Experimental Unicode mathematical typesetting: The unicode-math package

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

2013/03/16 v0.7d

Abstract

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

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

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

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

Part I
User documentation

Table of Contents

1 Introduction

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

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

2 Acknowledgements

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

Unicode math glyphs and their LaTeX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions; Philipp Stephani for extending the package to support LuaTeX. 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, Khaled Hosny, and Mariusz Wodzicki have been fantastic beta testers.

3 Getting started

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

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

Three OpenType maths fonts are included by default in TeX Live 2011: Latin Modern Math, Asana Math, and XITS Math. These can be loaded directly with their filename with both XeLaTeX and LuaLaTeX; resp.,

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

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

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

3.1 Package options

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

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

Note, however, that some package options 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:

Table 1: Package options.

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

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

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

3.2 Known issues

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

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

4 Unicode maths font setup

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

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

implements this for every every symbol and alphabetic variant. That means `x` to x, `\xi` to ξ, `\leq` to $\leq$, etc., `\mathscr{H}` to $\mathscr{H}$ 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.

Table 2: Maths font options.

Option	Description	See…
`range`	Style of letters	section §4.1
`script-font`	Font to use for sub- and super-scripts	section §4.2
`script-features`	Font features for sub- and super-scripts	section §4.2
`sscript-font`	Font to use for nested sub- and super-scripts	section §4.2
`sscript-features`	Font features for nested sub- and super-scripts	section §4.2

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

4.1 Using multiple fonts

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

`\setmathfont[range=`⟨*unicode range*⟩`,`⟨*font features*⟩`]{`⟨*font name*⟩`}`

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

4.1.1 Control over maths alphabets

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

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

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

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

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

4.2 Script and scriptscript fonts/features

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

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

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

4.3 Maths 'versions'

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

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

`\setmathfont[version=`⟨*version name*⟩`,`⟨*font features*⟩`]{`⟨*font name*⟩`}`

and to switch between maths versions mid-document use the standard LaTeX command `\mathversion{`⟨*version name*⟩`}`.

5 Maths input

XeTeX'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 have been known to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

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

The philosophy behind the interface to the mathematical alphabet symbols lies in LaTeX's attempt of separating content and formatting. Because input source

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

	Example	
Package option	Latin	Greek
`math-style=ISO`	(a,z,B,X)	$(\alpha,\beta,\mathit{\Gamma},\mathit{\Xi})$
`math-style=TeX`	(a,z,B,X)	$(\alpha,\beta,\Gamma,\Xi)$
`math-style=french`	$(a,z,\mathrm{B},\mathrm{X})$	$(\upalpha,\upbeta,\Gamma,\Xi)$
`math-style=upright`	$(\mathrm{a},\mathrm{z},\mathrm{B},\mathrm{X})$	$(\upalpha,\upbeta,\Gamma,\Xi)$

text may come from a variety of places, the upright and 'mathematical' italic Latin and Greek alphabets are *unified* from the point of view of having a specified meaning in the source text. That is, to get a mathematical 'x', either the ascii ('keyboard') letter `x` may be typed, or the actual Unicode character may be used. Similarly for Greek letters. The upright or italic forms are then chosen based on the `math-style` package option.

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

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

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

5.2 Bold style

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

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

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

To match the package options for non-bold characters, with option `bold-style=upright` all bold characters are upright, and `bold-style=literal` does not

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

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

change the upright/italic shape of the letter.

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

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

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

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical alphabet characters. These may be explicitly selected with the `\mathsfup`, `\mathsfit`, `\mathbfsfup`, 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 LaTeX's `\mathsf` is *upright* sans serif.

Therefore I reluctantly add the package options `[sans-style=upright]` and `[sans-style=italic]` to control the behaviour of `\mathsf`. The `upright` style sets up the command to use upright sans serif, including Greek; the `italic` style switches to using italic in both Latin and Greek alphabets. In other words, this option simply changes the meaning of `\mathsf` to either `\mathsfup` or `\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 either `\mathbfsfup` or `\mathbfsfit` based on `[sans-style=upright]` or `[sans-style=italic]`, respectively. And `[sans-style=literal]` causes `\mathbfsf` to retain the same italic or upright shape as the input, and turns it bold sans serif.

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

Font				Alphabet		
Style	Shape	Series	Switch	Latin	Greek	Numerals
Serif	Upright	Normal	`\mathup`	●	●	●
		Bold	`\mathbfup`	●	●	●
	Italic	Normal	`\mathit`	●	●	🔵
		Bold	`\mathbfit`	●	●	🔵
Sans serif	Upright	Normal	`\mathsfup`	●		●
	Italic	Normal	`\mathsfit`	●		🔵
	Upright	Bold	`\mathbfsfup`	●	●	●
	Italic	Bold	`\mathbfsfit`	●	●	🔵
Typewriter	Upright	Normal	`\mathtt`	●		●
Double-struck	Upright	Normal	`\mathbb`	●		●
	Italic	Normal	`\mathbbit`	●		
Script	Upright	Normal	`\mathscr`	●		
		Bold	`\matbfscr`	●		
Fraktur	Upright	Normal	`\mathfrak`	●		
		Bold	`\mathbffrac`	●		

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

5.4 All (the rest) of the mathematical alphabets

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

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,𝔸ℤ}, numerals 𝟘–𝟡, summation symbol ⅀, and four Greek letters only: {ℽℼℾℿ}.

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

Table 6: The various forms of nabla.

Description		Glyph
Upright	Serif	∇
	Bold serif	$\mathbf{\nabla}$
	Bold sans	$\boldsymbol{\nabla}$
Italic	Serif	$\mathit{\nabla}$
	Bold serif	$\boldsymbol{\mathit{\nabla}}$
	Bold sans	$\boldsymbol{\mathit{\nabla}}$

There are also five Latin *italic* double-struck letters: ⅅⅆⅇⅈⅉ. These can be accessed (if not with their literal characters or control sequences) with the `\mathbbit` alphabet switch, but note that only those five letters will give the expected output.

5.4.2 Caligraphic vs. Script variants

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

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

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

An example is shown below.

The Script style (`\mathscr`) in XITS Math is: $\mathscr{ABCXYZ}$

The Caligraphic style (`\mathcal`) in XITS Math is: $\mathcal{ABCXYZ}$

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, and ISO standards agree with this convention. The package options `nabla=upright` and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. This is then inherited through `\mathbf`; `\mathit` and `\mathup` can be used to force one way or the other.

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

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	𝟃

5.5.2 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.

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

TeX defines `\epsilon` to look like ϵ and `\varepsilon` to look like ε. By constrast, 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 an 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`.

[1] A 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.

$$\boxed{\mathrm{A}\,^{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -\ =\ (\ )\ \mathrm{i}\ \mathrm{n}}\,\mathrm{Z}}$$

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.

5.5.4 Primes

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

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

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

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

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

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

5.5.5 Unicode subscripts and superscripts

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

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

5.5.6 Colon

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

$$\boxed{\mathrm{A}_{0\,1\,2\,3\,4\,5\,6\,7\,8\,9\,+\,-\,=\,(\,)\,a\,e\,i\,o\,r\,u\,v\,x\,\beta\,\gamma\,\rho\,\varphi\,\chi}\mathrm{Z}}$$

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

Table 8: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	`\slash`
U+2044	FRACTION SLASH	⁄	`\fracslash`
U+2215	DIVISION SLASH	∕	`\divslash`
U+29F8	BIG SOLIDUS	⧸	`\xsol`
U+005C	REVERSE SOLIDUS	\	`\backslash`
U+2216	SET MINUS	∖	`\smallsetminus`
U+29F5	REVERSE SOLIDUS OPERATOR	⧵	`\setminus`
U+29F9	BIG REVERSE SOLIDUS	⧹	`\xbsol`

with punctuation spacing: '$a\colon b$'.

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

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

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

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 $\sum$).

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: $A \smallsetminus B$.[2] The LaTeX command name `\smallsetminus` is used for backwards compatibility.

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

Finally, U+29F9 big reverse solidus is a 'big operator' (like $\sum$).

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

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

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

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

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

However, we assume that there is only *one* stretchy slash in the font; this is assumed by default to be U+002F solidus. Writing `\left/` or `\left\slash` or `\left`fracslash 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 Growing and non-growing accents

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

[2] §4.4.5.11 http://www.w3.org/TR/MathML3/

[3] This 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	`\cdotp`	·			
U+22C5	`\cdot`	⋅			
U+2219	`\vysmblkcircle`	∙	∘	`\vysmwhtcircle`	U+2218
U+2022	`\smblkcircle`	•	◦	`\smwhtcircle`	U+25E6
U+2981	`\mdsmblkcircle`	⦁	⚬	`\mdsmwhtcircle`	U+26AC
U+26AB	`\mdblkcircle`	⚫	⚪	`\mdwhtcircle`	U+26AA
U+25CF	`\mdlgblkcircle`	●	○	`\mdlgwhtcircle`	U+25CB
U+2B24	`\lgblkcircle`	⬤	◯	`\lgwhtcircle`	U+25EF

Table 9: Filled and hollow Unicode circles.

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

5.5.9 Pre-drawn fraction characters

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

¼ ½ ¾ ↉ ⅐ ⅑ ⅒ ⅓ ⅔ ⅕ ⅖ ⅗ ⅘ ⅙ ⅚ ⅛ ⅜ ⅝ ⅞

For example, instead of writing '`\tfrac12 x`', you may consider it more readable to have '½x' in the source instead.

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

5.5.10 Circles

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

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

5.5.11 Triangles

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

Slot	Command	Glyph	Class
`U+25B5`	`\vartriangle`	▵	binary
`U+25B3`	`\bigtriangleup`	△	binary
`U+25B3`	`\triangle`	△	ordinary
`U+2206`	`\increment`	∆	ordinary
`U+0394`	`\mathup\Delta`	Δ	ordinary

Table 10: Different upwards pointing triangles.

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

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

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

6 Advanced

6.1 Warning messages

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

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

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

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

6.2 Programmer's interface

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

Part II

Package implementation

Table of Contents

7 Header code

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

```
⟨*load⟩
\luatex_if_engine:T { \RequirePackage{unicode-math-luatex} \endinput }
\xetex_if_engine:T  { \RequirePackage{unicode-math-xetex}  \endinput }
⟨/load⟩
```

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

```
⟨*preamble&!XE&!LU⟩

\usepackage{ifxetex,ifluatex}
\ifxetex
  \ifdim\number\XeTeXversion\XeTeXrevision in<0.9998in%
    \PackageError{unicode-math}{%
      Cannot run with this version of XeTeX!\MessageBreak
      You need XeTeX 0.9998 or newer.%
    }\@ehd
  \fi
\else\ifluatex
  \ifnum\luatexversion<64%
    \PackageError{unicode-math}{%
      Cannot run with this version of LuaTeX!\MessageBreak
      You need LuaTeX 0.64 or newer.%
    }\@ehd
  \fi
\else
  \PackageError{unicode-math}{%
    Cannot be run with pdfLaTeX!\MessageBreak
    Use XeLaTeX or LuaLaTeX instead.%
  }\@ehd
\fi\fi
```

Packages

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

Start using LaTeX3 — finally!

```
\ExplSyntaxOn
```

Extra expl3 variants

```
\cs_generate_variant:Nn \tl_put_right:Nn {cx}
\cs_generate_variant:Nn \seq_if_in:NnTF {NV}
\cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
\cs_generate_variant:Nn \prop_get:NnN {cxN}
\cs_generate_variant:Nn \prop_if_in:NnTF {cx}
```

Extra expansion command:

```
\cs_set:Npn \exp_args:NNcc #1#2#3#4 {
  \exp_after:wN #1 \exp_after:wN #2
    \cs:w #3 \exp_after:wN \cs_end:
    \cs:w #4 \cs_end:
}
```

Conditionals

```
\bool_new:N \l_um_ot_math_bool
\bool_new:N \l_um_init_bool
\bool_new:N \l_um_implicit_alph_bool
\bool_new:N \g_um_mainfont_already_set_bool
```

For `math-style`:

```
\bool_new:N \g_um_literal_bool
\bool_new:N \g_um_upLatin_bool
\bool_new:N \g_um_uplatin_bool
\bool_new:N \g_um_upGreek_bool
\bool_new:N \g_um_upgreek_bool
```

For `bold-style`:

```
\bool_new:N \g_um_bfliteral_bool
\bool_new:N \g_um_bfupLatin_bool
\bool_new:N \g_um_bfuplatin_bool
\bool_new:N \g_um_bfupGreek_bool
\bool_new:N \g_um_bfupgreek_bool
```

For `sans-style`:

```
\bool_new:N \g_um_upsans_bool
\bool_new:N \g_um_sfliteral_bool
```

For assorted package options:

```
\bool_new:N \g_um_upNabla_bool
\bool_new:N \g_um_uppartial_bool
\bool_new:N \g_um_literal_Nabla_bool
\bool_new:N \g_um_literal_partial_bool
\bool_new:N \g_um_texgreek_bool
\bool_set_true:N \g_um_texgreek_bool
\bool_new:N \l_um_smallfrac_bool
\bool_new:N \g_um_literal_colon_bool
```

Variables

```
\int_new:N \g_um_fam_int
```

```
\tl_const:Nn \c_um_math_alphabet_name_latin_tl {Latin,~lowercase}
\tl_const:Nn \c_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
\tl_const:Nn \c_um_math_alphabet_name_greek_tl {Greek,~lowercase}
\tl_const:Nn \c_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
\tl_const:Nn \c_um_math_alphabet_name_num_tl   {Numerals}
\tl_const:Nn \c_um_math_alphabet_name_misc_tl  {Misc.}
```

7.1 Extras

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

```
\prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F}
 {
  \etex_iffontchar:D \l_um_font #1 \scan_stop:
    \prg_return_true:
  \else:
    \prg_return_false:
  \fi:
 }
\cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
\cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
\cs_generate_variant:Nn \um_glyph_if_exist:nT  {c}
\cs_generate_variant:Nn \um_glyph_if_exist:nF  {c}
```

7.2 Function variants

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

7.3 Package options

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

```
\DeclareDocumentCommand \unimathsetup {m}
 {
  \keys_set:nn  {unicode-math} {#1}
 }
```

```
\cs_new:Nn \um_tl_map_dbl:nN
  {
    \__um_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
  }
\cs_new:Nn \__um_tl_map_dbl:Nnn
  {
    \quark_if_recursion_tail_stop:n {#2}
    \quark_if_recursion_tail_stop:n {#3}
    #1 {#2} {#3}
    \__um_tl_map_dbl:Nnn #1
 }
\cs_new:Nn \um_keys_choices:nn
 {
  \cs_set:Npn \um_keys_choices_fn:nn { \um_keys_choices_aux:nnn {#1} }
  \use:x
   {
    \exp_not:N \keys_define:nn {unicode-math}
     {
      #1 .choice: ,
      \um_tl_map_dbl:nN {#2} \um_keys_choices_fn:nn
     }
   }
 }
\cs_new:Nn \um_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
```

math-style

```
\um_keys_choices:nn {normal-style}
 {
   {ISO} {
       \bool_set_false:N \g_um_literal_bool
       \bool_set_false:N \g_um_upGreek_bool
       \bool_set_false:N \g_um_upgreek_bool
       \bool_set_false:N \g_um_upLatin_bool
       \bool_set_false:N \g_um_uplatin_bool }
   {TeX} {
       \bool_set_false:N \g_um_literal_bool
       \bool_set_true:N  \g_um_upGreek_bool
       \bool_set_false:N \g_um_upgreek_bool
       \bool_set_false:N \g_um_upLatin_bool
       \bool_set_false:N \g_um_uplatin_bool }
   {french} {
       \bool_set_false:N \g_um_literal_bool
       \bool_set_true:N  \g_um_upGreek_bool
       \bool_set_true:N  \g_um_upgreek_bool
       \bool_set_true:N  \g_um_upLatin_bool
       \bool_set_false:N \g_um_uplatin_bool }
   {upright} {
       \bool_set_false:N \g_um_literal_bool
       \bool_set_true:N  \g_um_upGreek_bool
       \bool_set_true:N  \g_um_upgreek_bool
       \bool_set_true:N  \g_um_upLatin_bool
       \bool_set_true:N  \g_um_uplatin_bool }
   {literal} {
       \bool_set_true:N  \g_um_literal_bool }
 }
```

```
\um_keys_choices:nn {math-style}
 {
  {ISO} {
      \unimathsetup { nabla=upright, partial=italic,
       normal-style=ISO, bold-style=ISO, sans-style=italic } }
  {TeX} {
      \unimathsetup { nabla=upright, partial=italic,
        normal-style=TeX, bold-style=TeX, sans-style=upright } }
  {french} {
      \unimathsetup { nabla=upright, partial=upright,
        normal-style=french, bold-style=upright, sans-style=upright } }
  {upright} {
      \unimathsetup { nabla=upright, partial=upright,
        normal-style=upright, bold-style=upright, sans-style=upright } }
  {literal} {
      \unimathsetup { colon=literal, nabla=literal, partial=literal,
        normal-style=literal, bold-style=literal, sans-style=literal } }
 }
```

bold-style

```
\um_keys_choices:nn {bold-style}
 {
  {ISO} {
      \bool_set_false:N \g_um_bfliteral_bool
      \bool_set_false:N \g_um_bfupGreek_bool
      \bool_set_false:N \g_um_bfupgreek_bool
      \bool_set_false:N \g_um_bfupLatin_bool
      \bool_set_false:N \g_um_bfuplatin_bool }
  {TeX} {
      \bool_set_false:N \g_um_bfliteral_bool
      \bool_set_true:N \g_um_bfupGreek_bool
      \bool_set_false:N \g_um_bfupgreek_bool
      \bool_set_true:N \g_um_bfupLatin_bool
      \bool_set_true:N \g_um_bfuplatin_bool }
  {upright} {
      \bool_set_false:N \g_um_bfliteral_bool
      \bool_set_true:N \g_um_bfupGreek_bool
      \bool_set_true:N \g_um_bfupgreek_bool
      \bool_set_true:N \g_um_bfupLatin_bool
      \bool_set_true:N \g_um_bfuplatin_bool }
  {literal} {
      \bool_set_true:N \g_um_bfliteral_bool }
 }
```

sans-style

```
\um_keys_choices:nn {sans-style}
 {
  {italic}  { \bool_set_false:N \g_um_upsans_bool    }
  {upright} { \bool_set_true:N  \g_um_upsans_bool    }
  {literal} { \bool_set_true:N  \g_um_sfliteral_bool }
 }
```

