775     \bool_if:NF \g_um_upsans_bool {
1776       \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1777       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1778     }
1779   }
1780   \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
1781 }
1782 \cs_new:Npn \um_config_mathbfsfit_misc:n #1 {
1783   \bool_if:NTF \g_um_literal_Nabla_bool {
1784     \um_map_single:nnn {Nabla}{bfsfit}{#1}
1785   }{

```

```

1786 \bool_if:NF \g_um_upNabla_bool {
1787   \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
1788 }
1789 }
1790 \bool_if:NTF \g_um_literal_partial_bool {
1791   \um_map_single:nnn {partial}{bfsfit}{#1}
1792 }{
1793   \bool_if:NF \g_um_uppartial_bool {
1794     \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
1795   }
1796 }
1797 \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{#1}
1798 \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{#1}
1799 \bool_if:NTF \g_um_literal_partial_bool {
1800   \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{#1}
1801 }{
1802   \bool_if:NF \g_um_uppartial_bool {
1803     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1804   }
1805 }
1806 \bool_if:NTF \g_um_literal_Nabla_bool {
1807   \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{#1}
1808 }{
1809   \bool_if:NF \g_um_upNabla_bool {
1810     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1811   }
1812 }
1813 }

```

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 `"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.

```

1814 \beginingroup
1815   \char_make_other:N \^
1816   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
1817     \lowercase{
1818       \tl_rescan:nn {

```

```

1819         \char_make_other:N \{
1820         \char_make_other:N \}
1821         \char_make_other:N \&
1822         \char_make_other:N \%
1823         \char_make_other:N \$
1824     }{
1825         \global\let#1=^^^^^#2
1826     }
1827 }
1828 }

```

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

```

1829 \gdef\um@scanactivedef"#1\@nil#2{
1830     \lowercase{
1831         \tl_rescan:nn{
1832             \ExplSyntaxOn
1833             \char_make_math_superscript:N\^
1834         }{
1835             \global\def^^^^^#1{#2}
1836         }
1837     }
1838 }
1839 \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`.

```

1840 \AtBeginDocument{
1841     \group_begin:
1842         \char_make_math_superscript:N\^
1843         \def\UnicodeMathSymbol#1#2#3#4{
1844             \bool_if:nF { \cs_if_eq_p:NN #3 \mathaccent ||
1845                 \cs_if_eq_p:NN #3 \mathopen ||
1846                 \cs_if_eq_p:NN #3 \mathclose } {
1847                 \um@scancharlet#2=#1\@nil\ignorespaces
1848             }
1849         }
1850         \char_make_other:N \#
1851         \@input{unicode-math-table.tex}
1852     \group_end:
1853 }

```

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

```

1854 \group_begin:
1855     \lccode`\*=\ \
1856     \char_make_escape:N \

```



```

1857 \char_make_other:N \\\
1858 |lowercase{
1859   |AtBeginDocument{
1860     |let|backslash=*
1861   }
1862 }
1863 |group_end:
Fix \backslash:

```

11 Epilogue

Lots of little things to tidy up.

11.0.15 Primes

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

```

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

```

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

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

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

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

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

Our algorithm is:

- Prime encountered; pcount=1.

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

```

1864 \muskip_new:N \g_um_primekern_muskip
1865 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
1866 \int_new:N \l_um_primecount_int

1867 \cs_new:Npn \um_nprimes:Nn #1#2 {
1868   ^{
1869     #1
1870     \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
1871   }
1872 }
1873 \cs_new:Npn \um_nprimes_select:nn #1#2 {
1874   \prg_case_int:nnn {#2}{
1875     {1} { ^{#1} }
1876     {2} {
1877       \um_glyph_if_exist:nTF {"2033} { ^{\um_prime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1878     }
1879     {3} {
1880       \um_glyph_if_exist:nTF {"2034} { ^{\um_prime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1881     }
1882     {4} {
1883       \um_glyph_if_exist:nTF {"2057} { ^{\um_prime_quad_mchar} } {\um_nprimes:Nn #1 {#2}}
1884     }
1885   }{
1886     \um_nprimes:Nn #1 {#2}
1887   }
1888 }
1889 \cs_new:Npn \um_nbackprimes_select:nn #1#2 {
1890   \prg_case_int:nnn {#2}{
1891     {1} { ^{#1} }
1892     {2} {
1893       \um_glyph_if_exist:nTF {"2033} { ^{\um_backprime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1894     }
1895     {3} {
1896       \um_glyph_if_exist:nTF {"2034} { ^{\um_backprime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1897     }
1898   }{
1899     \um_nprimes:Nn #1 {#2}
1900   }

```

1901 }

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

```
1902 \cs_new:Npn \um_scan_prime: {
1903   \int_zero:N \l_um_primecount_int
1904   \um_scanprime_collect:N \um_prime_single_mchar
1905 }
1906 \cs_new:Npn \um_scan_dprime: {
1907   \int_set:Nn \l_um_primecount_int {1}
1908   \um_scanprime_collect:N \um_prime_single_mchar
1909 }
1910 \cs_new:Npn \um_scan_trprime: {
1911   \int_set:Nn \l_um_primecount_int {2}
1912   \um_scanprime_collect:N \um_prime_single_mchar
1913 }
1914 \cs_new:Npn \um_scan_qprime: {
1915   \int_set:Nn \l_um_primecount_int {3}
1916   \um_scanprime_collect:N \um_prime_single_mchar
1917 }
1918 \cs_new:Npn \um_scanprime_collect:N #1 {
1919   \int_incr:N \l_um_primecount_int
1920   \peek_meaning_remove:NTF ' {
1921     \um_scanprime_collect:N #1
1922   }{
1923     \peek_meaning_remove:NTF \um_scan_prime: {
1924       \um_scanprime_collect:N #1
1925     }{
1926       \peek_meaning_remove:NTF ^^^^2032 {
1927         \um_scanprime_collect:N #1
1928       }{
1929         \peek_meaning_remove:NTF \um_scan_dprime: {
1930           \int_incr:N \l_um_primecount_int
1931           \um_scanprime_collect:N #1
1932         }{
1933           \peek_meaning_remove:NTF ^^^^2033 {
1934             \int_incr:N \l_um_primecount_int
1935             \um_scanprime_collect:N #1
1936           }{
1937             \peek_meaning_remove:NTF \um_scan_trprime: {
1938               \int_add:Nn \l_um_primecount_int {2}
1939               \um_scanprime_collect:N #1
1940             }{
1941               \peek_meaning_remove:NTF ^^^^2034 {
1942                 \int_add:Nn \l_um_primecount_int {2}
1943                 \um_scanprime_collect:N #1
1944               }{
1945                 \peek_meaning_remove:NTF \um_scan_qprime: {
```

```

1946         \int_add:Nn \l_um_primecount_int {3}
1947         \um_scanprime_collect:N #1
1948     }{
1949         \peek_meaning_remove:NTF ^^^^2057 {
1950             \int_add:Nn \l_um_primecount_int {3}
1951             \um_scanprime_collect:N #1
1952         }{
1953             \um_nprimes_select:nn {#1} {\l_um_primecount_int}
1954         }
1955     }
1956 }
1957 }
1958 }
1959 }
1960 }
1961 }
1962 }
1963 }
1964 \cs_new:Npn \um_scan_backprime: {
1965     \int_zero:N \l_um_primecount_int
1966     \um_scanbackprime_collect:N \um_backprime_single_mchar
1967 }
1968 \cs_new:Npn \um_scan_backdprime: {
1969     \int_set:Nn \l_um_primecount_int {1}
1970     \um_scanbackprime_collect:N \um_backprime_single_mchar
1971 }
1972 \cs_new:Npn \um_scan_backtrprime: {
1973     \int_set:Nn \l_um_primecount_int {2}
1974     \um_scanbackprime_collect:N \um_backprime_single_mchar
1975 }
1976 \cs_new:Npn \um_scanbackprime_collect:N #1 {
1977     \int_incr:N \l_um_primecount_int
1978     \peek_meaning_remove:NTF ` {
1979         \um_scanbackprime_collect:N #1
1980     }{
1981         \peek_meaning_remove:NTF \um_scan_backprime: {
1982             \um_scanbackprime_collect:N #1
1983         }{
1984             \peek_meaning_remove:NTF ^^^^2035 {
1985                 \um_scanbackprime_collect:N #1
1986             }{
1987                 \peek_meaning_remove:NTF \um_scan_backdprime: {
1988                     \int_incr:N \l_um_primecount_int
1989                     \um_scanbackprime_collect:N #1
1990                 }{
1991                     \peek_meaning_remove:NTF ^^^^2036 {

