

Experimental unicode mathematical typesetting: The unicode-math package

Will Robertson

2010/05/20 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 |
| 4 | Unicode maths font setup | 4 | 7.7 | Maths accents | 44 |
| 4.1 | Using multiple fonts | 5 | 8 | Font features | 45 |
| 4.2 | Script and scriptscript fonts/features | 6 | 8.1 | OpenType maths font features | 46 |
| 5 | Maths input | 6 | 8.2 | Script and scriptscript font options | 46 |
| 5.1 | Math ‘style’ | 6 | 8.3 | Range processing | 46 |
| 5.2 | Bold style | 7 | 8.4 | Resolving Greek symbol name control sequences | 52 |
| 5.3 | Sans serif style | 8 | 9 | Maths alphabets mapping definitions | 53 |
| 5.4 | All (the rest) of the mathematical alphabets | 9 | 9.1 | Alphabets | 57 |
| 5.5 | Miscellanea | 10 | 10 | Definitions of the math symbols | 71 |
| I | The unicode-math package | 16 | 11 | Epilogue | 73 |
| 6 | Things we need | 16 | 12 | Error messages | 85 |
| 6.1 | Options | 24 | 13 | stix table data extraction | 86 |
| 6.2 | Overcoming <code>\@onlypreamble</code> | 30 | A | Documenting maths support in the NFSS | 87 |
| 7 | Fundamentals | 30 | B | X_YTeX math font dimensions | 88 |
| 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 `srix` font comes to mind as a possible exception. It will therefore be necessary to delegate specific unicode ranges of glyphs to separate fonts:

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

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

4.1.1 Control over maths alphabets

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

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

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

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

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

4.2 Script and scriptscript fonts/features

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

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

5 Maths input

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

5.1 Math 'style'

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

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

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

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

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

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

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

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

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

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

5.3 Sans serif style

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

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

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

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

5.3.1 What about bold sans serif?

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

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

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

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

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

5.4 All (the rest) of the mathematical alphabets

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

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

5.4.1 Double-struck

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

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

There are also five Latin *italic* double-struck letters: $\mathbb{D}, \mathbb{d}, \mathbb{E}, \mathbb{e}, \mathbb{J}$. 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 \sum).

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

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

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

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

- `\solidus`;
- `\fracdash`;
- `\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 `\leftfracdash` 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}{2}` `\frac{1}{3}` `\frac{2}{3}` `\frac{1}{4}` `\frac{3}{4}` `\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{1}{2} x`, it’s more readable to have `\frac{1}{2}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.

\LaTeX 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 \TeX , 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 [2010/05/20 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}

```

Conditionals

```

26 \bool_new:N \l_um_fontspec_feature_bool
27 \bool_new:N \l_um_ot_math_bool
28 \bool_new:N \l_um_init_bool
29 \bool_new:N \l_um_implicit_alph_bool

```

For math-style:

```

30 \bool_new:N \g_um_literal_bool
31 \bool_new:N \g_um_upLatin_bool
32 \bool_new:N \g_um_uplatin_bool
33 \bool_new:N \g_um_upGreek_bool
34 \bool_new:N \g_um_upgreek_bool

```

For bold-style:

```

35 \bool_new:N \g_um_bfliteral_bool
36 \bool_new:N \g_um_bfupLatin_bool
37 \bool_new:N \g_um_bfuplatin_bool

```

```

38 \bool_new:N \g_um_bfupGreek_bool
39 \bool_new:N \g_um_bfupgreek_bool

```

For sans-style:

```

40 \bool_new:N \g_um_upsans_bool
41 \bool_new:N \g_um_sfliteral_bool

```

For assorted package options:

```

42 \bool_new:N \g_um_upNabla_bool
43 \bool_new:N \g_um_uppartial_bool
44 \bool_new:N \g_um_literal_Nabla_bool
45 \bool_new:N \g_um_literal_partial_bool
46 \bool_new:N \g_um_texgreek_bool
47 \bool_new:N \l_um_smallfrac_bool
48 \bool_new:N \g_um_literal_colon_bool

```

Variables

```

49 \int_new:N \g_um_fam_int

50 \tl_set:Nn \g_um_math_alphabet_name_latin_tl {Latin,~lowercase}
51 \tl_set:Nn \g_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
52 \tl_set:Nn \g_um_math_alphabet_name_greek_tl {Greek,~lowercase}
53 \tl_set:Nn \g_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
54 \tl_set:Nn \g_um_math_alphabet_name_num_tl {Numerals}
55 \tl_set:Nn \g_um_math_alphabet_name_misc_tl {Misc.}

```

6.0.12 Compatibility with LuaTeX

```

56 \xetex_or luatex:nn
57 { \cs_new:Nn \um_cs_compat:n { \cs_set_eq:cc {U#1} {XeTeX#1} } }
58 { \cs_new:Nn \um_cs_compat:n { \cs_set_eq:cc {U#1} {luatexU#1} } }
59 \um_cs_compat:n {mathcode}
60 \um_cs_compat:n {delcode}
61 \um_cs_compat:n {mathcodenum}
62 \um_cs_compat:n {mathcharnum}
63 \um_cs_compat:n {mathchardef}
64 \um_cs_compat:n {radical}
65 \um_cs_compat:n {mathaccent}
66 \um_cs_compat:n {delimiter}

```

6.0.13 Function variants

```

67 \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

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

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

```

68 \cs_new:Nn \usv_set:nnn {
69   \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}
70 }
71 \cs_new:Nn \um_to_usv:nn { g_um_#1_#2_usv }

```

Alphabets

```

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

```

Bold:

```

100 \usv_set:nnn {bf}{num}{"1D7CE}
101 \usv_set:nnn {bfup}{num}{"1D7CE}
102 \usv_set:nnn {bfit}{num}{"1D7CE}
103 \usv_set:nnn {bfup}{Latin}{"1D400}
104 \usv_set:nnn {bfup}{latin}{"1D41A}
105 \usv_set:nnn {bfup}{Greek}{"1D6A8}
106 \usv_set:nnn {bfup}{greek}{"1D6C2}
107 \usv_set:nnn {bfit}{Latin}{"1D468}
108 \usv_set:nnn {bfit}{latin}{"1D482}

```

```

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

126 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsf
127 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsf
128 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsf
129 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsf
130 \usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_L
131 \usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_l
132 \usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_G
133 \usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_g

```

Greek variants:

```

134 \usv_set:nnn {up}{varTheta}{ "3F4}
135 \usv_set:nnn {up}{Digamma}{ "3DC}
136 \usv_set:nnn {up}{varepsilon}{ "3F5}
137 \usv_set:nnn {up}{vartheta}{ "3D1}
138 \usv_set:nnn {up}{varkappa}{ "3F0}
139 \usv_set:nnn {up}{varphi}{ "3D5}
140 \usv_set:nnn {up}{varrho}{ "3F1}
141 \usv_set:nnn {up}{varpi}{ "3D6}
142 \usv_set:nnn {up}{digamma}{ "3DD}

```

Bold:

```

143 \usv_set:nnn {bfup}{varTheta}{ "1D6B9}
144 \usv_set:nnn {bfup}{Digamma}{ "1D7CA}
145 \usv_set:nnn {bfup}{varepsilon}{ "1D6DC}
146 \usv_set:nnn {bfup}{vartheta}{ "1D6DD}
147 \usv_set:nnn {bfup}{varkappa}{ "1D6DE}
148 \usv_set:nnn {bfup}{varphi}{ "1D6DF}
149 \usv_set:nnn {bfup}{varrho}{ "1D6E0}
150 \usv_set:nnn {bfup}{varpi}{ "1D6E1}
151 \usv_set:nnn {bfup}{digamma}{ "1D7CB}

```

Italic Greek variants:

```
152 \usv_set:nnn {it}{varTheta}{1D6F3}  
153 \usv_set:nnn {it}{varepsilon}{1D716}  
154 \usv_set:nnn {it}{vartheta}{1D717}  
155 \usv_set:nnn {it}{varkappa}{1D718}  
156 \usv_set:nnn {it}{varphi}{1D719}  
157 \usv_set:nnn {it}{varrho}{1D71A}  
158 \usv_set:nnn {it}{varpi}{1D71B}
```

Bold italic:

```
159 \usv_set:nnn {bfit}{varTheta}{1D72D}  
160 \usv_set:nnn {bfit}{varepsilon}{1D750}  
161 \usv_set:nnn {bfit}{vartheta}{1D751}  
162 \usv_set:nnn {bfit}{varkappa}{1D752}  
163 \usv_set:nnn {bfit}{varphi}{1D753}  
164 \usv_set:nnn {bfit}{varrho}{1D754}  
165 \usv_set:nnn {bfit}{varpi}{1D755}
```

Bold sans:

```
166 \usv_set:nnn {bfsfup}{varTheta}{1D767}  
167 \usv_set:nnn {bfsfup}{varepsilon}{1D78A}  
168 \usv_set:nnn {bfsfup}{vartheta}{1D78B}  
169 \usv_set:nnn {bfsfup}{varkappa}{1D78C}  
170 \usv_set:nnn {bfsfup}{varphi}{1D78D}  
171 \usv_set:nnn {bfsfup}{varrho}{1D78E}  
172 \usv_set:nnn {bfsfup}{varpi}{1D78F}
```

Bold sans italic:

```
173 \usv_set:nnn {bfsfit}{varTheta}{1D7A1}  
174 \usv_set:nnn {bfsfit}{varepsilon}{1D7C4}  
175 \usv_set:nnn {bfsfit}{vartheta}{1D7C5}  
176 \usv_set:nnn {bfsfit}{varkappa}{1D7C6}  
177 \usv_set:nnn {bfsfit}{varphi}{1D7C7}  
178 \usv_set:nnn {bfsfit}{varrho}{1D7C8}  
179 \usv_set:nnn {bfsfit}{varpi}{1D7C9}
```

Nabla:

```
180 \usv_set:nnn {up}{Nabla}{02207}  
181 \usv_set:nnn {it}{Nabla}{1D6FB}  
182 \usv_set:nnn {bfup}{Nabla}{1D6C1}  
183 \usv_set:nnn {bfit}{Nabla}{1D735}  
184 \usv_set:nnn {bfsfup}{Nabla}{1D76F}  
185 \usv_set:nnn {bfsfit}{Nabla}{1D7A9}
```

Partial:

```
186 \usv_set:nnn {up}{partial}{02202}  
187 \usv_set:nnn {it}{partial}{1D715}  
188 \usv_set:nnn {bfup}{partial}{1D6DB}
```

```

189 \usv_set:nnn {bfit} {partial}{1D74F}
190 \usv_set:nnn {bfsfup}{partial}{1D789}
191 \usv_set:nnn {bfsfit}{partial}{1D7C3}

```

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

```

192 \usv_set:nnn {up}{B}{\B}
193 \usv_set:nnn {up}{C}{\C}
194 \usv_set:nnn {up}{D}{\D}
195 \usv_set:nnn {up}{E}{\E}
196 \usv_set:nnn {up}{F}{\F}
197 \usv_set:nnn {up}{H}{\H}
198 \usv_set:nnn {up}{I}{\I}
199 \usv_set:nnn {up}{L}{\L}
200 \usv_set:nnn {up}{M}{\M}
201 \usv_set:nnn {up}{N}{\N}
202 \usv_set:nnn {up}{P}{\P}
203 \usv_set:nnn {up}{Q}{\Q}
204 \usv_set:nnn {up}{R}{\R}
205 \usv_set:nnn {up}{Z}{\Z}

206 \usv_set:nnn {it}{B}{1D435}
207 \usv_set:nnn {it}{C}{1D436}
208 \usv_set:nnn {it}{D}{1D437}
209 \usv_set:nnn {it}{E}{1D438}
210 \usv_set:nnn {it}{F}{1D439}
211 \usv_set:nnn {it}{H}{1D43B}
212 \usv_set:nnn {it}{I}{1D43C}
213 \usv_set:nnn {it}{L}{1D43F}
214 \usv_set:nnn {it}{M}{1D440}
215 \usv_set:nnn {it}{N}{1D441}
216 \usv_set:nnn {it}{P}{1D443}
217 \usv_set:nnn {it}{Q}{1D444}
218 \usv_set:nnn {it}{R}{1D445}
219 \usv_set:nnn {it}{Z}{1D44D}

220 \usv_set:nnn {up}{d}{\d}
221 \usv_set:nnn {up}{e}{\e}
222 \usv_set:nnn {up}{g}{\g}
223 \usv_set:nnn {up}{h}{\h}
224 \usv_set:nnn {up}{i}{\i}
225 \usv_set:nnn {up}{j}{\j}
226 \usv_set:nnn {up}{o}{\o}

227 \usv_set:nnn {it}{d}{1D451}
228 \usv_set:nnn {it}{e}{1D452}
229 \usv_set:nnn {it}{g}{1D454}
230 \usv_set:nnn {it}{h}{0210E}

```

²³¹ \usv_set:nnn {it}{i}{1D456}
²³² \usv_set:nnn {it}{j}{1D457}
²³³ \usv_set:nnn {it}{o}{1D45C}

Latin ‘h’:

²³⁴ \usv_set:nnn {bb} {h}{1D559}
²³⁵ \usv_set:nnn {tt} {h}{1D691}
²³⁶ \usv_set:nnn {scr} {h}{1D4BD}
²³⁷ \usv_set:nnn {frak} {h}{1D525}
²³⁸ \usv_set:nnn {bfup} {h}{1D421}
²³⁹ \usv_set:nnn {bfit} {h}{1D489}
²⁴⁰ \usv_set:nnn {sfup} {h}{1D5C1}
²⁴¹ \usv_set:nnn {sfit} {h}{1D629}
²⁴² \usv_set:nnn {bffrak}{h}{1D58D}
²⁴³ \usv_set:nnn {bfscr} {h}{1D4F1}
²⁴⁴ \usv_set:nnn {bfsfup}{h}{1D5F5}
²⁴⁵ \usv_set:nnn {bfsfit}{h}{1D65D}

Dotless ‘i’ and ‘j’:

²⁴⁶ \usv_set:nnn {up}{dotlessi}{00131}
²⁴⁷ \usv_set:nnn {up}{dotlessj}{00237}
²⁴⁸ \usv_set:nnn {it}{dotlessi}{1D6A4}
²⁴⁹ \usv_set:nnn {it}{dotlessj}{1D6A5}

Blackboard:

²⁵⁰ \usv_set:nnn {bb}{C}{2102}
²⁵¹ \usv_set:nnn {bb}{H}{210D}
²⁵² \usv_set:nnn {bb}{N}{2115}
²⁵³ \usv_set:nnn {bb}{P}{2119}
²⁵⁴ \usv_set:nnn {bb}{Q}{211A}
²⁵⁵ \usv_set:nnn {bb}{R}{211D}
²⁵⁶ \usv_set:nnn {bb}{Z}{2124}
²⁵⁷ \usv_set:nnn {up}{Pi} {"003A0}
²⁵⁸ \usv_set:nnn {up}{pi} {"003C0}
²⁵⁹ \usv_set:nnn {up}{Gamma} {"00393}
²⁶⁰ \usv_set:nnn {up}{gamma} {"003B3}
²⁶¹ \usv_set:nnn {up}{summation}{02211}
²⁶² \usv_set:nnn {it}{Pi} {"1D6F1}
²⁶³ \usv_set:nnn {it}{pi} {"1D70B}
²⁶⁴ \usv_set:nnn {it}{Gamma} {"1D6E4}
²⁶⁵ \usv_set:nnn {it}{gamma} {"1D6FE}
²⁶⁶ \usv_set:nnn {bb}{Pi} {"0213F}
²⁶⁷ \usv_set:nnn {bb}{pi} {"0213C}
²⁶⁸ \usv_set:nnn {bb}{Gamma} {"0213E}
²⁶⁹ \usv_set:nnn {bb}{gamma} {"0213D}
²⁷⁰ \usv_set:nnn {bb}{summation}{02140}

Italic blackboard:

```

271 \usv_set:nnn {bbit}{D}{ "2145}
272 \usv_set:nnn {bbit}{d}{ "2146}
273 \usv_set:nnn {bbit}{e}{ "2147}
274 \usv_set:nnn {bbit}{i}{ "2148}
275 \usv_set:nnn {bbit}{j}{ "2149}

```

Script exceptions:

```

276 \usv_set:nnn {scr}{B}{ "212C}
277 \usv_set:nnn {scr}{E}{ "2130}
278 \usv_set:nnn {scr}{F}{ "2131}
279 \usv_set:nnn {scr}{H}{ "210B}
280 \usv_set:nnn {scr}{I}{ "2110}
281 \usv_set:nnn {scr}{L}{ "2112}
282 \usv_set:nnn {scr}{M}{ "2133}
283 \usv_set:nnn {scr}{R}{ "211B}
284 \usv_set:nnn {scr}{e}{ "212F}
285 \usv_set:nnn {scr}{g}{ "210A}
286 \usv_set:nnn {scr}{o}{ "2134}

```

Fraktur exceptions:

```

287 \usv_set:nnn {frak}{C}{ "212D}
288 \usv_set:nnn {frak}{H}{ "210C}
289 \usv_set:nnn {frak}{I}{ "2111}
290 \usv_set:nnn {frak}{R}{ "211C}
291 \usv_set:nnn {frak}{Z}{ "2128}

```

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

```

292 \tl_new:Nn \g_um_mathup_latin_usv_range_tl {\a-`z}
293 \tl_new:Nn \g_um_mathup_Latin_usv_range_tl {\A-`Z}
294 \tl_new:Nn \g_um_mathup_greek_usv_range_tl {"3B1-"3C9,"3F5,"3D1,"3F0,"3D5,"3F1,"3D6,"3DD}
295 \tl_new:Nn \g_um_mathup_Greek_usv_range_tl {"391-"3A9,"3F4,"3DC}
296 \tl_new:Nn \g_um_mathup_num_usv_range_tl {\0-`9}
297 \tl_new:Nn \g_um_mathit_latin_usv_range_tl {"1D44E-"1D467,\g_um_it_h_usv}
298 \tl_new:Nn \g_um_mathit_Latin_usv_range_tl {"1D434-"1D44C}
299 \tl_new:Nn \g_um_mathit_greek_usv_range_tl {"1D6FC-"1D714,"1D716-1D71B}
300 \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.