Nabla and partial

```
\um_keys_choices:nn {nabla}
 {
  {upright} { \bool_set_false:N \g_um_literal_Nabla_bool
              \bool_set_true:N  \g_um_upNabla_bool    }
  {italic}  { \bool_set_false:N \g_um_literal_Nabla_bool
              \bool_set_false:N \g_um_upNabla_bool    }
  {literal} { \bool_set_true:N  \g_um_literal_Nabla_bool }
 }
```

```
\um_keys_choices:nn {partial}
 {
  {upright} { \bool_set_false:N \g_um_literal_partial_bool
              \bool_set_true:N  \g_um_uppartial_bool    }
  {italic}  { \bool_set_false:N \g_um_literal_partial_bool
              \bool_set_false:N \g_um_uppartial_bool    }
  {literal} { \bool_set_true:N  \g_um_literal_partial_bool }
 }
```

Epsilon and phi shapes

```
\um_keys_choices:nn {vargreek-shape}
 {
  {unicode} {\bool_set_false:N \g_um_texgreek_bool}
  {TeX}     {\bool_set_true:N  \g_um_texgreek_bool}
 }
```

Colon style

```
\um_keys_choices:nn {colon}
 {
  {literal} {\bool_set_true:N  \g_um_literal_colon_bool}
  {TeX}     {\bool_set_false:N \g_um_literal_colon_bool}
 }
```

Slash delimiter style

```
\um_keys_choices:nn {slash-delimiter}
 {
  {ascii} {\tl_set:Nn \g_um_slash_delimiter_usv {"002F}}
  {frac}  {\tl_set:Nn \g_um_slash_delimiter_usv {"2044}}
  {div}   {\tl_set:Nn \g_um_slash_delimiter_usv {"2215}}
 }
```

Active fraction style

```
\um_keys_choices:nn {active-frac}
 {
   {small}
   {
    \cs_if_exist:NTF \tfrac
     {
      \bool_set_true:N \l_um_smallfrac_bool
     }{
      \um_warning:n {no-tfrac}
      \bool_set_false:N \l_um_smallfrac_bool
     }
    \use:c {um_setup_active_frac:}
   }

   {normalsize}
   {
    \bool_set_false:N \l_um_smallfrac_bool
    \use:c {um_setup_active_frac:}
   }
 }
```

Debug/tracing

```
\keys_define:nn {unicode-math}
  {
    warnings-off .code:n =
      {
        \clist_map_inline:nn {#1}
          { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
      }
  }
\um_keys_choices:nn {trace}
 {
  {on}    {} % default
  {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
  {off}   { \msg_redirect_module:nnn { unicode-math } { log } { none } }
 }
\unimathsetup {math-style=TeX}
\unimathsetup {slash-delimiter=ascii}
\unimathsetup {trace=off}
\cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
\ProcessKeysOptions {unicode-math}
```

8 LuaLaTeX module

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

```
⟨/preamble&!XE&!LU⟩
⟨*lua⟩
```

```
local err, warn, info, log = luatexbase.provides_module({
  name        = "unicode-math",
  date        = "2012/04/23",
  version     = 0.1,
  description = "Unicode math typesetting for LuaLaTeX",
  author      = "Khaled Hosny, Will Robertson, Philipp Stephani",
  licence     = "LPPL v1.3+"
})
```

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

```
local function set_sscale_dimens(fontdata)
  local mc = fontdata.MathConstants
  if mc then
    fontdata.parameters[10] = mc.ScriptPercentScaleDown or 70
    fontdata.parameters[11] = mc.ScriptScriptPercentScaleDown or 50
  end
end
luatexbase.add_to_callback("luaotfload.patch_font", set_sscale_dimens, "uni-
code_math.set_sscale_dimens")
```

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

```
local function patch_cambria_domh(fontdata)
  local mc = fontdata.MathConstants
  local mh = 2800 / fontdata.units * fontdata.size
  if fontdata.psname == "CambriaMath" and mc then
    if mc.DisplayOperatorMinHeight < mh then
      mc.DisplayOperatorMinHeight = mh
    end
  end
end
luatexbase.add_to_callback("luaotfload.patch_font", patch_cambria_domh, "cam-
bria.domh")
⟨/lua⟩
```

(Error messages and warning definitions go here from the msg chunk defined in section §17 on page 102.)

9 Bifurcation

And here the split begins. Most of the code is still shared, but code for LuaTEX uses the 'LU' prefix and code for XƎTEX uses 'XE'.

```
⟨*package&(XE|LU)⟩
\ExplSyntaxOn
```

9.1 Engine differences

XeTeX before version 0.9999 did not support \U prefix for extended math primitives, and while LuaTeX had it from the start, prior 0.75.0 the LaTeX format did not provide them without the \luatex prefix.

```
⟨XE⟩\ifdim\number\XeTeXversion\XeTeXrevision in<0.9999in
⟨LU⟩\ifnum\luatexversion<75%
  \cs_new:Nn \um_cs_compat:n
⟨XE⟩    { \cs_set_eq:cc {U#1} {XeTeX#1}    }
⟨LU⟩    { \cs_set_eq:cc {U#1} {luatexU#1} }
  \um_cs_compat:n {mathcode}
  \um_cs_compat:n {delcode}
  \um_cs_compat:n {mathcodenum}
  \um_cs_compat:n {mathcharnum}
  \um_cs_compat:n {mathchardef}
  \um_cs_compat:n {radical}
  \um_cs_compat:n {mathaccent}
  \um_cs_compat:n {delimiter}
\fi
```

```
⟨*LU⟩
\RequirePackage { lualatex-math } [ 2011/08/07 ]
\RequirePackage { luatexbase }
\RequirePackage { luaotfload } [ 2010/11/26 ]
\RequireLuaModule { unicode-math } [ 2012/04/23 ]
⟨/LU⟩
```

9.2 Alphabet Unicode positions

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

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

```
\cs_new:Nn \usv_set:nnn {
  \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}
}
\cs_new:Nn \um_to_usv:nn { g_um_#1_#2_usv }
```

Alphabets

```
\usv_set:nnn {up}{num}{48}
\usv_set:nnn {up}{Latin}{65}
\usv_set:nnn {up}{latin}{97}
\usv_set:nnn {up}{Greek}{"391}
\usv_set:nnn {up}{greek}{"3B1}
\usv_set:nnn {it}{Latin}{"1D434}
\usv_set:nnn {it}{latin}{"1D44E}
\usv_set:nnn {it}{Greek}{"1D6E2}
\usv_set:nnn {it}{greek}{"1D6FC}
\usv_set:nnn {bb}{num}{"1D7D8}
```

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

```
\usv_set:nnn {bb}{Latin}{"1D538}
\usv_set:nnn {bb}{latin}{"1D552}
\usv_set:nnn {scr}{Latin}{"1D49C}
\usv_set:nnn {cal}{Latin}{"1D49C}
\usv_set:nnn {scr}{latin}{"1D4B6}
\usv_set:nnn {frak}{Latin}{"1D504}
\usv_set:nnn {frak}{latin}{"1D51E}
\usv_set:nnn {sf}{num}{"1D7E2}
\usv_set:nnn {sfup}{num}{"1D7E2}
\usv_set:nnn {sfit}{num}{"1D7E2}
\usv_set:nnn {sfup}{Latin}{"1D5A0}
\usv_set:nnn {sf}{Latin}{"1D5A0}
\usv_set:nnn {sfup}{latin}{"1D5BA}
\usv_set:nnn {sf}{latin}{"1D5BA}
\usv_set:nnn {sfit}{Latin}{"1D608}
\usv_set:nnn {sfit}{latin}{"1D622}
\usv_set:nnn {tt}{num}{"1D7F6}
\usv_set:nnn {tt}{Latin}{"1D670}
\usv_set:nnn {tt}{latin}{"1D68A}
```

Bold:

```
\usv_set:nnn {bf}{num}{"1D7CE}
\usv_set:nnn {bfup}{num}{"1D7CE}
\usv_set:nnn {bfit}{num}{"1D7CE}
\usv_set:nnn {bfup}{Latin}{"1D400}
\usv_set:nnn {bfup}{latin}{"1D41A}
\usv_set:nnn {bfup}{Greek}{"1D6A8}
\usv_set:nnn {bfup}{greek}{"1D6C2}
\usv_set:nnn {bfit}{Latin}{"1D468}
\usv_set:nnn {bfit}{latin}{"1D482}
\usv_set:nnn {bfit}{Greek}{"1D71C}
\usv_set:nnn {bfit}{greek}{"1D736}
\usv_set:nnn {bffrak}{Latin}{"1D56C}
\usv_set:nnn {bffrak}{latin}{"1D586}
\usv_set:nnn {bfscr}{Latin}{"1D4D0}
\usv_set:nnn {bfcal}{Latin}{"1D4D0}
\usv_set:nnn {bfscr}{latin}{"1D4EA}
\usv_set:nnn {bfsf}{num}{"1D7EC}
\usv_set:nnn {bfsfup}{num}{"1D7EC}
\usv_set:nnn {bfsfit}{num}{"1D7EC}
\usv_set:nnn {bfsfup}{Latin}{"1D5D4}
\usv_set:nnn {bfsfup}{latin}{"1D5EE}
\usv_set:nnn {bfsfup}{Greek}{"1D756}
\usv_set:nnn {bfsfup}{greek}{"1D770}
\usv_set:nnn {bfsfit}{Latin}{"1D63C}
\usv_set:nnn {bfsfit}{latin}{"1D656}
\usv_set:nnn {bfsfit}{Greek}{"1D790}
\usv_set:nnn {bfsfit}{greek}{"1D7AA}
```

```
\usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsfit_
\usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsfit_
\usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsfit_
```

```
\usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsfit_g
\usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_Latin
\usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_latin
\usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_Greek
\usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_greek
```

Greek variants:

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

Bold:

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

Italic Greek variants:

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

Bold italic:

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

Bold sans:

```
\usv_set:nnn {bfsfup}{varTheta}{"1D767}
\usv_set:nnn {bfsfup}{varepsilon}{"1D78A}
\usv_set:nnn {bfsfup}{vartheta}{"1D78B}
\usv_set:nnn {bfsfup}{varkappa}{"1D78C}
\usv_set:nnn {bfsfup}{varphi}{"1D78D}
\usv_set:nnn {bfsfup}{varrho}{"1D78E}
```

```
\usv_set:nnn {bfsfup}{varpi}{"1D78F}
```

Bold sans italic:

```
\usv_set:nnn {bfsfit}{varTheta}  {"1D7A1}
\usv_set:nnn {bfsfit}{varepsilon}{"1D7C4}
\usv_set:nnn {bfsfit}{vartheta}  {"1D7C5}
\usv_set:nnn {bfsfit}{varkappa}  {"1D7C6}
\usv_set:nnn {bfsfit}{varphi}    {"1D7C7}
\usv_set:nnn {bfsfit}{varrho}    {"1D7C8}
\usv_set:nnn {bfsfit}{varpi}     {"1D7C9}
```

Nabla:

```
\usv_set:nnn {up}    {Nabla}{"02207}
\usv_set:nnn {it}    {Nabla}{"1D6FB}
\usv_set:nnn {bfup}  {Nabla}{"1D6C1}
\usv_set:nnn {bfit}  {Nabla}{"1D735}
\usv_set:nnn {bfsfup}{Nabla}{"1D76F}
\usv_set:nnn {bfsfit}{Nabla}{"1D7A9}
```

Partial:

```
\usv_set:nnn {up}    {partial}{"02202}
\usv_set:nnn {it}    {partial}{"1D715}
\usv_set:nnn {bfup}  {partial}{"1D6DB}
\usv_set:nnn {bfit}  {partial}{"1D74F}
\usv_set:nnn {bfsfup}{partial}{"1D789}
\usv_set:nnn {bfsfit}{partial}{"1D7C3}
```

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

```
\usv_set:nnn {up}{B}{`\B}
\usv_set:nnn {up}{C}{`\C}
\usv_set:nnn {up}{D}{`\D}
\usv_set:nnn {up}{E}{`\E}
\usv_set:nnn {up}{F}{`\F}
\usv_set:nnn {up}{H}{`\H}
\usv_set:nnn {up}{I}{`\I}
\usv_set:nnn {up}{L}{`\L}
\usv_set:nnn {up}{M}{`\M}
\usv_set:nnn {up}{N}{`\N}
\usv_set:nnn {up}{P}{`\P}
\usv_set:nnn {up}{Q}{`\Q}
\usv_set:nnn {up}{R}{`\R}
\usv_set:nnn {up}{Z}{`\Z}

\usv_set:nnn {it}{B}{"1D435}
\usv_set:nnn {it}{C}{"1D436}
\usv_set:nnn {it}{D}{"1D437}
\usv_set:nnn {it}{E}{"1D438}
\usv_set:nnn {it}{F}{"1D439}
\usv_set:nnn {it}{H}{"1D43B}
\usv_set:nnn {it}{I}{"1D43C}
\usv_set:nnn {it}{L}{"1D43F}
```

```
\usv_set:nnn {it}{M}{"1D440}
\usv_set:nnn {it}{N}{"1D441}
\usv_set:nnn {it}{P}{"1D443}
\usv_set:nnn {it}{Q}{"1D444}
\usv_set:nnn {it}{R}{"1D445}
\usv_set:nnn {it}{Z}{"1D44D}
```

```
\usv_set:nnn {up}{d}{`\d}
\usv_set:nnn {up}{e}{`\e}
\usv_set:nnn {up}{g}{`\g}
\usv_set:nnn {up}{h}{`\h}
\usv_set:nnn {up}{i}{`\i}
\usv_set:nnn {up}{j}{`\j}
\usv_set:nnn {up}{o}{`\o}
```

```
\usv_set:nnn {it}{d}{"1D451}
\usv_set:nnn {it}{e}{"1D452}
\usv_set:nnn {it}{g}{"1D454}
\usv_set:nnn {it}{h}{"0210E}
\usv_set:nnn {it}{i}{"1D456}
\usv_set:nnn {it}{j}{"1D457}
\usv_set:nnn {it}{o}{"1D45C}
```

Latin 'h':

```
\usv_set:nnn {bb}    {h}{"1D559}
\usv_set:nnn {tt}    {h}{"1D691}
\usv_set:nnn {scr}   {h}{"1D4BD}
\usv_set:nnn {frak}  {h}{"1D525}
\usv_set:nnn {bfup}  {h}{"1D421}
\usv_set:nnn {bfit}  {h}{"1D489}
\usv_set:nnn {sfup}  {h}{"1D5C1}
\usv_set:nnn {sfit}  {h}{"1D629}
\usv_set:nnn {bffrak}{h}{"1D58D}
\usv_set:nnn {bfscr} {h}{"1D4F1}
\usv_set:nnn {bfsfup}{h}{"1D5F5}
\usv_set:nnn {bfsfit}{h}{"1D65D}
```

Dotless 'i' and 'j:

```
\usv_set:nnn {up}{dotlessi}{"00131}
\usv_set:nnn {up}{dotlessj}{"00237}
\usv_set:nnn {it}{dotlessi}{"1D6A4}
\usv_set:nnn {it}{dotlessj}{"1D6A5}
```

Blackboard:

```
\usv_set:nnn {bb}{C}{"2102}
\usv_set:nnn {bb}{H}{"210D}
\usv_set:nnn {bb}{N}{"2115}
\usv_set:nnn {bb}{P}{"2119}
\usv_set:nnn {bb}{Q}{"211A}
\usv_set:nnn {bb}{R}{"211D}
\usv_set:nnn {bb}{Z}{"2124}
\usv_set:nnn {up}{Pi}       {"003A0}
\usv_set:nnn {up}{pi}       {"003C0}
```

```
\usv_set:nnn {up}{Gamma}    {"00393}
\usv_set:nnn {up}{gamma}    {"003B3}
\usv_set:nnn {up}{summation}{"02211}
\usv_set:nnn {it}{Pi}       {"1D6F1}
\usv_set:nnn {it}{pi}       {"1D70B}
\usv_set:nnn {it}{Gamma}    {"1D6E4}
\usv_set:nnn {it}{gamma}    {"1D6FE}
\usv_set:nnn {bb}{Pi}       {"0213F}
\usv_set:nnn {bb}{pi}       {"0213C}
\usv_set:nnn {bb}{Gamma}    {"0213E}
\usv_set:nnn {bb}{gamma}    {"0213D}
\usv_set:nnn {bb}{summation}{"02140}
```

Italic blackboard:

```
\usv_set:nnn {bbit}{D}{"2145}
\usv_set:nnn {bbit}{d}{"2146}
\usv_set:nnn {bbit}{e}{"2147}
\usv_set:nnn {bbit}{i}{"2148}
\usv_set:nnn {bbit}{j}{"2149}
```

Script exceptions:

```
\usv_set:nnn {scr}{B}{"212C}
\usv_set:nnn {scr}{E}{"2130}
\usv_set:nnn {scr}{F}{"2131}
\usv_set:nnn {scr}{H}{"210B}
\usv_set:nnn {scr}{I}{"2110}
\usv_set:nnn {scr}{L}{"2112}
\usv_set:nnn {scr}{M}{"2133}
\usv_set:nnn {scr}{R}{"211B}
\usv_set:nnn {scr}{e}{"212F}
\usv_set:nnn {scr}{g}{"210A}
\usv_set:nnn {scr}{o}{"2134}

\usv_set:nnn {cal}{B}{"212C}
\usv_set:nnn {cal}{E}{"2130}
\usv_set:nnn {cal}{F}{"2131}
\usv_set:nnn {cal}{H}{"210B}
\usv_set:nnn {cal}{I}{"2110}
\usv_set:nnn {cal}{L}{"2112}
\usv_set:nnn {cal}{M}{"2133}
\usv_set:nnn {cal}{R}{"211B}
```

Fractur exceptions:

```
\usv_set:nnn {frak}{C}{"212D}
\usv_set:nnn {frak}{H}{"210C}
\usv_set:nnn {frak}{I}{"2111}
\usv_set:nnn {frak}{R}{"211C}
\usv_set:nnn {frak}{Z}{"2128}
```

9.3 STIX fonts

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

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

```
⟨/package&(XE|LU)⟩
⟨*stix⟩
```