```

```

1992         \int_incr:N \l_um_primecount_int
1993         \um_scanbackprime_collect:N #1
1994     }{
1995         \peek_meaning_remove:NTF \um_scan_backtrprime: {
1996             \int_add:Nn \l_um_primecount_int {2}
1997             \um_scanbackprime_collect:N #1
1998         }{
1999             \peek_meaning_remove:NTF ^^^^2037 {
2000                 \int_add:Nn \l_um_primecount_int {2}
2001                 \um_scanbackprime_collect:N #1
2002             }{
2003                 \um_nbackprimes_select:nn {#1} {\l_um_primecount_int}
2004             }
2005         }
2006     }
2007 }
2008 }
2009 }
2010 }
2011 }
2012 \AtBeginDocument {
2013     \cs_set_eq:NN \prime          \um_scan_prime:
2014     \cs_set_eq:NN \drime          \um_scan_dprime:
2015     \cs_set_eq:NN \trprime        \um_scan_trprime:
2016     \cs_set_eq:NN \qprime         \um_scan_qprime:
2017     \cs_set_eq:NN \backprime      \um_scan_backprime:
2018     \cs_set_eq:NN \backdprime     \um_scan_backdprime:
2019     \cs_set_eq:NN \backtrprime    \um_scan_backtrprime:
2020 }
2021 \group_begin:
2022     \char_make_active:N \'
2023     \char_make_active:N `
2024     \char_make_active:n {"2032}
2025     \char_make_active:n {"2033}
2026     \char_make_active:n {"2034}
2027     \char_make_active:n {"2057}
2028     \char_make_active:n {"2035}
2029     \char_make_active:n {"2036}
2030     \char_make_active:n {"2037}
2031 \AtBeginDocument{
2032     \cs_set_eq:NN '          \um_scan_prime:
2033     \cs_set_eq:NN ^^^^2032 \um_scan_prime:
2034     \cs_set_eq:NN ^^^^2033 \um_scan_dprime:
2035     \cs_set_eq:NN ^^^^2034 \um_scan_trprime:
2036     \cs_set_eq:NN ^^^^2057 \um_scan_qprime:
2037     \cs_set_eq:NN `          \um_scan_backprime:

```

```

2038 \cs_set_eq:NN ^^^^2035 \um_scan_backprime:
2039 \cs_set_eq:NN ^^^^2036 \um_scan_backdprime:
2040 \cs_set_eq:NN ^^^^2037 \um_scan_backtrprime:
2041 }
2042 \group_end:

```

11.0.16 Unicode radicals

`\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`.

```

2043 \cs_set_nopar:Npn \r@@t #1#2 {
2044   \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
2045   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen63\l_um_font}
2046   \raise \dimexpr(
2047     \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-
2048     \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
2049   )\relax
2050   \copy \rootbox
2051   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen64\l_um_font}
2052   \box \z@
2053 }

```

`\um_fontdimen_to_percent:nn` #1 : Font dimen number
 #2 : Font 'variable'
`\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.

```

2054 \cs_new:Npn \um_fontdimen_to_percent:nn #1#2 {
2055   0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
2056 }

```

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

```

2057 \cs_new:Npn \um_mathstyle_scale:Nnn #1#2#3 {
2058   \ifx#1\scriptstyle
2059     #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
2060   \else
2061     \ifx#1\scriptscriptstyle
2062       #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
2063     \else

```

```

2064     #2#3
2065     \fi
2066 \fi
2067 }

```

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 $\text{Xe}\text{L}\text{A}\text{TeX}$ 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:

```

2068 \prop_new:N \g_um_supers_prop
2069 \prop_new:N \g_um_subs_prop
2070
2071 \group_begin:
2072
2073 % Populate a property list with superscript characters; their mean-
2074 %   ing as their key,
2075 % for reasons that will become apparent soon, and their replace-
2076 %   ment as each key's value.
2077 % Then make the superscript active and bind it to the scanning function.
2078 %
2079 % \cs{scantokens} makes this process much simpler since we can acti-
2080 %   vate the char
2081 % and assign its meaning in one step.
2082 \cs_set:Npn \um_setup_active_superscript:nn #1#2 {
2083   \prop_gput:Nxn \g_um_supers_prop {\meaning #1} {#2}
2084   \char_make_active:N #1
2085   \char_gmake_mathactive:N #1
2086   \scantokens{
2087     \cs_gset:Npn #1 {
2088       \tl_set:Nn \l_um_ss_chain_tl {#2}
2089       \cs_set_eq:NN \um_sub_or_super:n \sp
2090       \tl_set:Nn \l_um_tmpa_tl {supers}
2091       \um_scan_ssript:
2092     }
2093   }
2094 }
2095
2096 \um_setup_active_superscript:nn {^^^2070} {0}
2097 \um_setup_active_superscript:nn {^^^00b9} {1}

```

```

2095 \um_setup_active_superscript:nn {^^^00b2} {2}
2096 \um_setup_active_superscript:nn {^^^00b3} {3}
2097 \um_setup_active_superscript:nn {^^^2074} {4}
2098 \um_setup_active_superscript:nn {^^^2075} {5}
2099 \um_setup_active_superscript:nn {^^^2076} {6}
2100 \um_setup_active_superscript:nn {^^^2077} {7}
2101 \um_setup_active_superscript:nn {^^^2078} {8}
2102 \um_setup_active_superscript:nn {^^^2079} {9}
2103 \um_setup_active_superscript:nn {^^^207a} {+}
2104 \um_setup_active_superscript:nn {^^^207b} {-}
2105 \um_setup_active_superscript:nn {^^^207c} {=}
2106 \um_setup_active_superscript:nn {^^^207d} {(}
2107 \um_setup_active_superscript:nn {^^^207e} {)}
2108 \um_setup_active_superscript:nn {^^^207i} {i}
2109 \um_setup_active_superscript:nn {^^^207f} {n}
2110
2111 % Ditto above.
2112 \cs_set:Npn \um_setup_active_subscript:nn #1#2 {
2113   \prop_gput:Nxn \g_um_subs_prop {\meaning #1} {#2}
2114   \char_make_active:N #1
2115   \char_gmake_mathactive:N #1
2116   \scantokens{
2117     \cs_gset:Npn #1 {
2118       \tl_set:Nn \l_um_ss_chain_tl {#2}
2119       \cs_set_eq:NN \um_sub_or_super:n \sb
2120       \tl_set:Nn \l_um_tmpa_tl {subs}
2121       \um_scan_sscript:
2122     }
2123   }
2124 }
2125
2126 \um_setup_active_subscript:nn {^^^2080} {0}
2127 \um_setup_active_subscript:nn {^^^2081} {1}
2128 \um_setup_active_subscript:nn {^^^2082} {2}
2129 \um_setup_active_subscript:nn {^^^2083} {3}
2130 \um_setup_active_subscript:nn {^^^2084} {4}
2131 \um_setup_active_subscript:nn {^^^2085} {5}
2132 \um_setup_active_subscript:nn {^^^2086} {6}
2133 \um_setup_active_subscript:nn {^^^2087} {7}
2134 \um_setup_active_subscript:nn {^^^2088} {8}
2135 \um_setup_active_subscript:nn {^^^2089} {9}
2136 \um_setup_active_subscript:nn {^^^208a} {+}
2137 \um_setup_active_subscript:nn {^^^208b} {-}
2138 \um_setup_active_subscript:nn {^^^208c} {=}
2139 \um_setup_active_subscript:nn {^^^208d} {(}
2140 \um_setup_active_subscript:nn {^^^208e} {)}

```



```

2141 \um_setup_active_subscript:nn {^^^2090} {a}
2142 \um_setup_active_subscript:nn {^^^2091} {e}
2143 \um_setup_active_subscript:nn {^^^1d62} {i}
2144 \um_setup_active_subscript:nn {^^^2092} {o}
2145 \um_setup_active_subscript:nn {^^^1d63} {r}
2146 \um_setup_active_subscript:nn {^^^1d64} {u}
2147 \um_setup_active_subscript:nn {^^^1d65} {v}
2148 \um_setup_active_subscript:nn {^^^2093} {x}
2149 \um_setup_active_subscript:nn {^^^1d66} {\beta}
2150 \um_setup_active_subscript:nn {^^^1d67} {\gamma}
2151 \um_setup_active_subscript:nn {^^^1d68} {\rho}
2152 \um_setup_active_subscript:nn {^^^1d69} {\phi}
2153 \um_setup_active_subscript:nn {^^^1d6a} {\chi}
2154
2155 \group_end:
2156
2157 % The scanning command, evident in its purpose:
2158 \cs_new:Npn \um_scan_sscript: {
2159   \um_scan_sscript:TF {
2160     \um_scan_sscript:
2161   }{
2162     \um_sub_or_super:n {\l_um_ss_chain_tl}
2163   }
2164 }
2165
2166 % The main theme here is stolen from the source to the vari-
2167   ous \cs{peek_} functions.
2168 % Consider this function as simply boilerplate:
2169 \cs_new:Npn \um_scan_sscript:TF #1#2 {
2170   \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
2171   \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
2172   \tl_set:Nx \l_peek_false_tl { \exp_not:n{\group_align_safe_end: #2}}
2173   \group_align_safe_begin:
2174   \peek_after:NN \um_peek_execute_branches_ss:
2175 }
2176
2177 % We do not skip spaces when scanning ahead, and we explicitly wish to
2178 % bail out on encountering a space or a brace.
2179 \cs_new:Npn \um_peek_execute_branches_ss: {
2180   \bool_if:nTF {
2181     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2182     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2183     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2184   }
2185   { \l_peek_false_tl }
2186   { \um_peek_execute_branches_ss_aux: }