```

301 \DeclareDocumentCommand \unimathsetup {m} {
302   \clist_clear:N \l_um_unknown_keys_clist
303   \keys_set:nn {unicode-math} {#1}
304 }

```


math-style

```
305 \keys_define:nn {unicode-math} {  
306   normal-style .choice_code:n =  
307   {  
308     \bool_set_false:N \g_um_literal_bool  
309     \ifcase \l_keys_choice_int  
310       \bool_set_false:N \g_um_upGreek_bool  
311       \bool_set_false:N \g_um_upgreek_bool  
312       \bool_set_false:N \g_um_upLatin_bool  
313       \bool_set_false:N \g_um_uplatin_bool  
314     \or  
315       \bool_set_true: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_true:N \g_um_upgreek_bool  
322       \bool_set_true: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_true:N \g_um_uplatin_bool  
329     \or  
330       \bool_set_true:N \g_um_literal_bool  
331     \fi  
332   } ,  
333   normal-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,  
334 }  
  
335 \keys_define:nn {unicode-math} {  
336   math-style .choice_code:n =  
337   {  
338     \ifcase \l_keys_choice_int  
339       \unimathsetup {  
340         normal-style=ISO,  
341         bold-style=ISO,  
342         sans-style=italic,  
343         nabla=italic,  
344         partial=italic,  
345       }  
346     \or  
347       \unimathsetup {  
348         normal-style=TeX,  
349         bold-style=TeX,
```

```

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

```

bold-style

```

383 \keys_define:nn {unicode-math} {
384   bold-style .choice_code:n = {
385     \bool_set_false:N \g_um_bfliteral_bool
386     \ifcase \l_keys_choice_int
387       \bool_set_false:N \g_um_bfupGreek_bool
388       \bool_set_false:N \g_um_bfupgreek_bool
389       \bool_set_false:N \g_um_bfupLatin_bool
390       \bool_set_false:N \g_um_bfuplatin_bool
391     \or
392       \bool_set_true:N \g_um_bfupGreek_bool
393       \bool_set_false:N \g_um_bfupgreek_bool

```

```

394     \bool_set_true:N \g_um_bfupLatin_bool
395     \bool_set_true:N \g_um_bfuplatin_bool
396   \or
397     \bool_set_true:N \g_um_bfupGreek_bool
398     \bool_set_true: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_bfliteral_bool
403   \fi
404 } ,
405 bold-style .generate_choices:n = {ISO,TeX,upright,literal} ,
406 }

```

sans-style

```

407 \keys_define:nn {unicode-math} {
408   sans-style .choice_code:n = {
409     \ifcase \l_keys_choice_int
410       \bool_set_false:N \g_um_upsans_bool
411     \or
412       \bool_set_true:N \g_um_upsans_bool
413     \or
414       \bool_set_true:N \g_um_sfliteral_bool
415     \fi
416   } ,
417   sans-style .generate_choices:n = {italic,upright,literal} ,
418 }

```

Nabla and partial

```

419 \keys_define:nn {unicode-math} {
420   nabla .choice_code:n = {
421     \bool_set_false:N \g_um_literal_Nabla_bool
422     \ifcase \l_keys_choice_int
423       \bool_set_true:N \g_um_upNabla_bool
424     \or
425       \bool_set_false:N \g_um_upNabla_bool
426     \or
427       \bool_set_true:N \g_um_literal_Nabla_bool
428     \fi
429   } ,
430   nabla .generate_choices:n = {upright,italic,literal} ,
431 }

432 \keys_define:nn {unicode-math} {
433   partial .choice_code:n = {
434     \bool_set_false:N \g_um_literal_partial_bool

```

```

435 \ifcase \l_keys_choice_int
436 \bool_set_true:N \g_um_uppartial_bool
437 \or
438 \bool_set_false:N \g_um_uppartial_bool
439 \or
440 \bool_set_true:N \g_um_literal_partial_bool
441 \fi
442 } ,
443 partial .generate_choices:n = {upright,italic,literal} ,
444 }

```

Epsilon and phi shapes

```

445 \keys_define:nn {unicode-math} {
446   vargreek-shape .choice: ,
447   vargreek-shape / unicode .code:n = {
448     \bool_set_false:N \g_um_texgreek_bool
449   } ,
450   vargreek-shape / TeX .code:n = {
451     \bool_set_true:N \g_um_texgreek_bool
452   }
453 }

```

Colon style

```

454 \keys_define:nn {unicode-math} {
455   colon .choice: ,
456   colon / literal .code:n = {
457     \bool_set_true:N \g_um_literal_colon_bool
458   } ,
459   colon / TeX .code:n = {
460     \bool_set_false:N \g_um_literal_colon_bool
461   }
462 }

```

Slash delimiter style

```

463 \keys_define:nn {unicode-math} {
464   slash-delimiter .choice: ,
465   slash-delimiter / ascii .code:n = {
466     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
467   } ,
468   slash-delimiter / frac .code:n = {
469     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
470   } ,
471   slash-delimiter / div .code:n = {
472     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
473   }

```

```
474 }
```

Active fraction style

```
475 \keys_define:nn {unicode-math} {
476   active-frac .choice: ,
477   active-frac / small .code:n = {
478     \cs_if_exist:NTF \tfrac {
479       \bool_set_true:N \l_um_smallfrac_bool
480     }{
481       \um_warning:n {no-tfrac}
482       \bool_set_false:N \l_um_smallfrac_bool
483     }
484     \use:c{um_setup_active_frac:}
485   } ,
486   active-frac / normalsize .code:n = {
487     \bool_set_false:N \l_um_smallfrac_bool
488     \use:c{um_setup_active_frac:}
489   }
490 }
```

Debug/tracing

```
491 \keys_define:nn {unicode-math} {
492   trace .choice: ,
493   trace / debug .code:n = {
494     \msg_redirect_module:nnn { unicode-math } { trace } { warning }
495   } ,
496   trace / on .code:n = {
497     \msg_redirect_module:nnn { unicode-math } { trace } { trace }
498   } ,
499   trace / off .code:n = {
500     \msg_redirect_module:nnn { unicode-math } { trace } { none }
501   } ,
502 }

503 \clist_new:N \l_um_unknown_keys_clist
504 \keys_define:nn {unicode-math} {
505   unknown .code:n = {
506     \clist_put_right:No \l_um_unknown_keys_clist {
507       \l_keys_key_tl = {#1}
508     }
509   }
510 }

511 \unimathsetup {math-style=TeX}
512 \unimathsetup {slash-delimiter=ascii}
513 \unimathsetup {trace=off}
514 \cs_if_exist:NT \tfrac {
```

```

515 \unimathsetup {active-frac=small}
516 }
517 \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.

```

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

```

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 `lftfssbas.dtx`) we want to redefine

```

535 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
536 \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 LATEX 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.

```

537 \cs_set:Nn \um_set_mathsymbol:nNNn {
538   \prg_case_tl:Nnn #3 {
539     \mathop {
540       \um_set_big_operator:nnn {#1} {#2} {#4}
541     }
542     \mathopen {
543       \tl_if_in:NnTF \l_um_radicals_tl {#2} {
544         \cs_gset:cpx {\cs_to_str:N #2 sign} { \um_radical:nn {#1} {#4} }
545       }{
546         \um_set_delcode:n {#4}
547         \um_set_mathcode:nnn {#4} \mathopen {#1}
548         \cs_gset:Npx #2 { \um_delimiter:Nnn \mathopen {#1} {#4} }
549       }
550     }
551     \mathclose {
552       \um_set_delcode:n {#4}
553       \um_set_mathcode:nnn {#4} \mathclose {#1}
554       \cs_gset:Npx #2 { \um_delimiter:Nnn \mathclose {#1} {#4} }
555     }
556     \mathfence {
557       \um_set_mathcode:nnn {#4} {#3} {#1}
558       \um_set_delcode:n {#4}
559       \cs_gset:cpx {l \cs_to_str:N #2} { \um_delimiter:Nnn \math-
\open {#1} {#4} }
560       \cs_gset:cpx {r \cs_to_str:N #2} { \um_delimiter:Nnn \math-
\close {#1} {#4} }
561     }
562     \mathaccent {
563       \cs_gset:Npx #2 { \um_accent:Nnn #3 {#1} {#4} }
564     }
565   }{
566     \um_set_mathcode:nnn {#4} {#3} {#1}
567   }
568 }

```

`\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:

$(\sum \rightarrow) \Sigma \rightarrow \sum_{\text{sym}} \rightarrow \sumop \nolimits$

$(\int \rightarrow) \int \rightarrow \int_{\text{sym}} \rightarrow \intop$

```

569 \cs_new:Nn \um_set_big_operator:nnn {
570   \group_begin:
571     \char_make_active:n {#3}
572     \char_gmake_mathactive:n {#3}
573     \um@scanactivedef #3 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }
574   \group_end:
575   \um_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
576   \cs_gset:cpx { \cs_to_str:N #2 _sym } {
577     \exp_not:c { \cs_to_str:N #2 op }
578     \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
579   }
580 }

\um_set_mathcode:nnnn
\um_set_mathcode:nnn 581 \cs_set:Nn \um_set_mathcode:nnnn {
\um_set_mathchar:NNnn 582 \Umathcode \intexpr_eval:n {#1} =
\um_set_mathchar:cNnn 583 \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
\um_radical:nn 584 }
\um_delimiter:Nnn 585 \cs_set:Nn \um_set_mathcode:nnn {
\um_accent:Nnn 586 \Umathcode \intexpr_eval:n {#1} =
587 \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#1} \scan_stop:
588 }
589 \cs_set:Nn \um_set_mathchar:NNnn {
590 \Umathchardef #1 =
591 \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
592 }
593 \cs_new:Nn \um_radical:nn {
594 \Uradical \csname sym#1\endcsname #2 \scan_stop:
595 }
596 \cs_new:Nn \um_delimiter:Nnn {
597 \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
598 }
599 \cs_new:Nn \um_accent:Nnn {
600 \Umathaccent \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:

```



```

601 }
602 \cs_generate_variant:Nn \um_set_mathchar:NNnn {c}

\char_gmake_mathactive:N
\char_gmake_mathactive:n
603 \cs_new:Nn \char_gmake_mathactive:N {
604   \global\mathcode `#1 = "8000 \scan_stop:
605 }
606 \cs_new:Nn \char_gmake_mathactive:n {
607   \global\mathcode #1 = "8000 \scan_stop:
608 }

```

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

```

609 \DeclareDocumentCommand \setmathfont { O{ } m } {

```

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

```

610   \let\glb@currsizel\relax

```

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

```

611   \bool_set_true:N \l_um_init_bool
612   \seq_clear:N \l_um_char_range_seq
613   \clist_clear:N \l_um_char_num_range_clist

```

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

```

614   \csname S@\f@size\endcsname

```

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

```

615   \tl_set:Nn \l_um_mversion_tf {normal}
616   \DeclareMathVersion{\l_um_mversion_tf}

```

Define default font features for the script and scriptscript font.

```

617   \tl_set:Nn \l_um_script_features_tl {ScriptStyle}
618   \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
619   \tl_set:Nn \l_um_script_font_tl {#2}
620   \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.

```
621 \clist_clear:N \l_um_unknown_keys_clist
622 \keys_set:nn {unicode-math} {#1}
623 \um_fontspec_select_font:n {#2}
```

Check for the correct number of `\fontdimens`:

```
624 %% \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
625 %% \bool_set_true:N \l_um_ot_math_bool
626 %% \else
627 %% \bool_set_false:N \l_um_ot_math_bool
628 %% \PackageWarningNoLine{unicode-math}{
629 %% The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
630 %% Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
631 %% in~ a~ substandard~ manner
632 %% }
633 %% \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

```
634 \bool_if:NTF \l_um_init_bool {
635 \tl_set:Nn \um_symfont_tl {um_allsym}
636 \msg_trace:nxx {unicode-math} {default-math-font} {#2}
637 \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_noparse:nnnn
638 \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
639 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
640 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
641 \cs_set_eq:NN \um_map_char:nn \um_map_char_noparse:nn
642 }{
643 \int_incr:N \g_um_fam_int
644 \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
645 \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_parse:nnnn
646 \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_parse:Nnn
647 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
648 \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
649 \cs_set_eq:NN \um_map_char:nn \um_map_char_parse:nn
650 }
```

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

```
651 \DeclareSymbolFont{\um_symfont_tl}
652 {\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`.

```

653 \input{unicode-math-table.tex}
654 \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.)

```

655 \um_remap_symbols:
656 \um_setup_mathactives:
657 \um_setup_delcodes:
658 \um_setup_alphabets:
659 }

```

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

```

660 \cs_new:Nn \um_fontspec_select_font:n {
661   \bool_set_true:N \l_um_fontspec_feature_bool
662   \fontspec_select:xn
663   {
664     BoldFont = {}, ItalicFont = {},
665     Script = Math,
666     SizeFeatures = {
667       {Size = \tf@size-} ,
668       {Size = \sf@size-\tf@size ,
669         Font = \l_um_script_font_tl ,
670         \l_um_script_features_tl
671       } ,
672       {Size = -\sf@size ,
673         Font = \l_um_sscript_font_tl ,
674         \l_um_sscript_features_tl
675       }
676     },
677     \l_um_unknown_keys_clist
678   }
679   {#1}
680   \tl_set_eq:NN \l_um_font \zf@basefont
681   \bool_set_false:N \l_um_fontspec_feature_bool
682 }

```

7.3.1 Functions for setting up symbols with mathcodes

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

`\um_process_symbol_parse:nnnn`

```

683 \cs_set:Nn \um_process_symbol_noparse:nnnn {
684   \um_set_mathsymbol:nNNn {\um_symfont_tl} #2#3{#1}
685 }

686 \cs_set:Nn \um_process_symbol_parse:nnnn {
687   \um@parse@term{#1}{#2}{#3}{
688     \um_process_symbol_noparse:nnnn{#1}{#2}{#3}{#4}
689   }
690 }

```

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

`\um_remap_symbol_noparse:nnn`

`\um_remap_symbol_parse:nnn`

```

691 \cs_new:Nn \um_remap_symbols: {
692   \um_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
693   \um_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "cen-
        tred asterisk"
694   \bool_if:NF \g_um_literal_colon_bool {
695     \um_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
696   }
697 }

```

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

```

698 \cs_new:Nn \um_remap_symbol_parse:nnn {
699   \um@parse@term {#3} {\@nil} {#2} {
700     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
701   }
702 }
703 \cs_new:Nn \um_remap_symbol_noparse:nnn {
704   \clist_map_inline:nn {#1} {
705     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
706   }
707 }

```

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:`

```

708 \cs_new:Nn \um_setup_mathactives: {
709   \um_make_mathactive:nNN {"2032} \um_prime_single_mchar \mathord
710   \um_make_mathactive:nNN {"2033} \um_prime_double_mchar \mathord
711   \um_make_mathactive:nNN {"2034} \um_prime_triple_mchar \mathord
712   \um_make_mathactive:nNN {"2057} \um_prime_quad_mchar \mathord
713   \um_make_mathactive:nNN {"2035} \um_backprime_single_mchar \mathord
714   \um_make_mathactive:nNN {"2036} \um_backprime_double_mchar \mathord

```

```

715 \um_make_mathactive:nNN {"2037} \um_backprime_triple_mchar \mathord
716 \um_make_mathactive:nNN {\`\' } \mathstraightquote \mathord
717 \um_make_mathactive:nNN {\`\' } \mathbacktick \mathord
718 }

```

`\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!

```

719 \cs_new:Nn \um_make_mathactive:nNN {
720 \um_set_mathchar:NNnn #2 #3 {\um_symfont_tl} {#1}
721 \char_gmake_mathactive:n {#1}
722 }

```

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:`

```

723 \cs_new:Nn \um_setup_delcodes: {
724 \um_set_delcode:nn {\`\/} {\g_um_slash_delimiter_usv}
725 \um_set_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
726 \um_set_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
727 \um_set_delcode:n {"005C} % backslash
728 \um_set_delcode:nn {\`<} {"27E8} % angle brackets with ascii notation
729 \um_set_delcode:nn {\`>} {"27E9} % angle brackets with ascii notation
730 \um_set_delcode:n {"2191} % up arrow
731 \um_set_delcode:n {"2193} % down arrow
732 \um_set_delcode:n {"2195} % updown arrow
733 \um_set_delcode:n {"219F} % up arrow twohead
734 \um_set_delcode:n {"21A1} % down arrow twohead
735 \um_set_delcode:n {"21A5} % up arrow from bar
736 \um_set_delcode:n {"21A7} % down arrow from bar
737 \um_set_delcode:n {"21A8} % updown arrow from bar
738 \um_set_delcode:n {"21BE} % up harpoon right
739 \um_set_delcode:n {"21BF} % up harpoon left
740 \um_set_delcode:n {"21C2} % down harpoon right
741 \um_set_delcode:n {"21C3} % down harpoon left
742 \um_set_delcode:n {"21C5} % arrows up down
743 \um_set_delcode:n {"21F5} % arrows down up
744 \um_set_delcode:n {"21C8} % arrows up up
745 \um_set_delcode:n {"21CA} % arrows down down
746 \um_set_delcode:n {"21D1} % double up arrow
747 \um_set_delcode:n {"21D3} % double down arrow

```

```

748 \um_set_delcode:n {"21D5} % double updown arrow
749 \um_set_delcode:n {"21DE} % up arrow double stroke
750 \um_set_delcode:n {"21DF} % down arrow double stroke
751 \um_set_delcode:n {"21E1} % up arrow dashed
752 \um_set_delcode:n {"21E3} % down arrow dashed
753 \um_set_delcode:n {"21E7} % up white arrow
754 \um_set_delcode:n {"21E9} % down white arrow
755 \um_set_delcode:n {"21EA} % up white arrow from bar
756 \um_set_delcode:n {"21F3} % updown white arrow
757 }

```

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

```

\um_set_delcode:n 758 \cs_new:Nn \um_set_delcode:nn {
759   \Udelcode#1 = \csname sym\um_symfont_t1\endcsname #2
760 }
761 \cs_new:Nn \um_set_delcode:n {
762   \Udelcode#1 = \csname sym\um_symfont_t1\endcsname #1
763 }

```

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.

```

764 \cs_set:Nn \um_mathmap_noparse:Nnn {
765   \clist_map_inline:nn {#2} {
766     \tl_put_right:cx {um_setup_\cs_to_str:N #1:} {
767       \um_set_mathcode:nnnn{##1}{\mathalpha}{\um_symfont_t1}{#3}
768     }
769   }
770 }

```

`\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.

```

771 \cs_set:Nn \um_mathmap_parse:Nnn {
772   \clist_if_in:NnT \l_um_char_num_range_clist {#3} {
773     \um_mathmap_noparse:Nnn {#1}{#2}{#3}

```

774 }
775 }

7.4 (Big) operators


















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


However, the limits aren't set automatically; that is, we want to define, a la Plain \TeX *etc.*, `\def\int{\intop\nolimits}`, so there needs to be a transformation from `\int` to `\intop` during the expansion of `\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 | $\sum\limits_{00}^1$ | <code>\Bbbsum</code> | DOUBLE-STRUCK N-ARY SUMMATION |
| U+0220F | $\prod\limits_{00}^1$ | <code>\prod</code> | PRODUCT OPERATOR |
| U+02210 | $\coprod\limits_{00}^1$ | <code>\coprod</code> | COPRODUCT OPERATOR |
| U+02211 | $\sum\limits_{00}^1$ | <code>\sum</code> | SUMMATION OPERATOR |
| U+0222B | $\int\limits_{00}^1$ | <code>\int</code> | INTEGRAL OPERATOR |
| U+0222C | $\iint\limits_{00}^1$ | <code>\iint</code> | DOUBLE INTEGRAL OPERATOR |
| U+0222D | $\iiint\limits_{00}^1$ | <code>\iiint</code> | TRIPLE INTEGRAL OPERATOR |
| U+0222E | $\oint\limits_{00}^1$ | <code>\oint</code> | CONTOUR INTEGRAL OPERATOR |
| U+0222F | $\oiint\limits_{00}^1$ | <code>\oiint</code> | DOUBLE CONTOUR INTEGRAL OPERATOR |
| U+02230 | $\oiint\limits_{00}^1$ | <code>\oiint</code> | TRIPLE CONTOUR INTEGRAL OPERATOR |
| U+02231 | $\int\limits_{00}^1$ | <code>\intclockwise</code> | CLOCKWISE INTEGRAL |
| U+02232 | $\oint\limits_{00}^1$ | <code>\varointclockwise</code> | CONTOUR INTEGRAL, CLOCKWISE |
| U+02233 | $\oint\limits_{00}^1$ | <code>\ointctrackwise</code> | CONTOUR INTEGRAL, ANTICLOCKWISE |
| U+022C0 | $\bigwedge\limits_{00}^1$ | <code>\bigwedge</code> | LOGICAL OR OPERATOR |
| U+022C1 | $\bigvee\limits_{00}^1$ | <code>\bigvee</code> | LOGICAL AND OPERATOR |
| U+022C2 | $\bigcap\limits_{00}^1$ | <code>\bigcap</code> | INTERSECTION OPERATOR |
| U+022C3 | $\bigcup\limits_{00}^1$ | <code>\bigcup</code> | UNION OPERATOR |

| | | | |
|---------|---|------------------------------|--|
| U+027D5 | $\overset{1}{\underset{0}{\times}}$ | <code>\leftouterjoin</code> | LEFT OUTER JOIN |
| U+027D6 | $\overset{1}{\underset{0}{\times}}$ | <code>\rightouterjoin</code> | RIGHT OUTER JOIN |
| U+027D7 | $\overset{1}{\underset{0}{\times}}$ | <code>\fullouterjoin</code> | FULL OUTER JOIN |
| U+027D8 | $\overset{1}{\underset{0}{\perp}}$ | <code>\bigbot</code> | LARGE UP TACK |
| U+027D9 | $\overset{1}{\underset{0}{\top}}$ | <code>\bigtop</code> | LARGE DOWN TACK |
| U+029F8 | $\overset{1}{\underset{0}{/}}$ | <code>\xsol</code> | BIG SOLIDUS |
| U+029F9 | $\overset{1}{\underset{0}{\backslash}}$ | <code>\xbsol</code> | BIG REVERSE SOLIDUS |
| U+02A00 | $\overset{1}{\underset{0}{\odot}}$ | <code>\bigodot</code> | N-ARY CIRCLED DOT OPERATOR |
| U+02A01 | $\overset{1}{\underset{0}{\oplus}}$ | <code>\bigoplus</code> | N-ARY CIRCLED PLUS OPERATOR |
| U+02A02 | $\overset{1}{\underset{0}{\otimes}}$ | <code>\bigotimes</code> | N-ARY CIRCLED TIMES OPERATOR |
| U+02A03 | $\overset{1}{\underset{0}{\cup\!\!\cdot}}$ | <code>\bigcupdot</code> | N-ARY UNION OPERATOR WITH DOT |
| U+02A04 | $\overset{1}{\underset{0}{\cup\!\!\oplus}}$ | <code>\bigcupplus</code> | N-ARY UNION OPERATOR WITH PLUS |
| U+02A05 | $\overset{1}{\underset{0}{\sqcap}}$ | <code>\bigsqcap</code> | N-ARY SQUARE INTERSECTION OPERATOR |
| U+02A06 | $\overset{1}{\underset{0}{\sqcup}}$ | <code>\bigsqcup</code> | N-ARY SQUARE UNION OPERATOR |
| U+02A07 | $\overset{1}{\underset{0}{\&}}$ | <code>\conjquant</code> | TWO LOGICAL AND OPERATOR |
| U+02A08 | $\overset{1}{\underset{0}{\vee}}$ | <code>\disjquant</code> | TWO LOGICAL OR OPERATOR |
| U+02A09 | $\overset{1}{\underset{0}{\times}}$ | <code>\bigtimes</code> | N-ARY TIMES OPERATOR |
| U+02A0B | $\overset{1}{\underset{0}{\int}}$ | <code>\sumint</code> | SUMMATION WITH INTEGRAL |
| U+02A0C | $\overset{1}{\underset{0}{\iiint}}$ | <code>\iiiint</code> | QUADRUPLE INTEGRAL OPERATOR |
| U+02A0D | $\overset{1}{\underset{0}{\int}}$ | <code>\intbar</code> | FINITE PART INTEGRAL |
| U+02A0E | $\overset{1}{\underset{0}{\int}}$ | <code>\intBar</code> | INTEGRAL WITH DOUBLE STROKE |
| U+02A0F | $\overset{1}{\underset{0}{\int}}$ | <code>\fint</code> | INTEGRAL AVERAGE WITH SLASH |
| U+02A10 | $\overset{1}{\underset{0}{\oint}}$ | <code>\cirfnint</code> | CIRCULATION FUNCTION |
| U+02A11 | $\overset{1}{\underset{0}{\int}}$ | <code>\awint</code> | ANTICLOCKWISE INTEGRATION LINE INTEGRATION WITH RECTANGULAR |
| U+02A12 | $\overset{1}{\underset{0}{\int}}$ | <code>\rppoint</code> | PATH AROUND POLE |

| | | | |
|---------|---|-------------------------------|---|
| U+02A13 |  | <code>\scpolint</code> | LINE INTEGRATION WITH SEMICIRCULAR PATH AROUND POLE |
| U+02A14 |  | <code>\npolint</code> | LINE INTEGRATION NOT INCLUDING THE POLE |
| U+02A15 |  | <code>\pointint</code> | INTEGRAL AROUND A POINT OPERATOR |
| U+02A16 |  | <code>\sqint</code> | QUATERNION INTEGRAL OPERATOR |
| U+02A17 |  | <code>\intlarhk</code> | INTEGRAL WITH LEFTWARDS ARROW WITH HOOK |
| U+02A18 |  | <code>\intx</code> | INTEGRAL WITH TIMES SIGN |
| U+02A19 |  | <code>\intcap</code> | INTEGRAL WITH INTERSECTION |
| U+02A1A |  | <code>\intcup</code> | INTEGRAL WITH UNION |
| U+02A1B |  | <code>\upint</code> | INTEGRAL WITH OVERBAR |
| U+02A1C |  | <code>\lowint</code> | INTEGRAL WITH UNDERBAR |
| U+02A1D |  | <code>\Join</code> | JOIN |
| U+02A1E |  | <code>\bigtriangleleft</code> | LARGE LEFT TRIANGLE OPERATOR |
| U+02A1F |  | <code>\zcmp</code> | Z NOTATION SCHEMA COMPOSITION |
| U+02A20 |  | <code>\zpipe</code> | Z NOTATION SCHEMA PIPING |
| U+02A21 |  | <code>\zproject</code> | Z NOTATION SCHEMA PROJECTION |
| U+02AFC |  | <code>\biginterleave</code> | LARGE TRIPLE VERTICAL BAR OPERATOR |
| U+02AFF |  | <code>\bigtalloblong</code> | N-ARY WHITE VERTICAL BAR |

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