Upright

```
\usv_set:nnn {stixsfup}{partial}{"E17C}
\usv_set:nnn {stixsfup}{Greek}{"E17D}
\usv_set:nnn {stixsfup}{greek}{"E196}
\usv_set:nnn {stixsfup}{varTheta}{"E18E}
\usv_set:nnn {stixsfup}{varepsilon}{"E1AF}
\usv_set:nnn {stixsfup}{vartheta}{"E1B0}
\usv_set:nnn {stixsfup}{varkappa}{0000} % ???
\usv_set:nnn {stixsfup}{varphi}{"E1B1}
\usv_set:nnn {stixsfup}{varrho}{"E1B2}
\usv_set:nnn {stixsfup}{varpi}{"E1B3}
\usv_set:nnn {stixupslash}{Greek}{"E2FC}
```

Italic

```
\usv_set:nnn {stixbbit}{A}{"E154}
\usv_set:nnn {stixbbit}{B}{"E155}
\usv_set:nnn {stixbbit}{E}{"E156}
\usv_set:nnn {stixbbit}{F}{"E157}
\usv_set:nnn {stixbbit}{G}{"E158}
\usv_set:nnn {stixbbit}{I}{"E159}
\usv_set:nnn {stixbbit}{J}{"E15A}
\usv_set:nnn {stixbbit}{K}{"E15B}
\usv_set:nnn {stixbbit}{L}{"E15C}
\usv_set:nnn {stixbbit}{M}{"E15D}
\usv_set:nnn {stixbbit}{O}{"E15E}
\usv_set:nnn {stixbbit}{S}{"E15F}
\usv_set:nnn {stixbbit}{T}{"E160}
\usv_set:nnn {stixbbit}{U}{"E161}
\usv_set:nnn {stixbbit}{V}{"E162}
\usv_set:nnn {stixbbit}{W}{"E163}
\usv_set:nnn {stixbbit}{X}{"E164}
\usv_set:nnn {stixbbit}{Y}{"E165}

\usv_set:nnn {stixbbit}{a}{"E166}
\usv_set:nnn {stixbbit}{b}{"E167}
\usv_set:nnn {stixbbit}{c}{"E168}
\usv_set:nnn {stixbbit}{f}{"E169}
\usv_set:nnn {stixbbit}{g}{"E16A}
\usv_set:nnn {stixbbit}{h}{"E16B}
\usv_set:nnn {stixbbit}{k}{"E16C}
\usv_set:nnn {stixbbit}{l}{"E16D}
\usv_set:nnn {stixbbit}{m}{"E16E}
\usv_set:nnn {stixbbit}{n}{"E16F}
\usv_set:nnn {stixbbit}{o}{"E170}
\usv_set:nnn {stixbbit}{p}{"E171}
```

```
\usv_set:nnn {stixbbit}{q}{"E172}
\usv_set:nnn {stixbbit}{r}{"E173}
\usv_set:nnn {stixbbit}{s}{"E174}
\usv_set:nnn {stixbbit}{t}{"E175}
\usv_set:nnn {stixbbit}{u}{"E176}
\usv_set:nnn {stixbbit}{v}{"E177}
\usv_set:nnn {stixbbit}{w}{"E178}
\usv_set:nnn {stixbbit}{x}{"E179}
\usv_set:nnn {stixbbit}{y}{"E17A}
\usv_set:nnn {stixbbit}{z}{"E17B}

\usv_set:nnn {stixsfit}{Numerals}{"E1B4}
\usv_set:nnn {stixsfit}{partial}{"E1BE}
\usv_set:nnn {stixsfit}{Greek}{"E1BF}
\usv_set:nnn {stixsfit}{greek}{"E1D8}
\usv_set:nnn {stixsfit}{varTheta}{"E1D0}
\usv_set:nnn {stixsfit}{varepsilon}{"E1F1}
\usv_set:nnn {stixsfit}{vartheta}{"E1F2}
\usv_set:nnn {stixsfit}{varkappa}{0000} % ???
\usv_set:nnn {stixsfit}{varphi}{"E1F3}
\usv_set:nnn {stixsfit}{varrho}{"E1F4}
\usv_set:nnn {stixsfit}{varpi}{"E1F5}

\usv_set:nnn {stixcal}{Latin}{"E22D}
\usv_set:nnn {stixcal}{num}{"E262}
\usv_set:nnn {scr}{num}{48}
\usv_set:nnn {it}{num}{48}

\usv_set:nnn {stixsfitslash}{Latin}{"E294}
\usv_set:nnn {stixsfitslash}{latin}{"E2C8}
\usv_set:nnn {stixsfitslash}{greek}{"E32C}
\usv_set:nnn {stixsfitslash}{varepsilon}{"E37A}
\usv_set:nnn {stixsfitslash}{vartheta}{"E35E}
\usv_set:nnn {stixsfitslash}{varkappa}{"E374}
\usv_set:nnn {stixsfitslash}{varphi}{"E360}
\usv_set:nnn {stixsfitslash}{varrho}{"E376}
\usv_set:nnn {stixsfitslash}{varpi}{"E362}
\usv_set:nnn {stixsfitslash}{digamma}{"E36A}
```

Bold

```
\usv_set:nnn {stixbfupslash}{Greek}{"E2FD}
\usv_set:nnn {stixbfupslash}{Digamma}{"E369}

\usv_set:nnn {stixbfbb}{A}{"E38A}
\usv_set:nnn {stixbfbb}{B}{"E38B}
\usv_set:nnn {stixbfbb}{E}{"E38D}
\usv_set:nnn {stixbfbb}{F}{"E38E}
\usv_set:nnn {stixbfbb}{G}{"E38F}
\usv_set:nnn {stixbfbb}{I}{"E390}
\usv_set:nnn {stixbfbb}{J}{"E391}
\usv_set:nnn {stixbfbb}{K}{"E392}
\usv_set:nnn {stixbfbb}{L}{"E393}
\usv_set:nnn {stixbfbb}{M}{"E394}
```

```
\usv_set:nnn {stixbfbb}{O}{"E395}
\usv_set:nnn {stixbfbb}{S}{"E396}
\usv_set:nnn {stixbfbb}{T}{"E397}
\usv_set:nnn {stixbfbb}{U}{"E398}
\usv_set:nnn {stixbfbb}{V}{"E399}
\usv_set:nnn {stixbfbb}{W}{"E39A}
\usv_set:nnn {stixbfbb}{X}{"E39B}
\usv_set:nnn {stixbfbb}{Y}{"E39C}

\usv_set:nnn {stixbfbb}{a}{"E39D}
\usv_set:nnn {stixbfbb}{b}{"E39E}
\usv_set:nnn {stixbfbb}{c}{"E39F}
\usv_set:nnn {stixbfbb}{f}{"E3A2}
\usv_set:nnn {stixbfbb}{g}{"E3A3}
\usv_set:nnn {stixbfbb}{h}{"E3A4}
\usv_set:nnn {stixbfbb}{k}{"E3A7}
\usv_set:nnn {stixbfbb}{l}{"E3A8}
\usv_set:nnn {stixbfbb}{m}{"E3A9}
\usv_set:nnn {stixbfbb}{n}{"E3AA}
\usv_set:nnn {stixbfbb}{o}{"E3AB}
\usv_set:nnn {stixbfbb}{p}{"E3AC}
\usv_set:nnn {stixbfbb}{q}{"E3AD}
\usv_set:nnn {stixbfbb}{r}{"E3AE}
\usv_set:nnn {stixbfbb}{s}{"E3AF}
\usv_set:nnn {stixbfbb}{t}{"E3B0}
\usv_set:nnn {stixbfbb}{u}{"E3B1}
\usv_set:nnn {stixbfbb}{v}{"E3B2}
\usv_set:nnn {stixbfbb}{w}{"E3B3}
\usv_set:nnn {stixbfbb}{x}{"E3B4}
\usv_set:nnn {stixbfbb}{y}{"E3B5}
\usv_set:nnn {stixbfbb}{z}{"E3B6}

\usv_set:nnn {stixbfsfup}{Numerals}{"E3B7}
```

Bold Italic

```
\usv_set:nnn {stixbfsfit}{Numerals}{"E1F6}

\usv_set:nnn {stixbfbbit}{A}{"E200}
\usv_set:nnn {stixbfbbit}{B}{"E201}
\usv_set:nnn {stixbfbbit}{E}{"E203}
\usv_set:nnn {stixbfbbit}{F}{"E204}
\usv_set:nnn {stixbfbbit}{G}{"E205}
\usv_set:nnn {stixbfbbit}{I}{"E206}
\usv_set:nnn {stixbfbbit}{J}{"E207}
\usv_set:nnn {stixbfbbit}{K}{"E208}
\usv_set:nnn {stixbfbbit}{L}{"E209}
\usv_set:nnn {stixbfbbit}{M}{"E20A}
\usv_set:nnn {stixbfbbit}{O}{"E20B}
\usv_set:nnn {stixbfbbit}{S}{"E20C}
\usv_set:nnn {stixbfbbit}{T}{"E20D}
\usv_set:nnn {stixbfbbit}{U}{"E20E}
\usv_set:nnn {stixbfbbit}{V}{"E20F}
```

```
\usv_set:nnn {stixbfbbit}{W}{"E210}
\usv_set:nnn {stixbfbbit}{X}{"E211}
\usv_set:nnn {stixbfbbit}{Y}{"E212}
\usv_set:nnn {stixbfbbit}{a}{"E213}
\usv_set:nnn {stixbfbbit}{b}{"E214}
\usv_set:nnn {stixbfbbit}{c}{"E215}
\usv_set:nnn {stixbfbbit}{e}{"E217}
\usv_set:nnn {stixbfbbit}{f}{"E218}
\usv_set:nnn {stixbfbbit}{g}{"E219}
\usv_set:nnn {stixbfbbit}{h}{"E21A}
\usv_set:nnn {stixbfbbit}{k}{"E21D}
\usv_set:nnn {stixbfbbit}{l}{"E21E}
\usv_set:nnn {stixbfbbit}{m}{"E21F}
\usv_set:nnn {stixbfbbit}{n}{"E220}
\usv_set:nnn {stixbfbbit}{o}{"E221}
\usv_set:nnn {stixbfbbit}{p}{"E222}
\usv_set:nnn {stixbfbbit}{q}{"E223}
\usv_set:nnn {stixbfbbit}{r}{"E224}
\usv_set:nnn {stixbfbbit}{s}{"E225}
\usv_set:nnn {stixbfbbit}{t}{"E226}
\usv_set:nnn {stixbfbbit}{u}{"E227}
\usv_set:nnn {stixbfbbit}{v}{"E228}
\usv_set:nnn {stixbfbbit}{w}{"E229}
\usv_set:nnn {stixbfbbit}{x}{"E22A}
\usv_set:nnn {stixbfbbit}{y}{"E22B}
\usv_set:nnn {stixbfbbit}{z}{"E22C}
\usv_set:nnn {stixbfcal}{Latin}{"E247}
\usv_set:nnn {stixbfitslash}{Latin}{"E295}
\usv_set:nnn {stixbfitslash}{latin}{"E2C9}
\usv_set:nnn {stixbfitslash}{greek}{"E32D}
\usv_set:nnn {stixsfitslash}{varepsilon}{"E37B}
\usv_set:nnn {stixsfitslash}{vartheta}{"E35F}
\usv_set:nnn {stixsfitslash}{varkappa}{"E375}
\usv_set:nnn {stixsfitslash}{varphi}{"E361}
\usv_set:nnn {stixsfitslash}{varrho}{"E377}
\usv_set:nnn {stixsfitslash}{varpi}{"E363}
\usv_set:nnn {stixsfitslash}{digamma}{"E36B}
⟨/stix⟩
⟨*package&(XE|LU)⟩
```

9.4 Overcoming \@onlypreamble

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

```
\tl_map_inline:nn {
  \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
  \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
  \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
  \version@list\version@elt\alpha@list\alpha@elt
```

```
 \restore@mathversion\init@restore@version\dorestore@version\process@table
  \new@mathversion\DeclareSymbolFont\group@list\group@elt
  \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
  \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
  \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
  \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
  \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
  \set@@mathdelimiter\DeclareMathRadical\mathchar@type
  \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
}{
  \tl_remove_once:Nn \@preamblecmds {\do#1}
}
```

10 Fundamentals

10.1 Enlarging the number of maths families

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

```
⟨*XE⟩
\def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
\let\newfam\new@mathgroup
⟨/XE⟩
```

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

10.2 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.

```
\cs_set:Nn \um_set_mathsymbol:nNNn {
  \tl_case:Nnn #3 {
    \mathop { \um_set_big_operator:nnn {#1} {#2} {#4} }
    \mathopen
      {
        \tl_if_in:NnTF \l_um_radicals_tl {#2}
          {
            \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
              { \um_radical:nn {#1} {#4} }
           \tl_set:cn {l_um_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #4}
          }
          {
            \um_set_delcode:nnn {#1} {#4} {#4}
            \um_set_mathcode:nnn {#4} \mathopen {#1}
            \cs_gset_protected_nopar:Npx #2
              { \um_delimiter:Nnn \mathopen {#1} {#4} }
          }
      }
    \mathclose
      {
        \um_set_delcode:nnn {#1} {#4} {#4}
        \um_set_mathcode:nnn {#4} \mathclose {#1}
        \cs_gset_protected_nopar:Npx #2
          { \um_delimiter:Nnn \mathclose {#1} {#4} }
      }
    \mathfence
      {
        \um_set_mathcode:nnn {#4} {#3} {#1}
        \um_set_delcode:nnn {#1} {#4} {#4}
        \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
          { \um_delimiter:Nnn \mathopen  {#1} {#4} }
        \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
          { \um_delimiter:Nnn \mathclose {#1} {#4} }
      }
    \mathaccent
    { \cs_gset_protected_nopar:Npx #2 { \um_accent:nnn {fixed} {#1} {#4} } }
    \mathbotaccent
            {  \cs_gset_protected_nopar:Npx  #2  {  \um_accent:nnn  {bot-
tom~ fixed} {#1} {#4} } }
    \mathover
      {
        \cs_set_protected_nopar:Npx #2 ##1
          { \mathop { \um_accent:nnn {} {#1} {#4} {##1} } \limits }
      }
    \mathunder
      {
        \cs_set_protected_nopar:Npx #2 ##1
          { \mathop { \um_accent:nnn {bottom} {#1} {#4} {##1} } \limits }
      }
  }{
    \um_set_mathcode:nnn {#4} {#3} {#1}
  }
}
```

```
\edef\mathfence{\string\mathfence}
\edef\mathover{\string\mathover}
\edef\mathunder{\string\mathunder}
\edef\mathbotaccent{\string\mathbotaccent}
```

`\um_set_big_operator:nnn`

#1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot

In the examples following, say we're defining for the symbol `\sum` (∑). 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 LaTeX/amsmath.)
- Define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

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

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

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

(`\int` →) $\int$ → `\int_sym` → `\intop`

```
\cs_new:Nn \um_set_big_operator:nnn {
  \group_begin:
    \char_set_catcode_active:n {#3}
    \char_gmake_mathactive:n {#3}
    \um_active_char_set:wc #3 \q_nil { \cs_to_str:N #2 _sym }
  \group_end:
  \um_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
  \cs_gset:cpx { \cs_to_str:N #2 _sym } {
    \exp_not:c { \cs_to_str:N #2 op   }
    \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
  }
}
```

`\um_set_mathcode:nnnn`
`\um_set_mathcode:nnn`
`\um_set_mathchar:NNnn`
`\um_set_mathchar:cNnn`
`\um_set_delcode:nnn`
`\um_radical:nn`
`\um_delimiter:Nnn`
`\um_accent:nnn`
`\um_accent_keyword:`

These are all wrappers for the primitive commands that take numerical input only.

```
\cs_set:Npn \um_set_mathcode:nnnn #1#2#3#4 {
  \Umathcode \int_eval:n {#1} =
    \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
}
\cs_set:Npn \um_set_mathcode:nnn #1#2#3 {
  \Umathcode \int_eval:n {#1} =
    \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
}
\cs_set:Npn \um_set_mathchar:NNnn #1#2#3#4 {
  \Umathchardef #1 =
    \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
}
\cs_new:Nn \um_set_delcode:nnn {
  \Udelcode#2 = \csname sym#1\endcsname #3
}
\cs_new:Nn \um_radical:nn {
  \Uradical \csname sym#1\endcsname #2 \scan_stop:
}
\cs_new:Nn \um_delimiter:Nnn {
  \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
}
\cs_new:Nn \um_accent:nnn {
 \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
}
\cs_generate_variant:Nn \um_set_mathchar:NNnn {c}
```

\char_gmake_mathactive:N
\char_gmake_mathactive:n

```
\cs_new:Nn \char_gmake_mathactive:N {
  \global\mathcode `#1 = "8000 \scan_stop:
}
\cs_new:Nn \char_gmake_mathactive:n {
  \global\mathcode #1 = "8000 \scan_stop:
}
```

10.3 The main `\setmathfont` macro

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

\setmathfont [#1]: font features
#2 : font name

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

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

```
       \cs_set_eq:NN \glb@currsize \scan_stop:
```

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

```
       \bool_set_true:N \l_um_init_bool
       \seq_clear:N \l_um_char_range_seq
       \clist_clear:N \l_um_char_num_range_clist
       \seq_clear:N \l_um_mathalph_seq
       \seq_clear:N \l_um_missing_alph_seq
```

- By default use the 'normal' math version

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

- Other range initialisations

```
         \tl_set:Nn \um_symfont_tl {operators}
         \cs_set_eq:NN \_um_sym:nnn \um_process_symbol_noparse:nnn
        \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
         \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
         \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
         \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
        \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_noparse:nn
        \cs_set_eq:NN \um_make_mathactive:nNN \um_make_mathactive_noparse:nNN
```

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

```
      \tl_set:Nn \l_um_script_features_tl  {Style=MathScript}
      \tl_set:Nn \l_um_sscript_features_tl {Style=MathScriptScript}
      \tl_set_eq:NN \l_um_script_font_tl   \l_um_fontname_tl
      \tl_set_eq:NN \l_um_sscript_font_tl  \l_um_fontname_tl

}
\DeclareDocumentCommand \setmathfont { O{} m } {
  \tl_set:Nn \l_um_fontname_tl {#2}
  \um_init:
```

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

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

Parse options and tell people what's going on:

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

Use fontspec to select a font to use.

```
  \um_fontspec_select_font:
```

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

```
  \cs_if_exist:cF { sym \um_symfont_tl }
    {
      \DeclareSymbolFont{\um_symfont_tl}
        {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
    }
  \SetSymbolFont{\um_symfont_tl}{\l_um_mversion_tl}
    {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}
```