```

```

2186 }
2187
2188 % This is the actual comparison code.
2189 % Because the peeking has already tokenised the next token,
2190 % it's too late to extract its charcode directly. Instead,
2191 % we look at its meaning, which remains a `character' even
2192 % though it is itself math-active. If the character is ever
2193 % made fully active, this will break our assumptions!
2194 %
2195 % If the char's meaning exists as a property list key, we
2196 % build up a chain of sub-/superscripts and iterate. (If not, exit and
2197 % typeset what we've already collected.)
2198 \cs_new:Npn \um_peek_execute_branches_ss_aux: {
2199   \prop_if_in:cxTF
2200     {g_um_\l_um_tmpa_tl _prop}
2201     {\meaning\l_peek_token}
2202     {
2203       \prop_get:cxN
2204         {g_um_\l_um_tmpa_tl _prop}
2205         {\meaning\l_peek_token}
2206         \l_um_tmpb_tl
2207         \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
2208         \l_peek_true_tl
2209     }
2210     {\l_peek_false_tl}
2211 }

```

11.0.18 Active fractions

Active fractions can be setup independently of any maths font definition; all it requires is a mapping from the unicode input chars to the relevant \LaTeX fraction declaration.

```

2212 \cs_new:Npn \um_setup_active_frac: {
2213   \group_begin:
2214   \um_define_active_frac:Nw ^^^^2152 1/{10}
2215   \um_define_active_frac:Nw ^^^^2151 1/9
2216   \um_define_active_frac:Nw ^^^^215b 1/8
2217   \um_define_active_frac:Nw ^^^^2150 1/7
2218   \um_define_active_frac:Nw ^^^^2159 1/6
2219   \um_define_active_frac:Nw ^^^^2155 1/5
2220   \um_define_active_frac:Nw ^^^^00bc 1/4
2221   \um_define_active_frac:Nw ^^^^2153 1/3
2222   \um_define_active_frac:Nw ^^^^215c 3/8
2223   \um_define_active_frac:Nw ^^^^2156 2/5
2224   \um_define_active_frac:Nw ^^^^00bd 1/2
2225   \um_define_active_frac:Nw ^^^^2157 3/5

```

```

2226 \um_define_active_frac:Nw ^^^^215d 5/8
2227 \um_define_active_frac:Nw ^^^^2154 2/3
2228 \um_define_active_frac:Nw ^^^^00be 3/4
2229 \um_define_active_frac:Nw ^^^^2158 4/5
2230 \um_define_active_frac:Nw ^^^^215a 5/6
2231 \um_define_active_frac:Nw ^^^^215e 7/8
2232 \group_end:
2233 }
2234 \cs_new:Npn \um_define_active_frac:Nw #1 #2/#3 {
2235   \char_make_active:n {`#1}
2236   \char_gmake_mathactive:N #1
2237   \tl_rescan:nn {
2238     \ExplSyntaxOn
2239   }{
2240     \cs_gset:Npx #1 {
2241       \bool_if:NTF \l_um_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2242       {#2} {#3}
2243     }
2244   }
2245 }
2246 \um_setup_active_frac:

```

11.0.19 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`:

```

2247 \def\operator@font{\um_switchto_mathup:}
2248 \def\to{\rightarrow}
2249 \def\overrightarrow{\vec}
2250 \def\le{\leq}
2251 \def\ge{\geq}
2252 \def\neq{\ne}
2253 \def\triangle{\mathord{\bigtriangleup}}
2254 \def\bigcirc{\mdlgwhtcircle}
2255 \def\circ{\vysmwhtcircle}
2256 \def\bullet{\smblkcircle}
2257 \def\mathyen{\yen}
2258 \def\mathsterling{\sterling}

```

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

```

2259 \@ifpackageloaded{amsmath}{
2260   % define their own colon, perhaps I should just steal it. (It does look much bet-
2261   % ter.)
2262 }{
2263   \cs_set_protected:Npn \colon {

```

```

2263     \bool_if:NTF \g_um_literal_colon_bool {;} { \mathpunct{:} }
2264   }
2265 }

\mathcal
2266 \def\mathcal{\mathscr}

\mathrm
2267 \def\mathrm{\mathup}
2268 \let\mathfence\mathord

\digamma I might end up just changing these in the table.
\Digamma 2269 \def\digamma{\updigamma}
2270 \def\Digamma{\upDigamma}

```

11.0.20 Compatibility

```

\um_patch_pkg:nn #1 : package
#2 : code
If <package> is loaded either already or later in the preamble, <code> is executed
(after the package is loaded in the latter case).
2271 \cs_new:Npn \um_patch_pkg:nn #1#2 {
2272   \@ifpackageloaded {#1} {
2273     #2
2274   }{
2275     \um_after_pkg:nn {#1} {#2}
2276   }
2277 }

```

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

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

```

2278 \um_patch_pkg:nn {url} {
2279   \tl_put_left:Nn \Url@FormatString { \um_switchto_mathup: }
2280   \tl_put_right:Nn \UrlSpecials {
2281     \do\{\mathchar`\}
2282     \do\'\mathchar`\'
2283     \do\$\mathchar`\$
2284     \do\&\mathchar`\&
2285   }
2286 }

```

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

```
2287 \um_patch_pkg:n {amsmath} {
2288   \tl_remove_in:Nn \@begindocumenthook {
2289     \mathchardef\std@minus\mathcode`\-\relax
2290     \mathchardef\std@equal\mathcode`\=\relax
2291   }
2292   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
2293   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
2294   \def\@cdots{\mathinner{\cdots}}
2295   \cs_set_eq:NN \dotso@ \cdots
2296 }
```

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

```
2297 \um_patch_pkg:n {amsopn} {
2298   \cs_set:Npn \newmcodes@ {
2299     \mathcode`\'39\scan_stop:
2300     \mathcode`\*42\scan_stop:
2301     \mathcode`\."613A\scan_stop:
2302     %% \ifnum\mathcode`\-=45 \else
2303     %%   \mathchardef\std@minus\mathcode`\-\relax
2304     %% \fi
2305     \mathcode`\-45\scan_stop:
2306     \mathcode`\ /47\scan_stop:
2307     \mathcode`\:"603A\scan_stop:
2308   }
2309 }
```

Symbols

```
2310 \cs_set:Npn \l {\Vert}
2311 \mathinner items:
2312 \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2313 \cs_set:Npn \cdots {\mathinner{\unicodcdots}}
```

Accents

```
2313 \AtBeginDocument{
2314   \def\widehat{\hat}
2315   \def\widetilde{\tilde}
2316 }
```

beamer At end of the package so the warnings are defined.

```
2317 \AtEndOfPackage{
2318   \@ifclassloaded{beamer}{
2319     \ifbeamer@suppressreplacements\else
2320       \um_warning:n {disable-beamer}
2321     \beamer@suppressreplacementstrue
2322   \fi
2323 }{}
2324 }
```

12 Error messages

Wrapper functions:

```
2325 \cs_new:Npn \um_warning:n { \msg_warning:nn {unicode-math} }
2326 \cs_new:Npn \um_trace:n { \msg_trace:nn {unicode-math} }
2327 \cs_new:Npn \um_trace:nx { \msg_trace:nnx {unicode-math} }
2328 \msg_new:nnn {unicode-math} {maths-feature-only}
2329 {
2330   The~ '#1'~ font~ feature~ can~ only~ be~ used~ for~ maths~ fonts.
2331 }
2332 \msg_new:nnn {unicode-math} {disable-beamer}
2333 {
2334   Disabling~ beamer's~ math~ setup.\\
2335   Please~ load~ beamer~ with~ the~ [professionalfonts]~ class~ option.
2336 }
2337 \msg_new:nnn {unicode-math} {no-tfrac}
2338 {
2339   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\\
2340   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
2341 }
2342 \msg_new:nnn {unicode-math} {default-math-font}
2343 {
2344   Defining~ the~ default~ maths~ font~ as~ '#1'.
2345 }
2346 \msg_new:nnn {unicode-math} {setup-implicit}
2347 {
2348   Setup~ alphabets:~ implicit~ mode.
2349 }
2350 \msg_new:nnn {unicode-math} {setup-explicit}
2351 {
2352   Setup~ alphabets:~ explicit~ mode.
2353 }
2354 \msg_new:nnn {unicode-math} {alph-initialise}
2355 {
2356   Initialising~ \@backslashchar math#1.
```

```

2357 }
2358 \msg_new:nnn {unicode-math} {setup-alph}
2359 {
2360   Setup~ alphabet:~ #1.
2361 }

    The end.
2362 \ExplSyntaxOff
2363 \errorcontextlines=999

```

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

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

```

2364 < See stix-extract.sh for now. >