```

776 \tl_new:Nn \l_um_nolimits_tl {
777   \int\iint\iiint\iiiiint\oint\oiint\oiint
778   \intclockwise\varointclockwise\ointctrclockwise\sumint
779   \intbar\intBar\fint\cirfnint\awint\rppolint
780   \scpolint\npolint\pointint\sqint\intlarhk\intx
781   \intcap\intcup\upint\lowint
782 }

```

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

```

783 \DeclareDocumentCommand \addnolimits {m} {

```

```

784 \tl_put_right:Nn \l_um_nolimits_tl {#1}
785 }

```

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

```

786 \DeclareDocumentCommand \removenolimits {m} {
787   \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
788 }

```

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.

```

789 \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>

```

790 \let\left@primitive\left
791 \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 | √ ⁴ | \fourthroot | FOURTH ROOT |
| U+02308 | ┌ | \lceil | LEFT CEILING |
| U+0230A | └ | \lfloor | LEFT FLOOR |
| U+0231C | ┐ | \ulcorner | UPPER LEFT CORNER |
| U+0231E | └ | \llcorner | LOWER LEFT CORNER LIGHT LEFT TORTOISE SHELL BRACKET |
| U+02772 | | \lbrbrak | ORNAMENT |
| U+027C5 | ⌿ | \lbag | LEFT S-SHAPED BAG DELIMITER |
| U+027CC |) | \longdivision | LONG DIVISION MATHEMATICAL LEFT WHITE SQUARE |
| U+027E6 | ⌈ | \lBrack | BRACKET |
| U+027E8 | ⟨ | \langle | MATHEMATICAL LEFT ANGLE BRACKET MATHEMATICAL LEFT DOUBLE ANGLE |
| U+027EA | ⌈ | \lAngle | BRACKET MATHEMATICAL LEFT WHITE TORTOISE |
| U+027EC | | \Lbrbrak | SHELL BRACKET |
| U+02983 | ⌈ | \lBrace | LEFT WHITE CURLY BRACKET |
| U+02985 | (| \lParen | LEFT WHITE PARENTHESIS |
| U+02987 | ⌈ | \llparenthesis | Z NOTATION LEFT IMAGE BRACKET |
| U+02989 | ⌈ | \llangle | Z NOTATION LEFT BINDING BRACKET |
| U+0298B | ┌ | \lbrackubar | LEFT SQUARE BRACKET WITH UNDERBAR LEFT SQUARE BRACKET WITH TICK IN TOP |
| U+0298D | ┌ | \lbrackultick | CORNER LEFT SQUARE BRACKET WITH TICK IN |
| U+0298F | └ | \lbracklltick | BOTTOM CORNER |
| U+02991 | ⟨ | \langedot | LEFT ANGLE BRACKET WITH DOT |
| U+02993 | ⋈ | \lparenless | LEFT ARC LESS-THAN BRACKET |
| U+02995 | ⋈ | \Lparenltr | DOUBLE LEFT ARC GREATER-THAN BRACKET |
| U+02997 | ⌈ | \lblkbrbrak | LEFT BLACK TORTOISE SHELL BRACKET |
| U+029D8 | ⌿ | \lvzigzag | LEFT WIGGLY FENCE |
| U+029DA | ⌿ | \Lvzigzag | LEFT DOUBLE WIGGLY FENCE |
| U+029FC | ⋈ | \lcurvyangle | LEFT POINTING CURVED ANGLE BRACKET |
| U+03014 | | \lbrbrak | LEFT BROKEN BRACKET |
| U+03018 | | \Lbrbrak | LEFT WHITE TORTOISE SHELL BRACKET |

And \mathclose:

| USV | Ex. | Macro | Description |
|---------|-----|-----------|----------------------|
| U+00029 |) | \rparen | RIGHT PARENTHESIS |
| U+0005D |] | \rbrack | RIGHT SQUARE BRACKET |
| U+0007D | } | \rbrace | RIGHT CURLY BRACKET |
| U+02309 | ┐ | \rceil | RIGHT CEILING |
| U+0230B | └ | \rfloor | RIGHT FLOOR |
| U+0231D | ┐ | \urcorner | UPPER RIGHT CORNER |

| | | | |
|---------|---|----------------|---|
| U+0231F | ┐ | \lrcorner | LOWER RIGHT CORNER LIGHT RIGHT TORTOISE SHELL BRACKET |
| U+02773 | | \rbrbrak | ORNAMENT |
| U+027C6 | Œ | \rbag | RIGHT S-SHAPED BAG DELIMITER MATHEMATICAL RIGHT WHITE SQUARE |
| U+027E7 | ⌋ | \rBrack | BRACKET |
| U+027E9 | ⌋ | \rangle | MATHEMATICAL RIGHT ANGLE BRACKET MATHEMATICAL RIGHT DOUBLE ANGLE |
| U+027EB | ⌋ | \rAngle | BRACKET MATHEMATICAL RIGHT WHITE TORTOISE |
| U+027ED | | \Rbrbrak | SHELL BRACKET |
| U+02984 | ⌋ | \RBrace | RIGHT WHITE CURLY BRACKET |
| U+02986 | ⌋ | \RParen | RIGHT WHITE PARENTHESIS |
| U+02988 | ⌋ | \rrparenthesis | Z NOTATION RIGHT IMAGE BRACKET |
| U+0298A | ⌋ | \rrangle | Z NOTATION RIGHT BINDING BRACKET |
| U+0298C | ⌋ | \rbrackubar | RIGHT SQUARE BRACKET WITH UNDERBAR RIGHT SQUARE BRACKET WITH TICK IN |
| U+0298E | ⌋ | \rbracklrtick | BOTTOM CORNER RIGHT SQUARE BRACKET WITH TICK IN TOP |
| U+02990 | ⌋ | \rbrackurtick | CORNER |
| U+02992 | ⌋ | \rangledot | RIGHT ANGLE BRACKET WITH DOT |
| U+02994 | ⌋ | \rparengtr | RIGHT ARC GREATER-THAN BRACKET |
| U+02996 | ⌋ | \Rparenless | DOUBLE RIGHT ARC LESS-THAN BRACKET |
| U+02998 | ⌋ | \rblkrbrak | RIGHT BLACK TORTOISE SHELL BRACKET |
| U+029D9 | ⌋ | \rvzigzag | RIGHT WIGGLY FENCE |
| U+029DB | ⌋ | \Rvzigzag | RIGHT DOUBLE WIGGLY FENCE |
| U+029FD | ⌋ | \rcurvyangle | RIGHT POINTING CURVED ANGLE BRACKET |
| U+03015 | | \rbrbrak | RIGHT BROKEN BRACKET |
| U+03019 | | \Rbrbrak | RIGHT WHITE TORTOISE SHELL BRACKET |

7.7 Maths accents

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

| USV | Ex. | Macro | Description |
|---------|-----|----------|-----------------------|
| U+00300 | ˘ | \grave | GRAVE ACCENT |
| U+00301 | ˆ | \acute | ACUTE ACCENT |
| U+00302 | ˆ | \hat | CIRCUMFLEX ACCENT |
| U+00303 | ˜ | \tilde | TILDE |
| U+00304 | ˉ | \bar | MACRON |
| U+00305 | ˆ | \overbar | OVERBAR EMBELLISHMENT |
| U+00306 | ˘ | \breve | BREVE |
| U+00307 | ˙ | \dot | DOT ABOVE |
| U+00308 | ¨ | \ddot | DIERESIS |

| | | | |
|---------|----------------|-------------------------------------|--|
| U+00309 | \acute{x} | <code>\ovhook</code> | COMBINING HOOK ABOVE |
| U+0030A | \mathring{x} | <code>\ocirc</code> | RING |
| U+0030C | \check{x} | <code>\check</code> | CARON |
| U+00310 | \breve{x} | <code>\candra</code> | CANDRABINDU (NON-SPACING) |
| U+00312 | \mathring{x} | <code>\oturnedcomma</code> | COMBINING TURNED COMMA ABOVE GREEK PSILI (SMOOTH BREATHING) |
| U+00313 | \mathring{x} | <code>\osmooth</code> | (NON-SPACING) GREEK DASIA (ROUGH BREATHING) |
| U+00314 | \mathring{x} | <code>\orough</code> | (NON-SPACING) |
| U+00315 | \mathring{x} | <code>\ocommatopright</code> | COMBINING COMMA ABOVE RIGHT |
| U+0031A | \mathring{x} | <code>\droang</code> | LEFT ANGLE ABOVE (NON-SPACING) |
| U+00330 | x | <code>\wideutilde</code> | UNDER TILDE ACCENT (MULTIPLE CHARACTERS AND NON-SPACING) |
| U+00331 | x | <code>\underbar</code> | COMBINING MACRON BELOW |
| U+00338 | x | <code>\not</code> | COMBINING LONG SOLIDUS OVERLAY |
| U+020D0 | \mathring{x} | <code>\leftharpoonaccent</code> | COMBINING LEFT HARPOON ABOVE |
| U+020D1 | \mathring{x} | <code>\rightharpoonaccent</code> | COMBINING RIGHT HARPOON ABOVE |
| U+020D2 | x | <code>\vertoverlay</code> | COMBINING LONG VERTICAL LINE OVERLAY |
| U+020D6 | \mathring{x} | <code>\overleftarrow</code> | COMBINING LEFT ARROW ABOVE |
| U+020D7 | \mathring{x} | <code>\vec</code> | COMBINING RIGHT ARROW ABOVE |
| U+020DB | \mathring{x} | <code>\dddot</code> | COMBINING THREE DOTS ABOVE |
| U+020DC | \mathring{x} | <code>\ddddot</code> | COMBINING FOUR DOTS ABOVE |
| U+020E1 | \mathring{x} | <code>\overleftrighthararrow</code> | COMBINING LEFT RIGHT ARROW ABOVE |
| U+020E7 | x | <code>\annuity</code> | COMBINING ANNUITY SYMBOL |
| U+020E8 | x | <code>\threeunderdot</code> | COMBINING TRIPLE UNDERDOT |
| U+020E9 | \mathring{x} | <code>\widebridgeabove</code> | COMBINING WIDE BRIDGE ABOVE COMBINING RIGHTWARDS HARPOON WITH |
| U+020EC | x | <code>\underrightharpoondown</code> | BARB DOWNWARDS COMBINING LEFTWARDS HARPOON WITH |
| U+020ED | x | <code>\underleftharpoondown</code> | BARB DOWNWARDS |
| U+020EE | x | <code>\underleftarrow</code> | COMBINING LEFT ARROW BELOW |
| U+020EF | x | <code>\underrightarrow</code> | COMBINING RIGHT ARROW BELOW |
| U+020F0 | x | <code>\asteraccent</code> | COMBINING ASTERISK ABOVE |

8 Font features

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

```

792 \newcommand\um@zf@feature[2]{
793   \define@key[zf]{options}{#1}[] {
794     \bool_if:NTF \l_um_fontspec_feature_bool {
795       #2

```

```

796     }{
797     \um_warning:n {maths-feature-only}
798     }
799   }
800 }

```

8.1 OpenType maths font features

```

801 \um@zf@feature{ScriptStyle}{
802   \zf@update@ff{+ssty=0}
803 }
804 \um@zf@feature{ScriptScriptStyle}{
805   \zf@update@ff{+ssty=1}
806 }

```

8.2 Script and scriptscript font options

```

807 \keys_define:nn {unicode-math}
808 {
809   script-features .tl_set:N = \l_um_script_features_tl ,
810   sscript-features .tl_set:N = \l_um_sscript_features_tl ,
811   script-font .tl_set:N = \l_um_script_font_tl ,
812   sscript-font .tl_set:N = \l_um_sscript_font_tl ,
813 }

```

8.3 Range processing

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

```

814 \seq_new:N \l_um_mathalph_seq
815 \seq_new:N \l_um_char_range_seq
816 \keys_define:nn {unicode-math} {
817   range .code:n = {
818     \bool_set_false:N \l_um_init_bool
819     \seq_clear:N \l_um_char_range_seq
820     \seq_clear:N \l_um_mathalph_seq
821     \clist_map_inline:nn {#1} {
822       \um_if_mathalph_decl:nTF {##1} {
823         \seq_put_right:Nx \l_um_mathalph_seq {
824           { \exp_not:V \l_um_tmpa_tl }
825           { \exp_not:V \l_um_tmpb_tl }
826           { \exp_not:V \l_um_tmpc_tl }
827         }
828       }{
829         \seq_put_right:Nn \l_um_char_range_seq {##1}
830       }
831     }
832   }
833 }

```

```

834 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
835   \tl_set:Nn \l_um_tmpa_tl {#1}
836   \tl_set:Nn \l_um_tmppb_tl {}
837   \tl_set:Nn \l_um_tmppc_tl {}
838   \tl_if_in:NnT \l_um_tmpa_tl {->} {
839     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
840   }
841   \tl_if_in:NnT \l_um_tmpa_tl {/#} {
842     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
843   }
844   \seq_if_in:NVTF \g_um_mathalph_seq \l_um_tmpa_tl {
845     \prg_return_true:
846   }{
847     \prg_return_false:
848   }
849 }
850 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
851   \tl_set:Nn \l_um_tmpa_tl {#1}
852   \tl_set:Nn \l_um_tmppc_tl {#2}
853 }
854 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
855   \tl_set:Nn \l_um_tmpa_tl {#1}
856   \tl_set:Nn \l_um_tmppb_tl {#2}
857 }

```

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:

```

858 \newcommand\um@parse@term[4]{
859   \seq_map_variable:NNn \l_um_char_range_seq \@ii {
860     \unless\ifx\@ii\empty
861       \@tempswafalse

```

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

```

862     \expandafter\um@firstchar\expandafter{\@ii}
863     \ifx\@tempa\um@backslash
864       \expandafter\ifx\@ii#2\relax
865         \@tempswatrue
866       \else
867         \expandafter\ifx\@ii#3\relax
868           \@tempswatrue
869         \fi
870       \fi

```

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

```

871       \else
872         \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
873       \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.

```

874       \if@tempswa
875         \clist_put_right:Nx \l_um_char_num_range_clist { \int-
876           expr_eval:n {#1} }
877       \fi
878     \fi
879   }
880 }
881 \def\um@firstof#1#2\@nil{#1}
882 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
883 \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`.

```

884 \def\um@parse@range#1-#2-#3\@nil#4\@nil{
885   \def\@tempa{#1}
886   \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 | <code>#1-#2-#3 = X-\@marker-{\}</code> |


```

887 \expandafter\ifx\expandafter\@marker\@tempb\relax
888 \intexpr_compare:nT {#4=#1} \@tempswattrue
889 \else


---


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


---


890 \ifx\@empty\@tempb
891 \intexpr_compare:nT {#4>\#1-1} \@tempswattrue
892 \else


---


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


---


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


---


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


---


895 \else
896 \intexpr_compare:nT {#4>\#1-1} {
897 \intexpr_compare:nT {#4<\#2+1} \@tempswattrue
898 }
899 \fi
900 \fi
901 \fi
902 }

```

\um_map_char:nn **#1** : Starting input char (single)
#2 : Starting output char
Loops through character ranges setting \mathcode.

```

903 \cs_set:Nn \um_map_chars_range:nnn {
904 \prg_stepwise_inline:nnnn {0}{1}{\#1-1} {
905 \um_map_char:nn {\#2+\#1}{\#3+\#1}
906 }
907 }
908 \cs_new:Nn \um_map_char_noparse:nn {
909 \um_set_mathcode:nnnn {\#1}{\mathalpha}{\um_symfont_t1}{\#2}
910 }
911 \cs_new:Nn \um_map_char_parse:nn {
912 \um@parse@term {\#1} {\@nil} {\mathalpha} {
913 \um_map_char_noparse:nn {\#1}{\#2}
914 }

```

```

915 }
916 \cs_set:Nn \um_map_chars_Latin:nn {
917   \clist_map_inline:nn {#1} {
918     \um_map_chars_range:ncc {26} { \um_to_usv:nn{##1}{Latin} }
919     { \um_to_usv:nn {#2}{Latin} }
920   }
921 }
922 \cs_set:Nn \um_map_chars_latin:nn {
923   \clist_map_inline:nn {#1} {
924     \um_map_chars_range:ncc {26} { \um_to_usv:nn{##1}{latin} }
925     { \um_to_usv:nn {#2}{latin} }
926   }
927 }
928 \cs_set:Nn \um_map_chars_greek:nn {
929   \clist_map_inline:nn {#1} {
930     \um_map_chars_range:ncc {25} { \um_to_usv:nn {##1} {greek} }
931     { \um_to_usv:nn {#2} {greek} }
932     \um_map_char_single:cc { \um_to_usv:nn {##1} {varepsilon} }
933     { \um_to_usv:nn {#2} {varepsilon} }
934     \um_map_char_single:cc { \um_to_usv:nn {##1} {vartheta} }
935     { \um_to_usv:nn {#2} {vartheta} }
936     \um_map_char_single:cc { \um_to_usv:nn {##1} {varkappa} }
937     { \um_to_usv:nn {#2} {varkappa} }
938     \um_map_char_single:cc { \um_to_usv:nn {##1} {varphi} }
939     { \um_to_usv:nn {#2} {varphi} }
940     \um_map_char_single:cc { \um_to_usv:nn {##1} {varrho} }
941     { \um_to_usv:nn {#2} {varrho} }
942     \um_map_char_single:cc { \um_to_usv:nn {##1} {varpi} }
943     { \um_to_usv:nn {#2} {varpi} }
944   }
945 }
946 \cs_set:Nn \um_map_chars_Greek:nn {
947   \clist_map_inline:nn {#1} {
948     \um_map_chars_range:ncc {25} { \um_to_usv:nn{##1}{Greek} }
949     { \um_to_usv:nn {#2}{Greek} }
950     \um_map_char_single:cc { \um_to_usv:nn{##1}{varTheta} }
951     { \um_to_usv:nn {#2}{varTheta} }
952   }
953 }
954 \cs_set:Nn \um_map_chars_numbers:nn {
955   \um_map_chars_range:ncc {10} { \um_to_usv:nn{##1}{num} }
956   { \um_to_usv:nn{##2}{num} }
957 }
958 \cs_set:Nn \um_map_single:nnn {
959   \clist_map_inline:nn {#2} {
960     \um_map_char_single:cc { \um_to_usv:nn{##1}{#1} }