Set the bold math version.

```
  \tl_set:Nn \l_um_tmpa_tl {normal}
  \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
    {
    \SetSymbolFont{\um_symfont_tl}{bold}
      {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
    }
```

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

```
  \bool_if:nT {\l_um_ot_math_bool && !\g_um_mainfont_already_set_bool} {
    \bool_set_true:N \g_um_mainfont_already_set_bool
    \um_declare_math_sizes:
    \um_setup_legacy_fam_two:
    \um_setup_legacy_fam_three:
  }
```

And now we input every single maths char.

```
  \um_input_math_symbol_table:
```

Finally,

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

```
  \um_remap_symbols:
  \um_setup_mathactives:
  \um_setup_accents:
  \um_setup_delcodes:
  \um_setup_alphabets:
  \um_setup_negations:
```

Prevent spaces, and that's it:

```
  \ignorespaces
}
```

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

```
\cs_new:Nn \um_declare_math_sizes:
  {
    \dim_compare:nF { \fontdimen 10 \l_um_font == 0pt }
      {
        \DeclareMathSizes { \f@size } { \f@size }
          { \um_fontdimen_to_scale:nn {10} {\l_um_font} }
          { \um_fontdimen_to_scale:nn {11} {\l_um_font} }
      }
  }
```

`\um_setup_legacy_fam_two:`

```
\cs_new:Nn \um_setup_legacy_fam_two:
  {
    \fontspec_set_family:Nxn \l_um_family_tl
      {
      \l_um_font_keyval_tl,
      Scale=1.00001,
      FontAdjustment={
        \fontdimen8\font= \um_get_fontparam:nn {43} {FractionNumeratorDis-
playStyleShiftUp}\relax
          \fontdimen9\font= \um_get_fontparam:nn {42} {FractionNumerator-
ShiftUp}\relax
       \fontdimen10\font=\um_get_fontparam:nn {32} {StackTopShiftUp}\relax
           \fontdimen11\font=\um_get_fontparam:nn {45} {FractionDenomina-
torDisplayStyleShiftDown}\relax
        \fontdimen12\font=\um_get_fontparam:nn {44} {FractionDenominator-
ShiftDown}\relax
               \fontdimen13\font=\um_get_fontparam:nn  {21}  {Superscript-
ShiftUp}\relax
```

```
              \fontdimen14\font=\um_get_fontparam:nn  {21}  {Superscript-
ShiftUp}\relax
           \fontdimen15\font=\um_get_fontparam:nn  {22}  {SuperscriptShif-
tUpCramped}\relax
            \fontdimen16\font=\um_get_fontparam:nn  {18}  {SubscriptShift-
Down}\relax
         \fontdimen17\font=\um_get_fontparam:nn {18} {SubscriptShiftDown-
WithSuperscript}\relax
        \fontdimen18\font=\um_get_fontparam:nn {24} {SuperscriptBaseline-
DropMax}\relax
       \fontdimen19\font=\um_get_fontparam:nn {20} {SubscriptBaselineDrop-
Min}\relax
       \fontdimen20\font=0pt\relax % delim1 = FractionDelimiterDisplaySize
        \fontdimen21\font=0pt\relax % delim2 = FractionDelimiterSize
        \fontdimen22\font=\um_get_fontparam:nn {15} {AxisHeight}\relax
      }
      } {\l_um_fontname_tl}
    \SetSymbolFont{symbols}{\l_um_mversion_tl}
      {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}

    \tl_set:Nn \l_um_tmpa_tl {normal}
    \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
      {
      \SetSymbolFont{symbols}{bold}
        {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
      }
  }
```

\um_setup_legacy_fam_three:

```
\cs_new:Nn \um_setup_legacy_fam_three:
  {
    \fontspec_set_family:Nxn \l_um_family_tl
      {
      \l_um_font_keyval_tl,
      Scale=0.99999,
      FontAdjustment={
          \fontdimen8\font= \um_get_fontparam:nn {48} {FractionRuleThick-
ness}\relax
      \fontdimen9\font= \um_get_fontparam:nn {28} {UpperLimitGapMin}\relax
      \fontdimen10\font=\um_get_fontparam:nn {30} {LowerLimitGapMin}\relax
        \fontdimen11\font=\um_get_fontparam:nn {29} {UpperLimitBaselineR-
iseMin}\relax
         \fontdimen12\font=\um_get_fontparam:nn {31} {LowerLimitBaseline-
DropMin}\relax
        \fontdimen13\font=0pt\relax
      }
    } {\l_um_fontname_tl}
    \SetSymbolFont{largesymbols}{\l_um_mversion_tl}
      {\encodingdefault}{\l_um_family_tl}{\mddefault}{\updefault}

    \tl_set:Nn \l_um_tmpa_tl {normal}
    \tl_if_eq:NNT \l_um_mversion_tl \l_um_tmpa_tl
      {
      \SetSymbolFont{largesymbols}{bold}
        {\encodingdefault}{\l_um_family_tl}{\bfdefault}{\updefault}
      }
  }
```

```
\cs_new:Nn \um_get_fontparam:nn
⟨XE⟩  { \the\fontdimen#1\l_um_font\relax }
⟨LU⟩  { \directlua{fontspec.mathfontdimen("l_um_font","#2")} }
```

Backward compatibility alias.

```
\cs_set_eq:NN \resetmathfont \setmathfont
```

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

```
\cs_new:Nn \um_fontspec_select_font: {
  \tl_set:Nx \l_um_font_keyval_tl {
⟨LU⟩      Renderer = Basic,
    BoldItalicFont = {}, ItalicFont = {},
    Script = Math,
    SizeFeatures = {
      {Size = \tf@size-} ,
      {Size = \sf@size-\tf@size ,
       Font = \l_um_script_font_tl ,
       \l_um_script_features_tl
      } ,
      {Size = -\sf@size ,
       Font = \l_um_sscript_font_tl ,
       \l_um_sscript_features_tl
      }
    },
    \l_um_unknown_keys_clist
  }
  \fontspec_set_fontface:NNxn \l_um_font \l_um_family_tl
    {\l_um_font_keyval_tl} {\l_um_fontname_tl}
```

Check whether we're using a real maths font:

```
  \group_begin:
    \fontfamily{\l_um_family_tl}\selectfont
    \fontspec_if_script:nF {math} {\bool_gset_false:N \l_um_ot_math_bool}
  \group_end:
}
```

10.3.1 Functions for setting up symbols with mathcodes

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

```
\cs_set:Npn \um_process_symbol_noparse:nnn #1#2#3 {
  \um_set_mathsymbol:nNNn {\um_symfont_tl} #2#3{#1}
}
```

```
\cs_set:Npn \um_process_symbol_parse:nnn #1#2#3 {
  \um_if_char_spec:nNNT{#1}{#2}{#3}{
    \um_process_symbol_noparse:nnn {#1}{#2}{#3}
  }
}
```

\um_remap_symbols:
\um_remap_symbol_noparse:nnn
\um_remap_symbol_parse:nnn

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

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

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

```
\cs_new:Nn \um_remap_symbol_parse:nnn {
  \um_if_char_spec:nNNT {#3} {\@nil} {#2} {
    \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
  }
}
\cs_new:Nn \um_remap_symbol_noparse:nnn {
  \clist_map_inline:nn {#1} {
    \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
  }
}
```

10.3.2 Active math characters

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

\um_setup_mathactives:

```
\cs_new:Npn \um_setup_mathactives: {
  \um_make_mathactive:nNN {"2032} \um_prime_single_mchar \mathord
  \um_make_mathactive:nNN {"2033} \um_prime_double_mchar \mathord
  \um_make_mathactive:nNN {"2034} \um_prime_triple_mchar \mathord
  \um_make_mathactive:nNN {"2057} \um_prime_quad_mchar   \mathord
  \um_make_mathactive:nNN {"2035} \um_backprime_single_mchar \mathord
  \um_make_mathactive:nNN {"2036} \um_backprime_double_mchar \mathord
  \um_make_mathactive:nNN {"2037} \um_backprime_triple_mchar \mathord
  \um_make_mathactive:nNN {`\'} \mathstraightquote \mathord
  \um_make_mathactive:nNN {`\`} \mathbacktick      \mathord
}
```

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

```
\cs_new:Nn \um_make_mathactive_parse:nNN
  {
    \um_if_char_spec:nNNT {#1} #2 #3
      { \um_make_mathactive_noparse:nNN {#1} #2 #3 }
  }
\cs_new:Nn \um_make_mathactive_noparse:nNN
  {
    \um_set_mathchar:NNnn #2 #3 {\um_symfont_tl} {#1}
    \char_gmake_mathactive:n {#1}
  }
```

10.3.3 Delimiter codes

\um_assign_delcode:nn

```
\cs_new:Nn \um_assign_delcode_noparse:nn {
  \um_set_delcode:nnn \um_symfont_tl {#1} {#2}
}
\cs_new:Nn \um_assign_delcode_parse:nn {
  \um_if_char_spec:nNNT {#2}{\@nil}{\@nil} {
    \um_assign_delcode_noparse:nn {#1} {#2}
  }
}
```

\um_assign_delcode:n Shorthand.

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

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

\um_setup_delcodes:

```
\cs_new:Npn \um_setup_delcodes: {
  \um_assign_delcode:nn {`\.} {\c_zero} % ensure \left. and \right. work
  \um_assign_delcode:nn {`\/}   {\g_um_slash_delimiter_usv}
  \um_assign_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
  \um_assign_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
  \um_assign_delcode:n {"005C} % backslash
  \um_assign_delcode:nn {`\<} {"27E8} % angle brackets with ascii notation
  \um_assign_delcode:nn {`\>} {"27E9} % angle brackets with ascii notation
  \um_assign_delcode:n {"2191} % up arrow
  \um_assign_delcode:n {"2193} % down arrow
  \um_assign_delcode:n {"2195} % updown arrow
  \um_assign_delcode:n {"219F} % up arrow twohead
  \um_assign_delcode:n {"21A1} % down arrow twohead
  \um_assign_delcode:n {"21A5} % up arrow from bar
  \um_assign_delcode:n {"21A7} % down arrow from bar
  \um_assign_delcode:n {"21A8} % updown arrow from bar
  \um_assign_delcode:n {"21BE} % up harpoon right
  \um_assign_delcode:n {"21BF} % up harpoon left
  \um_assign_delcode:n {"21C2} % down harpoon right
  \um_assign_delcode:n {"21C3} % down harpoon left
  \um_assign_delcode:n {"21C5} % arrows up down
  \um_assign_delcode:n {"21F5} % arrows down up
  \um_assign_delcode:n {"21C8} % arrows up up
  \um_assign_delcode:n {"21CA} % arrows down down
  \um_assign_delcode:n {"21D1} % double up arrow
  \um_assign_delcode:n {"21D3} % double down arrow
  \um_assign_delcode:n {"21D5} % double updown arrow
  \um_assign_delcode:n {"21DE} % up arrow double stroke
  \um_assign_delcode:n {"21DF} % down arrow double stroke
  \um_assign_delcode:n {"21E1} % up arrow dashed
  \um_assign_delcode:n {"21E3} % down arrow dashed
  \um_assign_delcode:n {"21E7} % up white arrow
  \um_assign_delcode:n {"21E9} % down white arrow
  \um_assign_delcode:n {"21EA} % up white arrow from bar
  \um_assign_delcode:n {"21F3} % updown white arrow
}
```

10.4 (Big) operators

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

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

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

```
\tl_new:N \l_um_nolimits_tl
\tl_set:Nn \l_um_nolimits_tl {
  \int\iint\iiint\iiiint\oint\oiint\oiiint
  \intclockwise\varointclockwise\ointctrclockwise\sumint
  \intbar\intBar\fint\cirfnint\awint\rppolint
  \scpolint\npolint\pointint\sqint\intlarhk\intx
  \intcap\intcup\upint\lowint
}
```

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

```
\DeclareDocumentCommand \addnolimits {m} {
  \tl_put_right:Nn \l_um_nolimits_tl {#1}
}
```

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

```
\DeclareDocumentCommand \removenolimits {m} {
  \tl_remove_all:Nn \l_um_nolimits_tl {#1}
}
```

10.5 Radicals

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

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

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

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

10.6 Maths accents

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

10.7 Common interface for font parameters

XeTeX and LuaTeX have different interfaces for math font parameters. We use LuaTeX's interface because it's much better, but rename the primitives to be more LaTeX3-like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the LuaTeX names, with underscores inserted between words. For every parameter `\Umath⟨LuaTeX name⟩`, we define an expandable getter command `\um_⟨LaTeX3 name⟩:N` and a protected setter command `\um_set_⟨LaTeX3 name⟩:Nn`. The getter command takes one of the style primitives (`\displaystyle` etc.) and expands to the font parameter, which is a ⟨*dimension*⟩. The setter command takes a style primitive and a dimension expression, which is parsed with `\dim_eval:n`.

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

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

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

`\um_new_cramped_style:N`

`#1` : command

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

```
\cs_new_protected_nopar:Nn \um_new_cramped_style:N
⟨XE⟩  { \quark_new:N #1 }
⟨LU⟩  { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
```

`\crampeddisplaystyle`
`\crampedtextstyle`
`\crampedscriptstyle`
`\crampedscriptscriptstyle`

The cramped style commands.

```
\um_new_cramped_style:N \crampeddisplaystyle
\um_new_cramped_style:N \crampedtextstyle
\um_new_cramped_style:N \crampedscriptstyle
\um_new_cramped_style:N \crampedscriptscriptstyle
```

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

`\um_font_dimen:Nnnnn`

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

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

```
⟨*XE⟩
  \cs_new_nopar:Npn \um_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
    \fontdimen
    \cs_if_eq:NNTF #1 \displaystyle {
      #2 \textfont
    } {
      \cs_if_eq:NNTF #1 \crampeddisplaystyle {
        #3 \textfont
      } {
        \cs_if_eq:NNTF #1 \textstyle {
          #4 \textfont
        } {
          \cs_if_eq:NNTF #1 \crampedtextstyle {
            #5 \textfont
          } {
            \cs_if_eq:NNTF #1 \scriptstyle {
              #4 \scriptfont
            } {
              \cs_if_eq:NNTF #1 \crampedscriptstyle {
                #5 \scriptfont
              } {
                \cs_if_eq:NNTF #1 \scriptscriptstyle {
                  #4 \scriptscriptfont
                } {
```

Should we check here if the style is invalid?

```
                  #5 \scriptscriptfont
                }
              }
            }
          }
        }
      }
    }
```

Which family to use?

```
    \c_two
  }
⟨/XE⟩
```

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

`\um_font_param:nnnnn`

#1 : name
#2 : font dimension for non-cramped display style
#3 : font dimension for cramped display style
#4 : font dimension for non-cramped non-display styles
#5 : font dimension for cramped non-display styles

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTEX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XƎTEX font dimension numbers must be integer constants.

```
\cs_new_protected_nopar:Nn \um_font_param:nnnnn
⟨*XE⟩
{
  \um_font_param_aux:ccnnnn { um_ #1 :N } { um_set_ #1 :N }
    { #2 } { #3 } { #4 } { #5 }
}
⟨/XE⟩
⟨*LU⟩
{
  \tl_set:Nn \l_um_tmpa_tl { #1 }
  \tl_remove_all:Nn \l_um_tmpa_tl { _ }
  \um_font_param_aux:ccc { um_ #1 :N } { um_set_ #1 :N }
    { luatexUmath \l_um_tmpa_tl }
}
⟨/LU⟩
```

`\um_font_param:nnn`

`#1` : name
`#2` : font dimension for display style
`#3` : font dimension for non-display styles

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XeTeX font dimension numbers must be integer constants.

```
\cs_new_protected_nopar:Npn \um_font_param:nnn #1 #2 #3 {
  \um_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
}
```

`\um_font_param:nn`

`#1` : name
`#2` : font dimension

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The XeTeX font dimension number must be an integer constant.

```
\cs_new_protected_nopar:Npn \um_font_param:nn #1 #2 {
  \um_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
}
```

`\um_font_param:n`

`#1` : name

This macro defines getter and setter functions for the font parameter ⟨*name*⟩, which is considered unavailable in XeTeX. The LuaTeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`.

```
\cs_new_protected_nopar:Nn \um_font_param:n
⟨XE⟩  { }
⟨LU⟩  { \um_font_param:nnnnn { #1 } { 0 } { 0 } { 0 } { 0 } }
```

`\um_font_param_aux:NNnnnn`
`\um_font_param_aux:NNN`

Auxiliary macros for generating font parameter accessor macros.

```
⟨*XE⟩
\cs_new_protected_nopar:Nn \um_font_param_aux:NNnnnn
  {
    \cs_new_nopar:Npn #1 ##1 {
      \um_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
    }
    \cs_new_protected_nopar:Npn #2 ##1 ##2 {
      #1 ##1 \dim_eval:n { ##2 }
    }
  }
\cs_generate_variant:Nn \um_font_param_aux:NNnnnn { cc }
⟨/XE⟩
⟨*LU⟩
\cs_new_protected_nopar:Nn \um_font_param_aux:NNN
  {
    \cs_new_nopar:Npn #1 ##1 {
      #3 ##1
    }
    \cs_new_protected_nopar:Npn #2 ##1 ##2 {
      #3 ##1 \dim_eval:n { ##2 }
    }
  }
\cs_generate_variant:Nn \um_font_param_aux:NNN { ccc }
⟨/LU⟩
```