```

A Documenting maths support in the NFSS

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

Maths symbol fonts Fonts for symbols: 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>}{NFSS decl.}
\SetMathAlphabet{<cmd>}{<maths version>}{NFSS decl.}
```

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

Delimiters and radicals use wrappers around \TeX ’s `\delimiter`/`\radical` primitives, which are re-designed in $\X_{\mathbb{E}}\TeX$. 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 $\X_{\mathbb{E}}\TeX$.

Accents are not included yet.

Summary For symbols, something like:

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

For characters, something like:

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


B X_YTeX math font dimensions

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

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

\fontdimen	Dimension name	Description
19	SUBSCRIPTTOPMAX	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: $\frac{1}{5}$ x-height.
20	SUBSCRIPTBASELINEDROPMIN	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.
21	SUPERSCRIPSHIFTUP	Standard shift up applied to superscript elements. Suggested: $os2.ySuperscriptYOffset$.
22	SUPERSCRIPSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPBOTTOMMIN	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height.
24	SUPERSCRIPBASELINEDROP-MAX	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SUBSUPERSCRIPGAPMIN	Minimum gap between the superscript and subscript ink. Suggested: $4 \times$ default rule thickness.
26	SUPERSCRIPBOTTOMMAX-WITHSUBSCRIPT	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: $\frac{1}{5}$ x-height.
27	SPACEAFTERSCRIP	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.

\fontdimen	Dimension name	Description
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	STACKTOPDISPLAYSTYLESHIFTUP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	STACKBOTTOMDISPLAYSTYLE-SHIFTDOWN	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	STACKGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.
37	STACKDISPLAYSTYLEGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	STRETCHSTACKTOPSHIFTUP	Standard shift up applied to the top element of the stretch stack.
39	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.

\fontdimen	Dimension name	Description
43	FRACTIONNUMERATOR- DISPLAYSTYLESHIFTUP	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FRACTIONDENOMINATORSHIFT- DOWN	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FRACTIONDENOMINATOR- DISPLAYSTYLESHIFTDOWN	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FRACTIONNUMERATORGAP- MIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	FRACTIONNUMDISPLAYSTYLE- GAPMIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
48	FRACTIONRULETHICKNESS	Thickness of the fraction bar. Suggested: default rule thickness.
49	FRACTIONDENOMINATORGAP- MIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FRACTIONDENOMDISPLAY- STYLEGAPMIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
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.

\fontdimen	Dimension name	Description
55	OVERBAREXTRAASCENDER	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UNDERBARVERTICALGAP	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UNDERBARRULETHICKNESS	Thickness of underbar. Suggested: default rule thickness.
58	UNDERBAREXTRADESCENDER	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RADICALVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RADICALDISPLAYSTYLE- VERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
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			
\#	1850	\`	727, 2023, 2281
\\$	1823, 2283	\	1856, 2310
\%	1822	Numbers	
\&	1821, 2284	\0	301
\'	726, 2022, 2282, 2299	\9	301
*	703, 1855, 2300		
\-	702, 2289, 2292, 2302, 2303, 2305	\	2339
\.	2301	A	
\/	734, 2306	\A	298
\:	705, 2307	\a	297
\<	738	\addnolimits	793
\=	2290, 2293	\addtoversion	526
\>	739	\AfterPackage	19
\@DeclareMathDelimiter	534	\alloc@	540
\@DeclareMathSizes	525	\alpha@elt	527
\@backslashchar	1180, 2356	\alpha@list	527
\@begindocumenthook	2288	\AtBeginDocument	
\@ccclvi	540		1031, 1840, 2012, 2031, 2313
\@cdots	2294	\AtEndOfPackage	1074, 2317
\@ehd	8	\AtEndPackage	16
\@empty	870, 900, 903	\awint	789
\@ifclassloaded	15, 2318	B	
\@ifpackageloaded	2259, 2272	\B	197
\@ii	869, 870, 872, 874, 877, 882	\backdprime	2018
\@input	662, 1851	\backprime	2017
\@marker	882, 897	\backtrprime	2019
\@nil	578, 709, 882, 891–894, 927, 1816, 1829, 1847	\beamer@suppressreplacementstrue	2321
\@preamblecmds	538	\begingroup	1814
\@tempa	873, 893, 895, 903	\beta	2149
\@tempb	896, 897, 900	\bgroup	1063
\@tempswafalse	871	\bigcirc	2254
\@tempwattrue	875, 878, 898, 901, 904, 907	\bigtriangleup	2253
\@xDeclareMathDelimiter	534	\bool_if:NF	
\@xxDeclareMathDelimiter	533		704, 1279, 1290, 1305, 1315, 1325, 1332, 1430, 1442, 1459, 1467, 1474, 1482, 1494, 1506, 1516,
\\	1855, 1857, 2334, 2339		
\{	1819		
\}	1820		
\^	1815, 1833, 1842		

1339, 1342, 1352, 1355, 1362, 1369, 1380, 1386, 1394, 1397, 1401, 1413, 1425, 1437, 1449, 1452, 1455, 1458, 1473, 1488, 1500, 1512, 1544, 1548, 1563, 1578, 1590, 1602, 1638, 1641, 1644, 1647, 1650, 1654, 1666, 1678, 1690, 1702, 1734, 1746, 1758, 1770, 1782, 1867, 1873, 1889, 1902, 1906, 1910, 1914, 1918, 1964, 1968, 1972, 1976, 2054, 2057, 2158, 2168, 2178, 2198, 2212, 2234, 2271, 2325–2327	\def 540, 801, 891, 893–896, 1835, 1843, 2247–2258, 2266, 2267, 2269, 2270, 2292–2294, 2314, 2315
\cs_set:cpn 1058	\define@key 803
\cs_set:Npn 542, 586, 590, 594, 693, 696, 774, 781, 860, 864, 913, 931, 936, 941, 952, 958, 966, 1192, 2079, 2112, 2298, 2310–2312	\define@mathalphabet 526
\cs_set_eq:cc 62, 63	\define@mathgroup 526
\cs_set_eq:cN 1194	\Digamma 2269
\cs_set_eq:NN 16, 19, 646–650, 654–658, 663, 1120, 1121, 1134, 2013–2019, 2032–2040, 2086, 2119, 2295	\digamma 2269
\cs_set_nopar:Npn 2043	\dimexpr 633, 2046, 2055
\cs_set_protected:cpx 1061	\do 538, 2281–2284
\cs_set_protected:Npn 2262	\dorestore@version 528
\cs_to_str:N 549, 564, 565, 578, 580–582, 776, 1084, 1153	\dotsb@ 2295
\csname 578, 588, 592, 596, 599, 602, 605, 624, 769, 772, 1066	\dp 2048
	\drime 2014
D	E
\D 199	\E 200
\d 225	\e 226
\DeclareDocumentCommand 306, 622, 793, 796	\edef 892, 893
\DeclareMathAccent 532	\egroup 1059, 1065
\DeclareMathAlphabet 531	\else 4, 635, 876, 881, 899, 902, 905, 2060, 2063, 2302, 2319
\DeclareMathDelimiter 533	\else: 1200
\DeclareMathRadical 535	\encodingdefault 661
\DeclareMathSizes 524	\endcsname 578, 588, 592, 596, 599, 602, 605, 624, 769, 772, 1066
\DeclareMathSymbol 532	\endgroup 1839
\DeclareMathVersion 526, 626	\epsilon 1043