```

```

961 { \um_to_usv:nn {#3}{#1} }
962 }
963 }
964 \cs_set:Nn \um_map_char_single:nn { \um_map_char:nn {#1}{#2} }
965 \cs_generate_variant:Nn \um_map_char_single:nn {cc}
966 \cs_generate_variant:Nn \um_map_chars_range:nnn {ncc}

\um_set_mathalphabet_char:Nnn #1 : Maths alphabet
#2 : Input char (single)
#3 : Output char
Loops through character ranges setting \mathcode.
967 \cs_new:Nn \um_set_mathalphabet_char:Nnn {
968   \um_mathmap:Nnn {#1} {#2} {#3}
969 }

\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.
970 \cs_new:Nn \um_set_mathalph_range:nnn {
971   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {
972     \um_mathmap:Nnn {#2} { ##1 + #3 } { ##1 + #4 }
973   }
974 }
975 \cs_new:Nn \um_set_mathalphabet_pos:Nnnn {
976   \cs_if_exist:cT { \um_to_usv:nn {#4}{#2} } {
977     \clist_map_inline:nn {#3} {
978       \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {#2} }
979       { \um_to_usv:nn {#4} {#2} }
980     }
981   }
982 }
983 \cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
984   \clist_map_inline:nn {#2} {
985     \um_set_mathalph_range:ncc {10} #1 { \um_to_usv:nn {##1} {num} }
986     { \um_to_usv:nn {#3} {num} }
987   }
988 }
989 \cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
990   \clist_map_inline:nn {#2} {
991     \um_set_mathalph_range:ncc {26} #1 { \um_to_usv:nn {##1} {Latin} }
992     { \um_to_usv:nn {#3} {Latin} }
993   }
994 }
995 \cs_new:Nn \um_set_mathalphabet_latin:Nnn {
996   \clist_map_inline:nn {#2} {

```

```

997 \um_set_mathalph_range:nNcc {26} #1 { \um_to_usv:nn {##1} {latin} }
998 { \um_to_usv:nn {#3} {latin} }
999 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {h} }
1000 { \um_to_usv:nn {#3} {h} }
1001 }
1002 }
1003 \cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
1004 \clist_map_inline:nn {#2} {
1005 \um_set_mathalph_range:nNcc {25} #1 { \um_to_usv:nn {##1} {Greek} }
1006 { \um_to_usv:nn {#3} {Greek} }
1007 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varTheta} }
1008 { \um_to_usv:nn {#3} {varTheta} }
1009 }
1010 }
1011 \cs_new:Nn \um_set_mathalphabet_greek:Nnn {
1012 \clist_map_inline:nn {#2} {
1013 \um_set_mathalph_range:nNcc {25} #1 { \um_to_usv:nn {##1} {greek} }
1014 { \um_to_usv:nn {#3} {greek} }
1015 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varepsilon} }
1016 { \um_to_usv:nn {#3} {varepsilon} }
1017 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {vartheta} }
1018 { \um_to_usv:nn {#3} {vartheta} }
1019 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varkappa} }
1020 { \um_to_usv:nn {#3} {varkappa} }
1021 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varphi} }
1022 { \um_to_usv:nn {#3} {varphi} }
1023 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varrho} }
1024 { \um_to_usv:nn {#3} {varrho} }
1025 \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {##1} {varpi} }
1026 { \um_to_usv:nn {#3} {varpi} }
1027 }
1028 }
1029 \cs_generate_variant:Nn \um_set_mathalphabet_char:Nnn {Ncc}
1030 \cs_generate_variant:Nn \um_set_mathalph_range:nNnn {nNcc}

```

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:Nn \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` 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:Nn \um_prepare_alph:n {
1057   \cs_set:cpn {um_#1:n} ##1 {
1058     \use:c {um_setup_#1:} ##1 \egroup
1059   }
1060   \cs_set_protected:cpx {#1} {
1061     \exp_not:n{
1062       \bgroup
1063       \mode_if_math:F {
1064         \egroup\expandafter
1065         \non@alpherr\expandafter{\csname #1\endcsname\space}
1066       }
1067     }
1068     \exp_not:c {um_#1:n}
1069   }
1070 }

```

This is every math alphabet known to unicode-math:

`\g_um_mathalph_seq`

```

1071 \seq_new:N \g_um_mathalph_seq
1072 \tl_map_inline:nn {
1073   \mathup\mathit\mathbb\mathbbi
1074   \mathscr\mathfrak\mathtt
1075   \mathsf\mathsfup\mathsfit
1076   \mathbf\mathbfup\mathbfit
1077   \mathbfscr\mathbffrac
1078   \mathbfsf\mathbfsfup\mathbfsfit
1079 }{
1080   \seq_put_right:Nn \g_um_mathalph_seq {#1}
1081   \exp_args:Nf \um_prepare_alph:n {\cs_to_str:N #1}
1082 }

1083 \seq_new:N \g_um_default_mathalph_seq
1084 \clist_map_inline:nn {
1085   {\mathup    } {latin, Latin, greek, Greek, num, misc} {\mathup    } ,
1086   {\mathit    } {latin, Latin, greek, Greek, misc}     {\mathit    } ,
1087   {\mathbb    } {latin, Latin, num, misc}               {\mathbb    } ,
1088   {\mathbbi   } {misc}                                 {\mathbbi   } ,
1089   {\mathscr   } {latin, Latin}                         {\mathscr   } ,
1090   {\mathfrak  } {latin, Latin}                         {\mathfrak  } ,
1091   {\mathtt    } {latin, Latin, num}                    {\mathtt    } ,
1092   {\mathsfup  } {latin, Latin, num}                    {\mathsfup  } ,
1093   {\mathsfit  } {latin, Latin}                        {\mathsfit  } ,

```

```

1094 {\mathbfup } {latin, Latin, greek, Greek, num, misc} {\mathbfup } ,
1095 {\mathbfit } {latin, Latin, greek, Greek, misc} {\mathbfit } ,
1096 {\mathbfscr } {latin, Latin} {\mathbfscr } ,
1097 {\mathbffrak} {latin, Latin} {\mathbffrak} ,
1098 {\mathbfsfup} {latin, Latin, greek, Greek, num, misc} {\mathbfsfup} ,
1099 {\mathbfsfit} {latin, Latin, greek, Greek, misc} {\mathbfsfit}
1100 }{
1101 \seq_put_right:Nn \g_um_default_mathalph_seq {#1}
1102 }

```

\um_setup_alphabets: Variables:

```

1103 \seq_new:N \l_um_missing_alph_seq
1104 \cs_new:Nn \um_setup_alphabets: {
1105 \seq_clear:N \l_um_missing_alph_seq
1106 \seq_if_empty:NTF \l_um_mathalph_seq {
1107 \um_trace:n {setup-implicit}
1108 \seq_set_eq:NN \l_um_mathalph_seq \g_um_default_mathalph_seq
1109 \bool_set_true:N \l_um_implicit_alph_bool
1110 \um_maybe_init_alphabet:n {sf}
1111 \um_maybe_init_alphabet:n {bf}
1112 \um_maybe_init_alphabet:n {bfsf}
1113 }{
1114 \um_trace:n {setup-explicit}
1115 \bool_set_false:N \l_um_implicit_alph_bool
1116 \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
1117 }
1118 \seq_map_inline:Nn \l_um_mathalph_seq {
1119 \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
1120 \tl_set:No \l_um_tmpb_tl { \use_ii:nnn ##1 }
1121 \tl_set:No \l_um_remap_alphabet_tl { \use_iii:nnn ##1 }
1122 \tl_if_empty:NTF \l_um_remap_alphabet_tl {
1123 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \token_to_str:N \l_um_tmpa_tl}
1124 }{
1125 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \token_to_str:N \l_um_remap_alphabet_tl}
1126 }
1127 \tl_set:Nx \l_um_remap_alphabet_tl {\exp_after:wN \use_none:nnnnn \l_um_remap_alphabet_tl}
1128 \tl_if_empty:NT \l_um_tmpb_tl {
1129 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
1130 \tl_set:Nn \l_um_tmpb_tl { latin, Latin, greek, Greek, num, misc }
1131 }
1132 \um_setup_math_alphabet:VVV \l_um_tmpa_tl \l_um_tmpb_tl \l_um_remap_alphabet_tl
1133 }
1134 \um_warn_missing_alphabets:
1135 }
1136 \cs_new:Nn \um_warn_missing_alphabets: {
1137 \seq_if_empty:NF \l_um_missing_alph_seq {

```

```

1138 \typeout{
1139   Package~unicode-math~Warning:~
1140   missing~math~alphabets~in~font~ \fontname\l_um_font
1141 }
1142 \seq_map_inline:Nn \l_um_missing_alph_seq {
1143   \typeout{\space\space\space\space##1}
1144 }
1145 }
1146 }

```

`\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`)

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

```

1147 \cs_new:Nn \um_setup_math_alphabet:Nnn {
1148   \tl_set:Nx \l_um_tmpa_tl {\cs_to_str:N #1}
1149   \tl_set:Nx \l_um_tmpb_tl {\exp_after:wN \use_none:nnnn \l_um_tmpa_tl}
1150   \clist_map_inline:nn {#2} {
1151     \cs_if_exist:cT {um_config_ \l_um_tmpa_tl _##1:n} {
1152       \tl_if_eq:nnTF {##1}{misc} {
1153         \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
1154         \clist_map_break:
1155       }{
1156         \um_glyph_if_exist:cT { \um_to_usv:nn {#3}{##1} }{
1157           \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
1158           \clist_map_break:
1159         }
1160       }
1161     }
1162   }
1163   \clist_map_inline:nn {#2} {
1164     \cs_if_exist:cT {um_config_ \l_um_tmpa_tl _##1:n} {
1165       \tl_if_eq:nnTF {##1}{misc} {
1166         \um_trace:nx {setup-alph} {\l_um_tmpa_tl~(##1)}
1167         \use:c {um_config_ \l_um_tmpa_tl _##1:n} {#3}
1168       }{
1169         \um_glyph_if_exist:cTF { \um_to_usv:nn {#3}{##1} } {
1170           \um_trace:nx {setup-alph} {\l_um_tmpa_tl~(##1)}
1171           \use:c {um_config_ \l_um_tmpa_tl _##1:n} {#3}
1172         }{
1173           \bool_if:NTF \l_um_implicit_alph_bool {
1174             \seq_put_right:Nx \l_um_missing_alph_seq {
1175               \@backslashchar
1176               \l_um_tmpa_tl\space(\tl_use:c{g_um_math_alphabet_name_##1_tl})
1177             }

```



```

1178         }{
1179         \use:c {um_config_ \l_um_tmpa_tl _##1:n} {up}
1180         }
1181     }
1182 }
1183 }
1184 }
1185 }
1186 \cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {NV,VVV}

1187 \cs_set:Nn \um_init_alphabet:n {
1188     \um_trace:nx {alph-initialise} {#1}
1189     \cs_set_eq:cN {um_setup_math#1:} \prg_do_nothing:
1190 }

```

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

```

1191 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
1192     \etex_iffontchar:D \l_um_font #1 \scan_stop:
1193     \prg_return_true:
1194     \else:
1195     \prg_return_false:
1196     \fi:
1197 }
1198 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
1199 \cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
1200 \cs_generate_variant:Nn \um_glyph_if_exist:nT {c}
1201 \cs_generate_variant:Nn \um_glyph_if_exist:nF {c}

```

9.1 Alphabets

9.1.1 Upright: `\mathup`

```

1202 \cs_new:Nn \um_config_mathup_num:n {
1203     \um_map_chars_numbers:nn {up}{#1}
1204     \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}
1205 }
1206 \cs_new:Nn \um_config_mathup_Latin:n {
1207     \bool_if:NTF \g_um_literal_bool {
1208         \um_map_chars_Latin:nn {up} {#1}
1209     }{
1210         \bool_if:NT \g_um_upLatin_bool {
1211             \um_map_chars_Latin:nn {up,it} {#1}
1212         }
1213     }
1214     \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
1215 }

```

```

1216 \cs_new:Nn \um_config_mathup_latin:n {
1217   \bool_if:NTF \g_um_literal_bool {
1218     \um_map_chars_latin:nn {up} {#1}
1219   }{
1220     \bool_if:NT \g_um_uplatin_bool {
1221       \um_map_chars_latin:nn {up,it} {#1}
1222       \um_map_single:nnn {h} {up,it} {#1}
1223       \um_map_single:nnn {dotlessi} {up,it} {#1}
1224       \um_map_single:nnn {dotlessj} {up,it} {#1}
1225     }
1226   }
1227   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
1228 }
1229 \cs_new:Nn \um_config_mathup_Greek:n {
1230   \bool_if:NTF \g_um_literal_bool {
1231     \um_map_chars_Greek:nn {up}{#1}
1232   }{
1233     \bool_if:NT \g_um_upGreek_bool {
1234       \um_map_chars_Greek:nn {up,it}{#1}
1235     }
1236   }
1237   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
1238 }
1239 \cs_new:Nn \um_config_mathup_greek:n {
1240   \bool_if:NTF \g_um_literal_bool {
1241     \um_map_chars_greek:nn {up} {#1}
1242   }{
1243     \bool_if:NT \g_um_upgreek_bool {
1244       \um_map_chars_greek:nn {up,it} {#1}
1245     }
1246   }
1247   \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
1248 }
1249 \cs_new:Nn \um_config_mathup_misc:n {
1250   \bool_if:NTF \g_um_literal_Nabla_bool {
1251     \um_map_single:nnn {Nabla}{up}{up}
1252   }{
1253     \bool_if:NT \g_um_upNabla_bool {
1254       \um_map_single:nnn {Nabla}{up,it}{up}
1255     }
1256   }
1257   \bool_if:NTF \g_um_literal_partial_bool {
1258     \um_map_single:nnn {partial}{up}{up}
1259   }{
1260     \bool_if:NT \g_um_uppartial_bool {
1261       \um_map_single:nnn {partial}{up,it}{up}

```

```

1262     }
1263   }
1264   \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it} {#1}
1265   \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it} {#1}
1266   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessi} {up,it} {#1}
1267   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessj} {up,it} {#1}
1268 }

```

9.1.2 Italic: \mathit

```

1269 \cs_new:Nn \um_config_mathit_Latin:n {
1270   \bool_if:NTF \g_um_literal_bool {
1271     \um_map_chars_Latin:nn {it} {#1}
1272   }{
1273     \bool_if:NF \g_um_upLatin_bool {
1274       \um_map_chars_Latin:nn {up,it} {#1}
1275     }
1276   }
1277   \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
1278 }
1279 \cs_new:Nn \um_config_mathit_latin:n {
1280   \bool_if:NTF \g_um_literal_bool {
1281     \um_map_chars_latin:nn {it} {#1}
1282     \um_map_single:nnn {h}{it}{#1}
1283   }{
1284     \bool_if:NF \g_um_uplatin_bool {
1285       \um_map_chars_latin:nn {up,it} {#1}
1286       \um_map_single:nnn {h}{up,it}{#1}
1287       \um_map_single:nnn {dotlessi}{up,it}{#1}
1288       \um_map_single:nnn {dotlessj}{up,it}{#1}
1289     }
1290   }
1291   \um_set_mathalphabet_latin:Nnn \mathit {up,it} {#1}
1292   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessi} {up,it} {#1}
1293   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessj} {up,it} {#1}
1294 }
1295 \cs_new:Nn \um_config_mathit_Greek:n {
1296   \bool_if:NTF \g_um_literal_bool {
1297     \um_map_chars_Greek:nn {it}{#1}
1298   }{
1299     \bool_if:NF \g_um_upGreek_bool {
1300       \um_map_chars_Greek:nn {up,it}{#1}
1301     }
1302   }
1303   \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
1304 }
1305 \cs_new:Nn \um_config_mathit_greek:n {

```

```

1306 \bool_if:NTF \g_um_literal_bool {
1307   \um_map_chars_greek:nn {it} {#1}
1308 }{
1309   \bool_if:NF \g_um_upgreek_bool {
1310     \um_map_chars_greek:nn {it,up} {#1}
1311   }
1312 }
1313 \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
1314 }
1315 \cs_new:Nn \um_config_mathit_misc:n {
1316   \bool_if:NTF \g_um_literal_Nabla_bool {
1317     \um_map_single:nnn {Nabla}{it}{it}
1318   }{
1319     \bool_if:NF \g_um_upNabla_bool {
1320       \um_map_single:nnn {Nabla}{up,it}{it}
1321     }
1322   }
1323   \bool_if:NTF \g_um_literal_partial_bool {
1324     \um_map_single:nnn {partial}{it}{it}
1325   }{
1326     \bool_if:NF \g_um_uppartial_bool {
1327       \um_map_single:nnn {partial}{up,it}{it}
1328     }
1329   }
1330   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
1331   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{#1}
1332 }

```

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

```

1333 \cs_new:Nn \um_config_mathbb_latin:n {
1334   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
1335 }
1336 \cs_new:Nn \um_config_mathbb_Latin:n {
1337   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
1338   \um_set_mathalphabet_pos:Nnnn \mathbb {C} {up,it} {#1}
1339   \um_set_mathalphabet_pos:Nnnn \mathbb {H} {up,it} {#1}
1340   \um_set_mathalphabet_pos:Nnnn \mathbb {N} {up,it} {#1}
1341   \um_set_mathalphabet_pos:Nnnn \mathbb {P} {up,it} {#1}
1342   \um_set_mathalphabet_pos:Nnnn \mathbb {Q} {up,it} {#1}
1343   \um_set_mathalphabet_pos:Nnnn \mathbb {R} {up,it} {#1}
1344   \um_set_mathalphabet_pos:Nnnn \mathbb {Z} {up,it} {#1}
1345 }
1346 \cs_new:Nn \um_config_mathbb_num:n {
1347   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
1348 }
1349 \cs_new:Nn \um_config_mathbb_misc:n {

```

```

1350 \um_set_mathalphabet_pos:Nnnn \mathbb      {\Pi} {up,it} {#1}
1351 \um_set_mathalphabet_pos:Nnnn \mathbb      {\pi} {up,it} {#1}
1352 \um_set_mathalphabet_pos:Nnnn \mathbb      {\Gamma} {up,it} {#1}
1353 \um_set_mathalphabet_pos:Nnnn \mathbb      {\gamma} {up,it} {#1}
1354 \um_set_mathalphabet_pos:Nnnn \mathbb      {\textstyle\sum} {up} {#1}
1355 }
1356 \cs_new:Nn \um_config_mathbbbit_misc:n {
1357   \um_set_mathalphabet_pos:Nnnn \mathbbbit {D} {up,it} {#1}
1358   \um_set_mathalphabet_pos:Nnnn \mathbbbit {d} {up,it} {#1}
1359   \um_set_mathalphabet_pos:Nnnn \mathbbbit {e} {up,it} {#1}
1360   \um_set_mathalphabet_pos:Nnnn \mathbbbit {i} {up,it} {#1}
1361   \um_set_mathalphabet_pos:Nnnn \mathbbbit {j} {up,it} {#1}
1362 }

```

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

```

1363 \cs_new:Nn \um_config_mathscr_Latin:n {
1364   \um_set_mathalphabet_Latin:Nnn \mathscr    {up,it}{#1}
1365   \um_set_mathalphabet_pos:Nnnn \mathscr    {B}{up,it}{#1}
1366   \um_set_mathalphabet_pos:Nnnn \mathscr    {E}{up,it}{#1}
1367   \um_set_mathalphabet_pos:Nnnn \mathscr    {F}{up,it}{#1}
1368   \um_set_mathalphabet_pos:Nnnn \mathscr    {H}{up,it}{#1}
1369   \um_set_mathalphabet_pos:Nnnn \mathscr    {I}{up,it}{#1}
1370   \um_set_mathalphabet_pos:Nnnn \mathscr    {L}{up,it}{#1}
1371   \um_set_mathalphabet_pos:Nnnn \mathscr    {M}{up,it}{#1}
1372   \um_set_mathalphabet_pos:Nnnn \mathscr    {R}{up,it}{#1}
1373 }
1374 \cs_new:Nn \um_config_mathscr_latin:n {
1375   \um_set_mathalphabet_latin:Nnn \mathscr    {up,it}{#1}
1376   \um_set_mathalphabet_pos:Nnnn \mathscr    {e}{up,it}{#1}
1377   \um_set_mathalphabet_pos:Nnnn \mathscr    {g}{up,it}{#1}
1378   \um_set_mathalphabet_pos:Nnnn \mathscr    {o}{up,it}{#1}
1379 }

```

9.1.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1380 \cs_new:Nn \um_config_mathfrak_Latin:n {
1381   \um_set_mathalphabet_Latin:Nnn \mathfrak   {up,it}{#1}
1382   \um_set_mathalphabet_pos:Nnnn \mathfrak   {C}{up,it}{#1}
1383   \um_set_mathalphabet_pos:Nnnn \mathfrak   {H}{up,it}{#1}
1384   \um_set_mathalphabet_pos:Nnnn \mathfrak   {I}{up,it}{#1}
1385   \um_set_mathalphabet_pos:Nnnn \mathfrak   {R}{up,it}{#1}
1386   \um_set_mathalphabet_pos:Nnnn \mathfrak   {Z}{up,it}{#1}
1387 }
1388 \cs_new:Nn \um_config_mathfrak_latin:n {
1389   \um_set_mathalphabet_latin:Nnn \mathfrak   {up,it}{#1}
1390 }