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

```
\um_font_param:nn { axis } { 15 }
\um_font_param:nn { operator_size } { 13 }
\um_font_param:n { fraction_del_size }
\um_font_param:nnn { fraction_denom_down } { 45 } { 44 }
\um_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
\um_font_param:nnn { fraction_num_up } { 43 } { 42 }
\um_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
\um_font_param:nn { fraction_rule } { 48 }
\um_font_param:nn { limit_above_bgap } { 29 }
\um_font_param:n { limit_above_kern }
\um_font_param:nn { limit_above_vgap } { 28 }
\um_font_param:nn { limit_below_bgap } { 31 }
\um_font_param:n { limit_below_kern }
\um_font_param:nn { limit_below_vgap } { 30 }
\um_font_param:nn { over_delimiter_vgap } { 41 }
\um_font_param:nn { over_delimiter_bgap } { 38 }
\um_font_param:nn { under_delimiter_vgap } { 40 }
\um_font_param:nn { under_delimiter_bgap } { 39 }
\um_font_param:nn { overbar_kern } { 55 }
\um_font_param:nn { overbar_rule } { 54 }
\um_font_param:nn { overbar_vgap } { 53 }
\um_font_param:n { quad }
\um_font_param:nn { radical_kern } { 62 }
\um_font_param:nn { radical_rule } { 61 }
\um_font_param:nnn { radical_vgap } { 60 } { 59 }
\um_font_param:nn { radical_degree_before } { 63 }
\um_font_param:nn { radical_degree_after } { 64 }
\um_font_param:nn { radical_degree_raise } { 65 }
\um_font_param:nn { space_after_script } { 27 }
\um_font_param:nnn { stack_denom_down } { 35 } { 34 }
\um_font_param:nnn { stack_num_up } { 33 } { 32 }
\um_font_param:nnn { stack_vgap } { 37 } { 36 }
\um_font_param:nn { sub_shift_down } { 18 }
\um_font_param:nn { sub_shift_drop } { 20 }
\um_font_param:n { subsup_shift_down }
\um_font_param:nn { sub_top_max } { 19 }
\um_font_param:nn { subsup_vgap } { 25 }
\um_font_param:nn { sup_bottom_min } { 23 }
\um_font_param:nn { sup_shift_drop } { 24 }
\um_font_param:nnnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
\um_font_param:nn { supsub_bottom_max } { 26 }
\um_font_param:nn { underbar_kern } { 58 }
\um_font_param:nn { underbar_rule } { 57 }
\um_font_param:nn { underbar_vgap } { 56 }
```

```
\um_font_param:n { connector_overlap_min }
```

11 Font features

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

```
\def\new@mathversion#1{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \@font@info{Redeclaring math version
               `\expandafter\@gobblefour\string#1'}%
  \else
    \expandafter\newcount\csname c@\expandafter
                                \@gobble\string#1\endcsname
    \def\version@elt{\noexpand\version@elt\noexpand}%
    \edef\version@list{\version@list\version@elt#1}%
  \fi
  \toks@{}%
  \count@\z@
  \def\group@elt##1##2{%
       \advance\count@\@ne
       \addto@hook\toks@{\getanddefine@fonts##1##2}%
       }%
  \group@list
  \global\csname c@\expandafter\@gobble\string#1\endcsname\count@
  \def\alpha@elt##1##2##3{%
       \ifx##2\no@alphabet@error
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\no@alphabet@error##1}}%
       \else
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\select@group##1##2##3}}%
       \fi
          }%
  \alpha@list
  \xdef#1{\the\toks@}%
}
```

11.1 Math version

```
\keys_define:nn {unicode-math}
  {
    version .code:n =
      {
        \tl_set:Nn \l_um_mversion_tl {#1}
        \DeclareMathVersion{\l_um_mversion_tl}
      }
  }
```

11.2 Script and scriptscript font options

```
\keys_define:nn {unicode-math}
{
  script-features  .tl_set:N =  \l_um_script_features_tl ,
  sscript-features .tl_set:N = \l_um_sscript_features_tl ,
       script-font .tl_set:N =      \l_um_script_font_tl ,
      sscript-font .tl_set:N =     \l_um_sscript_font_tl ,
}
```

11.3 Range processing

```
\seq_new:N \l_um_mathalph_seq
\seq_new:N \l_um_char_range_seq
\seq_new:N \l_um_mclass_range_seq
\seq_new:N \l_um_cmd_range_seq
\keys_define:nn {unicode-math} {
  range .code:n = {
    \bool_set_false:N \l_um_init_bool
```

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

```
    \int_incr:N \g_um_fam_int
    \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
    \cs_set_eq:NN \_um_sym:nnn \um_process_symbol_parse:nnn
    \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_parse:Nnn
    \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
    \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
    \cs_set_eq:NN \um_map_char_single:nn \um_map_char_parse:nn
    \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_parse:nn
    \cs_set_eq:NN \um_make_mathactive:nNN \um_make_mathactive_parse:nNN
```

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

```
    \seq_clear:N \l_um_char_range_seq
    \seq_clear:N \l_um_mclass_range_seq
    \seq_clear:N \l_um_cmd_range_seq
    \seq_clear:N \l_um_mathalph_seq
    \clist_map_inline:nn {#1} {
      \um_if_mathalph_decl:nTF {##1} {
        \seq_put_right:Nx \l_um_mathalph_seq {
          { \exp_not:V \l_um_tmpa_tl }
          { \exp_not:V \l_um_tmpb_tl }
          { \exp_not:V \l_um_tmpc_tl }
        }
      }{
```

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

```
        \seq_if_in:NnTF \g_um_mathclasses_seq {##1}
          { \seq_put_right:Nn \l_um_mclass_range_seq {##1} }
          {
```

```
          \bool_if:nTF { \tl_if_single_p:n {##1} && \token_if_cs_p:N ##1 }
            { \seq_put_right:Nn \l_um_cmd_range_seq {##1} }
            { \seq_put_right:Nn \l_um_char_range_seq {##1} }
          }
      }
    }
  }
}
\seq_new:N \g_um_mathclasses_seq
\seq_set_from_clist:Nn \g_um_mathclasses_seq
  {
    \mathord,\mathalpha,\mathop,\mathbin,\mathrel,
    \mathopen,\mathclose,\mathpunct,\mathaccent,
    \mathfence,\mathover,\mathunder,\mathbotaccent
  }
```

`\um_if_mathalph_decl:nTF`

Possible forms of input:
`\mathscr`
`\mathscr->\mathup`
`\mathscr/{Latin}`
`\mathscr/{Latin}->\mathup`
Outputs:
`tmpa`: math style (*e.g.*, `\mathscr`)
`tmpb`: alphabets (*e.g.*, `Latin`)
`tmpc`: remap style (*e.g.*, `\mathup`). Defaults to `tmpa`.

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

```
\prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
  \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {#1} }
  \tl_clear:N \l_um_tmpb_tl
  \tl_clear:N \l_um_tmpc_tl
  \tl_if_in:NnT \l_um_tmpa_tl {->} {
    \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
  }
  \tl_if_in:NnT \l_um_tmpa_tl {/} {
    \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
  }
 \tl_if_empty:NT \l_um_tmpc_tl { \tl_set_eq:NN \l_um_tmpc_tl \l_um_tmpa_tl }
  \seq_if_in:NVTF \g_um_mathstyles_seq \l_um_tmpa_tl {
    \prg_return_true:
  }{
    \prg_return_false:
  }
}
\cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
  \tl_set:Nn \l_um_tmpa_tl {#1}
  \tl_if_single:nTF {#2}
    { \tl_set:Nn \l_um_tmpc_tl {#2} }
    { \exp_args:NNc \tl_set:Nn \l_um_tmpc_tl {math#2} }
}
```

```
\cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
  \tl_set:Nn \l_um_tmpa_tl {#1}
  \tl_set:Nn \l_um_tmpb_tl {#2}
}
```

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

`\um_if_char_spec:nNNT`

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

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

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

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

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

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

```
\cs_new:Nn \um_if_char_spec:nNNT
  {

    % math class:
    \seq_if_in:NnT \l_um_mclass_range_seq {#3}
      { \use_none_delimit_by_q_nil:w }

    % command name:
    \seq_if_in:NnT \l_um_cmd_range_seq {#2}
      { \use_none_delimit_by_q_nil:w }

    % character slot:
    \seq_map_inline:Nn \l_um_char_range_seq
      {
        \um_int_if_slot_in_range:nnT {#1} {##1}
          { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
      }

    % this executes if no match was found:
    \use_none:nnn
    \q_nil
    \use:n
      {
        \clist_put_right:Nx \l_um_char_num_range_clist { \int_eval:n {#1} }
         #4
      }
  }
```

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

Four cases, four argument types:

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

```
\cs_new:Nn \um_int_if_slot_in_range:nnT
  { \um_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
\cs_set:Npn \um_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
  {
    \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
      {
    \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
      {
    \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
      {
    \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
      } } }
  }
```

11.4 Resolving Greek symbol name control sequences

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

```
\AtBeginDocument{\um_resolve_greek:}
\cs_new:Npn \um_resolve_greek: {
  \clist_map_inline:nn {
    Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
    alpha,beta,gamma,delta,        zeta,eta,theta,iota,kappa,lambda,
    Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
    mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,    chi,psi,omega,
    varTheta,
    varsigma,vartheta,varkappa,varrho,varpi
  }{
    \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
  }
  \tl_set:Nn \epsilon {
    \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
  }
  \tl_set:Nn \phi {
    \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
  }
  \tl_set:Nn \varepsilon {
    \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
  }
  \tl_set:Nn \varphi {
    \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
  }
}
```

12 Maths alphabets mapping definitions

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

Implicit mode:

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

Explicit mode:

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

12.1 Initialising math styles

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

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

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

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

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

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

```
\AtEndOfPackage{
\clist_map_inline:nn {
  {\mathup    } {latin,Latin,greek,Greek,num,misc} {\mathup    }  ,
  {\mathit    } {latin,Latin,greek,Greek,misc}     {\mathit    }  ,
  {\mathbb    } {latin,Latin,num,misc}             {\mathbb    }  ,
  {\mathbbit  } {misc}                             {\mathbbit  }  ,
  {\mathscr   } {latin,Latin}                      {\mathscr   }  ,
  {\mathcal   } {Latin}                            {\mathscr   }  ,
  {\mathbfcal } {Latin}                            {\mathbfscr }  ,
  {\mathfrak  } {latin,Latin}                      {\mathfrak  }  ,
  {\mathtt    } {latin,Latin,num}                  {\mathtt    }  ,
  {\mathsfup  } {latin,Latin,num}                  {\mathsfup  }  ,
  {\mathsfit  } {latin,Latin}                      {\mathsfit  }  ,
  {\mathbfup  } {latin,Latin,greek,Greek,num,misc} {\mathbfup  }  ,
  {\mathbfit  } {latin,Latin,greek,Greek,misc}     {\mathbfit  }  ,
  {\mathbfscr } {latin,Latin}                      {\mathbfscr }  ,
  {\mathbffrak} {latin,Latin}                      {\mathbffrak}  ,
  {\mathbfsfup} {latin,Latin,greek,Greek,num,misc} {\mathbfsfup}  ,
  {\mathbfsfit} {latin,Latin,greek,Greek,misc}     {\mathbfsfit}
}{
  \seq_put_right:Nn \g_um_default_mathalph_seq {#1}
  \exp_after:wN \um_new_mathstyle:N \use_i:nnn #1
}
```

These are 'false' mathstyles that inherit other definitions:

```
\um_new_mathstyle:N \mathsf
\um_new_mathstyle:N \mathbf
\um_new_mathstyle:N \mathbfsf
```

```
}
```

12.2 Defining the math style macros

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

\um_prepare_mathstyle:n #1 : math style name (e.g., it or bb)
Define the high level math alphabet macros (\mathit, etc.) in terms of unicode-math definitions. Use \bgroup/\egroup so s'scripts scan the whole thing.

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

```
\cs_new:Nn \um_prepare_mathstyle:n {
  \um_init_alphabet:x {#1}
  \cs_set:cpn {_um_math#1_aux:n} ##1 {
    \use:c {um_switchto_math#1:} ##1 \egroup
  }
  \cs_set_protected:cpx {math#1} {
    \exp_not:n{
      \bgroup
      \mode_if_math:F
        {
          \egroup\expandafter
          \non@alpherr\expandafter{\csname math#1\endcsname\space}
        }
      \tl_set:Nn \l_um_mathstyle_tl {#1}
    }
    \exp_not:c {_um_math#1_aux:n}
  }
}
\tl_new:N \l_um_mathstyle_tl
\cs_generate_variant:Nn \um_prepare_mathstyle:n {f}
```

`\um_init_alphabet:n`

#1 : math alphabet name (e.g., it or bb)
This macro initialises the macros used to set up a math alphabet. First used with the math alphabet macro is first defined, but then used later when redefining a particular maths alphabet.

```
\cs_set:Npn \um_init_alphabet:n #1 {
  \um_log:nx {alph-initialise} {#1}
  \cs_set_eq:cN {um_switchto_math#1:} \prg_do_nothing:
}
\cs_generate_variant:Nn \um_init_alphabet:n {x}
```

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

```
\cs_new:Npn \um_maybe_init_alphabet:V {
  \exp_args:NV \um_maybe_init_alphabet:n
}
```

12.3 Defining the math alphabets per style

Variables:

```
\seq_new:N \l_um_missing_alph_seq
```

`\um_setup_alphabets:`

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

```
\cs_new:Npn \um_setup_alphabets: {
```

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

```
  \seq_if_empty:NTF \l_um_mathalph_seq {
    \um_log:n {setup-implicit}
    \seq_set_eq:NN \l_um_mathalph_seq \g_um_default_mathalph_seq
    \bool_set_true:N \l_um_implicit_alph_bool
    \um_maybe_init_alphabet:n  {sf}
    \um_maybe_init_alphabet:n  {bf}
    \um_maybe_init_alphabet:n  {bfsf}
  }
```

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

```
  {
    \um_log:n {setup-explicit}
    \bool_set_false:N \l_um_implicit_alph_bool
    \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
    \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
  }
```

Now perform the mapping:

```
  \seq_map_inline:Nn \l_um_mathalph_seq {
    \tl_set:No \l_um_tmpa_tl { \use_i:nnn   ##1 }
    \tl_set:No \l_um_tmpb_tl { \use_ii:nnn  ##1 }
    \tl_set:No \l_um_remap_style_tl { \use_iii:nnn ##1 }
    \tl_set:Nx \l_um_remap_style_tl {
      \exp_after:wN \exp_after:wN \exp_after:wN \use_none:nnnnn
      \exp_after:wN \token_to_str:N \l_um_remap_style_tl
    }
    \tl_if_empty:NT \l_um_tmpb_tl {
      \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
      \tl_set:Nn \l_um_tmpb_tl { latin,Latin,greek,Greek,num,misc }
    }
    \um_setup_math_alphabet:VVV
      \l_um_tmpa_tl \l_um_tmpb_tl \l_um_remap_style_tl
  }
 \seq_if_empty:NF \l_um_missing_alph_seq { \um_log:n { missing-alphabets } }
}
```

`\um_setup_math_alphabet:Nnn`

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

```
\cs_new:Nn \um_setup_math_alphabet:Nnn {
  \tl_set:Nx \l_um_style_tl {
    \exp_after:wN \use_none:nnnnn \token_to_str:N #1
  }
```

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

```
  \clist_map_inline:nn {#2} {
    \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {##1} }
    \cs_if_exist:cT {um_config_ \l_um_style_tl _\l_um_tmpa_tl :n} {
      \str_if_eq_x:nnTF {\l_um_tmpa_tl}{misc} {
        \um_maybe_init_alphabet:V \l_um_style_tl
        \clist_map_break:
      }{
        \um_glyph_if_exist:cT { \um_to_usv:nn {#3}{\l_um_tmpa_tl} }{
          \um_maybe_init_alphabet:V \l_um_style_tl
          \clist_map_break:
        }
      }
    }
  }
```

…and then loop through them defining the individual ranges:

```
  \clist_map_inline:nn {#2} {
    \tl_set:Nx \l_um_tmpa_tl { \tl_trim_spaces:n {##1} }
    \cs_if_exist:cT {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {
      \str_if_eq_x:nnTF {\l_um_tmpa_tl}{misc} {
        \um_log:nx {setup-alph} {math \l_um_style_tl~(\l_um_tmpa_tl)}
        \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {#3}
      }{
        \um_glyph_if_exist:cTF { \um_to_usv:nn {#3}{\l_um_tmpa_tl} } {
          \um_log:nx {setup-alph} {math \l_um_style_tl~(\l_um_tmpa_tl)}
          \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {#3}
        }{
          \bool_if:NTF \l_um_implicit_alph_bool {
            \seq_put_right:Nx \l_um_missing_alph_seq {
              \@backslashchar math \l_um_style_tl \space
              (\tl_use:c{c_um_math_alphabet_name_ \l_um_tmpa_tl _tl})
            }
          }{
            \use:c {um_config_ \l_um_style_tl _ \l_um_tmpa_tl :n} {up}
          }
        }
      }
    }
  }
}
\cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {VVV}
```

12.4 Mapping 'naked' math characters

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

12.4.1 Functions

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

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

`\um_map_char_noparse:nn`
`\um_map_char_parse:nn`

```
\cs_new:Nn \um_map_char_noparse:nn {
  \um_set_mathcode:nnnn {#1}{\mathalpha}{\um_symfont_tl}{#2}
}
```

```
\cs_new:Nn \um_map_char_parse:nn {
  \um_if_char_spec:nNNT {#1} {\@nil} {\mathalpha} {
    \um_map_char_noparse:nn {#1}{#2}
  }
}
```

\um_map_single:nnn

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

```
\cs_new:Nn \um_map_char_single:nnn {
  \um_map_char_single:cc { \um_to_usv:nn {#1}{#3} }
                         { \um_to_usv:nn {#2}{#3} }
}
\cs_set:Npn \um_map_single:nnn #1#2#3 {
  \cs_if_exist:cT { \um_to_usv:nn {#3} {#1} }
  {
    \clist_map_inline:nn {#2} {
      \um_map_char_single:nnn {##1} {#3} {#1}
    }
  }
}
```

\um_map_chars_range:nnnn

#1 : Number of chars (26)
#2 : From style, one or more (it)
#3 : To style (up)
#4 : Alphabet name (Latin)

First the function with numbers:

```
\cs_set:Npn \um_map_chars_range:nnn #1#2#3 {
  \int_step_inline:nnnn {0}{1}{#1-1} {
    \um_map_char_single:nn {#2+##1}{#3+##1}
  }
}
\cs_generate_variant:Nn \um_map_chars_range:nnn {ncc}
```

And the wrapper with names:

```
\cs_new:Nn \um_map_chars_range:nnnn {
  \um_map_chars_range:ncc {#1} { \um_to_usv:nn {#2}{#4} }
                               { \um_to_usv:nn {#3}{#4} }
}
```