\DeclareSymbolFont 529, 660	\errorcontextlines 2363
\DeclareSymbolFontAlphabet 536	\etex_iffontchar:D 1198
\DeclareSymbolFontAlphabet@ 536	\exp_after:wN 27, 28, 849, 852, 1128, 1130, 1132, 1154
	\exp_args:Ncc 961
	\exp_args:NNcc 26, 1021
	\exp_args:NV 1158, 1162
	\exp_not:c 582, 1041, 1069
	\exp_not:N 2241
	\exp_not:n 583, 1062, 2169, 2171
	\exp_not:V 834–836
	\expandafter 872, 874, 877, 882, 892, 893, 897, 1065, 1066
	\ExplSyntaxOff 2362
	\ExplSyntaxOn 14, 1832, 2238
	F
	\F 201
	\f@size 624

<code>\fi</code>	9, 336, 384, 408, 420, 433, 446, 642, 879, 880, 883, 887, 888, 909–911, 2065, 2066, 2304, 2322
<code>\fi:</code>	1202
<code>\fint</code>	789
<code>\fontdimen</code>	633, 2045, 2051, 2055
<code>\fontname</code>	1145
<code>\fontspec_select:nn</code>	72
<code>\fontspec_select:xn</code>	672
<code>\frac</code>	2241
G	
<code>\g</code>	227
<code>\g_um_bfit_Greek_usv</code>	137
<code>\g_um_bfit_greek_usv</code>	138
<code>\g_um_bfit_Latin_usv</code>	135
<code>\g_um_bfit_latin_usv</code>	136
<code>\g_um_bfliteral_bool</code> 40, 390, 407, 1463, 1478, 1490, 1502, 1553, 1568, 1580, 1592
<code>\g_um_bfsfit_Greek_usv</code>	133
<code>\g_um_bfsfit_greek_usv</code>	134
<code>\g_um_bfsfit_Latin_usv</code>	131
<code>\g_um_bfsfit_latin_usv</code>	132
<code>\g_um_bfsfup_Greek_usv</code>	133
<code>\g_um_bfsfup_greek_usv</code>	134
<code>\g_um_bfsfup_Latin_usv</code>	131
<code>\g_um_bfsfup_latin_usv</code>	132
<code>\g_um_bfup_Greek_usv</code>	137
<code>\g_um_bfup_greek_usv</code>	138
<code>\g_um_bfup_Latin_usv</code>	135
<code>\g_um_bfup_latin_usv</code>	136
<code>\g_um_bfupGreek_bool</code> 43, 137, 392, 397, 402, 1494, 1584
<code>\g_um_bfupgreek_bool</code> 44, 138, 393, 398, 403, 1506, 1596
<code>\g_um_bfupLatin_bool</code>	41, 135, 394, 399, 404, 1459, 1467, 1549, 1557
<code>\g_um_bfuplatin_bool</code>	42, 136, 395, 400, 405, 1474, 1482, 1564, 1572
<code>\g_um_default_mathalph_seq</code> 1087, 1105, 1112
<code>\g_um_fam_int</code>	54, 652, 653
<code>\g_um_it_h_usv</code>	302
<code>\g_um_literal_bool</code> 35, 313, 335, 1213, 1223, 1236, 1246, 1276, 1286, 1302, 1312
<code>\g_um_literal_colon_bool</code> 53, 462, 465, 704, 2263
<code>\g_um_literal_Nabla_bool</code>	49, 426, 432, 1256, 1322, 1513, 1536, 1603, 1630, 1703, 1726, 1783, 1806
<code>\g_um_literal_partial_bool</code>	50, 439, 445, 1263, 1329, 1520, 1529, 1610, 1623, 1710, 1719, 1790, 1799
<code>\g_um_math_alphabet_name_Greek_tl</code>	58
<code>\g_um_math_alphabet_name_greek_tl</code>	57
<code>\g_um_math_alphabet_name_Latin_tl</code>	56
<code>\g_um_math_alphabet_name_latin_tl</code>	55
<code>\g_um_math_alphabet_name_misc_tl</code>	60
<code>\g_um_math_alphabet_name_num_tl</code>	59
<code>\g_um_mathalph_seq</code>	854, <u>1073</u>
<code>\g_um_mathit_Greek_usv_range_tl</code>	305
<code>\g_um_mathit_greek_usv_range_tl</code>	304
<code>\g_um_mathit_Latin_usv_range_tl</code>	303
<code>\g_um_mathit_latin_usv_range_tl</code>	302
<code>\g_um_mathup_Greek_usv_range_tl</code>	300
<code>\g_um_mathup_greek_usv_range_tl</code>	299
<code>\g_um_mathup_Latin_usv_range_tl</code>	298
<code>\g_um_mathup_latin_usv_range_tl</code>	297
<code>\g_um_mathup_num_usv_range_tl</code>	301
<code>\g_um_primekern_muskip</code>	1864, 1865, 1870
<code>\g_um_sfliteral_bool</code>	46, 419, 1402, 1414, 1426, 1438, 1655, 1667, 1679, 1691, 1735, 1747, 1759, 1771
<code>\g_um_slash_delimiter_usv</code> 471, 474, 477, 734–736
<code>\g_um_subs_prop</code>	2069, 2113
<code>\g_um_supers_prop</code>	2068, 2080
<code>\g_um_texgreek_bool</code>	51, 453, 456, 1044, 1047, 1050, 1053
<code>\g_um_upGreek_bool</code>	38, 133, 315, 320, 325, 330, 1239, 1305
<code>\g_um_upgreek_bool</code>	39, 134, 316, 321, 326, 331, 1249, 1315
<code>\g_um_upLatin_bool</code>	36, 131, 317, 322, 327, 332, 1216, 1279
<code>\g_um_uplatin_bool</code>	37, 132, 318, 323, 328, 333, 1226, 1290

<code>\g_um_upNabla_bool</code>	47,	<code>\init@restore@version</code>	528
	428, 430, 1259, 1325, 1516, 1539,	<code>\int</code>	787
	1606, 1633, 1706, 1729, 1786, 1809	<code>\int_add:Nn</code>	
<code>\g_um_uppartial_bool</code>	48,		1938, 1942, 1946, 1950, 1996, 2000
	441, 443, 1266, 1332, 1523, 1532,	<code>\int_incr:N</code>	652,
	1613, 1626, 1713, 1722, 1793, 1802		1919, 1930, 1934, 1977, 1988, 1992
<code>\g_um_upsans_bool</code>	45, 415, 417, 1406,	<code>\int_new:N</code>	54, 1866
	1418, 1430, 1442, 1659, 1671,	<code>\int_set:Nn</code>	1907, 1911, 1915, 1969, 1973
	1683, 1695, 1739, 1751, 1763, 1775	<code>\int_use:N</code>	653
<code>\gamma</code>	2150	<code>\int_zero:N</code>	1903, 1965
<code>\gdef</code>	1829	<code>\intBar</code>	789
<code>\ge</code>	2251	<code>\intbar</code>	789
<code>\geq</code>	2251	<code>\intcap</code>	791
<code>\get@cdp</code>	530	<code>\intclockwise</code>	788
<code>\glb@currsz</code>	615	<code>\intcup</code>	791
<code>\global</code>	609, 612, 1825, 1835	<code>\intexpr_compare:nT</code>	
<code>\group@elt</code>	529		898, 901, 904, 906, 907
<code>\group@list</code>	529	<code>\intexpr_eval:n</code>	
<code>\group_align_safe_begin:</code>	2172		587, 588, 591, 592, 596, 885
<code>\group_align_safe_end:</code>	2171	<code>\intlarhk</code>	790
<code>\group_begin:</code>		<code>\intx</code>	790
	575, 1841, 1854, 2021, 2071, 2213		
<code>\group_end:</code>	579, 1852, 2042, 2155, 2232		
		J	
H		<code>\j</code>	230
<code>\H</code>	202		
<code>\h</code>	228	K	
<code>\hat</code>	2314	<code>\kern</code>	2045, 2051
<code>\hbox</code>	2044	<code>\keys_define:nn</code>	
<code>\ht</code>	2047		310, 340, 388, 412, 424, 437,
			450, 459, 468, 480, 496, 509, 817, 826
I		<code>\keys_set:nn</code>	308, 631
<code>\I</code>	203	<code>\KV_remove_surrounding_spaces:nw</code>	845
<code>\i</code>	229		
<code>\if@tempwa</code>	884	L	
<code>\ifbeamer@suppressreplacements</code>	2319	<code>\L</code>	204
<code>\ifcase</code>	314, 343, 391, 414, 427, 440	<code>\l_keys_choice_int</code>	
<code>\ifdim</code>	633		314, 343, 391, 414, 427, 440
<code>\ifluatex</code>	4	<code>\l_keys_key_tl</code>	512
<code>\ifnum</code>	2302	<code>\l_peek_false_tl</code>	2171, 2184, 2210
<code>\ifx</code>	870, 873,	<code>\l_peek_token</code>	2180–2182, 2201, 2205
	874, 877, 897, 900, 903, 2058, 2061	<code>\l_peek_true_aux_tl</code>	2169
<code>\ifxetex</code>	4	<code>\l_peek_true_tl</code>	2170, 2208
<code>\ignorespaces</code>	668, 1847	<code>\l_um_char_num_range_clist</code>	
<code>\iiiint</code>	787		618, 782, 885
<code>\iiint</code>	787	<code>\l_um_char_range_seq</code>	
<code>\iint</code>	787		617, 825, 829, 839, 869
		<code>\l_um_font</code>	633, 690, 1145, 1198,
			2045, 2047, 2048, 2051, 2059, 2062

<code>\l_umspec_feature_bool</code>		M
31, 671, 691, 804		<code>\M</code> 205