```

9.1.6 Sans serif upright: `\mathsfup`

```
1391 \cs_new:Nn \um_config_mathsfup_num:n {
1392   \um_set_mathalphabet_numbers:Nnn \mathsf {up}{#1}
1393   \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
1394 }
1395 \cs_new:Nn \um_config_mathsfup_Latin:n {
1396   \bool_if:NTF \g_um_sfliteral_bool {
1397     \um_map_chars_Latin:nn {sfup} {#1}
1398     \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
1399   }{
1400     \bool_if:NT \g_um_upsans_bool {
1401       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1402       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1403     }
1404   }
1405   \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}
1406 }
1407 \cs_new:Nn \um_config_mathsfup_latin:n {
1408   \bool_if:NTF \g_um_sfliteral_bool {
1409     \um_map_chars_latin:nn {sfup} {#1}
1410     \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
1411   }{
1412     \bool_if:NT \g_um_upsans_bool {
1413       \um_map_chars_latin:nn {sfup,sfit} {#1}
1414       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1415     }
1416   }
1417   \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
1418 }
```

9.1.7 Sans serif italic: `\mathsfit`

```
1419 \cs_new:Nn \um_config_mathsfit_Latin:n {
1420   \bool_if:NTF \g_um_sfliteral_bool {
1421     \um_map_chars_Latin:nn {sfit} {#1}
1422     \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
1423   }{
1424     \bool_if:NF \g_um_upsans_bool {
1425       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1426       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1427     }
1428   }
1429   \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{#1}
1430 }
1431 \cs_new:Nn \um_config_mathsfit_latin:n {
1432   \bool_if:NTF \g_um_sfliteral_bool {
1433     \um_map_chars_latin:nn {sfit} {#1}
```

```

1434 \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
1435 }{
1436 \bool_if:NF \g_um_upsans_bool {
1437 \um_map_chars_latin:nn {sfup,sfit} {#1}
1438 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1439 }
1440 }
1441 \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1442 }

```

9.1.8 Typewriter or monospaced: `\mathtt`

```

1443 \cs_new:Nn \um_config_mathtt_num:n {
1444 \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
1445 }
1446 \cs_new:Nn \um_config_mathtt_Latin:n {
1447 \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
1448 }
1449 \cs_new:Nn \um_config_mathtt_latin:n {
1450 \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
1451 }

```

9.1.9 Bold Italic: `\mathbfit`

```

1452 \cs_new:Nn \um_config_mathbfit_Latin:n {
1453 \bool_if:NF \g_um_bfupLatin_bool {
1454 \um_map_chars_Latin:nn {bfup,bfit} {#1}
1455 }
1456 \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
1457 \bool_if:NTF \g_um_bfliteral_bool {
1458 \um_map_chars_Latin:nn {bfit} {#1}
1459 \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
1460 }{
1461 \bool_if:NF \g_um_bfupLatin_bool {
1462 \um_map_chars_Latin:nn {bfup,bfit} {#1}
1463 \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1464 }
1465 }
1466 }
1467 \cs_new:Nn \um_config_mathbfit_latin:n {
1468 \bool_if:NF \g_um_bfuplatin_bool {
1469 \um_map_chars_latin:nn {bfup,bfit} {#1}
1470 }
1471 \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
1472 \bool_if:NTF \g_um_bfliteral_bool {
1473 \um_map_chars_latin:nn {bfit} {#1}
1474 \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
1475 }{
1476 \bool_if:NF \g_um_bfuplatin_bool {

```

```

1477     \um_map_chars_latin:nn {bfup,bfit} {#1}
1478     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1479   }
1480 }
1481 }
1482 \cs_new:Nn \um_config_mathbfit_Greek:n {
1483   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
1484   \bool_if:NTF \g_um_bfliteral_bool {
1485     \um_map_chars_Greek:nn {bfit}{#1}
1486     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
1487   }{
1488     \bool_if:NF \g_um_bfupGreek_bool {
1489       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1490       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1491     }
1492   }
1493 }
1494 \cs_new:Nn \um_config_mathbfit_greek:n {
1495   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
1496   \bool_if:NTF \g_um_bfliteral_bool {
1497     \um_map_chars_greek:nn {bfit} {#1}
1498     \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
1499   }{
1500     \bool_if:NF \g_um_bfupgreek_bool {
1501       \um_map_chars_greek:nn {bfit,bfup} {#1}
1502       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1503     }
1504   }
1505 }
1506 \cs_new:Nn \um_config_mathbfit_misc:n {
1507   \bool_if:NTF \g_um_literal_Nabla_bool {
1508     \um_map_single:nnn {Nabla}{bfit}{#1}
1509   }{
1510     \bool_if:NF \g_um_upNabla_bool {
1511       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1512     }
1513   }
1514   \bool_if:NTF \g_um_literal_partial_bool {
1515     \um_map_single:nnn {partial}{bfit}{#1}
1516   }{
1517     \bool_if:NF \g_um_uppartial_bool {
1518       \um_map_single:nnn {partial}{bfup,bfit}{#1}
1519     }
1520   }
1521   \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{#1}
1522   \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{#1}

```



```

1523 \bool_if:NTF \g_um_literal_partial_bool {
1524   \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{#1}
1525 }{
1526   \bool_if:NF \g_um_uppartial_bool {
1527     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1528   }
1529 }
1530 \bool_if:NTF \g_um_literal_Nabla_bool {
1531   \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {it}{#1}
1532 }{
1533   \bool_if:NF \g_um_upNabla_bool {
1534     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1535   }
1536 }
1537 }

```

9.1.10 Bold Upright: `\mathbfup`

```

1538 \cs_new:Nn \um_config_mathbfup_num:n {
1539   \um_set_mathalphabet_numbers:Nnn \mathbf {up}{#1}
1540   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
1541 }
1542 \cs_new:Nn \um_config_mathbfup_Latin:n {
1543   \bool_if:NT \g_um_bfupLatin_bool {
1544     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1545   }
1546   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
1547   \bool_if:NTF \g_um_bfliteral_bool {
1548     \um_map_chars_Latin:nn {bfup} {#1}
1549     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
1550   }{
1551     \bool_if:NT \g_um_bfupLatin_bool {
1552       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1553       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1554     }
1555   }
1556 }
1557 \cs_new:Nn \um_config_mathbfup_latin:n {
1558   \bool_if:NT \g_um_bfuplatin_bool {
1559     \um_map_chars_latin:nn {bfup,bfit} {#1}
1560   }
1561   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
1562   \bool_if:NTF \g_um_bfliteral_bool {
1563     \um_map_chars_latin:nn {bfup} {#1}
1564     \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
1565   }{
1566     \bool_if:NT \g_um_bfuplatin_bool {

```

```

1567     \um_map_chars_latin:nn {bfup,bfit} {#1}
1568     \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1569   }
1570 }
1571 }
1572 \cs_new:Nn \um_config_mathbfup_Greek:n {
1573   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
1574   \bool_if:NTF \g_um_bfliteral_bool {
1575     \um_map_chars_Greek:nn {bfup}{#1}
1576     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}
1577   }{
1578     \bool_if:NT \g_um_bfupGreek_bool {
1579       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1580       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1581     }
1582   }
1583 }
1584 \cs_new:Nn \um_config_mathbfup_greek:n {
1585   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
1586   \bool_if:NTF \g_um_bfliteral_bool {
1587     \um_map_chars_greek:nn {bfup} {#1}
1588     \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
1589   }{
1590     \bool_if:NT \g_um_bfupgreek_bool {
1591       \um_map_chars_greek:nn {bfup,bfit} {#1}
1592       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1593     }
1594   }
1595 }
1596 \cs_new:Nn \um_config_mathbfup_misc:n {
1597   \bool_if:NTF \g_um_literal_Nabla_bool {
1598     \um_map_single:nnn {Nabla}{bfup}{#1}
1599   }{
1600     \bool_if:NT \g_um_upNabla_bool {
1601       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1602     }
1603   }
1604   \bool_if:NTF \g_um_literal_partial_bool {
1605     \um_map_single:nnn {partial}{bfup}{#1}
1606   }{
1607     \bool_if:NT \g_um_uppartial_bool {
1608       \um_map_single:nnn {partial}{bfup,bfit}{#1}
1609     }
1610   }
1611   \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{#1}
1612   \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{#1}

```

```

1613 \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{#1}
1614 \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{#1}
1615 \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{#1}
1616 \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{#1}
1617 \bool_if:NTF \g_um_literal_partial_bool {
1618   \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{#1}
1619 }{
1620   \bool_if:NT \g_um_uppartial_bool {
1621     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1622   }
1623 }
1624 \bool_if:NTF \g_um_literal_Nabla_bool {
1625   \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{#1}
1626 }{
1627   \bool_if:NT \g_um_upNabla_bool {
1628     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1629   }
1630 }
1631 }

```

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

```

1632 \cs_new:Nn \um_config_mathbffrak_Latin:n {
1633   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
1634 }
1635 \cs_new:Nn \um_config_mathbffrak_latin:n {
1636   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
1637 }

```

9.1.12 Bold script or calligraphic: `\mathbfscr`

```

1638 \cs_new:Nn \um_config_mathbfscr_Latin:n {
1639   \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{#1}
1640 }
1641 \cs_new:Nn \um_config_mathbfscr_latin:n {
1642   \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{#1}
1643 }

```

9.1.13 Bold upright sans serif: `\mathbfsfup`

```

1644 \cs_new:Nn \um_config_mathbfsfup_num:n {
1645   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{#1}
1646   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
1647 }
1648 \cs_new:Nn \um_config_mathbfsfup_Latin:n {
1649   \bool_if:NTF \g_um_sfliteral_bool {
1650     \um_map_chars_Latin:nn {bfsfup} {#1}
1651     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
1652   }{

```

```

1653     \bool_if:NT \g_um_upsans_bool {
1654         \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1655         \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1656     }
1657 }
1658 \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
1659 }
1660 \cs_new:Nn \um_config_mathbfsfup_latin:n {
1661     \bool_if:NTF \g_um_sfliteral_bool {
1662         \um_map_chars_latin:nn {bfsfup} {#1}
1663         \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
1664     }{
1665         \bool_if:NT \g_um_upsans_bool {
1666             \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1667             \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1668         }
1669     }
1670     \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
1671 }
1672 \cs_new:Nn \um_config_mathbfsfup_greek:n {
1673     \bool_if:NTF \g_um_sfliteral_bool {
1674         \um_map_chars_greek:nn {bfsfup}{#1}
1675         \um_set_mathalphabet_greek:Nnn \mathbfsf {up}{#1}
1676     }{
1677         \bool_if:NT \g_um_upsans_bool {
1678             \um_map_chars_greek:nn {bfsfup,bfsfit}{#1}
1679             \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it}{#1}
1680         }
1681     }
1682     \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it}{#1}
1683 }
1684 \cs_new:Nn \um_config_mathbfsfup_greek:n {
1685     \bool_if:NTF \g_um_sfliteral_bool {
1686         \um_map_chars_greek:nn {bfsfup} {#1}
1687         \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
1688     }{
1689         \bool_if:NT \g_um_upsans_bool {
1690             \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1691             \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1692         }
1693     }
1694     \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
1695 }
1696 \cs_new:Nn \um_config_mathbfsfup_misc:n {
1697     \bool_if:NTF \g_um_literal_Nabla_bool {
1698         \um_map_single:nnn {Nabla}{bfsfup}{#1}

```

```

1699 }{
1700   \bool_if:NT \g_um_upNabla_bool {
1701     \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
1702   }
1703 }
1704 \bool_if:NTF \g_um_literal_partial_bool {
1705   \um_map_single:nnn {partial}{bfsfup}{#1}
1706 }{
1707   \bool_if:NT \g_um_uppartial_bool {
1708     \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
1709   }
1710 }
1711 \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{#1}
1712 \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{#1}
1713 \bool_if:NTF \g_um_literal_partial_bool {
1714   \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{#1}
1715 }{
1716   \bool_if:NT \g_um_uppartial_bool {
1717     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1718   }
1719 }
1720 \bool_if:NTF \g_um_literal_Nabla_bool {
1721   \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{#1}
1722 }{
1723   \bool_if:NT \g_um_upNabla_bool {
1724     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1725   }
1726 }
1727 }

```

9.1.14 Bold italic sans serif: `\mathbfsfit`

```

1728 \cs_new:Nn \um_config_mathbfsfit_Latin:n {
1729   \bool_if:NTF \g_um_sfliteral_bool {
1730     \um_map_chars_Latin:nn {bfsfit} {#1}
1731     \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
1732   }{
1733     \bool_if:NF \g_um_upsans_bool {
1734       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1735       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1736     }
1737   }
1738   \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
1739 }
1740 \cs_new:Nn \um_config_mathbfsfit_latin:n {
1741   \bool_if:NTF \g_um_sfliteral_bool {
1742     \um_map_chars_latin:nn {bfsfit} {#1}

```

```

1743 \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
1744 }{
1745 \bool_if:NF \g_um_upsans_bool {
1746 \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1747 \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1748 }
1749 }
1750 \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{#1}
1751 }
1752 \cs_new:Nn \um_config_mathbfsfit_Greek:n {
1753 \bool_if:NTF \g_um_sfliteral_bool {
1754 \um_map_chars_Greek:nn {bfsfit}{#1}
1755 \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{#1}
1756 }{
1757 \bool_if:NF \g_um_upsans_bool {
1758 \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
1759 \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
1760 }
1761 }
1762 \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{#1}
1763 }
1764 \cs_new:Nn \um_config_mathbfsfit_greek:n {
1765 \bool_if:NTF \g_um_sfliteral_bool {
1766 \um_map_chars_greek:nn {bfsfit} {#1}
1767 \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
1768 }{
1769 \bool_if:NF \g_um_upsans_bool {
1770 \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1771 \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1772 }
1773 }
1774 \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
1775 }
1776 \cs_new:Nn \um_config_mathbfsfit_misc:n {
1777 \bool_if:NTF \g_um_literal_Nabla_bool {
1778 \um_map_single:nnn {Nabla}{bfsfit}{#1}
1779 }{
1780 \bool_if:NF \g_um_upNabla_bool {
1781 \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
1782 }
1783 }
1784 \bool_if:NTF \g_um_literal_partial_bool {
1785 \um_map_single:nnn {partial}{bfsfit}{#1}
1786 }{
1787 \bool_if:NF \g_um_uppartial_bool {
1788 \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}

```

```

1789     }
1790   }
1791   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{#1}
1792   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{#1}
1793   \bool_if:NTF \g_um_literal_partial_bool {
1794     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{#1}
1795   }{
1796     \bool_if:NF \g_um_uppartial_bool {
1797       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1798     }
1799   }
1800   \bool_if:NTF \g_um_literal_Nabla_bool {
1801     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{#1}
1802   }{
1803     \bool_if:NF \g_um_upNabla_bool {
1804       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1805     }
1806   }
1807 }

```

10 Definitions of the math symbols

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

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

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

```

1808 \begingroup
1809   \char_make_other:N \^
1810   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
1811     \lowercase{
1812       \tl_rescan:nn {
1813         \char_make_other:N \{
1814         \char_make_other:N \}
1815         \char_make_other:N \&
1816         \char_make_other:N \%
1817         \char_make_other:N \$
1818       }{
1819         \global\let#1=^^^#2
1820       }
1821     }

```

```
1822 }
```

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

```
1823 \gdef\um@scanactivedef"#1\@nil#2{
1824   \lowercase{
1825     \tl_rescan:nn{
1826       \ExplSyntaxOn
1827       \char_make_math_superscript:N\^
1828     }{
1829       \global\def^^^^#1{#2}
1830     }
1831   }
1832 }
1833 \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`.

```
1834 \AtBeginDocument{
1835   \group_begin:
1836     \char_make_math_superscript:N\^
1837     \def\UnicodeMathSymbol#1#2#3#4{
1838       \bool_if:nF { \cs_if_eq_p:NN #3 \mathaccent ||
1839                     \cs_if_eq_p:NN #3 \mathopen   ||
1840                     \cs_if_eq_p:NN #3 \mathclose } {
1841         \um@scancharlet#2=#1\@nil\ignorespaces
1842       }
1843     }
1844     \char_make_other:N \#
1845     \@input{unicode-math-table.tex}
1846   \group_end:
1847 }
```

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

```
1848 \group_begin:
1849   \lccode`\*=\ \ \
1850   \char_make_escape:N \ |
1851   \char_make_other:N \ \
1852   |lowercase{
1853     |AtBeginDocument{
1854       |let|backslash=*
1855     }
1856   }
1857 |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.

```
1858 \muskip_new:N \g_um_primekern_muskip
1859 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
1860 \int_new:N \l_um_primecount_int
```

```

1861 \cs_new:Nn \um_nprimes:Nn {
1862   ^{
1863     #1
1864     \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
1865   }
1866 }
1867 \cs_new:Nn \um_nprimes_select:nn {
1868   \prg_case_int:nnn {#2}{
1869     {1} { ^{#1} }
1870     {2} {
1871       \um_glyph_if_exist:nTF {"2033} { ^{\um_prime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1872     }
1873     {3} {
1874       \um_glyph_if_exist:nTF {"2034} { ^{\um_prime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1875     }
1876     {4} {
1877       \um_glyph_if_exist:nTF {"2057} { ^{\um_prime_quad_mchar} } {\um_nprimes:Nn #1 {#2}}
1878     }
1879   }{
1880     \um_nprimes:Nn #1 {#2}
1881   }
1882 }
1883 \cs_new:Nn \um_nbackprimes_select:nn {
1884   \prg_case_int:nnn {#2}{
1885     {1} { ^{#1} }
1886     {2} {
1887       \um_glyph_if_exist:nTF {"2033} { ^{\um_backprime_double_mchar} } {\um_nprimes:Nn #1 {#2}}
1888     }
1889     {3} {
1890       \um_glyph_if_exist:nTF {"2034} { ^{\um_backprime_triple_mchar} } {\um_nprimes:Nn #1 {#2}}
1891     }
1892   }{
1893     \um_nprimes:Nn #1 {#2}
1894   }
1895 }

```

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

```

1896 \cs_new:Nn \um_scan_prime: {
1897   \int_zero:N \l_um_primecount_int
1898   \um_scanprime_collect:N \um_prime_single_mchar
1899 }
1900 \cs_new:Nn \um_scan_dprime: {
1901   \int_set:Nn \l_um_primecount_int {1}
1902   \um_scanprime_collect:N \um_prime_single_mchar
1903 }
1904 \cs_new:Nn \um_scan_trprime: {
1905   \int_set:Nn \l_um_primecount_int {2}

```

```

1906 \um_scanprime_collect:N \um_prime_single_mchar
1907 }
1908 \cs_new:Nn \um_scan_qprime: {
1909   \int_set:Nn \l_um_primecount_int {3}
1910   \um_scanprime_collect:N \um_prime_single_mchar
1911 }
1912 \cs_new:Nn \um_scanprime_collect:N {
1913   \int_incr:N \l_um_primecount_int
1914   \peek_meaning_remove:NTF ' {
1915     \um_scanprime_collect:N #1
1916   }{
1917     \peek_meaning_remove:NTF \um_scan_prime: {
1918       \um_scanprime_collect:N #1
1919     }{
1920       \peek_meaning_remove:NTF ^^^^2032 {
1921         \um_scanprime_collect:N #1
1922       }{
1923         \peek_meaning_remove:NTF \um_scan_dprime: {
1924           \int_incr:N \l_um_primecount_int
1925           \um_scanprime_collect:N #1
1926         }{
1927           \peek_meaning_remove:NTF ^^^^2033 {
1928             \int_incr:N \l_um_primecount_int
1929             \um_scanprime_collect:N #1
1930           }{
1931             \peek_meaning_remove:NTF \um_scan_trprime: {
1932               \int_add:Nn \l_um_primecount_int {2}
1933               \um_scanprime_collect:N #1
1934             }{
1935               \peek_meaning_remove:NTF ^^^^2034 {
1936                 \int_add:Nn \l_um_primecount_int {2}
1937                 \um_scanprime_collect:N #1
1938               }{
1939                 \peek_meaning_remove:NTF \um_scan_qprime: {
1940                   \int_add:Nn \l_um_primecount_int {3}
1941                   \um_scanprime_collect:N #1
1942                 }{
1943                   \peek_meaning_remove:NTF ^^^^2057 {
1944                     \int_add:Nn \l_um_primecount_int {3}
1945                     \um_scanprime_collect:N #1
1946                   }{
1947                     \um_nprimes_select:nn {#1} {\l_um_primecount_int}
1948                   }
1949                 }
1950             }
1951           }

```

```

1952     }
1953   }
1954 }
1955 }
1956 }
1957 }
1958 \cs_new:Nn \um_scan_backprime: {
1959   \int_zero:N \l_um_primecount_int
1960   \um_scanbackprime_collect:N \um_backprime_single_mchar
1961 }
1962 \cs_new:Nn \um_scan_backdprime: {
1963   \int_set:Nn \l_um_primecount_int {1}
1964   \um_scanbackprime_collect:N \um_backprime_single_mchar
1965 }
1966 \cs_new:Nn \um_scan_backtrprime: {
1967   \int_set:Nn \l_um_primecount_int {2}
1968   \um_scanbackprime_collect:N \um_backprime_single_mchar
1969 }
1970 \cs_new:Nn \um_scanbackprime_collect:N {
1971   \int_incr:N \l_um_primecount_int
1972   \peek_meaning_remove:NTF ` {
1973     \um_scanbackprime_collect:N #1
1974   }{
1975     \peek_meaning_remove:NTF \um_scan_backprime: {
1976       \um_scanbackprime_collect:N #1
1977     }{
1978       \peek_meaning_remove:NTF ^^^^2035 {
1979         \um_scanbackprime_collect:N #1
1980       }{
1981         \peek_meaning_remove:NTF \um_scan_backdprime: {
1982           \int_incr:N \l_um_primecount_int
1983           \um_scanbackprime_collect:N #1
1984         }{
1985           \peek_meaning_remove:NTF ^^^^2036 {
1986             \int_incr:N \l_um_primecount_int
1987             \um_scanbackprime_collect:N #1
1988           }{
1989             \peek_meaning_remove:NTF \um_scan_backtrprime: {
1990               \int_add:Nn \l_um_primecount_int {2}
1991               \um_scanbackprime_collect:N #1
1992             }{
1993               \peek_meaning_remove:NTF ^^^^2037 {
1994                 \int_add:Nn \l_um_primecount_int {2}
1995                 \um_scanbackprime_collect:N #1
1996               }{
1997                 \um_nbackprimes_select:nn {#1} {\l_um_primecount_int}

```

```

1998     }
1999   }
2000 }
2001   }
2002 }
2003 }
2004 }
2005 }

2006 \AtBeginDocument {
2007   \cs_set_eq:NN \prime      \um_scan_prime:
2008   \cs_set_eq:NN \drime      \um_scan_dprime:
2009   \cs_set_eq:NN \trprime    \um_scan_trprime:
2010   \cs_set_eq:NN \qprime     \um_scan_qprime:
2011   \cs_set_eq:NN \backprime  \um_scan_backprime:
2012   \cs_set_eq:NN \backdprime \um_scan_backdprime:
2013   \cs_set_eq:NN \backtrprime \um_scan_backtrprime:
2014 }
2015 \group_begin:
2016   \char_make_active:N \'
2017   \char_make_active:N `
2018   \char_make_active:n {"2032}
2019   \char_make_active:n {"2033}
2020   \char_make_active:n {"2034}
2021   \char_make_active:n {"2057}
2022   \char_make_active:n {"2035}
2023   \char_make_active:n {"2036}
2024   \char_make_active:n {"2037}
2025 \AtBeginDocument{
2026   \cs_set_eq:NN '      \um_scan_prime:
2027   \cs_set_eq:NN ^^^^2032 \um_scan_prime:
2028   \cs_set_eq:NN ^^^^2033 \um_scan_dprime:
2029   \cs_set_eq:NN ^^^^2034 \um_scan_trprime:
2030   \cs_set_eq:NN ^^^^2057 \um_scan_qprime:
2031   \cs_set_eq:NN `      \um_scan_backprime:
2032   \cs_set_eq:NN ^^^^2035 \um_scan_backprime:
2033   \cs_set_eq:NN ^^^^2036 \um_scan_backdprime:
2034   \cs_set_eq:NN ^^^^2037 \um_scan_backtrprime:
2035 }
2036 \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.