12.4.2 Functions for alphabets

```
\cs_new:Nn \um_map_chars_Latin:nn {
  \clist_map_inline:nn {#1} {
    \um_map_chars_range:nnnn {26} {##1} {#2} {Latin}
  }
}
```

```
\cs_new:Nn \um_map_chars_latin:nn {
  \clist_map_inline:nn {#1} {
    \um_map_chars_range:nnnn {26} {##1} {#2} {latin}
  }
}
```

```
\cs_new:Nn \um_map_chars_greek:nn {
  \clist_map_inline:nn {#1} {
    \um_map_chars_range:nnnn {25} {##1} {#2} {greek}
    \um_map_char_single:nnn {##1} {#2} {varepsilon}
    \um_map_char_single:nnn {##1} {#2} {vartheta}
    \um_map_char_single:nnn {##1} {#2} {varkappa}
    \um_map_char_single:nnn {##1} {#2} {varphi}
    \um_map_char_single:nnn {##1} {#2} {varrho}
    \um_map_char_single:nnn {##1} {#2} {varpi}
  }
}
```

```
\cs_new:Nn \um_map_chars_Greek:nn {
  \clist_map_inline:nn {#1} {
    \um_map_chars_range:nnnn {25} {##1} {#2} {Greek}
    \um_map_char_single:nnn {##1} {#2} {varTheta}
  }
}
```

```
\cs_new:Nn \um_map_chars_numbers:nn {
  \um_map_chars_range:nnnn {10} {#1} {#2} {num}
}
```

12.5 Mapping chars inside a math style

12.5.1 Functions for setting up the maths alphabets

\um_set_mathalphabet_char:Nnn

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

```
\cs_new:Npn \um_set_mathalphabet_char:Ncc {
  \exp_args:NNcc \um_set_mathalphabet_char:Nnn
}
```

\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.

```
\cs_new:Nn \um_mathmap_noparse:Nnn {
  \clist_map_inline:nn {#2} {
    \tl_put_right:cx {um_switchto_\cs_to_str:N #1:} {
      \um_set_mathcode:nnnn{##1}{\mathalpha}{\um_symfont_tl}{#3}
    }
  }
}
```

\um_mathmap_parse:Nnn

#1 : Maths alphabet, *e.g.*, \mathbb
#2 : Input slot(s), *e.g.*, the slot for 'A' (comma separated)
#3 : Output slot, *e.g.*, the slot for '𝔸'
When \um_if_char_spec:nNNT is executed, it populates the \l_um_char_num_-range_clist macro with slot numbers corresponding to the specified range. This range is used to conditionally add \um_set_mathcode:nnnn declaractions to the maths alphabet definition.

```
\cs_new:Nn \um_mathmap_parse:Nnn {
  \clist_if_in:NnT \l_um_char_num_range_clist {#3} {
    \um_mathmap_noparse:Nnn {#1}{#2}{#3}
  }
}
```

um_set_mathalphabet_char:Nnnn

#1 : math style command
#2 : input math alphabet name
#3 : output math alphabet name
#4 : char name to map

```
\cs_new:Npn \um_set_mathalphabet_char:Nnnn #1#2#3#4 {
  \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {#2} {#4} }
                                   { \um_to_usv:nn {#3} {#4} }
}
```

\um_set_mathalph_range:nNnn

#1 : Number of iterations
#2 : Maths alphabet
#3 : Starting input char (single)
#4 : Starting output char

Loops through character ranges setting \mathcode. First the version that uses numbers:

```
\cs_new:Npn \um_set_mathalph_range:nNnn #1#2#3#4 {
  \int_step_inline:nnnn {0}{1}{#1-1}
    { \um_set_mathalphabet_char:Nnn {#2} { ##1 + #3 } { ##1 + #4 } }
}
\cs_generate_variant:Nn \um_set_mathalph_range:nNnn {nNcc}
```

Then the wrapper version that uses names:

```
\cs_new:Npn \um_set_mathalph_range:nNnnn #1#2#3#4#5 {
  \um_set_mathalph_range:nNcc {#1} #2 { \um_to_usv:nn {#3} {#5} }
                                      { \um_to_usv:nn {#4} {#5} }
}
```

12.5.2 Individual mapping functions for different alphabets

```
\cs_new:Npn \um_set_mathalphabet_pos:Nnnn #1#2#3#4 {
  \cs_if_exist:cT { \um_to_usv:nn {#4}{#2} } {
    \clist_map_inline:nn {#3}
      { \um_set_mathalphabet_char:Nnnn #1 {##1} {#4} {#2} }
  }
}
```

```
\cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
  \clist_map_inline:nn {#2}
    { \um_set_mathalph_range:nNnnn {10} #1  {##1} {#3} {num} }
}
```

```
\cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
  \clist_map_inline:nn {#2}
    { \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {Latin} }
}
```

```
\cs_new:Nn \um_set_mathalphabet_latin:Nnn {
  \clist_map_inline:nn {#2} {
    \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {latin}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {h}
  }
}
```

```
\cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
  \clist_map_inline:nn {#2} {
    \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {Greek}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varTheta}
  }
}
```

```
\cs_new:Nn \um_set_mathalphabet_greek:Nnn {
  \clist_map_inline:nn {#2} {
    \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {greek}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varepsilon}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {vartheta}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varkappa}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varphi}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varrho}
    \um_set_mathalphabet_char:Nnnn    #1 {##1} {#3} {varpi}
  }
}
```

12.6 Alphabets

12.6.1 Upright: `\mathup`

```
\cs_new:Nn \um_config_up_num:n {
  \um_map_chars_numbers:nn {up}{#1}
  \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}
}
\cs_new:Nn \um_config_up_Latin:n
  {
    \bool_if:NTF \g_um_literal_bool { \um_map_chars_Latin:nn {up} {#1} }
  {
    \bool_if:NT \g_um_upLatin_bool { \um_map_chars_Latin:nn {up,it} {#1} }
  }
  \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
}
\cs_new:Nn \um_config_up_latin:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_latin:nn {up} {#1} }
  {
    \bool_if:NT \g_um_uplatin_bool {
      \um_map_chars_latin:nn        {up,it} {#1}
      \um_map_single:nnn        {h} {up,it} {#1}
      \um_map_single:nnn {dotlessi} {up,it} {#1}
      \um_map_single:nnn {dotlessj} {up,it} {#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
}
\cs_new:Nn \um_config_up_Greek:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_Greek:nn {up}{#1} }
  {
    \bool_if:NT \g_um_upGreek_bool { \um_map_chars_Greek:nn {up,it}{#1} }
  }
  \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
}
\cs_new:Nn \um_config_up_greek:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_greek:nn {up} {#1} }
  {
    \bool_if:NT \g_um_upgreek_bool {
      \um_map_chars_greek:nn {up,it} {#1}
    }
  }
  \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
}
\cs_new:Nn \um_config_up_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{up}{up}
  }{
    \bool_if:NT \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{up,it}{up}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{up}{up}
  }{
    \bool_if:NT \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{up,it}{up}
    }
  }
  \um_set_mathalphabet_pos:Nnnn \mathup  {partial} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathup    {Nabla} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathup {dotlessi} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathup {dotlessj} {up,it} {#1}
}
```

12.6.2 Italic: `\mathit`

```
\cs_new:Nn \um_config_it_Latin:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_Latin:nn {it} {#1} }
  {
    \bool_if:NF \g_um_upLatin_bool { \um_map_chars_Latin:nn {up,it} {#1} }
  }
  \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
}
\cs_new:Nn \um_config_it_latin:n {
  \bool_if:NTF \g_um_literal_bool {
    \um_map_chars_latin:nn {it} {#1}
    \um_map_single:nnn {h}{it}{#1}
  }{
    \bool_if:NF \g_um_uplatin_bool {
      \um_map_chars_latin:nn {up,it} {#1}
      \um_map_single:nnn {h}{up,it}{#1}
      \um_map_single:nnn {dotlessi}{up,it}{#1}
      \um_map_single:nnn {dotlessj}{up,it}{#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathit            {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathit {dotlessi} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathit {dotlessj} {up,it} {#1}
}
\cs_new:Nn \um_config_it_Greek:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_Greek:nn {it}{#1}
  }{
    \bool_if:NF \g_um_upGreek_bool { \um_map_chars_Greek:nn {up,it}{#1} }
  }
  \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
}
\cs_new:Nn \um_config_it_greek:n {
  \bool_if:NTF \g_um_literal_bool { \um_map_chars_greek:nn {it} {#1} }
  {
    \bool_if:NF \g_um_upgreek_bool { \um_map_chars_greek:nn {it,up} {#1} }
  }
  \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
}
\cs_new:Nn \um_config_it_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{it}{it}
  }{
    \bool_if:NF \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{up,it}{it}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{it}{it}
  }{
    \bool_if:NF \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{up,it}{it}
    }
  }
  \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn \mathit {Nabla}   {up,it}{#1}
}
```

12.6.3 Blackboard or double-struck: \mathbb and \mathbbit

```
\cs_new:Nn \um_config_bb_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
}
\cs_new:Nn \um_config_bb_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {C} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {H} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {N} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {P} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {Q} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {R} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbb {Z} {up,it} {#1}
}
\cs_new:Nn \um_config_bb_num:n {
  \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
}
\cs_new:Nn \um_config_bb_misc:n {
  \um_set_mathalphabet_pos:Nnnn \mathbb        {Pi} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbb        {pi} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbb     {Gamma} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbb     {gamma} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbb {summation} {up} {#1}
}
\cs_new:Nn \um_config_bbit_misc:n {
  \um_set_mathalphabet_pos:Nnnn \mathbbit {D} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbbit {d} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbbit {e} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbbit {i} {up,it} {#1}
  \um_set_mathalphabet_pos:Nnnn \mathbbit {j} {up,it} {#1}
}
```

12.6.4 Script and caligraphic: \mathscr and \mathcal

```
\cs_new:Nn \um_config_scr_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathscr    {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {B}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {E}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {F}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {H}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {I}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {L}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {M}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {R}{up,it}{#1}
}
\cs_new:Nn \um_config_scr_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathscr    {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {e}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {g}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathscr {o}{up,it}{#1}
}
```

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

```
\cs_new:Nn \um_config_cal_Latin:n {
  \um_set_mathalphabet_Latin:Nnn  \mathcal  {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {B}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {E}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {F}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {H}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {I}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {L}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {M}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn   \mathcal {R}{up,it}{#1}
}
```

12.6.5 Fractur or fraktur or blackletter: \mathfrak

```
\cs_new:Nn \um_config_frak_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathfrak    {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathfrak {C}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathfrak {H}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathfrak {I}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathfrak {R}{up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathfrak {Z}{up,it}{#1}
}
\cs_new:Nn \um_config_frak_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{#1}
}
```

12.6.6 Sans serif upright: \mathsfup

```
\cs_new:Nn \um_config_sfup_num:n {
  \um_set_mathalphabet_numbers:Nnn \mathsf   {up}{#1}
  \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
}
\cs_new:Nn \um_config_sfup_Latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Latin:nn {sfup} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_Latin:nn {sfup,sfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}
}
\cs_new:Nn \um_config_sfup_latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_latin:nn {sfup} {#1}
    \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_latin:nn {sfup,sfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
}
```

12.6.7 Sans serif italic: \mathsfit

```
\cs_new:Nn \um_config_sfit_Latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Latin:nn {sfit} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_Latin:nn {sfup,sfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{#1}
}
\cs_new:Nn \um_config_sfit_latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_latin:nn {sfit} {#1}
    \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_latin:nn {sfup,sfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathsfit {up,it}{#1}
}
```

12.6.8 Typewriter or monospaced: `\mathtt`

```
\cs_new:Nn \um_config_tt_num:n {
  \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
}
\cs_new:Nn \um_config_tt_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
}
\cs_new:Nn \um_config_tt_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
}
```

12.6.9 Bold Italic: `\mathbfit`

```
\cs_new:Nn \um_config_bfit_Latin:n {
  \bool_if:NF \g_um_bfupLatin_bool {
    \um_map_chars_Latin:nn {bfup,bfit} {#1}
  }
  \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_Latin:nn {bfit} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
  }{
    \bool_if:NF \g_um_bfupLatin_bool {
      \um_map_chars_Latin:nn {bfup,bfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
    }
  }
```

```
}
\cs_new:Nn \um_config_bfit_latin:n {
  \bool_if:NF \g_um_bfuplatin_bool {
    \um_map_chars_latin:nn {bfup,bfit} {#1}
  }
  \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_latin:nn {bfit} {#1}
    \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
  }{
    \bool_if:NF \g_um_bfuplatin_bool {
      \um_map_chars_latin:nn {bfup,bfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
    }
  }
}
\cs_new:Nn \um_config_bfit_Greek:n {
  \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_Greek:nn {bfit}{#1}
    \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
  }{
    \bool_if:NF \g_um_bfupGreek_bool {
      \um_map_chars_Greek:nn {bfup,bfit}{#1}
      \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
    }
  }
}
\cs_new:Nn \um_config_bfit_greek:n {
  \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_greek:nn {bfit} {#1}
    \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
  }{
    \bool_if:NF \g_um_bfupgreek_bool {
      \um_map_chars_greek:nn {bfit,bfup} {#1}
      \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
    }
  }
}
\cs_new:Nn \um_config_bfit_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{bfit}{#1}
  }{
    \bool_if:NF \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{bfit}{#1}
  }{
    \bool_if:NF \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{bfup,bfit}{#1}
    }
  }
  \um_set_mathalphabet_pos:Nnnn  \mathbfit {partial} {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfit {Nabla}   {up,it}{#1}
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {it}{#1}
  }{
    \bool_if:NF \g_um_uppartial_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up,it}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {it}{#1}
  }{
    \bool_if:NF \g_um_upNabla_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up,it}{#1}
    }
  }
}
```

12.6.10 Bold Upright: `\mathbfup`

```
\cs_new:Nn \um_config_bfup_num:n {
  \um_set_mathalphabet_numbers:Nnn \mathbf   {up}{#1}
  \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
}
\cs_new:Nn \um_config_bfup_Latin:n {
  \bool_if:NT \g_um_bfupLatin_bool {
    \um_map_chars_Latin:nn {bfup,bfit} {#1}
  }
  \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_Latin:nn {bfup} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
  }{
    \bool_if:NT \g_um_bfupLatin_bool {
      \um_map_chars_Latin:nn {bfup,bfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
    }
  }
}
\cs_new:Nn \um_config_bfup_latin:n {
  \bool_if:NT \g_um_bfuplatin_bool {
    \um_map_chars_latin:nn {bfup,bfit} {#1}
  }
  \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_latin:nn {bfup} {#1}
    \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
  }{
    \bool_if:NT \g_um_bfuplatin_bool {
      \um_map_chars_latin:nn {bfup,bfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
    }
  }
}
\cs_new:Nn \um_config_bfup_Greek:n {
  \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_Greek:nn {bfup}{#1}
    \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}
  }{
    \bool_if:NT \g_um_bfupGreek_bool {
      \um_map_chars_Greek:nn {bfup,bfit}{#1}
      \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
    }
  }
}
\cs_new:Nn \um_config_bfup_greek:n {
  \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
  \bool_if:NTF \g_um_bfliteral_bool {
    \um_map_chars_greek:nn {bfup} {#1}
    \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
  }{
    \bool_if:NT \g_um_bfupgreek_bool {
      \um_map_chars_greek:nn {bfup,bfit} {#1}
      \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
    }
  }
}
\cs_new:Nn \um_config_bfup_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{bfup}{#1}
  }{
    \bool_if:NT \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{bfup}{#1}
  }{
    \bool_if:NT \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{bfup,bfit}{#1}
    }
  }
  \um_set_mathalphabet_pos:Nnnn  \mathbfup {partial} {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfup {Nabla}   {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfup {digamma} {up}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfup {Digamma} {up}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbf   {digamma} {up}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbf   {Digamma} {up}{#1}
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up}{#1}
  }{
    \bool_if:NT \g_um_uppartial_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up,it}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up}{#1}
  }{
    \bool_if:NT \g_um_upNabla_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up,it}{#1}
    }
  }
}
```

12.6.11 Bold fractur or fraktur or blackletter: \mathbffrak

```
\cs_new:Nn \um_config_bffrak_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
}
\cs_new:Nn \um_config_bffrak_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
}
```

12.6.12 Bold script or calligraphic: \mathbfscr

```
\cs_new:Nn \um_config_bfscr_Latin:n {
  \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{#1}
}
\cs_new:Nn \um_config_bfscr_latin:n {
  \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{#1}
}
\cs_new:Nn \um_config_bfcal_Latin:n {
  \um_set_mathalphabet_Latin:Nnn   \mathbfcal  {up,it}{#1}
}
```

12.6.13 Bold upright sans serif: \mathbfsfup

```
\cs_new:Nn \um_config_bfsfup_num:n {
  \um_set_mathalphabet_numbers:Nnn \mathbfsf   {up}{#1}
  \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
}
\cs_new:Nn \um_config_bfsfup_Latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Latin:nn {bfsfup} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfup_latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_latin:nn {bfsfup} {#1}
    \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfup_Greek:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Greek:nn {bfsfup}{#1}
    \um_set_mathalphabet_Greek:Nnn \mathbfsf {up}{#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
      \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Greek:Nnn \mathbfsfup {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfup_greek:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_greek:nn {bfsfup} {#1}
    \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
  }{
    \bool_if:NT \g_um_upsans_bool {
      \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
    }
  }
  \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
}
\cs_new:Nn \um_config_bfsfup_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{bfsfup}{#1}
  }{
    \bool_if:NT \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{bfsfup}{#1}
  }{
    \bool_if:NT \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
    }
  }
  \um_set_mathalphabet_pos:Nnnn  \mathbfsfup {partial} {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfsfup {Nabla}   {up,it}{#1}
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up}{#1}
  }{
    \bool_if:NT \g_um_uppartial_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up,it}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up}{#1}
  }{
    \bool_if:NT \g_um_upNabla_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up,it}{#1}
    }
  }
}
```

12.6.14 Bold italic sans serif: \mathbfsfit

```
\cs_new:Nn \um_config_bfsfit_Latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Latin:nn {bfsfit} {#1}
    \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfit_latin:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_latin:nn {bfsfit} {#1}
    \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfit_Greek:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_Greek:nn {bfsfit}{#1}
    \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
      \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
    }
  }
  \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{#1}
}
\cs_new:Nn \um_config_bfsfit_greek:n {
  \bool_if:NTF \g_um_sfliteral_bool {
    \um_map_chars_greek:nn {bfsfit} {#1}
    \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
  }{
    \bool_if:NF \g_um_upsans_bool {
      \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
      \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
    }
  }
  \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
}
\cs_new:Nn \um_config_bfsfit_misc:n {
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_map_single:nnn {Nabla}{bfsfit}{#1}
  }{
    \bool_if:NF \g_um_upNabla_bool {
      \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_map_single:nnn {partial}{bfsfit}{#1}
  }{
    \bool_if:NF \g_um_uppartial_bool {
      \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
    }
  }
  \um_set_mathalphabet_pos:Nnnn  \mathbfsfit {partial} {up,it}{#1}
  \um_set_mathalphabet_pos:Nnnn  \mathbfsfit {Nabla}   {up,it}{#1}
  \bool_if:NTF \g_um_literal_partial_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {it}{#1}
  }{
    \bool_if:NF \g_um_uppartial_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up,it}{#1}
    }
  }
  \bool_if:NTF \g_um_literal_Nabla_bool {
    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {it}{#1}
  }{
    \bool_if:NF \g_um_upNabla_bool {
      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up,it}{#1}
    }
  }
}
```

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

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

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

```
\cs_new:Npn \um_symbol_setup: {
  \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4 {
    \exp_not:n {\_um_sym:nnn{##1}{##2}{##3}}
  }
}
```

```
\CatchFileEdef \g_um_mathtable_tl {unicode-math-table.tex} {\um_symbol_setup:}
```

`\um_input_math_symbol_table:` This function simply expands to the token list containing all the data.

```
\cs_new:Nn \um_input_math_symbol_table: {\g_um_mathtable_tl}
```

14 Definitions of the active math characters

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

`\um_cs_set_eq_active_char:Nw`
`\um_active_char_set:wc`

We need to do some trickery to transform the `\_um_sym:nnn` argument `"ABCDEF` into the XƎTEX '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.

```
\group_begin:
  \char_set_catcode_other:N \^
  \cs_gset:Npn \um_cs_set_eq_active_char:Nw #1 = "#2 \q_nil {
    \tex_lowercase:D {
      \tl_rescan:nn {
        \ExplSyntaxOn
        \char_set_catcode_other:N \{
        \char_set_catcode_other:N \}
        \char_set_catcode_other:N \&
        \char_set_catcode_other:N \%
        \char_set_catcode_other:N \$
      }{
        \cs_gset_eq:NN #1 ^^^^^#2
      }
    }
  }
```

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

```
  \cs_new:Npn \um_active_char_set:wc "#1 \q_nil #2 {
    \tex_lowercase:D {
      \tl_rescan:nn { \ExplSyntaxOn }
        { \cs_gset_protected_nopar:Npx ^^^^^#1 { \exp_not:c {#2} } }
    }
  }
\group_end:
```

Now give _um_sym:nnn a definition in terms of \um_cs_set_eq_active_-char:Nw and we're good to go.