<code>\l_um_implicit_alph_bool</code>		<code>\m@th</code> 2044
34, 1113, 1119, 1178		<code>\mathaccent</code> 567, 1844
<code>\l_um_init_bool</code> 33, 616, 643, 828		<code>\mathalpha</code> 777, 924, 927
<code>\l_um_mathalph_seq</code>		<code>\mathbacktick</code> 727
619, 824, 830, 833, 1110, 1112, 1123		<code>\mathbb</code> 1076, 1091,
<code>\l_um_missing_alph_seq</code>		1340, 1343–1350, 1353, 1356–1360
. 1107, 1109, 1142, 1147, 1179		<code>\mathbbbit</code> 1076, 1092, 1363–1367
<code>\l_um_mversion_tf</code> 625, 626		<code>\mathbf</code> 1079, 1465,
<code>\l_um_nolimits_tl</code> . . . 583, <u>786</u> , 794, 797		1469, 1480, 1484, 1492, 1496,
<code>\l_um_ot_math_bool</code> 32, 634, 636		1504, 1508, 1530, 1533, 1537,
<code>\l_um_primecount_int</code> 1866,		1540, 1545, 1555, 1559, 1570,
1903, 1907, 1911, 1915, 1919,		1574, 1582, 1586, 1594, 1598,
1930, 1934, 1938, 1942, 1946,		1621, 1622, 1624, 1627, 1631, 1634
1950, 1953, 1965, 1969, 1973,		<code>\mathbffrak</code> 1080, 1101, 1639, 1642
1977, 1988, 1992, 1996, 2000, 2003		<code>\mathbfbit</code> 1079, 1099,
<code>\l_um_radicals_tl</code> 548, 799		1462, 1477, 1489, 1501, 1527, 1528
<code>\l_um_remap_alphabet_tl</code>		<code>\mathbfbscr</code> 1080, 1100, 1645, 1648
. 1126–1128, 1130, 1132, 1137		<code>\mathbfbsf</code> 1081, 1651, 1657, 1661, 1669,
<code>\l_um_script_features_tl</code> 627, 680, 819		1673, 1681, 1685, 1693, 1697,
<code>\l_um_script_font_tl</code> . . . 629, 679, 821		1720, 1723, 1727, 1730, 1737,
<code>\l_um_smallfrac_bool</code>		1741, 1749, 1753, 1761, 1765,
. 52, 484, 487, 492, 2241		1773, 1777, 1800, 1803, 1807, 1810
<code>\l_um_ss_chain_tl</code> 2085, 2118, 2162, 2207		<code>\mathbfbsffit</code> 1081, 1103,
<code>\l_um_sscript_features_tl</code> 628, 684, 820		1744, 1756, 1768, 1780, 1797, 1798
<code>\l_um_sscript_font_tl</code> . . . 630, 683, 822		<code>\mathbfbsfup</code> 1081, 1102, 1652,
<code>\l_um_tmpa_tl</code> 834, 845, 848,		1664, 1676, 1688, 1700, 1717, 1718
849, 851, 852, 854, 861, 865, 1124,		<code>\mathbfbup</code> 1079, 1098, 1546,
1128, 1137, 1153, 1154, 1156,		1552, 1567, 1579, 1591, 1617–1620
1169, 1171, 1172, 1175, 1176,		<code>\mathbin</code> 702, 703
1181, 1184, 2087, 2120, 2200, 2204		<code>\mathcal</code> <u>2266</u>
<code>\l_um_tmpb_tl</code>		<code>\mathchar</code> 2281–2284
. 835, 846, 866, 1125, 1133, 1135,		<code>\mathchar@type</code> 535, 588, 592, 596, 602, 605
1137, 1154, 1158, 1162, 2206, 2207		<code>\mathchardef</code> 2289, 2290, 2303
<code>\l_um_tmpc_tl</code> 836, 847, 862		<code>\mathclose</code> 556, 558, 559, 565, 1846
<code>\l_um_unknown_keys_clist</code>		<code>\mathcode</code> 609, 612,
. 307, 508, 511, 620, 687		2289, 2290, 2299–2303, 2305–2307
<code>\lccode</code> 1855		<code>\mathellipsis</code> 2311
<code>\le</code> 2250		<code>\mathfence</code> 561, 2268
<code>\left</code> <u>800</u>		<code>\mathfrak</code> . . 1077, 1094, 1387–1392, 1395
<code>\left@primitive</code> 800, 801		<code>\mathgroup</code> 540
<code>\leq</code> 2250		<code>\mathinner</code> 2294, 2311, 2312
<code>\let</code> 541, 615, 800, 1825, 2268		<code>\mathit</code> 1076, 1090, 1283,
<code>\lowercase</code> 1817, 1830		1297–1299, 1309, 1319, 1336, 1337
<code>\lowint</code> 791		<code>\mathop</code> 544, 580

<code>\mathopen</code> .. 547, 552, 553, 564, 801, 1845	<code>\newmathalphabet@@</code> 525
<code>\mathord</code> 719–727, 2253, 2268	<code>\newmathalphabet@@@</code> 525
<code>\mathpunct</code> 2263	<code>\newmcodes@</code> 2298
<code>\mathrel</code> 705	<code>\nolimits</code> 583
<code>\mathrm</code> 2267	<code>\non@alpherr</code> 1066
<code>\mathscr</code> 1077,	<code>\npolint</code> 790
1093, 1370–1378, 1381–1384, 2266	
<code>\mathsf</code> 1078, 1398, 1404, 1408,	
1416, 1420, 1428, 1432, 1440, 1444	
<code>\mathsf{fit}</code> 1078, 1097, 1435, 1447	
<code>\mathsf{fup}</code> .. 1078, 1096, 1399, 1411, 1423	
<code>\mathsterling</code> 2258	
<code>\mathstraightquote</code> 726	
<code>\mathtt</code> 1077, 1095, 1450, 1453, 1456	
<code>\mathup</code> 1076, 1089, 1210, 1220,	
1233, 1243, 1253, 1270–1273, 2267	
<code>\mathyen</code> 2257	
<code>\mddefault</code> 661	
<code>\mdlgwhtcircle</code> 2254	
<code>\meaning</code> 2080, 2113, 2201, 2205	
<code>\MessageBreak</code> 6	
<code>\mitepsilon</code> 1044, 1050	
<code>\mitphi</code> 1047, 1053	
<code>\mitvarepsilon</code> 1044, 1050	
<code>\mitvarphi</code> 1047, 1053	
<code>\mode_if_math:F</code> 1064	
<code>\msg_new:nnn</code> 2328, 2332,	
2337, 2342, 2346, 2350, 2354, 2358	
<code>\msg_redirect_module:nnn</code> 499, 502, 505	
<code>\msg_trace:nn</code> 2326	
<code>\msg_trace:nnx</code> 645, 2327	
<code>\msg_warning:nn</code> 2325	
<code>\mskip</code> 1870	
<code>\muskip_gset:Nn</code> 1865	
<code>\muskip_new:N</code> 1864	
N	
<code>\N</code> 206	
<code>\ne</code> 2252	
<code>\neq</code> 2252	
<code>\new@mathalphabet</code> 531	
<code>\new@mathgroup</code> 524, 540, 541	
<code>\new@mathversion</code> 529	
<code>\new@symbolfont</code> 530	
<code>\newcommand</code> 802, 868	
<code>\newfam</code> 541	
<code>\newmathalphabet</code> 525	
	O
	<code>\o</code> 231
	<code>\oiint</code> 787
	<code>\oiint</code> 787
	<code>\oint</code> 787
	<code>\ointctrclockwise</code> 788
	<code>\operator@font</code> 2247
	<code>\or</code> 319, 324,
	329, 334, 351, 359, 367, 375, 396,
	401, 406, 416, 418, 429, 431, 442, 444
	<code>\overrightarrow</code> 2249
	P
	<code>\P</code> 207
	<code>\PackageError</code> 5
	<code>\PackageWarningNoLine</code> 637
	<code>\peek_after:NN</code> 2173
	<code>\peek_meaning_remove:NTF</code>
	.. 1920, 1923, 1926, 1929, 1933,
	1937, 1941, 1945, 1949, 1978,
	1981, 1984, 1987, 1991, 1995, 1999
	<code>\phi</code> 1046, 2152
	<code>\pointint</code> 790
	<code>\prg_case_int:nnn</code> 1874, 1890
	<code>\prg_case_tl:Nnn</code> 543
	<code>\prg_do_nothing:</code> 1194
	<code>\prg_new_conditional:Nnn</code> .. 844, 1197
	<code>\prg_replicate:nn</code> 1870
	<code>\prg_return_false:</code> 857, 1201
	<code>\prg_return_true:</code> 855, 1199
	<code>\prg_stepwise_inline:nnnn</code> .. 914, 975
	<code>\prime</code> 2013
	<code>\process@table</code> 528
	<code>\ProcessKeysOptions</code> 522
	<code>\prop_get:cxN</code> 2203
	<code>\prop_get:NnN</code> 24
	<code>\prop_gput:Nnn</code> 23
	<code>\prop_gput:Nxn</code> 2080, 2113
	<code>\prop_if_in:cxTF</code> 2199
	<code>\prop_if_in:NnTF</code> 25

\prop_new:N	2068, 2069	\setbox	2044
\protect	2339	\SetMathAlphabet	531
\ProvidesPackage	1	\SetMathAlphabet@	531
Q		\setmathfont	614
\Q	208	\SetSymbolFont	530
\q_nil	845, 849, 852, 860, 864	\SetSymbolFont@	530
\qprime	2016	\sf@size	678, 682
R		\smbkcircle	2256
\R	209	\sp	2086
\r@@t	2043	\space	1066, 1148, 1181