```

2037 \cs_set_nopar:Npn \r@@t #1#2 {
2038   \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
2039   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen63\l_um_font}
2040   \raise \dimexpr(
2041     \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-
2042     \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
2043   )\relax
2044   \copy \rootbox
2045   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen64\l_um_font}
2046   \box \z@
2047 }

```

`\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.

```

2048 \cs_new:Nn \um_fontdimen_to_percent:nn {
2049   0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
2050 }

```

`\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.

```

2051 \cs_new:Nn \um_mathstyle_scale:Nnn {
2052   \ifx#1\scriptstyle
2053     #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
2054   \else
2055     \ifx#1\scriptscriptstyle
2056       #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
2057     \else
2058       #2#3
2059     \fi
2060   \fi
2061 }

```

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 \XeTeX 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:

```

2062 \prop_new:N \g_um_supers_prop
2063 \prop_new:N \g_um_subs_prop
2064
2065 \group_begin:
2066
2067 % Populate a property list with superscript characters; their mean-
      ing as their key,
2068 % for reasons that will become apparent soon, and their replace-
      ment as each key's value.
2069 % Then make the superscript active and bind it to the scanning function.
2070 %
2071 % \cs{scantokens} makes this process much simpler since we can acti-
      vate the char
2072 % and assign its meaning in one step.
2073 \cs_set:Nn \um_setup_active_superscript:nn {
2074   \prop_gput:Nxn \g_um_supers_prop {\meaning #1} {#2}
2075   \char_make_active:N #1
2076   \char_gmake_mathactive:N #1
2077   \scantokens{
2078     \cs_gset:Npn #1 {
2079       \tl_set:Nn \l_um_ss_chain_tl {#2}
2080       \cs_set_eq:NN \um_sub_or_super:n \sp
2081       \tl_set:Nn \l_um_tmpa_tl {supers}
2082       \um_scan_sscript:
2083     }
2084   }
2085 }
2086
2087 \um_setup_active_superscript:nn {^^^2070} {0}
2088 \um_setup_active_superscript:nn {^^^00b9} {1}
2089 \um_setup_active_superscript:nn {^^^00b2} {2}
2090 \um_setup_active_superscript:nn {^^^00b3} {3}
2091 \um_setup_active_superscript:nn {^^^2074} {4}
2092 \um_setup_active_superscript:nn {^^^2075} {5}
2093 \um_setup_active_superscript:nn {^^^2076} {6}
2094 \um_setup_active_superscript:nn {^^^2077} {7}
2095 \um_setup_active_superscript:nn {^^^2078} {8}
2096 \um_setup_active_superscript:nn {^^^2079} {9}
2097 \um_setup_active_superscript:nn {^^^207a} {+}
2098 \um_setup_active_superscript:nn {^^^207b} {-}
2099 \um_setup_active_superscript:nn {^^^207c} {=}
2100 \um_setup_active_superscript:nn {^^^207d} {(}
2101 \um_setup_active_superscript:nn {^^^207e} {)}

```

```

2102 \um_setup_active_superscript:nn {^^^2071} {i}
2103 \um_setup_active_superscript:nn {^^^207f} {n}
2104
2105 % Ditto above.
2106 \cs_set:Nn \um_setup_active_subscript:nn {
2107   \prop_gput:Nxn \g_um_subs_prop {\meaning #1} {#2}
2108   \char_make_active:N #1
2109   \char_gmake_mathactive:N #1
2110   \scantokens{
2111     \cs_gset:Npn #1 {
2112       \tl_set:Nn \l_um_ss_chain_tl {#2}
2113       \cs_set_eq:NN \um_sub_or_super:n \sb
2114       \tl_set:Nn \l_um_tmpa_tl {subs}
2115       \um_scan_sscript:
2116     }
2117   }
2118 }
2119
2120 \um_setup_active_subscript:nn {^^^2080} {0}
2121 \um_setup_active_subscript:nn {^^^2081} {1}
2122 \um_setup_active_subscript:nn {^^^2082} {2}
2123 \um_setup_active_subscript:nn {^^^2083} {3}
2124 \um_setup_active_subscript:nn {^^^2084} {4}
2125 \um_setup_active_subscript:nn {^^^2085} {5}
2126 \um_setup_active_subscript:nn {^^^2086} {6}
2127 \um_setup_active_subscript:nn {^^^2087} {7}
2128 \um_setup_active_subscript:nn {^^^2088} {8}
2129 \um_setup_active_subscript:nn {^^^2089} {9}
2130 \um_setup_active_subscript:nn {^^^208a} {+}
2131 \um_setup_active_subscript:nn {^^^208b} {-}
2132 \um_setup_active_subscript:nn {^^^208c} {=}
2133 \um_setup_active_subscript:nn {^^^208d} {(}
2134 \um_setup_active_subscript:nn {^^^208e} {)}
2135 \um_setup_active_subscript:nn {^^^2090} {a}
2136 \um_setup_active_subscript:nn {^^^2091} {e}
2137 \um_setup_active_subscript:nn {^^^1d62} {i}
2138 \um_setup_active_subscript:nn {^^^2092} {o}
2139 \um_setup_active_subscript:nn {^^^1d63} {r}
2140 \um_setup_active_subscript:nn {^^^1d64} {u}
2141 \um_setup_active_subscript:nn {^^^1d65} {v}
2142 \um_setup_active_subscript:nn {^^^2093} {x}
2143 \um_setup_active_subscript:nn {^^^1d66} {\beta}
2144 \um_setup_active_subscript:nn {^^^1d67} {\gamma}
2145 \um_setup_active_subscript:nn {^^^1d68} {\rho}
2146 \um_setup_active_subscript:nn {^^^1d69} {\phi}
2147 \um_setup_active_subscript:nn {^^^1d6a} {\chi}

```



```

2148
2149 \group_end:
2150
2151 % The scanning command, evident in its purpose:
2152 \cs_new:Nn \um_scan_sscript: {
2153   \um_scan_sscript:TF {
2154     \um_scan_sscript:
2155   }{
2156     \um_sub_or_super:n {\l_um_ss_chain_tl}
2157   }
2158 }
2159
2160 % The main theme here is stolen from the source to the vari-
2161   ous \cs{peek_} functions.
2162 % Consider this function as simply boilerplate:
2163 \cs_new:Nn \um_scan_sscript:TF {
2164   \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
2165   \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
2166   \tl_set:Nx \l_peek_false_tl { \exp_not:n{\group_align_safe_end: #2}}
2167   \group_align_safe_begin:
2168   \peek_after:NN \um_peek_execute_branches_ss:
2169 }
2170
2171 % We do not skip spaces when scanning ahead, and we explicitly wish to
2172 % bail out on encountering a space or a brace.
2173 \cs_new:Npn \um_peek_execute_branches_ss: {
2174   \bool_if:nTF {
2175     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2176     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2177     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2178   }
2179   { \l_peek_false_tl }
2180   { \um_peek_execute_branches_ss_aux: }
2181 }
2182
2183 % This is the actual comparison code.
2184 % Because the peeking has already tokenised the next token,
2185 % it's too late to extract its charcode directly. Instead,
2186 % we look at its meaning, which remains a `character' even
2187 % though it is itself math-active. If the character is ever
2188 % made fully active, this will break our assumptions!
2189 %
2190 % If the char's meaning exists as a property list key, we
2191 % build up a chain of sub-/superscripts and iterate. (If not, exit and
2192 % typeset what we've already collected.)
2193 \cs_new:Nn \um_peek_execute_branches_ss_aux: {

```

```

2193 \prop_if_in:cxTF
2194   {g_um_\l_um_tmpa_tl _prop}
2195   {\meaning\l_peek_token}
2196   {
2197     \prop_get:cxN
2198     {g_um_\l_um_tmpa_tl _prop}
2199     {\meaning\l_peek_token}
2200     \l_um_tmpb_tl
2201     \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
2202     \l_peek_true_tl
2203   }
2204   {\l_peek_false_tl}
2205 }

```

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.

```

2206 \cs_new:Nn \um_setup_active_frac: {
2207   \group_begin:
2208   \um_define_active_frac:Nw ^^^^2152 1/{10}
2209   \um_define_active_frac:Nw ^^^^2151 1/9
2210   \um_define_active_frac:Nw ^^^^215b 1/8
2211   \um_define_active_frac:Nw ^^^^2150 1/7
2212   \um_define_active_frac:Nw ^^^^2159 1/6
2213   \um_define_active_frac:Nw ^^^^2155 1/5
2214   \um_define_active_frac:Nw ^^^^00bc 1/4
2215   \um_define_active_frac:Nw ^^^^2153 1/3
2216   \um_define_active_frac:Nw ^^^^215c 3/8
2217   \um_define_active_frac:Nw ^^^^2156 2/5
2218   \um_define_active_frac:Nw ^^^^00bd 1/2
2219   \um_define_active_frac:Nw ^^^^2157 3/5
2220   \um_define_active_frac:Nw ^^^^215d 5/8
2221   \um_define_active_frac:Nw ^^^^2154 2/3
2222   \um_define_active_frac:Nw ^^^^00be 3/4
2223   \um_define_active_frac:Nw ^^^^2158 4/5
2224   \um_define_active_frac:Nw ^^^^215a 5/6
2225   \um_define_active_frac:Nw ^^^^215e 7/8
2226   \group_end:
2227 }
2228 \cs_new:Npn \um_define_active_frac:Nw #1 #2/#3 {
2229   \char_make_active:n {\`#1}
2230   \char_gmake_mathactive:N #1
2231   \tl_rescan:nn {
2232     \ExplSyntaxOn

```

```

2233 }{
2234   \cs_gset:Npx #1 {
2235     \bool_if:NTF \l_um_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2236     {#2} {#3}
2237   }
2238 }
2239 }
2240 \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`:

```

2241 \def\operator@font{\um_setup_mathup:}
2242 \def\to{\rightarrow}
2243 \def\overrightarrow{\vec}
2244 \def\le{\leq}
2245 \def\ge{\geq}
2246 \def\neq{\ne}
2247 \def\triangle{\mathord{\bigtriangleup}}
2248 \def\bigcirc{\mdlgwhtcircle}
2249 \def\circ{\vysmwhtcircle}
2250 \def\bullet{\smblkcircle}
2251 \def\mathyen{\yen}
2252 \def\mathsterling{\sterling}

```

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

```

2253 \@ifpackageloaded{amsmath}{
2254   % define their own colon, perhaps I should just steal it. (It does look much bet-
2255   % ter.)
2256 }{
2257   \cs_set_protected:Npn \colon {
2258     \bool_if:NTF \g_um_literal_colon_bool {:\} { \mathpunct{:\} }
2259   }
2260 }

```

`\mathcal`

```

2260 \def\mathcal{\mathscr}

```

`\mathrm`

```

2261 \def\mathrm{\mathup}
2262 \let\mathfence\mathord

```

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

```

\Digamma 2263 \def\digamma{\updigamma}
2264 \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).

```
2265 \cs_new:Nn \um_patch_pkg:nn {
2266   \@ifpackageloaded {#1} {
2267     #2
2268   }{
2269     \um_after_pkg:nn {#1} {#2}
2270   }
2271 }
```

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_setup_mathup:}` but activates automatically so old documents that might change the `\url` font still work correctly.

```
2272 \um_patch_pkg:nn {url} {
2273   \tl_put_left:Nn \Url@FormatString { \um_setup_mathup: }
2274   \tl_put_right:Nn \UrlSpecials {
2275     \do\`{\mathchar`\` }
2276     \do\'\{\mathchar`\' }
2277     \do\$\{\mathchar`\$ }
2278     \do\&\{\mathchar`\& }
2279   }
2280 }
```

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:

```
2281 \um_patch_pkg:nn {amsmath} {
2282   \tl_remove_in:Nn \@begindocumenthook {
2283     \mathchardef\std@minus\mathcode`\-\relax
2284     \mathchardef\std@equal\mathcode`\=\relax
2285   }
2286   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
2287   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
2288   \def\@cdots{\mathinner{\cdots}}
2289   \cs_set_eq:NN \dotsb@ \cdots
2290 }
```

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:

```

2291 \um_patch_pkg:nn {amsopn} {
2292   \cs_set:Npn \newmcodes@ {
2293     \mathcode`\'39\scan_stop:
2294     \mathcode`\'*42\scan_stop:
2295     \mathcode`\."613A\scan_stop:
2296     %% \ifnum\mathcode`\'=-45 \else
2297     %%   \mathchardef\std@minus\mathcode`\'-\relax
2298     %% \fi
2299     \mathcode`\'-45\scan_stop:
2300     \mathcode`\'/47\scan_stop:
2301     \mathcode`\':"603A\scan_stop:
2302   }
2303 }

```

Symbols

```

2304 \cs_set:Npn \lvert {\Vert}
\mathinner items:
2305 \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2306 \cs_set:Npn \cdots {\mathinner{\unicodecdots}}

```

Accents

```

2307 \AtBeginDocument{
2308   \def\widehat{\hat}
2309   \def\widetilde{\tilde}
2310 }

```

beamer

```

2311 \@ifclassloaded{beamer}{
2312   \ifbeamer@suppressreplacements\else
2313     \um_warning:n {disable-beamer}
2314     \beamer@suppressreplacementstrue
2315   \fi
2316 }{}

```

12 Error messages

Wrapper functions:

```

2317 \cs_new:Npn \um_warning:n { \msg_warning:nn {unicode-math} }
2318 \cs_new:Npn \um_trace:n { \msg_trace:nn {unicode-math} }
2319 \cs_new:Npn \um_trace:nx { \msg_trace:nxx {unicode-math} }

```

```

2320 \msg_new:nnn {unicode-math} {maths-feature-only}
2321 {
2322   The '#1' font feature can only be used for maths fonts.
2323 }
2324 \msg_new:nnn {unicode-math} {disable-beamer}
2325 {
2326   Disabling beamer's math setup.\
2327   Please load beamer with the [professionalfonts] class option.
2328 }
2329 \msg_new:nnn {unicode-math} {no-tfrac}
2330 {
2331   Small fraction command \protect\tfrac\ not defined.\
2332   Load amsmath or define it manually before loading unicode-math.
2333 }
2334 \msg_new:nnn {unicode-math} {default-math-font}
2335 {
2336   Defining the default maths font as '#1'.
2337 }
2338 \msg_new:nnn {unicode-math} {setup-implicit}
2339 {
2340   Setup alphabets: implicit mode.
2341 }
2342 \msg_new:nnn {unicode-math} {setup-explicit}
2343 {
2344   Setup alphabets: explicit mode.
2345 }
2346 \msg_new:nnn {unicode-math} {alph-initialise}
2347 {
2348   Initialising \@backslashchar math#1.
2349 }
2350 \msg_new:nnn {unicode-math} {setup-alph}
2351 {
2352   Setup~ alphabet:~ #1.
2353 }

    The end.
2354 \ExplSyntaxOff
2355 \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). 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 \LaTeX . 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!)

2356 < See `stix-extract.sh` for now. >

A Documenting maths support in the NFSS

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

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

$\backslash\text{DeclareSymbolFont}\{\langle name \rangle\}\langle NFSS\ decl. \rangle$

Declares a named maths font such as operators from which symbols are defined with $\backslash\text{DeclareMathSymbol}$.

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

$\backslash\text{DeclareMathAlphabet}\{\langle cmd \rangle\}\langle NFSS\ decl. \rangle$

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

$\backslash\text{DeclareSymbolFontAlphabet}\{\langle cmd \rangle\}\{\langle name \rangle\}$

Alternative (and optimisation) for $\backslash\text{DeclareMathAlphabet}$ 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 $\backslash\text{mathversion}\{\langle maths\ version \rangle\}$ command.

$\backslash\text{SetSymbolFont}\{\langle name \rangle\}\{\langle maths\ version \rangle\}\langle NFSS\ decl. \rangle$

$\backslash\text{SetMathAlphabet}\{\langle cmd \rangle\}\{\langle maths\ version \rangle\}\langle NFSS\ decl. \rangle$

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

Delimiters and radicals use wrappers around \TeX ’s $\backslash\text{delimiter}/\backslash\text{radical}$ primitives, which are re-designed in \LaTeX . The syntax used in \LaTeX ’s NFSS is therefore not so relevant here.

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

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

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

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

Accents are not included yet.