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

```
\AtBeginDocument{\um_define_math_chars:}
\cs_new:Nn \um_define_math_chars: {
  \group_begin:
    \char_set_catcode_math_superscript:N \^
    \cs_set:Npn \_um_sym:nnn ##1##2##3 {
      \bool_if:nF { \cs_if_eq_p:NN ##3 \mathaccent ||
                    \cs_if_eq_p:NN ##3 \mathopen   ||
                    \cs_if_eq_p:NN ##3 \mathclose  ||
                    \cs_if_eq_p:NN ##3 \mathover   ||
                    \cs_if_eq_p:NN ##3 \mathunder  ||
                    \cs_if_eq_p:NN ##3 \mathbotaccent } {
        \um_cs_set_eq_active_char:Nw ##2 = ##1 \q_nil \ignorespaces
      }
    }
    \char_set_catcode_other:N \#
    \um_input_math_symbol_table:
  \group_end:
}
```

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

```
\group_begin:
  \lccode`\*=`\\
  \char_set_catcode_escape:N \|
  \char_set_catcode_other:N \\
  |lowercase{
    |AtBeginDocument{
      |let|backslash=*
    }
  }
|group_end:
```

Fix \backslash:

15 Fall-back font

Want to load Latin Modern Math if nothing else.

```
\AtBeginDocument { \um_load_lm_if_necessary: }
\cs_new:Nn \um_load_lm_if_necessary:
  {
    \cs_if_exist:NF \l_um_fontname_tl
      {
        % XXX: update this when lmmath-bold.otf is released
       \setmathfont[BoldFont={latinmodern-math.otf}]{latinmodern-math.otf}
      }
  }
```

16 Epilogue

Lots of little things to tidy up.

16.1 Primes

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

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

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

> $x′ \quad x″ \quad x‴ \quad x⁗$

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

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

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

Our algorithm is:

- Prime encountered; pcount=1.
- Scan ahead; if prime: pcount:=pcount+1; repeat.
- If not prime, stop scanning.
- If pcount=1, `\prime`, end.
- If pcount=2, check `\dprime`; if it exists, use it, end; if not, goto last step.
- Ditto pcount=3 & `\trprime`.

- Ditto pcount=4 & `\qprime`.
- If pcount>4 or the glyph doesn't exist, insert pcount `\primes` with `\primekern` between each.

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

```
\cs_new:Nn \um_arg_i_before_egroup:n {#1\egroup}
\cs_new:Nn \um_superscript:n {
  ^\bgroup #1
  \peek_meaning_remove:NTF ^ \um_arg_i_before_egroup:n \egroup
}
```

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

```
\cs_new:Nn \um_nprimes:Nn {
  \um_superscript:n {
    #1
    \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
  }
}
\cs_new:Nn \um_nprimes_select:nn {
  \int_case:nnn {#2}{
    {1} { \um_superscript:n {#1} }
    {2} {
      \um_glyph_if_exist:nTF {"2033}
        { \um_superscript:n {\um_prime_double_mchar} }
        { \um_nprimes:Nn #1 {#2} }
    }
    {3} {
      \um_glyph_if_exist:nTF {"2034}
        { \um_superscript:n {\um_prime_triple_mchar} }
        { \um_nprimes:Nn #1 {#2} }
    }
    {4} {
      \um_glyph_if_exist:nTF {"2057}
        { \um_superscript:n {\um_prime_quad_mchar} }
        { \um_nprimes:Nn #1 {#2} }
    }
  }{
    \um_nprimes:Nn #1 {#2}
  }
}
\cs_new:Nn \um_nbackprimes_select:nn {
  \int_case:nnn {#2}{
    {1} { \um_superscript:n {#1} }
    {2} {
      \um_glyph_if_exist:nTF {"2036}
        { \um_superscript:n {\um_backprime_double_mchar} }
        { \um_nprimes:Nn #1 {#2} }
    }
    {3} {
      \um_glyph_if_exist:nTF {"2037}
        { \um_superscript:n {\um_backprime_triple_mchar} }
        { \um_nprimes:Nn #1 {#2} }
    }
  }{
    \um_nprimes:Nn #1 {#2}
  }
}
```

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

```
\cs_new:Npn \um_scan_prime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_zero:N \l_um_primecount_int
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_dprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_set:Nn \l_um_primecount_int {1}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_trprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_set:Nn \l_um_primecount_int {2}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_qprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_set:Nn \l_um_primecount_int {3}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_sup_prime: {
  \int_zero:N \l_um_primecount_int
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_sup_dprime: {
  \int_set:Nn \l_um_primecount_int {1}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_sup_trprime: {
  \int_set:Nn \l_um_primecount_int {2}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Npn \um_scan_sup_qprime: {
  \int_set:Nn \l_um_primecount_int {3}
  \um_scanprime_collect:N \um_prime_single_mchar
}
\cs_new:Nn \um_scanprime_collect:N {
  \int_incr:N \l_um_primecount_int
  \peek_meaning_remove:NTF ' {
    \um_scanprime_collect:N #1
  }{
    \peek_meaning_remove:NTF \um_scan_prime: {
      \um_scanprime_collect:N #1
    }{
      \peek_meaning_remove:NTF ^^^^2032 {
        \um_scanprime_collect:N #1
      }{
        \peek_meaning_remove:NTF \um_scan_dprime: {
          \int_incr:N \l_um_primecount_int
          \um_scanprime_collect:N #1
        }{
          \peek_meaning_remove:NTF ^^^^2033 {
            \int_incr:N \l_um_primecount_int
            \um_scanprime_collect:N #1
          }{
            \peek_meaning_remove:NTF \um_scan_trprime: {
              \int_add:Nn \l_um_primecount_int {2}
              \um_scanprime_collect:N #1
            }{
              \peek_meaning_remove:NTF ^^^^2034 {
                \int_add:Nn \l_um_primecount_int {2}
                \um_scanprime_collect:N #1
              }{
                \peek_meaning_remove:NTF \um_scan_qprime: {
                  \int_add:Nn \l_um_primecount_int {3}
                  \um_scanprime_collect:N #1
                }{
                  \peek_meaning_remove:NTF ^^^^2057 {
                    \int_add:Nn \l_um_primecount_int {3}
                    \um_scanprime_collect:N #1
                  }{
                    \um_nprimes_select:nn {#1} {\l_um_primecount_int}
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
\cs_new:Npn \um_scan_backprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_zero:N \l_um_primecount_int
  \um_scanbackprime_collect:N \um_backprime_single_mchar
}
\cs_new:Npn \um_scan_backdprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_set:Nn \l_um_primecount_int {1}
  \um_scanbackprime_collect:N \um_backprime_single_mchar
```

```
}
\cs_new:Npn \um_scan_backtrprime: {
  \cs_set_eq:NN \um_superscript:n \use:n
  \int_set:Nn \l_um_primecount_int {2}
  \um_scanbackprime_collect:N \um_backprime_single_mchar
}
\cs_new:Npn \um_scan_sup_backprime: {
  \int_zero:N \l_um_primecount_int
  \um_scanbackprime_collect:N \um_backprime_single_mchar
}
\cs_new:Npn \um_scan_sup_backdprime: {
  \int_set:Nn \l_um_primecount_int {1}
  \um_scanbackprime_collect:N \um_backprime_single_mchar
}
\cs_new:Npn \um_scan_sup_backtrprime: {
  \int_set:Nn \l_um_primecount_int {2}
  \um_scanbackprime_collect:N \um_backprime_single_mchar
}
\cs_new:Nn \um_scanbackprime_collect:N {
  \int_incr:N \l_um_primecount_int
  \peek_meaning_remove:NTF ` {
    \um_scanbackprime_collect:N #1
  }{
    \peek_meaning_remove:NTF \um_scan_backprime: {
      \um_scanbackprime_collect:N #1
    }{
      \peek_meaning_remove:NTF ^^^^2035 {
        \um_scanbackprime_collect:N #1
      }{
        \peek_meaning_remove:NTF \um_scan_backdprime: {
          \int_incr:N \l_um_primecount_int
          \um_scanbackprime_collect:N #1
        }{
          \peek_meaning_remove:NTF ^^^^2036 {
            \int_incr:N \l_um_primecount_int
            \um_scanbackprime_collect:N #1
          }{
            \peek_meaning_remove:NTF \um_scan_backtrprime: {
              \int_add:Nn \l_um_primecount_int {2}
              \um_scanbackprime_collect:N #1
            }{
              \peek_meaning_remove:NTF ^^^^2037 {
                \int_add:Nn \l_um_primecount_int {2}
                \um_scanbackprime_collect:N #1
              }{
                \um_nbackprimes_select:nn {#1} {\l_um_primecount_int}
              }
            }
          }
        }
      }
    }
  }
}
\AtBeginDocument{\um_define_prime_commands: \um_define_prime_chars:}
\cs_new:Nn \um_define_prime_commands: {
  \cs_set_eq:NN \prime       \um_prime_single_mchar
  \cs_set_eq:NN \dprime      \um_prime_double_mchar
  \cs_set_eq:NN \trprime     \um_prime_triple_mchar
  \cs_set_eq:NN \qprime      \um_prime_quad_mchar
  \cs_set_eq:NN \backprime   \um_backprime_single_mchar
  \cs_set_eq:NN \backdprime  \um_backprime_double_mchar
  \cs_set_eq:NN \backtrprime \um_backprime_triple_mchar
}
\group_begin:
  \char_set_catcode_active:N \'
  \char_set_catcode_active:N \`
  \char_set_catcode_active:n {"2032}
  \char_set_catcode_active:n {"2033}
  \char_set_catcode_active:n {"2034}
  \char_set_catcode_active:n {"2057}
  \char_set_catcode_active:n {"2035}
  \char_set_catcode_active:n {"2036}
  \char_set_catcode_active:n {"2037}
  \cs_gset:Nn \um_define_prime_chars: {
    \cs_set_eq:NN '        \um_scan_sup_prime:
    \cs_set_eq:NN ^^^^2032 \um_scan_sup_prime:
    \cs_set_eq:NN ^^^^2033 \um_scan_sup_dprime:
    \cs_set_eq:NN ^^^^2034 \um_scan_sup_trprime:
    \cs_set_eq:NN ^^^^2057 \um_scan_sup_qprime:
    \cs_set_eq:NN `        \um_scan_sup_backprime:
    \cs_set_eq:NN ^^^^2035 \um_scan_sup_backprime:
    \cs_set_eq:NN ^^^^2036 \um_scan_sup_backdprime:
    \cs_set_eq:NN ^^^^2037 \um_scan_sup_backtrprime:
  }
\group_end:
```

16.2 Unicode radicals

```
\AtBeginDocument{\um_redefine_radical:}
\cs_new:Nn \um_redefine_radical:
⟨*XE⟩
 {
  \@ifpackageloaded { amsmath } { } {
```

`\r@@t`

#1 : A mathstyle (for `\mathpalette`)
#2 : Leading superscript for the sqrt sign
A re-implementation of LaTeX's hard-coded n-root sign using the appropriate `\fontdimens`.

```
    \cs_set_nopar:Npn \r@@t ##1 ##2 {
      \hbox_set:Nn \l_tmpa_box {
        \c_math_toggle_token
        \m@th
        ##1
        \sqrtsign { ##2 }
        \c_math_toggle_token
      }
      \um_mathstyle_scale:Nnn ##1 { \kern } {
        \fontdimen 63 \l_um_font
      }
      \box_move_up:nn {
        (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
        * \number \fontdimen 65 \l_um_font / 100
      } {
        \box_use:N \rootbox
      }
      \um_mathstyle_scale:Nnn ##1 { \kern } {
        \fontdimen 64 \l_um_font
      }
      \box_use_clear:N \l_tmpa_box
    }

  }
}
⟨/XE⟩
⟨*LU⟩
{
  \@ifpackageloaded { amsmath } { } {
```

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

```
      \cs_set:Npn \root ##1 \of ##2 {
        \luatexUroot \l_um_radical_sqrt_tl { ##1 } { ##2 }
      }

    }
  }
⟨/LU⟩
```

\um_fontdimen_to_percent:nn
\um_fontdimen_to_scale: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. \um_fontdimen_to_percent:nn takes a font dimension number and outputs the decimal value of the associated parameter. \um_fontdimen_to_scale:nn returns a dimension correspond to the current font size relative proportion based on that percentage.

```
\cs_new:Nn \um_fontdimen_to_percent:nn {
  \strip@pt\dimexpr\fontdimen#1#2*65536/100\relax
}
\cs_new:Nn \um_fontdimen_to_scale:nn
  {
    \um_fontdimen_to_percent:nn {#1} {#2} \dimexpr \f@size pt\relax
  }
```

\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.

```
\cs_new:Nn \um_mathstyle_scale:Nnn {
  \ifx#1\scriptstyle
    #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
  \else
    \ifx#1\scriptscriptstyle
      #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
    \else
      #2#3
    \fi
  \fi
}
```

16.3 Unicode sub- and super-scripts

The idea here is to enter a scanning state after a superscript or subscript is encountered. If subsequent superscripts or subscripts (resp.) are found, they are lumped together. Each sub/super has a corresponding regular size glyph which is used by 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?

```
\prop_new:N \g_um_supers_prop
\prop_new:N \g_um_subs_prop
\group_begin:
```

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

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

```
\cs_new:Nn \um_setup_active_superscript:nn {
  \prop_gput:Nxn \g_um_supers_prop   {\meaning #1} {#2}
  \char_set_catcode_active:N #1
  \char_gmake_mathactive:N #1
  \scantokens{
    \cs_gset:Npn #1 {
      \tl_set:Nn \l_um_ss_chain_tl {#2}
      \cs_set_eq:NN \um_sub_or_super:n \sp
      \tl_set:Nn \l_um_tmpa_tl {supers}
      \um_scan_sscript:
    }
  }
}
```

Bam:

```
\um_setup_active_superscript:nn {^^^^2070} {0}
\um_setup_active_superscript:nn {^^^^00b9} {1}
\um_setup_active_superscript:nn {^^^^00b2} {2}
\um_setup_active_superscript:nn {^^^^00b3} {3}
\um_setup_active_superscript:nn {^^^^2074} {4}
\um_setup_active_superscript:nn {^^^^2075} {5}
\um_setup_active_superscript:nn {^^^^2076} {6}
\um_setup_active_superscript:nn {^^^^2077} {7}
\um_setup_active_superscript:nn {^^^^2078} {8}
\um_setup_active_superscript:nn {^^^^2079} {9}
\um_setup_active_superscript:nn {^^^^207a} {+}
\um_setup_active_superscript:nn {^^^^207b} {-}
\um_setup_active_superscript:nn {^^^^207c} {=}
\um_setup_active_superscript:nn {^^^^207d} {(}
\um_setup_active_superscript:nn {^^^^207e} {)}
\um_setup_active_superscript:nn {^^^^2071} {i}
\um_setup_active_superscript:nn {^^^^207f} {n}
```

Subscripts Ditto above.

```
\cs_new:Nn \um_setup_active_subscript:nn {
  \prop_gput:Nxn \g_um_subs_prop   {\meaning #1} {#2}
  \char_set_catcode_active:N #1
  \char_gmake_mathactive:N #1
  \scantokens{
    \cs_gset:Npn #1 {
      \tl_set:Nn \l_um_ss_chain_tl {#2}
      \cs_set_eq:NN \um_sub_or_super:n \sb
      \tl_set:Nn \l_um_tmpa_tl {subs}
      \um_scan_sscript:
    }
  }
}
```