\raise	2046	\sqint	790
\relax	615, 633, 874, 877, 897, 2049, 2055, 2289, 2290, 2292, 2293, 2303	\sqrt	799
\removenolimits	796	\sqrtsign	2044
\RequirePackage	10–13, 18	\std@equal	2290, 2293
\restore@mathversion	528	\std@minus	2289, 2292, 2303
\rho	2151	\sterling	2258
\rightarrow	2248	\string	892, 893
\rootbox	2050	\strip@pt	2055
\rppolint	789	\sumint	788
S		T	
\sb	2119	\tf@size	677, 678
\scan_stop:	588, 592, 596, 599, 602, 605, 609, 612, 1198, 2299–2301, 2305–2307	\tfrac	483, 519, 2241, 2339
\scantokens	2083, 2116	\thinmuskip	1865
\scpolint	790	\tilde	2315
\scriptscriptstyle	2061	\tl_if_empty:NT	1133
\scriptstyle	2058	\tl_if_empty:NTF	1127
\seq_clear:N	617, 619, 829, 830, 1109	\tl_if_eq:nnTF	1157, 1170
\seq_if_empty:NF	1142	\tl_if_in:NnT	583, 848, 851
\seq_if_empty:NTF	1110	\tl_if_in:NnTF	548
\seq_if_in:NnTF	22	\tl_map_inline:nn	523, 1075
\seq_if_in:NVTF	854	\tl_new:Nn	297–305, 786, 799
\seq_map_inline:Nn	1123, 1147	\tl_put_left:Nn	2279
\seq_map_variable:NNn	869	\tl_put_right:cx	776
\seq_new:N	824, 825, 1073, 1087, 1107	\tl_put_right:Nn	21, 794, 2280
\seq_put_right:Nn	839, 1083, 1105	\tl_put_right:NV	2207
\seq_put_right:Nx	833, 1179	\tl_remove_all_in:Nn	797
\seq_set_eq:NN	1112	\tl_remove_in:Nn	538, 2288
\set@@mathdelimiter	535	\tl_rescan:nn	1818, 1831, 2237
\set@mathaccent	532	\tl_set:cn	74
\set@mathchar	532	\tl_set:cx	1041
\set@mathdelimiter	534	\tl_set:Nf	845
\set@mathsymbol	533	\tl_set:Nn	55–60, 471, 474, 477, 625, 627–630, 644, 846, 847, 861, 862, 865, 866, 1043, 1046, 1049, 1052, 1135, 2085, 2087, 2118, 2120
		\tl_set:No	1124–1126

<code>\tl_set:Nx</code>	653, 1128,	<code>\um_config_mathbfsfit_latin:n</code> ..	1746
1130, 1132, 1153, 1154, 2169, 2171		<code>\um_config_mathbfsfit_misc:n</code> ...	1782
<code>\tl_set_eq:NN</code>	690, 2170	<code>\um_config_mathbfsfup_greek:n</code> ..	1678
<code>\tl_use:c</code>	1181	<code>\um_config_mathbfsfup_greek:n</code> ..	1690
<code>\to</code>	2248	<code>\um_config_mathbfsfup_Latin:n</code> ..	1654
<code>\token_if_eq_catcode_p:NN</code> .	2180, 2181	<code>\um_config_mathbfsfup_latin:n</code> ..	1666
<code>\token_if_eq_meaning_p:NN</code>	2182	<code>\um_config_mathbfsfup_misc:n</code> ...	1702
<code>\token_to_str:N</code>	1128, 1130	<code>\um_config_mathbfsfup_num:n</code> ...	1650
<code>\triangle</code>	2253	<code>\um_config_mathbfup_greek:n</code> ...	1578
<code>\trprime</code>	2015	<code>\um_config_mathbfup_greek:n</code> ...	1590
<code>\typeout</code>	1143, 1148	<code>\um_config_mathbfup_Latin:n</code> ...	1548
U			
<code>\Udelcode</code>	769, 772	<code>\um_config_mathbfup_latin:n</code> ...	1563
<code>\Udelimiter</code>	602	<code>\um_config_mathbfup_misc:n</code>	1602
<code>\um@backslash</code>	873, 892	<code>\um_config_mathbfup_num:n</code>	1544
<code>\um@firstchar</code>	872, 893	<code>\um_config_mathfrak_Latin:n</code> ...	1386
<code>\um@firstof</code>	891–893	<code>\um_config_mathfrak_latin:n</code> ...	1394
<code>\um@parse@range</code>	882, 894	<code>\um_config_mathit_greek:n</code>	1301
<code>\um@parse@term</code>	697, 709, 868, 927	<code>\um_config_mathit_greek:n</code>	1311
<code>\um@radicals</code>	799	<code>\um_config_mathit_Latin:n</code>	1275
<code>\um@scanactivedef</code>	578, 1814	<code>\um_config_mathit_latin:n</code>	1285
<code>\um@scancharlet</code>	1814, 1847	<code>\um_config_mathit_misc:n</code>	1321
<code>\um@zf@feature</code>	802, 811, 814	<code>\um_config_mathscr_Latin:n</code>	1369
<code>\um_accent:Nnn</code>	568, 586	<code>\um_config_mathscr_latin:n</code>	1380
<code>\um_after_pkg:nn</code>	16, 19, 2275	<code>\um_config_mathsf_Latin:n</code> ...	1425
<code>\um_backprime_double_mchar</code>	724, 1893	<code>\um_config_mathsf_Latin:n</code> ...	1437
<code>\um_backprime_single_mchar</code>	723, 1966, 1970, 1974	<code>\um_config_mathsfup_Latin:n</code> ...	1401
<code>\um_backprime_triple_mchar</code>	725, 1896	<code>\um_config_mathsfup_latin:n</code> ...	1413
<code>\um_config_mathbb_Latin:n</code>	1342	<code>\um_config_mathsfup_num:n</code>	1397
<code>\um_config_mathbb_latin:n</code>	1339	<code>\um_config_mathhtt_Latin:n</code>	1452
<code>\um_config_mathbb_misc:n</code>	1355	<code>\um_config_mathhtt_latin:n</code>	1455
<code>\um_config_mathbb_num:n</code>	1352	<code>\um_config_mathhtt_num:n</code>	1449
<code>\um_config_mathbbit_misc:n</code>	1362	<code>\um_config_mathup_greek:n</code>	1235
<code>\um_config_mathbffrak_Latin:n</code> ..	1638	<code>\um_config_mathup_greek:n</code>	1245
<code>\um_config_mathbffrak_latin:n</code> ..	1641	<code>\um_config_mathup_Latin:n</code>	1212
<code>\um_config_mathbfit_greek:n</code> ...	1488	<code>\um_config_mathup_latin:n</code>	1222
<code>\um_config_mathbfit_greek:n</code> ...	1500	<code>\um_config_mathup_misc:n</code>	1255
<code>\um_config_mathbfit_Latin:n</code> ...	1458	<code>\um_config_mathup_num:n</code>	1208
<code>\um_config_mathbfit_latin:n</code> ...	1473	<code>\um_cs_compat:n</code>	61, 64–71
<code>\um_config_mathbfit_misc:n</code>	1512	<code>\um_define_active_frac:Nw</code>	2214–2231, 2234
<code>\um_config_mathbfscr_Latin:n</code> ...	1644	<code>\um_delimiter:Nnn</code> 553, 559, 564, 565, 586	
<code>\um_config_mathbfscr_latin:n</code> ...	1647	<code>\um_fontdimen_to_percent:nn</code>	2047, 2048, 2054, 2059, 2062
<code>\um_config_mathbfsfit_greek:n</code> ..	1758	<code>\um_fontspec_select_font:</code>	670
<code>\um_config_mathbfsfit_greek:n</code> ..	1770	<code>\um_fontspec_select_font:n</code> .	632, 670
<code>\um_config_mathbfsfit_Latin:n</code> ..	1734	<code>\um_glyph_if_exist:cT</code>	1161

\um_glyph_if_exist:cTF	1174	1517, 1521, 1524, 1604, 1607,
\um_glyph_if_exist:n	1197	1611, 1614, 1704, 1707, 1711,
\um_glyph_if_exist:nF	1207	1714, 1784, 1787, 1791, 1794
\um_glyph_if_exist:nT	1206	\um_mathmap_noparse:Nnn
\um_glyph_if_exist:nTF 647, <u>774</u> , 783, 1120
. <u>1197</u> , 1877, 1880, 1883, 1893, 1896		\um_mathmap_parse:Nnn
\um_glyph_if_exist_p:n	1204 655, <u>781</u>
\um_if_mathalph_decl:n	844	\um_mathstyle_scale:Nnn
\um_if_mathalph_decl:nTF	832 2045, 2051, <u>2057</u>
\um_init:	614, 623	\um_maybe_init_alphabet:n
\um_init_alphabet:n	649, 1134, 1192, 1196 649, 657, 1114–1116, 1134, 1158, 1162
\um_init_alphabet:x	1057	\um_nbackprimes_select:nn .
\um_make_mathactive:nNN .	719–727, <u>729</u>	1889, 2003
\um_map_char_noparse:nn		\um_nprimes:Nn
..... 650, 923, 928, 1121		1867, 1877,
\um_map_char_parse:nn	658, 926	1880, 1883, 1886, 1893, 1896, 1899
\um_map_char_single:cc	961, 963	\um_nprimes_select:nn
\um_map_char_single:nn		1873, 1953
..... 650, 658, 915, 961, 1121		\um_patch_pkg:nn <u>2271</u> , 2278, 2287, 2297
\um_map_char_single:nnn		\um_peek_execute_branches_ss: ...
..... 944–949, 955, 962, 970	 2173, 2178