Summary For symbols, something like:

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

For characters, something like:

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

B \XETEX math font dimensions

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

| <code>\fontdimen</code> | Dimension name | Description |
|-------------------------|---|---|
| 10 | <code>SCRIPTPERCENTSCALEDOWN</code> | Percentage of scaling down for script level 1. Suggested value: 80%. |
| 11 | <code>SCRIPTSCRIPTPERCENTSCALEDOWN</code> | Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%. |
| 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. |

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

| \fontdimen | Dimension name | Description |
|------------|------------------------------------|--|
| 23 | SUPERSCRIPBTOTTOMMIN | Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height. |
| 24 | SUPERSCRIPTBASELINEDROP-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 | SUPERSCRIPBTOTTOMMAX-WITHSUBSCRIPT | The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: $\frac{1}{5}$ x-height. |
| 27 | SPACEAFTERSCRIPT | Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font. |
| 28 | UPPERLIMITGAPMIN | Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator. |
| 29 | UPPERLIMITBASELINERISEMIN | Minimum distance between baseline of upper limit and (ink) top of the base operator. |
| 30 | LOWERLIMITGAPMIN | Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator. |
| 31 | LOWERLIMITBASELINEDROP-MIN | Minimum distance between baseline of the lower limit and (ink) bottom of the base operator. |
| 32 | STACKTOPSHIFTUP | Standard shift up applied to the top element of a stack. |
| 33 | STACKTOPDISPLAYSTYLESHIFT-UP | Standard shift up applied to the top element of a stack in display style. |
| 34 | STACKBOTTOMSHIFTDOWN | Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction. |

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

| \fontdimen | Dimension name | Description |
|------------|----------------------------------|--|
| 46 | FRACTIONNUMERATORGAP-MIN | Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness |
| 47 | FRACTIONNUMDISPLAYSTYLE-GAPMIN | Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness. |
| 48 | FRACTIONRULETHICKNESS | Thickness of the fraction bar. Suggested: default rule thickness. |
| 49 | FRACTIONDENOMINATORGAP-MIN | Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness |
| 50 | FRACTIONDENOMDISPLAY-STYLEGAPMIN | Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness. |
| 51 | SKEWEDFRACTION-HORIZONTALGAP | Horizontal distance between the top and bottom elements of a skewed fraction. |
| 52 | SKEWEDFRACTIONVERTICAL-GAP | Vertical distance between the ink of the top and bottom elements of a skewed fraction. |
| 53 | OVERBARVERTICALGAP | Distance between the overbar and the (ink) top of the base. Suggested: 3×default rule thickness. |
| 54 | OVERBARRULETHICKNESS | Thickness of overbar. Suggested: default rule thickness. |
| 55 | OVERBAREXTRAASCENDER | Extra white space reserved above the overbar. Suggested: default rule thickness. |
| 56 | UNDERBARVERTICALGAP | Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness. |
| 57 | UNDERBARRULETHICKNESS | Thickness of underbar. Suggested: default rule thickness. |
| 58 | UNDERBAREXTRADESCENDER | Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness. |

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

Index

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

| Symbols | | |
|----------------------------------|-------------------------------------|--|
| \# | 1844 | |
| \\$ | 1817, 2277 | |
| \% | 1816 | |
| \& | 1815, 2278 | |
| \' | 716, 2016, 2276, 2293 | |
| * | 693, 1849, 2294 | |
| \- | 692, 2283, 2286, 2296, 2297, 2299 | |
| \. | 2295 | |
| \/ | 724, 2300 | |
| \: | 695, 2301 | |
| \< | 728 | |
| \= | 2284, 2287 | |
| \> | 729 | |
| \@DeclareMathDelimiter | 529 | |
| \@DeclareMathSizes | 520 | |
| \@backslashchar | 1175, 2348 | |
| \@begindocumenthook | 2282 | |
| \@cclvi | 535 | |
| \@cdots | 2288 | |
| \@ehd | 8 | |
| \@empty | 860, 890, 893 | |
| \@ifclassloaded | 15, 2311 | |
| \@ifpackageloaded | 2253, 2266 | |
| \@ii | 859, 860, 862, 864, 867, 872 | |
| \@input | 653, 1845 | |
| \@marker | 872, 887 | |
| \@nil | 573, 699, | |
| | 872, 881–884, 912, 1810, 1823, 1841 | |
| \@preamblecmds | 533 | |
| \@tempa | 863, 883, 885, 893 | |
| \@tempb | 886, 887, 890 | |
| \@tempswafalse | 861 | |
| \@tempwattrue | 865, 868, 888, 891, 894, 897 | |
| \@xDeclareMathDelimiter | 529 | |
| \@xxDeclareMathDelimiter | 528 | |
| \ | 1849, 1851, 2326, 2331 | |
| \{ | 1813 | |
| \} | 1814 | |
| \^ | 1809, 1827, 1836 | |
| \` | 717, 2017, 2275 | |
| \ | 1850, 2304 | |
| Numbers | | |
| \0 | 296 | |
| \9 | 296 | |
| \ | 2331 | |
| A | | |
| \A | 293 | |
| \a | 292 | |
| \addnolimits | <u>783</u> | |
| \addtoversion | 521 | |
| \AfterPackage | 19 | |
| \alloc@ | 535 | |
| \alpha@elt | 522 | |
| \alpha@list | 522 | |
| \AtBeginDocument | | |
| | 1031, 1834, 2006, 2025, 2307 | |
| \AtEndPackage | 16 | |
| \awint | 779 | |
| B | | |
| \B | 192 | |
| \backdprime | 2012 | |
| \backprime | 2011 | |
| \backtrprime | 2013 | |
| \beamer@suppressreplacementstrue | 2314 | |
| \begingroup | 1808 | |
| \beta | 2143 | |
| \bgroup | 1062 | |
| \bigcirc | 2248 | |
| \bigtriangleup | 2247 | |
| \bool_if:NF | | |
| | 694, 1273, 1284, 1299, 1309, 1319, | |
| | 1326, 1424, 1436, 1453, 1461, | |
| | 1468, 1476, 1488, 1500, 1510, | |
| | 1517, 1526, 1533, 1733, 1745, | |
| | 1757, 1769, 1780, 1787, 1796, 1803 | |

| | | | |
|------------------------------------|-------------------------------|-----------------------------------|---------------------------------------|
| \bool_if:nF | 1838 | \char_make_active:n | 571, 2018–2024, 2229 |
| \bool_if:NT | | \char_make_escape:N | 1850 |
| .. 1210, 1220, 1233, 1243, 1253, | | \char_make_math_superscript:N ... | |
| 1260, 1400, 1412, 1543, 1551, | | | 1827, 1836 |
| 1558, 1566, 1578, 1590, 1600, | | \char_make_other:N | |
| 1607, 1620, 1627, 1653, 1665, | | | 1809, 1813–1817, 1844, 1851 |
| 1677, 1689, 1700, 1707, 1716, 1723 | | \chardef | 535 |
| \bool_if:NTF | 126–133, | \chi | 2147 |
| 634, 794, 1044, 1047, 1050, 1053, | | \circ | 2249 |
| 1173, 1207, 1217, 1230, 1240, | | \cirfnint | 779 |
| 1250, 1257, 1270, 1280, 1296, | | \clist_clear:N | 302, 613, 621 |
| 1306, 1316, 1323, 1396, 1408, | | \clist_if_in:NnT | 772 |
| 1420, 1432, 1457, 1472, 1484, | | \clist_map_break: | 1154, 1158 |
| 1496, 1507, 1514, 1523, 1530, | | \clist_map_inline:nn | |
| 1547, 1562, 1574, 1586, 1597, | | | 704, 765, 821, 917, 923, |
| 1604, 1617, 1624, 1649, 1661, | | | 929, 947, 959, 977, 984, 990, 996, |
| 1673, 1685, 1697, 1704, 1713, | | | 1004, 1012, 1033, 1084, 1150, 1163 |
| 1720, 1729, 1741, 1753, 1765, | | \clist_new:N | 503 |
| 1777, 1784, 1793, 1800, 2235, 2257 | | \clist_put_right:No | 506 |
| \bool_if:nTF | 2173 | \clist_put_right:Nx | 875 |
| \bool_new:N | 26–48 | \colon | <u>2253</u> |
| \bool_set_false:N ... | 308, 310–313, | \copy | 2044 |
| 316–318, 323, 385, 387–390, 393, | | \cs | 2071, 2160 |
| 410, 421, 425, 434, 438, 448, | | \cs_generate_variant:Nn | |
| 460, 482, 487, 627, 681, 818, 1115 | | | 21–25, 67, 602, |
| \bool_set_true:N | | | 965, 966, 1029, 1030, 1186, 1198–1201 |
| | 315, 320–322, 325–328, | \cs_gset:cpx | 544, 559, 560, 576 |
| 330, 392, 394, 395, 397–400, 402, | | \cs_gset:Npn | 1810, 2078, 2111 |
| 412, 414, 423, 427, 436, 440, | | \cs_gset:Npx | 548, 554, 563, 2234 |
| 451, 457, 479, 611, 625, 661, 1109 | | \cs_if_eq_p:NN | 1838–1840 |
| \box | 2046 | \cs_if_exist:cT | 976, 1151, 1164 |
| \bullet | 2250 | \cs_if_exist:NT | 514 |
| | | \cs_if_exist:NTF | 478 |
| C | | \cs_new:Nn | 57, 58, 68, 71, |
| \C | 193 | | 569, 593, 596, 599, 603, 606, 660, |
| \c_group_begin_token | 2174 | | 691, 698, 703, 708, 719, 723, 758, |
| \c_group_end_token | 2175 | | 761, 908, 911, 967, 970, 975, 983, |
| \c_peek_true_remove_next_tl ... | 2164 | | 989, 995, 1003, 1011, 1032, 1056, |
| \c_space_token | 2176 | | 1104, 1136, 1147, 1202, 1206, |
| \cdots | 2288, 2289, 2306 | | 1216, 1229, 1239, 1249, 1269, |
| \cdp@elt | 519 | | 1279, 1295, 1305, 1315, 1333, |
| \cdp@list | 519 | | 1336, 1346, 1349, 1356, 1363, |
| \char_gmake_mathactive:N | | | 1374, 1380, 1388, 1391, 1395, |
| | <u>603</u> , 2076, 2109, 2230 | | 1407, 1419, 1431, 1443, 1446, |
| \char_gmake_mathactive:n | 572, <u>603</u> , 721 | | 1449, 1452, 1467, 1482, 1494, |
| \char_make_active:N | | | 1506, 1538, 1542, 1557, 1572, |
| | 2016, 2017, 2075, 2108 | | |

| | | | |
|---------------------------------------|--------------|-----------------------------------|---|
| \fontname | 1140 | \g_um_literal_partial_bool .. | 45, |
| \fontspec_select:nn | 67 | | 434, 440, 1257, 1323, 1514, 1523, |
| \fontspec_select:xn | 662 | | 1604, 1617, 1704, 1713, 1784, 1793 |
| \frac | 2235 | \g_um_math_alphabet_name_Greek_tl | |
| | | | 53 |
| G | | \g_um_math_alphabet_name_greek_tl | |
| \g | 222 | | 52 |
| \g_um_bfit_Greek_usv | 132 | \g_um_math_alphabet_name_Latin_tl | |
| \g_um_bfit_greek_usv | 133 | | 51 |
| \g_um_bfit_Latin_usv | 130 | \g_um_math_alphabet_name_latin_tl | |
| \g_um_bfit_latin_usv | 131 | | 50 |
| \g_um_bfliteral_bool | | \g_um_math_alphabet_name_misc_tl | 55 |
| 35, 385, 402, 1457, 1472, | | \g_um_math_alphabet_name_num_tl . | 54 |
| 1484, 1496, 1547, 1562, 1574, 1586 | | \g_um_mathalph_seq | 844, 1071 |
| \g_um_bfsfit_Greek_usv | 128 | \g_um_mathit_Greek_usv_range_tl . | 300 |
| \g_um_bfsfit_greek_usv | 129 | \g_um_mathit_greek_usv_range_tl . | 299 |
| \g_um_bfsfit_Latin_usv | 126 | \g_um_mathit_Latin_usv_range_tl . | 298 |
| \g_um_bfsfit_latin_usv | 127 | \g_um_mathit_latin_usv_range_tl . | 297 |
| \g_um_bfsfup_Greek_usv | 128 | \g_um_mathup_Greek_usv_range_tl . | 295 |
| \g_um_bfsfup_greek_usv | 129 | \g_um_mathup_greek_usv_range_tl . | 294 |
| \g_um_bfsfup_Latin_usv | 126 | \g_um_mathup_Latin_usv_range_tl . | 293 |
| \g_um_bfsfup_latin_usv | 127 | \g_um_mathup_latin_usv_range_tl . | 292 |
| \g_um_bfup_Greek_usv | 132 | \g_um_mathup_num_usv_range_tl ... | 296 |
| \g_um_bfup_greek_usv | 133 | \g_um_primekern_muskip | 1858, 1859, 1864 |
| \g_um_bfup_Latin_usv | 130 | \g_um_sfliteral_bool . | 41, 414, 1396, |
| \g_um_bfup_latin_usv | 131 | | 1408, 1420, 1432, 1649, 1661, |
| \g_um_bfupGreek_bool | | | 1673, 1685, 1729, 1741, 1753, 1765 |
| .. 38, 132, 387, 392, 397, 1488, 1578 | | \g_um_slash_delimiter_usv | |
| \g_um_bfupgreek_bool | | | 466, 469, 472, 724–726 |
| .. 39, 133, 388, 393, 398, 1500, 1590 | | \g_um_subs_prop | 2063, 2107 |
| \g_um_bfupLatin_bool | 36, 130, | \g_um_supers_prop | 2062, 2074 |
| 389, 394, 399, 1453, 1461, 1543, 1551 | | \g_um_texgreek_bool | |
| \g_um_bfuplatin_bool | 37, 131, | | 46, 448, 451, 1044, 1047, 1050, 1053 |
| 390, 395, 400, 1468, 1476, 1558, 1566 | | \g_um_upGreek_bool | |
| \g_um_default_mathalph_seq | | | 33, 128, 310, 315, 320, 325, 1233, 1299 |
| 1083, 1101, 1108 | | \g_um_upgreek_bool | |
| \g_um_fam_int | 49, 643, 644 | | 34, 129, 311, 316, 321, 326, 1243, 1309 |
| \g_um_it_h_usv | 297 | \g_um_upLatin_bool | |
| \g_um_literal_bool | | | 31, 126, 312, 317, 322, 327, 1210, 1273 |
| 30, 308, 330, 1207, 1217, | | \g_um_uplatin_bool | |
| 1230, 1240, 1270, 1280, 1296, 1306 | | | 32, 127, 313, 318, 323, 328, 1220, 1284 |
| \g_um_literal_colon_bool | | \g_um_upNabla_bool | 42, |
| 48, 457, 460, 694, 2257 | | | 423, 425, 1253, 1319, 1510, 1533, |
| \g_um_literal_Nabla_bool | 44, | | 1600, 1627, 1700, 1723, 1780, 1803 |
| 421, 427, 1250, 1316, 1507, 1530, | | \g_um_uppartial_bool | 43, |
| 1597, 1624, 1697, 1720, 1777, 1800 | | | 436, 438, 1260, 1326, 1517, 1526, |

| | |
|--|--|
| 1607, 1620, 1707, 1716, 1787, 1796 | \int_incr:N 643, 1913, 1924, 1928, 1971, 1982, 1986 |
| \g_um_upsans_bool 40, 410, 412, 1400, 1412, 1424, 1436, 1653, 1665, 1677, 1689, 1733, 1745, 1757, 1769 | \int_new:N 49, 1860 |
| \gamma 2144 | \int_set:Nn 1901, 1905, 1909, 1963, 1967 |
| \gdef 1823 | \int_use:N 644 |
| \ge 2245 | \int_zero:N 1897, 1959 |
| \geq 2245 | \intBar 779 |
| \get@cdp 525 | \intbar 779 |
| \glb@currsz 610 | \intcap 781 |
| \global 604, 607, 1819, 1829 | \intclockwise 778 |
| \group@elt 524 | \intcup 781 |
| \group@list 524 | \intexpr_compare:nT 888, 891, 894, 896, 897 |
| \group_align_safe_begin: 2166 | \intexpr_eval:n 582, 583, 586, 587, 591, 875 |
| \group_align_safe_end: 2165 | \intlarhk 780 |
| \group_begin: 570, 1835, 1848, 2015, 2065, 2207 | \intx 780 |
| \group_end: . 574, 1846, 2036, 2149, 2226 | |
| H | J |
| \H 197 | \j 225 |
| \h 223 | K |
| \hat 2308 | \kern 2039, 2045 |
| \hbox 2038 | \keys_define:nn 305, 335, 383, 407, 419, 432, 445, 454, 463, 475, 491, 504, 807, 816 |
| \ht 2041 | \keys_set:nn 303, 622 |
| I | L |
| \I 198 | \L 199 |
| \i 224 | \l_keys_choice_int 309, 338, 386, 409, 422, 435 |
| \if@tempswa 874 | \l_keys_key_tl 507 |
| \ifbeamer@suppressreplacements . 2312 | \l_peek_false_tl ... 2165, 2178, 2204 |
| \ifcase 309, 338, 386, 409, 422, 435 | \l_peek_token ... 2174–2176, 2195, 2199 |
| \ifdim 624 | \l_peek_true_aux_tl 2163 |
| \ifluatex 4 | \l_peek_true_tl 2164, 2202 |
| \ifnum 2296 | \l_um_char_num_range_clist 613, 772, 875 |
| \ifx 860, 863, 864, 867, 887, 890, 893, 2052, 2055 | \l_um_char_range_seq 612, 815, 819, 829, 859 |
| \ifxetex 4 | \l_um_font 624, 680, 1140, 1192, 2039, 2041, 2042, 2045, 2053, 2056 |
| \ignorespaces 1841 | \l_um_fontspec_feature_bool 26, 661, 681, 794 |
| \iiiint 777 | \l_um_implicit_alph_bool 29, 1109, 1115, 1173 |
| \iiint 777 | |
| \iint 777 | |
| \init@restore@version 523 | |
| \int 777 | |
| \int_add:Nn 1932, 1936, 1940, 1944, 1990, 1994 | |

| | |
|--|---|
| <code>\l_um_init_bool</code> | 28, 611, 634, 818 |
| <code>\l_um_mathalph_seq</code> | 814, 820, 823, 1106, 1108, 1118 |
| <code>\l_um_missing_alph_seq</code> | 1103, 1105, 1137, 1142, 1174 |
| <code>\l_um_mversion_tf</code> | 615, 616 |
| <code>\l_um_nolimits_tl</code> | 578, 776, 784, 787 |
| <code>\l_um_ot_math_bool</code> | 27, 625, 627 |
| <code>\l_um_primecount_int</code> | 1860, 1897, 1901, 1905, 1909, 1913, 1924, 1928, 1932, 1936, 1940, 1944, 1947, 1959, 1963, 1967, 1971, 1982, 1986, 1990, 1994, 1997 |
| <code>\l_um_radicals_tl</code> | 543, 789 |
| <code>\l_um_remap_alphabet_tl</code> | 1121–1123, 1125, 1127, 1132 |
| <code>\l_um_script_features_tl</code> | 617, 670, 809 |
| <code>\l_um_script_font_tl</code> | 619, 669, 811 |
| <code>\l_um_smallfrac_bool</code> | 47, 479, 482, 487, 2235 |
| <code>\l_um_ss_chain_tl</code> | 2079, 2112, 2156, 2201 |
| <code>\l_um_sscript_features_tl</code> | 618, 674, 810 |
| <code>\l_um_sscript_font_tl</code> | 620, 673, 812 |
| <code>\l_um_tmpa_tl</code> | 824, 835, 838, 839, 841, 842, 844, 851, 855, 1119, 1123, 1132, 1148, 1149, 1151, 1164, 1166, 1167, 1170, 1171, 1176, 1179, 2081, 2114, 2194, 2198 |
| <code>\l_um_tmpb_tl</code> | . 825, 836, 856, 1120, 1128, 1130, 1132, 1149, 1153, 1157, 2200, 2201 |
| <code>\l_um_tmpe_tl</code> | 826, 837, 852 |
| <code>\l_um_unknown_keys_clist</code> | 302, 503, 506, 621, 677 |
| <code>\lccode</code> | 1849 |
| <code>\le</code> | 2244 |
| <code>\left</code> | 790 |
| <code>\left@primitive</code> | 790, 791 |
| <code>\leq</code> | 2244 |
| <code>\let</code> | 536, 610, 790, 1819, 2262 |
| <code>\lowercase</code> | 1811, 1824 |
| <code>\lowint</code> | 781 |
| M | |
| <code>\M</code> | 200 |
| <code>\m@th</code> | 2038 |
| <code>\mathaccent</code> | 562, 1838 |
| <code>\mathalpha</code> | 767, 909, 912 |
| <code>\mathbacktick</code> | 717 |
| <code>\mathbb</code> | 1073, 1087, 1334, 1337–1344, 1347, 1350–1354 |
| <code>\mathbbbit</code> | 1073, 1088, 1357–1361 |
| <code>\mathbf</code> | 1076, 1459, 1463, 1474, 1478, 1486, 1490, 1498, 1502, 1524, 1527, 1531, 1534, 1539, 1549, 1553, 1564, 1568, 1576, 1580, 1588, 1592, 1615, 1616, 1618, 1621, 1625, 1628 |
| <code>\mathbffrac</code> | 1077, 1097, 1633, 1636 |
| <code>\mathbfrit</code> | 1076, 1095, 1456, 1471, 1483, 1495, 1521, 1522 |
| <code>\mathbfscr</code> | 1077, 1096, 1639, 1642 |
| <code>\mathbfssf</code> | 1078, 1645, 1651, 1655, 1663, 1667, 1675, 1679, 1687, 1691, 1714, 1717, 1721, 1724, 1731, 1735, 1743, 1747, 1755, 1759, 1767, 1771, 1794, 1797, 1801, 1804 |
| <code>\mathbfssfit</code> | 1078, 1099, 1738, 1750, 1762, 1774, 1791, 1792 |
| <code>\mathbfssfup</code> | 1078, 1098, 1646, 1658, 1670, 1682, 1694, 1711, 1712 |
| <code>\mathbfup</code> | 1076, 1094, 1540, 1546, 1561, 1573, 1585, 1611–1614 |
| <code>\mathbin</code> | 692, 693 |
| <code>\mathcal</code> | 2260 |
| <code>\mathchar</code> | 2275–2278 |
| <code>\mathchar@type</code> | 530, 583, 587, 591, 597, 600 |
| <code>\mathchardef</code> | 2283, 2284, 2297 |
| <code>\mathclose</code> | 551, 553, 554, 560, 1840 |
| <code>\mathcode</code> | 604, 607, 2283, 2284, 2293–2297, 2299–2301 |
| <code>\mathellipsis</code> | 2305 |
| <code>\mathfence</code> | 556, 2262 |
| <code>\mathfrak</code> | 1074, 1090, 1381–1386, 1389 |
| <code>\mathgroup</code> | 535 |
| <code>\mathinner</code> | 2288, 2305, 2306 |
| <code>\mathit</code> | 1073, 1086, 1277, 1291–1293, 1303, 1313, 1330, 1331 |
| <code>\mathop</code> | 539, 575 |
| <code>\mathopen</code> | 542, 547, 548, 559, 791, 1839 |
| <code>\mathord</code> | 709–717, 2247, 2262 |
| <code>\mathpunct</code> | 2257 |
| <code>\mathrel</code> | 695 |