\um_map_chars_Greek:nn .	952, 1237,	\um_peek_execute_branches_ss_aux:
1240, 1303, 1306, 1491, 1495,	 2185, 2198
1581, 1585, 1680, 1684, 1760, 1764		\um_prepare_alph:f
\um_map_chars_greek:nn .	941, 1247,	1084
1250, 1313, 1316, 1503, 1507,		\um_prepare_alph:n, \um_prepare_alph:f
1593, 1597, 1692, 1696, 1772, 1776	 <u>1056</u>
\um_map_chars_Latin:nn		\um_prime_double_mchar
..... 931, 1214, 1217,		720, 1877
1277, 1280, 1403, 1407, 1427,		\um_prime_quad_mchar
1431, 1460, 1464, 1468, 1550,		722, 1883
1554, 1558, 1656, 1660, 1736, 1740		\um_prime_single_mchar
\um_map_chars_latin:nn 719, 1904, 1908, 1912, 1916
..... 936, 1224, 1227,		\um_prime_triple_mchar
1287, 1291, 1415, 1419, 1439,		721, 1880
1443, 1475, 1479, 1483, 1565,		\um_process_symbol_noparse:nnnn .
1569, 1573, 1668, 1672, 1748, 1752	 646, <u>693</u>
\um_map_chars_numbers:nn ..	958, 1209	\um_process_symbol_parse:nnnn
\um_map_chars_range:ncc	920	654, <u>693</u>
\um_map_chars_range:nnn	913, 918	\um_radical:nn
\um_map_chars_range:nnnn		549, <u>586</u>
..... <u>919</u> , 933, 938, 943, 954, 959		\um_remap_symbol:nnn
\um_map_single:nnn 648, 656, 702, 703, 705
... <u>961</u> , 1228–1230, 1257, 1260,		\um_remap_symbol_noparse:nnn
1264, 1267, 1288, 1292–1294,		648, <u>701</u>
1323, 1326, 1330, 1333, 1514,		\um_remap_symbol_parse:nnn
	 656, <u>701</u> , 708
		\um_remap_symbols:
		664, <u>701</u>
		\um_resolve_greek:
		<u>1031</u>
		\um_scan_backdprime:
	 1968, 1987, 2018, 2039
		\um_scan_backprime:
	 1964, 1981, 2017, 2037, 2038
		\um_scan_backtrprime:
	 1972, 1995, 2019, 2040
		\um_scan_dprime: 1906, 1929, 2014, 2034

<code>\um_scan_prime:</code>	1567, 1570, 1574, 1642, 1648,
..... 1902, 1923, 2013, 2032, 2033	1669, 1673, 1676, 1749, 1753, 1756
<code>\um_scan_qprime:</code> 1914, 1945, 2016, 2036	<code>\um_set_mathalphabet_numbers:Nnn</code>
<code>\um_scan_sscript:</code> 2088, 2121, 2158, 2160 987, 1210, 1353, 1398,
<code>\um_scan_sscript:TF</code> 2159, 2168	1399, 1450, 1545, 1546, 1651, 1652
<code>\um_scan_trprime:</code> 1910, 1937, 2015, 2035	<code>\um_set_mathalphabet_pos:Nnnn</code> 980,
<code>\um_scanbackprime_collect:N</code>	1270–1273, 1298, 1299, 1336,
.. 1966, 1970, 1974, 1976, 1979,	1337, 1344–1350, 1356–1360,
1982, 1985, 1989, 1993, 1997, 2001	1363–1367, 1371–1378, 1382–1384,
<code>\um_scanprime_collect:N</code>	1388–1392, 1527, 1528, 1530,
..... 1904, 1908, 1912,	1533, 1537, 1540, 1617–1622,
1916, 1918, 1921, 1924, 1927,	1624, 1627, 1631, 1634, 1717,
1931, 1935, 1939, 1943, 1947, 1951	1718, 1720, 1723, 1727, 1730,
<code>\um_set_big_operator:nnn</code> ... 545, <u>574</u>	1797, 1798, 1800, 1803, 1807, 1810
<code>\um_set_delcode:n</code>	<code>\um_set_mathchar:cNnn</code> 580, <u>586</u>
... 551, 557, 563, 737, 740–766, <u>768</u>	<code>\um_set_mathchar:NNnn</code> <u>586</u> , 730
<code>\um_set_delcode:nn</code> 734–736, 738, 739, <u>768</u>	<code>\um_set_mathcode:nnn</code>
<code>\um_set_mathalph_range:nNcc</code> ... 1028 552, 558, 562, 571, 586
<code>\um_set_mathalph_range:Nnn</code> <u>974</u>	<code>\um_set_mathcode:nnnn</code> <u>586</u> , 715, 777, 924
<code>\um_set_mathalph_range:nNnn</code> 974, 979	<code>\um_set_mathsymbol:nNNn</code> 542, 694
<code>\um_set_mathalph_range:nNnnn</code>	<code>\um_setup_active_frac:</code> 2212, 2246
.... 989, 994, 999, 1005, 1011, 1027	<code>\um_setup_active_subscript:nn</code> ...
<code>\um_set_mathalphabet_char:Ncc</code> 2112, 2126–2153
..... 1020, 1024	<code>\um_setup_active_superscript:nn</code> .
<code>\um_set_mathalphabet_char:Nnn</code> 2079, 2093–2109
..... 647, 655, 976, 1021, 1120	<code>\um_setup_alphabets:</code> 667, <u>1107</u>
<code>\um_set_mathalphabet_char:Nnnn</code> ..	<code>\um_setup_delcodes:</code> 666, <u>733</u>
.. 983, 1000, 1006, 1012–1017, 1023	<code>\um_setup_math_alphabet:Nnn</code> ... <u>1152</u>
<code>\um_set_mathalphabet_Greek:Nnn</code> ..	<code>\um_setup_math_alphabet:VVV</code> ... 1137
..... 1003, 1243, 1309, 1489,	<code>\um_setup_mathactives:</code> 665, <u>718</u>
1492, 1496, 1579, 1582, 1586,	<code>\um_split_arrow:w</code> 849, 860
1681, 1685, 1688, 1761, 1765, 1768	<code>\um_split_slash:w</code> 852, 864
<code>\um_set_mathalphabet_greek:Nnn</code> ..	<code>\um_sub_or_super:n</code> .. 2086, 2119, 2162
..... 1009, 1253, 1319, 1501,	<code>\um_switchto_mathup:</code> 2247, 2279
1504, 1508, 1591, 1594, 1598,	<code>\um_symfont_tl</code> 644, 653,
1693, 1697, 1700, 1773, 1777, 1780	660, 694, 715, 730, 769, 772, 777, 924
<code>\um_set_mathalphabet_Latin:Nnn</code> ..	<code>\um_to_usv:nn</code> 74,
992, 1220, 1283, 1343, 1370, 1387,	76, 920, 921, 963, 964, 967, 981,
1404, 1408, 1411, 1428, 1432,	1024, 1025, 1028, 1029, 1161, 1174
1435, 1453, 1462, 1465, 1469,	<code>\um_trace:n</code> 1111, 1118, 2326
1552, 1555, 1559, 1639, 1645,	<code>\um_trace:nx</code> 1171, 1175, 1193, 2327
1657, 1661, 1664, 1737, 1741, 1744	<code>\um_warn_missing_alphabets:</code> 1139, 1141
<code>\um_set_mathalphabet_latin:Nnn</code> ..	<code>\um_warning:n</code> 486, 807, 2320, 2325
997, 1233, 1297, 1340, 1381, 1395,	<code>\Umathaccent</code> 605
1416, 1420, 1423, 1440, 1444,	<code>\Umathchardef</code> 595
1447, 1456, 1477, 1480, 1484,	<code>\Umathcharnum</code> 2292, 2293

<code>\Umathcode</code>	587, 591		
<code>\Umathcodenum</code>	2292, 2293		
<code>\unicodecdots</code>	2312		
<code>\unicodeellipsis</code>	2311		
<code>\UnicodeMathSymbol</code> .	646, 654, 663, 1843		
<code>\unimathsetup</code>	306,		
	344, 352, 360, 368, 376, 516–518, 520		
<code>\unless</code>	870		
<code>\updefault</code>	661		
<code>\upDigamma</code>	2270		
<code>\updigamma</code>	2269		
<code>\upint</code>	791		
<code>\Uradical</code>	599		
<code>\Url@FormatString</code>	2279		
<code>\UrlSpecials</code>	2280		
<code>\use:c</code> ...	489, 493, 1059, 1172, 1176, 1184		
<code>\use_i:nnn</code>	1124		
<code>\use_ii:nnn</code>	1125		
<code>\use_iii:nnn</code>	1126		
<code>\use_none:n</code>	657		
<code>\use_none:nnnn</code>	663, 1057, 1154		
<code>\use_none:nnnnn</code>	1132		
<code>\usepackage</code>	3		
<code>\usv_set:nnn</code>	73, 77–296		
		V	
		<code>\varepsilon</code>	1049
		<code>\varointclockwise</code>	788
		<code>\varphi</code>	1052
		<code>\vec</code>	2249
		<code>\version@elt</code>	527
		<code>\version@list</code>	527
		<code>\Vert</code>	2310
		<code>\ysmwhtcircle</code>	2255
		W	
		<code>\widehat</code>	2314
		<code>\widetilde</code>	2315
		X	
		<code>\xetex_or luatex:nnn</code>	61
		Y	
		<code>\yen</code>	2257
		Z	
		<code>\Z</code>	210, 298
		<code>\z</code>	297
		<code>\z@</code>	2044, 2047, 2048, 2052
		<code>\zf@basefont</code>	690
		<code>\zf@family</code>	661
		<code>\zf@update@ff</code>	812, 815