100

| | |
|---|---------------------|
| Q | |
| \Q | 203 |
| \q_nil | 839, 842, 850, 854 |
| \qprime | 2010 |
| R | |
| \R | 204 |
| \r@@t | 2037 |
| \raise | 2040 |
| \relax .. 610, 624, 864, 867, 887, 2043, | |
| 2049, 2283, 2284, 2286, 2287, 2297 | |
| \removenolimits | 786 |
| \RequirePackage | 10–13, 18 |
| \restore@mathversion | 523 |
| \rho | 2145 |
| \rightarrow | 2242 |
| \rootbox | 2044 |
| \rppolint | 779 |
| S | |
| \sb | 2113 |
| \scan_stop: 583, 587, 591, 594, 597, 600, | |
| 604, 607, 1192, 2293–2295, 2299–2301 | |
| \scantokens | 2077, 2110 |
| \scpolint | 780 |
| \scriptscriptstyle | 2055 |
| \scriptstyle | 2052 |
| \seq_clear:N | 612, 819, 820, 1105 |
| \seq_if_empty:N | 1137 |
| \seq_if_empty:NTF | 1106 |
| \seq_if_in:NnTF | 22 |
| \seq_if_in:NVTF | 844 |
| \seq_map_inline:Nn | 1118, 1142 |
| \seq_map_variable:NNn | 859 |
| \seq_new:N ... 814, 815, 1071, 1083, 1103 | |
| \seq_put_right:Nn 829, 1080, 1101 | |
| \seq_put_right:Nx | 823, 1174 |
| \seq_set_eq:NN | 1108 |
| \set@@mathdelimiter | 530 |
| \set@mathaccent | 527 |
| \set@mathchar | 527 |
| \set@mathdelimiter | 529 |
| \set@mathsymbol | 528 |
| \setbox | 2038 |
| \SetMathAlphabet | 526 |
| \SetMathAlphabet@ | 526 |
| \setmathfont | 609 |
| \SetSymbolFont | 525 |
| \SetSymbolFont@ | 525 |
| \sf@size | 668, 672 |
| \smbkcircle | 2250 |
| \sp | 2080 |
| \space | 1065, 1143, 1176 |
| \sqint | 780 |
| \sqrt | 789 |
| \sqrtsign | 2038 |
| \std@equal | 2284, 2287 |
| \std@minus | 2283, 2286, 2297 |
| \sterling | 2252 |
| \string | 882, 883 |
| \strip@pt | 2049 |
| \sumint | 778 |
| T | |
| \tf@size | 667, 668 |
| \tfrac..... 478, 514, 2235, 2331 | |
| \thinmuskip | 1859 |
| \tilde | 2309 |
| \tl_if_empty:NT | 1128 |
| \tl_if_empty:NTF | 1122 |
| \tl_if_eq:nnTF | 1152, 1165 |
| \tl_if_in:NnT | 578, 838, 841 |
| \tl_if_in:NnTF | 543 |
| \tl_map_inline:nn | 518, 1072 |
| \tl_new:Nn | 292–300, 776, 789 |
| \tl_put_left:Nn | 2273 |
| \tl_put_right:cx | 766 |
| \tl_put_right:Nn | 21, 784, 2274 |
| \tl_put_right:NV | 2201 |
| \tl_remove_all_in:Nn | 787 |
| \tl_remove_in:Nn | 533, 2282 |
| \tl_rescan:nn | 1812, 1825, 2231 |
| \tl_set:cn | 69 |
| \tl_set:cx | 1041 |
| \tl_set:Nn 50–55, 466, 469, 472, | |
| 615, 617–620, 635, 835–837, 851, | |
| 852, 855, 856, 1043, 1046, 1049, | |
| 1052, 1130, 2079, 2081, 2112, 2114 | |
| \tl_set:No | 1119–1121 |
| \tl_set:Nx | 644, 1123, |
| 1125, 1127, 1148, 1149, 2163, 2165 | |
| \tl_set_eq:NN | 680, 2164 |
| \tl_use:c | 1176 |
| \to | 2242 |

| | | | |
|-------------------------------|-----------------------|-------------------------------|------------------------------|
| \token_if_eq_catcode_p:NN | 2174, 2175 | \um_config_mathbfsfup_latin:n | 1660 |
| \token_if_eq_meaning_p:NN | 2176 | \um_config_mathbfsfup_misc:n | 1696 |
| \token_to_str:N | 1123, 1125 | \um_config_mathbfsfup_num:n | 1644 |
| \triangle | 2247 | \um_config_mathbfup_Greek:n | 1572 |
| \trprime | 2009 | \um_config_mathbfup_greek:n | 1584 |
| \typeout | 1138, 1143 | \um_config_mathbfup_Latin:n | 1542 |
| U | | | |
| \Udelcode | 759, 762 | \um_config_mathbfup_latin:n | 1557 |
| \Udelimiter | 597 | \um_config_mathbfup_misc:n | 1596 |
| \um@backslash | 863, 882 | \um_config_mathbfup_num:n | 1538 |
| \um@firstchar | 862, 883 | \um_config_mathfrak_Latin:n | 1380 |
| \um@firsttof | 881–883 | \um_config_mathfrak_latin:n | 1388 |
| \um@parse@range | 872, 884 | \um_config_mathit_Greek:n | 1295 |
| \um@parse@term | 687, 699, 858, 912 | \um_config_mathit_greek:n | 1305 |
| \um@radicals | 789 | \um_config_mathit_Latin:n | 1269 |
| \um@scanactivedef | 573, 1808 | \um_config_mathit_latin:n | 1279 |
| \um@scancharlet | 1808, 1841 | \um_config_mathit_misc:n | 1315 |
| \um@zf@feature | 792, 801, 804 | \um_config_mathscr_Latin:n | 1363 |
| \um_accent:Nnn | 563, 581 | \um_config_mathscr_latin:n | 1374 |
| \um_after_pkg:nn | 16, 19, 2269 | \um_config_mathsf_Latin:n | 1419 |
| \um_backprime_double_mchar | 714, 1887 | \um_config_mathsf_Latin:n | 1431 |
| \um_backprime_single_mchar | 713, 1960, 1964, 1968 | \um_config_mathsfup_Latin:n | 1395 |
| \um_backprime_triple_mchar | 715, 1890 | \um_config_mathsfup_latin:n | 1407 |
| \um_config_mathbb_Latin:n | 1336 | \um_config_mathsfup_num:n | 1391 |
| \um_config_mathbb_latin:n | 1333 | \um_config_mathtt_Latin:n | 1446 |
| \um_config_mathbb_misc:n | 1349 | \um_config_mathtt_latin:n | 1449 |
| \um_config_mathbb_num:n | 1346 | \um_config_mathtt_num:n | 1443 |
| \um_config_mathbbbit_misc:n | 1356 | \um_config_mathup_Greek:n | 1229 |
| \um_config_mathbffrak_Latin:n | 1632 | \um_config_mathup_greek:n | 1239 |
| \um_config_mathbffrak_latin:n | 1635 | \um_config_mathup_Latin:n | 1206 |
| \um_config_mathbfit_Greek:n | 1482 | \um_config_mathup_latin:n | 1216 |
| \um_config_mathbfit_greek:n | 1494 | \um_config_mathup_misc:n | 1249 |
| \um_config_mathbfit_Latin:n | 1452 | \um_config_mathup_num:n | 1202 |
| \um_config_mathbfit_latin:n | 1467 | \um_cs_compat:n | 57–66 |
| \um_config_mathbfit_misc:n | 1506 | \um_define_active_frac:Nw | 2208–2225, 2228 |
| \um_config_mathbfscr_Latin:n | 1638 | \um_delimiter:Nnn | 548, 554, 559, 560, 581 |
| \um_config_mathbfscr_latin:n | 1641 | \um_fontdimen_to_percent:nn | 2041, 2042, 2048, 2053, 2056 |
| \um_config_mathbfsfit_Greek:n | 1752 | \um_fontspec_select_font: | 660 |
| \um_config_mathbfsfit_greek:n | 1764 | \um_fontspec_select_font:n | 623, 660 |
| \um_config_mathbfsfit_Latin:n | 1728 | \um_glyph_if_exist:cT | 1156 |
| \um_config_mathbfsfit_latin:n | 1740 | \um_glyph_if_exist:cTF | 1169 |
| \um_config_mathbfsfit_misc:n | 1776 | \um_glyph_if_exist:n | 1191 |
| \um_config_mathbfsfup_Greek:n | 1672 | \um_glyph_if_exist:nF | 1201 |
| \um_config_mathbfsfup_greek:n | 1684 | \um_glyph_if_exist:nT | 1200 |
| \um_config_mathbfsfup_Latin:n | 1648 | | |

| | |
|---|---|
| <code>\um_glyph_if_exist:nTF</code> | <code>\um_mathstyle_scale:Nnn</code> |
| . 1191 , 1871, 1874, 1877, 1887, 1890 | 2039, 2045, 2051 |
| <code>\um_glyph_if_exist_p:n</code> | <code>\um_maybe_init_alphabet:n</code> |
| 1198 | 640, 648, 1110–1112, 1129, 1153, 1157 |
| <code>\um_if_mathalph_decl:n</code> | <code>\um_nbackprimes_select:nn</code> . 1883, 1997 |
| 834 | <code>\um_nprimes:Nn</code> |
| <code>\um_if_mathalph_decl:nTF</code> | 1861, 1871, |
| 822 | 1874, 1877, 1880, 1887, 1890, 1893 |
| <code>\um_init_alphabet:n</code> .. | <code>\um_nprimes_select:nn</code> |
| 640, 1129, 1187 | 1867, 1947 |
| <code>\um_make_mathactive:nNN</code> . 709–717, 719 | <code>\um_patch_pkg:nn</code> 2265 , 2272, 2281, 2291 |
| <code>\um_map_char:nn</code> | <code>\um_peek_execute_branches:ss</code> . . |
| 641, 649, 905, 964 | 2167, 2172 |
| <code>\um_map_char:nn</code> | <code>\um_peek_execute_branches_ss_aux:</code> |
| 903 | 2179, 2192 |
| <code>\um_map_char_noparse:nn</code> . 641, 908, 913 | <code>\um_prepare_alph:n</code> |
| <code>\um_map_char_parse:nn</code> | 1056, 1081 |
| 649, 911 | <code>\um_prime_double_mchar</code> |
| <code>\um_map_char_single:cc</code> | 710, 1871 |
| 932, 934, 936, 938, 940, 942, 950, 960 | <code>\um_prime_quad_mchar</code> |
| <code>\um_map_char_single:nn</code> | 712, 1877 |
| 964, 965 | <code>\um_prime_single_mchar</code> |
| <code>\um_map_chars_Greek:nn</code> . 946, 1231, | 709, 1898, 1902, 1906, 1910 |
| 1234, 1297, 1300, 1485, 1489, | <code>\um_prime_triple_mchar</code> |
| 1575, 1579, 1674, 1678, 1754, 1758 | 711, 1874 |
| <code>\um_map_chars_greek:nn</code> . 928, 1241, | <code>\um_process_symbol_noparse:nnnn</code> . |
| 1244, 1307, 1310, 1497, 1501, | 637, 683 |
| 1587, 1591, 1686, 1690, 1766, 1770 | <code>\um_process_symbol_parse:nnnn</code> 645, 683 |
| <code>\um_map_chars_Latin:nn</code> | <code>\um_radical:nn</code> |
| 916, 1208, 1211, | 544, 581 |
| 1271, 1274, 1397, 1401, 1421, | <code>\um_remap_symbol:nnn</code> |
| 1425, 1454, 1458, 1462, 1544, | 639, 647, 692, 693, 695 |
| 1548, 1552, 1650, 1654, 1730, 1734 | <code>\um_remap_symbol_noparse:nnn</code> 639, 691 |
| <code>\um_map_chars_latin:nn</code> | <code>\um_remap_symbol_parse:nnn</code> |
| 922, 1218, 1221, | 647, 691 , 698 |
| 1281, 1285, 1409, 1413, 1433, | <code>\um_remap_symbols</code> |
| 1437, 1469, 1473, 1477, 1559, | 655, 691 |
| 1563, 1567, 1662, 1666, 1742, 1746 | <code>\um_resolve_greek</code> |
| <code>\um_map_chars_numbers:nn</code> .. | 1031 |
| 954, 1203 | <code>\um_scan_backdprime</code> |
| <code>\um_map_chars_range:ncc</code> | 1962, 1981, 2012, 2033 |
| 918, 924, 930, 948, 955 | <code>\um_scan_backprime</code> |
| <code>\um_map_chars_range:nnn</code> | 1958, 1975, 2011, 2031, 2032 |
| 903, 966 | <code>\um_scan_backtrprime</code> |
| <code>\um_map_single:nnn</code> | 1966, 1989, 2013, 2034 |
| ... 958, 1222–1224, 1251, 1254, | <code>\um_scan_dprime</code> . 1900, 1923, 2008, 2028 |
| 1258, 1261, 1282, 1286–1288, | <code>\um_scan_prime</code> |
| 1317, 1320, 1324, 1327, 1508, | 1896, 1917, 2007, 2026, 2027 |
| 1511, 1515, 1518, 1598, 1601, | <code>\um_scan_qprime</code> . 1908, 1939, 2010, 2030 |
| 1605, 1608, 1698, 1701, 1705, | <code>\um_scan_sscript</code> . 2082, 2115, 2152, 2154 |
| 1708, 1778, 1781, 1785, 1788 | <code>\um_scan_sscript:TF</code> |
| <code>\um_mathmap:Nnn</code> 638, 646, 968, 972, 1116 | 2153, 2162 |
| <code>\um_mathmap_noparse:Nnn</code> | <code>\um_scan_trprime</code> . 1904, 1931, 2009, 2029 |
| 638, 764 , 773, 1116 | <code>\um_scanbackprime_collect:N</code> |
| <code>\um_mathmap_parse:Nnn</code> | ... 1960, 1964, 1968, 1970, 1973, |
| 646, 771 | 1976, 1979, 1983, 1987, 1991, 1995 |

| | |
|---|---|
| <code>\um_scanprime_collect:N</code> | 1618, 1621, 1625, 1628, 1711, |
| 1898, 1902, 1906, | 1712, 1714, 1717, 1721, 1724, |
| 1910, 1912, 1915, 1918, 1921, | 1791, 1792, 1794, 1797, 1801, 1804 |
| 1925, 1929, 1933, 1937, 1941, 1945 | <code>\um_set_mathchar:cNnn</code> 575, <u>581</u> |
| <code>\um_set_big_operator:nnn</code> . . . 540, <u>569</u> | <code>\um_set_mathchar:NNnn</code> <u>581</u> , 720 |
| <code>\um_set_delcode:n</code> | <code>\um_set_mathcode:nnn</code> |
| . . . 546, 552, 558, 727, 730–756, <u>758</u> | 547, 553, 557, 566, <u>581</u> |
| <code>\um_set_delcode:nn</code> 724–726, 728, 729, <u>758</u> | <code>\um_set_mathcode:nnnn</code> <u>581</u> , 705, 767, 909 |
| <code>\um_set_mathalph_range:nNcc</code> | <code>\um_set_mathsymbol:nNnn</code> <u>537</u> , 684 |
| 985, 991, 997, 1005, 1013 | <code>\um_setup_active_frac:</code> 2206, 2240 |
| <code>\um_set_mathalph_range:Nnn</code> <u>970</u> | <code>\um_setup_active_subscript:nn</code> . . . |
| <code>\um_set_mathalph_range:nNnn</code> 970, 1030 | 2106, 2120–2147 |
| <code>\um_set_mathalphabet_char:Ncc</code> . . . | <code>\um_setup_active_superscript:nn</code> . |
| 978, 999, 1007, | 2073, 2087–2103 |
| 1015, 1017, 1019, 1021, 1023, 1025 | <code>\um_setup_alphabets:</code> 658, <u>1103</u> |
| <code>\um_set_mathalphabet_char:Nnn</code> . . . | <code>\um_setup_delcodes:</code> 657, <u>723</u> |
| 967, 1029 | <code>\um_setup_math_alphabet:Nnn</code> . . . <u>1147</u> |
| <code>\um_set_mathalphabet_Greek:Nnn</code> . . | <code>\um_setup_math_alphabet:VVV</code> . . . 1132 |
| 1003, 1237, 1303, 1483, | <code>\um_setup_mathactives:</code> 656, <u>708</u> |
| 1486, 1490, 1573, 1576, 1580, | <code>\um_setup_mathup:</code> 2241, 2273 |
| 1675, 1679, 1682, 1755, 1759, 1762 | <code>\um_split_arrow:w</code> 839, 850 |
| <code>\um_set_mathalphabet_greek:Nnn</code> . . | <code>\um_split_slash:w</code> 842, 854 |
| 1011, 1247, 1313, 1495, | <code>\um_sub_or_super:n</code> . . . 2080, 2113, 2156 |
| 1498, 1502, 1585, 1588, 1592, | <code>\um_symfont_tl</code> 635, 644, |
| 1687, 1691, 1694, 1767, 1771, 1774 | 651, 684, 705, 720, 759, 762, 767, 909 |
| <code>\um_set_mathalphabet_Latin:Nnn</code> . . | <code>\um_to_usv:nn</code> 69, 71, 918, |
| 989, 1214, 1277, 1337, 1364, 1381, | 919, 924, 925, 930–943, 948–951, |
| 1398, 1402, 1405, 1422, 1426, | 955, 956, 960, 961, 976, 978, 979, |
| 1429, 1447, 1456, 1459, 1463, | 985, 986, 991, 992, 997–1000, |
| 1546, 1549, 1553, 1633, 1639, | 1005–1008, 1013–1026, 1156, 1169 |
| 1651, 1655, 1658, 1731, 1735, 1738 | <code>\um_trace:n</code> 1107, 1114, 2318 |
| <code>\um_set_mathalphabet_latin:Nnn</code> . . | <code>\um_trace:nx</code> 1166, 1170, 1188, 2319 |
| 995, 1227, 1291, 1334, 1375, 1389, | <code>\um_warn_missing_alphabets:</code> 1134, 1136 |
| 1410, 1414, 1417, 1434, 1438, | <code>\um_warning:n</code> 481, 797, 2313, 2317 |
| 1441, 1450, 1471, 1474, 1478, | <code>\Umathaccent</code> 600 |
| 1561, 1564, 1568, 1636, 1642, | <code>\Umathchardef</code> 590 |
| 1663, 1667, 1670, 1743, 1747, 1750 | <code>\Umathcharnum</code> 2286, 2287 |
| <code>\um_set_mathalphabet_numbers:Nnn</code> | <code>\Umathcode</code> 582, 586 |
| 983, 1204, 1347, 1392, | <code>\Umathcodenum</code> 2286, 2287 |
| 1393, 1444, 1539, 1540, 1645, 1646 | <code>\unicodecdots</code> 2306 |
| <code>\um_set_mathalphabet_pos:Nnnn</code> 975, | <code>\unicodeellipsis</code> 2305 |
| 1264–1267, 1292, 1293, 1330, | <code>\UnicodeMathSymbol</code> . 637, 645, 654, 1837 |
| 1331, 1338–1344, 1350–1354, | <code>\unimathsetup</code> 301, |
| 1357–1361, 1365–1372, 1376–1378, | 339, 347, 355, 363, 371, 511–513, 515 |
| 1382–1386, 1521, 1522, 1524, | <code>\unless</code> 860 |
| 1527, 1531, 1534, 1611–1616, | <code>\updefault</code> 652 |

| | | | |
|---|------------|---------------------------|------------------------|
| \upDigamma | 2264 | \version@elt | 522 |
| \updigamma | 2263 | \version@list | 522 |
| \upint | 781 | \Vert | 2304 |
| \Uradical | 594 | \vysmwhtcircle | 2249 |
| \Url@FormatString | 2273 | | |
| \UrlSpecials | 2274 | W | |
| \use:c ... 484, 488, 1058, 1167, 1171, 1179 | | \widehat | 2308 |
| \use_i:nnn | 1119 | \widetilde | 2309 |
| \use_ii:nnn | 1120 | | |
| \use_iii:nnn | 1121 | X | |
| \use_none:n | 648 | \xetex_or luatex:nn | 56 |
| \use_none:nnnn | 654, 1149 | | |
| \use_none:nnnnn | 1127 | Y | |
| \usepackage | 3 | \yen | 2251 |
| \usv_set:nnn | 68, 72–291 | | |
| | | Z | |
| V | | \Z | 205, 293 |
| \varepsilon | 1049 | \z | 292 |
| \varointclockwise | 778 | \z@ | 2038, 2041, 2042, 2046 |
| \varphi | 1052 | \zf@basefont | 680 |
| \vec | 2243 | \zf@family | 652 |
| | | \zf@update@ff | 802, 805 |