A few more subscripts than superscripts:

```
\um_setup_active_subscript:nn {^^^^2080} {0}
\um_setup_active_subscript:nn {^^^^2081} {1}
\um_setup_active_subscript:nn {^^^^2082} {2}
\um_setup_active_subscript:nn {^^^^2083} {3}
\um_setup_active_subscript:nn {^^^^2084} {4}
\um_setup_active_subscript:nn {^^^^2085} {5}
\um_setup_active_subscript:nn {^^^^2086} {6}
\um_setup_active_subscript:nn {^^^^2087} {7}
\um_setup_active_subscript:nn {^^^^2088} {8}
\um_setup_active_subscript:nn {^^^^2089} {9}
\um_setup_active_subscript:nn {^^^^208a} {+}
\um_setup_active_subscript:nn {^^^^208b} {-}
\um_setup_active_subscript:nn {^^^^208c} {=}
```

```
\um_setup_active_subscript:nn {^^^^208d} {(}
\um_setup_active_subscript:nn {^^^^208e} {)}
\um_setup_active_subscript:nn {^^^^2090} {a}
\um_setup_active_subscript:nn {^^^^2091} {e}
\um_setup_active_subscript:nn {^^^^1d62} {i}
\um_setup_active_subscript:nn {^^^^2092} {o}
\um_setup_active_subscript:nn {^^^^1d63} {r}
\um_setup_active_subscript:nn {^^^^1d64} {u}
\um_setup_active_subscript:nn {^^^^1d65} {v}
\um_setup_active_subscript:nn {^^^^2093} {x}
\um_setup_active_subscript:nn {^^^^1d66} {\beta}
\um_setup_active_subscript:nn {^^^^1d67} {\gamma}
\um_setup_active_subscript:nn {^^^^1d68} {\rho}
\um_setup_active_subscript:nn {^^^^1d69} {\phi}
\um_setup_active_subscript:nn {^^^^1d6a} {\chi}
```

```
\group_end:
```

The scanning command, evident in its purpose:

```
\cs_new:Npn \um_scan_sscript: {
  \um_scan_sscript:TF {
    \um_scan_sscript:
  }{
    \um_sub_or_super:n {\l_um_ss_chain_tl}
  }
}
```

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

```
\cs_new:Npn \um_scan_sscript:TF #1#2 {
  \tl_set:Nx \__peek_true_aux:w { \exp_not:n{ #1 } }
  \tl_set_eq:NN \__peek_true:w \__peek_true_remove:w
  \tl_set:Nx \__peek_false:w { \exp_not:n { \group_align_safe_end: #2 } }
  \group_align_safe_begin:
    \peek_after:Nw \um_peek_execute_branches_ss:
}
```

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

```
\cs_new:Npn \um_peek_execute_branches_ss: {
  \bool_if:nTF {
    \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
    \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
    \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
  }
  { \__peek_false:w  }
  { \um_peek_execute_branches_ss_aux: }
}
```

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

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

```
\cs_new:Npn \um_peek_execute_branches_ss_aux: {
  \prop_if_in:cxTF
    {g_um_\l_um_tmpa_tl _prop} {\meaning\l_peek_token}
    {
      \prop_get:cxN
        {g_um_\l_um_tmpa_tl _prop} {\meaning\l_peek_token} \l_um_tmpb_tl
      \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
      \__peek_true:w
    }
    { \__peek_false:w }
}
```

16.3.1 Active fractions

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

```
\cs_new:Npn \um_define_active_frac:Nw #1 #2/#3 {
  \char_set_catcode_active:N #1
  \char_gmake_mathactive:N #1
  \tl_rescan:nn {
    \catcode`\_=11\relax
    \catcode`\:=11\relax
  }{
    \cs_gset:Npx #1 {
     \bool_if:NTF \l_um_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
          {#2} {#3}
    }
  }
}
```

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

```
\cs_new:Npn \um_setup_active_frac: {
  \group_begin:
  \um_define_active_frac:Nw  ^^^^2189  0/3
  \um_define_active_frac:Nw  ^^^^2152  1/{10}
  \um_define_active_frac:Nw  ^^^^2151  1/9
  \um_define_active_frac:Nw  ^^^^215b  1/8
  \um_define_active_frac:Nw  ^^^^2150  1/7
  \um_define_active_frac:Nw  ^^^^2159  1/6
  \um_define_active_frac:Nw  ^^^^2155  1/5
  \um_define_active_frac:Nw  ^^^^00bc  1/4
  \um_define_active_frac:Nw  ^^^^2153  1/3
  \um_define_active_frac:Nw  ^^^^215c  3/8
  \um_define_active_frac:Nw  ^^^^2156  2/5
  \um_define_active_frac:Nw  ^^^^00bd  1/2
  \um_define_active_frac:Nw  ^^^^2157  3/5
  \um_define_active_frac:Nw  ^^^^215d  5/8
  \um_define_active_frac:Nw  ^^^^2154  2/3
  \um_define_active_frac:Nw  ^^^^00be  3/4
  \um_define_active_frac:Nw  ^^^^2158  4/5
  \um_define_active_frac:Nw  ^^^^215a  5/6
  \um_define_active_frac:Nw  ^^^^215e  7/8
  \group_end:
}
\um_setup_active_frac:
```

16.4 Synonyms and all the rest

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

```
\def\to{\rightarrow}
\def\le{\leq}
\def\ge{\geq}
\def\neq{\ne}
\def\triangle{\mathord{\bigtriangleup}}
\def\bigcirc{\mdlgwhtcircle}
\def\circ{\vysmwhtcircle}
\def\bullet{\smblkcircle}
\def\mathyen{\yen}
\def\mathsterling{\sterling}
\def\diamond{\smwhtdiamond}
\def\emptyset{\varnothing}
\def\hbar{\hslash}
\def\land{\wedge}
\def\lor{\vee}
\def\owns{\ni}
\def\gets{\leftarrow}
\def\mathring{\ocirc}
\def\lnot{\neg}
\def\longdivision{\longdivisionsign}
```

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

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

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

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

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

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

`\mathrm`

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

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

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

16.5 Compatibility

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

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

`\um_check_and_fix:NNnnnn`

#1	:	command
#2	:	factory command
#3	:	parameter text
#4	:	expected replacement text
#5	:	new replacement text for LuaTeX
#6	:	new replacement text for XeTeX

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

```
\cs_new_protected_nopar:Npn \um_check_and_fix:NNnnnn #1 #2 #3 #4 #5 #6 {
  \cs_if_exist:NT #1 {
    \token_if_macro:NTF #1 {
      \group_begin:
      #2 \um_tmpa:w #3 { #4 }
      \cs_if_eq:NNTF #1 \um_tmpa:w {
        \msg_info:nnx { unicode-math } { patch-macro }
          { \token_to_str:N #1 }
        \group_end:
        #2 #1 #3
⟨XE⟩          { #6 }
⟨LU⟩          { #5 }
      } {
        \msg_warning:nnxxx { unicode-math } { wrong-meaning }
          { \token_to_str:N #1 } { \token_to_meaning:N #1 }
          { \token_to_meaning:N \um_tmpa:w }
        \group_end:
      }
    } {
      \msg_warning:nnx { unicode-math } { macro-expected }
        { \token_to_str:N #1 }
    }
  }
}
```

`\um_check_and_fix:NNnnn`

#1	:	command
#2	:	factory command
#3	:	parameter text
#4	:	expected replacement text
#5	:	new replacement text

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

```
\cs_new_protected_nopar:Npn \um_check_and_fix:NNnnn #1 #2 #3 #4 #5 {
  \um_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
}
```

`um_check_and_fix_luatex:NNnnn`
`um_check_and_fix_luatex:cNnnn`

#1	:	command
#2	:	factory command
#3	:	parameter text
#4	:	expected replacement text
#5	:	new replacement text

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

```
\cs_new_protected_nopar:Npn \um_check_and_fix_luatex:NNnnn #1 #2 #3 #4 #5 {
  \luatex_if_engine:T {
    \um_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
  }
}
\cs_generate_variant:Nn \um_check_and_fix_luatex:NNnnn { c }
```

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

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

```
\AtEndOfPackageFile * {url} {
  \tl_put_left:Nn \Url@FormatString { \um_switchto_mathup: }
  \tl_put_right:Nn \UrlSpecials {
    \do\`{\mathchar`\`}
    \do\'{\mathchar`\'}
    \do\${\mathchar`\$}
    \do\&{\mathchar`\&}
  }
}
```

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:

```
\AtEndOfPackageFile * {amsmath} {
⟨*XE⟩
    \tl_remove_once:Nn \@begindocumenthook {
      \mathchardef\std@minus\mathcode`\-\relax
      \mathchardef\std@equal\mathcode`\=\relax
    }
    \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
    \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
⟨/XE⟩
  \cs_set:Npn \@cdots {\mathinner{\cdots}}
  \cs_set_eq:NN \dotsb@ \cdots
```

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

```
⟨*XE⟩
    \def \resetMathstrut@ {%
      \setbox\z@\hbox{$($}%)
      \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
    }
```

The subarray environment uses inappropriate font dimensions.

```
    \um_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 } {
      \vcenter
      \bgroup
      \Let@
      \restore@math@cr
      \default@tag
      \baselineskip \fontdimen 10~ \scriptfont \tw@
      \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
      \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
      \lineskiplimit \lineskip
      \ialign
      \bgroup
      \ifx c #1 \hfil \fi
      $ \m@th \scriptstyle ## $
      \hfil
      \crcr
    } {
      \vcenter
```

```
        \c_group_begin_token
        \Let@
        \restore@math@cr
        \default@tag
        \skip_set:Nn \baselineskip {
```

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

```
          \um_stack_num_up:N \scriptstyle
          + \um_stack_denom_down:N \scriptstyle
        }
```

Here we use the minimum stack gap.

```
        \lineskip \um_stack_vgap:N \scriptstyle
        \lineskiplimit \lineskip
        \ialign
        \c_group_begin_token
        \token_if_eq_meaning:NNT c #1 { \hfil }
        \c_math_toggle_token
        \m@th
        \scriptstyle
        \c_parameter_token \c_parameter_token
        \c_math_toggle_token
        \hfil
        \crcr
      }
⟨/XE⟩
```

The roots need a complete rework.

```
\um_check_and_fix_luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 } {
    \setbox \rootbox \hbox {
      $ \m@th \scriptscriptstyle { #1 } $
    }
    \mathchoice
      { \r@@t \displaystyle      { #2 } }
      { \r@@t \textstyle         { #2 } }~
      { \r@@t \scriptstyle       { #2 } }
      { \r@@t \scriptscriptstyle { #2 } }
    \egroup
  } {
    \bool_if:nTF {
      \int_compare_p:nNn { \uproot@ } = { \c_zero }
      && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
    } {
      \luatexUroot \l_um_radical_sqrt_tl { #1 } { #2 }
    } {
      \hbox_set:Nn \rootbox {
        \c_math_toggle_token
        \m@th
        \scriptscriptstyle { #1 }
        \c_math_toggle_token
      }
      \mathchoice
      { \r@@t \displaystyle      { #2 } }
      { \r@@t \textstyle         { #2 } }
      { \r@@t \scriptstyle       { #2 } }
      { \r@@t \scriptscriptstyle { #2 } }
    }
    \c_group_end_token
  }
  \um_check_and_fix:NNnnnn \r@@t \cs_set_nopar:Npn { #1 #2 } {
    \setboxz@h { $ \m@th #1 \sqrtsign { #2 } $ }
    \dimen@ \ht\z@
    \advance \dimen@ -\dp\z@
    \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
    \advance \dimen@ by 1.667 \wd\@ne
    \mkern -\leftroot@ mu
    \mkern 5mu
    \raise .6\dimen@ \copy\rootbox
    \mkern -10mu
    \mkern \leftroot@ mu
    \boxz@
  } {
    \hbox_set:Nn \l_tmpa_box {
      \c_math_toggle_token
      \m@th
      #1
      \mskip \uproot@ mu
      \c_math_toggle_token
    }
    \luatexUroot \l_um_radical_sqrt_tl {
      \box_move_up:nn { \box_wd:N \l_tmpa_box } {
        \hbox:n {
          \c_math_toggle_token
          \m@th
          \mkern -\leftroot@ mu
          \box_use:N \rootbox
          \mkern \leftroot@ mu
          \c_math_toggle_token
        }
      }
    } {
      #2
    }
  } {
    \hbox_set:Nn \l_tmpa_box {
      \c_math_toggle_token
      \m@th
      #1
      \sqrtsign { #2 }
      \c_math_toggle_token
    }
    \hbox_set:Nn \l_tmpb_box {
      \c_math_toggle_token
      \m@th
      #1
      \mskip \uproot@ mu
      \c_math_toggle_token
    }
    \mkern -\leftroot@ mu
    \um_mathstyle_scale:Nnn #1 { \kern } {
      \fontdimen 63 \l_um_font
    }
    \box_move_up:nn {
      \box_wd:N \l_tmpb_box
      + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
      * \number \fontdimen 65 \l_um_font / 100
    } {
      \box_use:N \rootbox
    }
    \um_mathstyle_scale:Nnn #1 { \kern } {
      \fontdimen 64 \l_um_font
    }
    \mkern \leftroot@ mu
    \box_use_clear:N \l_tmpa_box
  }
}
```

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

```
⟨*XE⟩
\AtEndOfPackageFile * {amsopn} {
  \cs_set:Npn \newmcodes@ {
    \mathcode`\'39\scan_stop:
    \mathcode`\*42\scan_stop:
    \mathcode`\."613A\scan_stop:
%%  \ifnum\mathcode`\-=45 \else
%%    \mathchardef\std@minus\mathcode`\-\relax
%%  \fi
    \mathcode`\-45\scan_stop:
    \mathcode`\/47\scan_stop:
    \mathcode`\:"603A\scan_stop:
  }
}
⟨/XE⟩
```

Symbols

```
\cs_set:Npn \| {\Vert}
```

`\mathinner` items:

```
\cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
\cs_set:Npn \cdots {\mathinner{\unicodecdots}}
```

Accents

```
\cs_new_protected_nopar:Nn \um_setup_accents: {
  \cs_gset_protected_nopar:Npx \widehat {
    \um_accent:nnn {} { \um_symfont_tl } { "0302 }
  }
  \cs_gset_protected_nopar:Npx \widetilde {
    \um_accent:nnn {} { \um_symfont_tl } { "0303 }
  }
  \cs_gset_protected_nopar:Npx \overleftarrow {
    \um_accent:nnn {} { \um_symfont_tl } { "20D6 }
  }
  \cs_gset_protected_nopar:Npx \overrightarrow {
    \um_accent:nnn {} { \um_symfont_tl } { "20D7 }
  }
  \cs_gset_protected_nopar:Npx \overleftrightarrow {
    \um_accent:nnn {} { \um_symfont_tl } { "20E1 }
  }
  \cs_gset_protected_nopar:Npx \wideutilde {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "0330 }
  }
  \cs_gset_protected_nopar:Npx \underrightharpoondown {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "20EC }
  }
  \cs_gset_protected_nopar:Npx \underleftharpoondown {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "20ED }
  }
  \cs_gset_protected_nopar:Npx \underleftarrow {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "20EE }
  }
  \cs_gset_protected_nopar:Npx \underrightarrow {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "20EF }
  }
  \cs_gset_protected_nopar:Npx \underleftrightarrow {
    \um_accent:nnn {bottom} { \um_symfont_tl } { "034D }
  }
}
```

```
\cs_set_eq:NN \um_text_slash: \slash
\cs_set_protected:Npn \slash {
  \mode_if_math:TF {\mathslash} {\um_text_slash:}
}
```

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

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

This code is based on an answer to a TeX – Stack Exchange question by Enrico Gregorio[5].

```
\tl_new:N \l_not_token_name_tl

\cs_new:Npn \not_newnot:N #1 {
   \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
   \exp_args:Nx \tl_if_empty:nF { \tl_tail:V \l_not_token_name_tl } {
     \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }
   }
   \cs_if_exist:cTF { n \l_not_token_name_tl } {
     \use:c { n \l_not_token_name_tl }
   } {
     \cs_if_exist:cTF { not \l_not_token_name_tl } {
       \use:c { not \l_not_token_name_tl }
     } {
       \not_oldnot: #1 %\l_not_token_name_tl
     }
   }
}

\cs_new_protected_nopar:Nn \um_setup_negations: {
  \cs_set_eq:NN \not_oldnot: \not
  \cs_set_eq:NN \not \not_newnot:N

  \cs_gset:cpn { not= }    { \neq }
  \cs_gset:cpn { not< }    { \nless }
  \cs_gset:cpn { not> }    { \ngtr }
  \cs_gset:Npn  \ngets     { \nleftarrow }
  \cs_gset:Npn  \nsimeq    { \nsime }
  \cs_gset:Npn  \nequal    { \ne }
  \cs_gset:Npn  \nle       { \nleq }
  \cs_gset:Npn  \nge       { \ngeq }
  \cs_gset:Npn  \ngreater  { \ngtr }
  \cs_gset:Npn  \nforksnot { \forks }
}
```

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

```
\AtEndOfPackageFile * { mathtools } {
⟨*XE⟩
    \newfam \g_um_empty_fam
    \um_check_and_fix:NNnnn
        \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
    {
      \sbox \z@ {
        $
        \m@th
        #1
```

[5]http://tex.stackexchange.com/a/47260/729

```
        \nulldelimiterspace = \z@
        \radical \z@ { #2 }
        $
      }
      \ifx #1 \displaystyle
        \dimen@ = \fontdimen 8 \textfont 3
        \advance \dimen@ .25 \fontdimen 5 \textfont 2
      \else
        \dimen@ = 1.25 \fontdimen 8
        \ifx #1 \textstyle
          \textfont
        \else
          \ifx #1 \scriptstyle
            \scriptfont
          \else
            \scriptscriptfont
          \fi
        \fi
        3
      \fi
      \advance \dimen@ -\ht\z@
      \ht\z@ = -\dimen@
      \box\z@
    }
```

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

```
    {
      \hbox_set:Nn \l_tmpa_box {
        \color@setgroup
        \c_math_toggle_token
        \m@th
        #1
        \dim_zero:N \nulldelimiterspace
        \XeTeXradical \g_um_empty_fam \c_zero { #2 }
        \c_math_toggle_token
        \color@endgroup
      }
      \box_set_ht:Nn \l_tmpa_box {
        \box_ht:N \l_tmpa_box
```

Here we use the radical vertical gap.

```
        - \um_radical_vgap:N #1
      }
      \box_use_clear:N \l_tmpa_box
    }
⟨/XE⟩
```

\overbracket
\underbracket

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

```
\AtEndOfPackageFile * { mathtools } {
    \let\MToverbracket =\overbracket
    \let\MTunderbracket=\underbracket

    \AtBeginDocument {
        \msg_warning:nn { unicode-math } { mathtools-overbracket }

\def\downbracketfill#1#2{%
```

Original definition used the height of \braceld which is not available with Unicode fonts, so we are hard coding the 5/18ex suggested by mathtools's documentation.

```
            \edef\l_MT_bracketheight_fdim{.27ex}%
            \downbracketend{#1}{#2}
            \leaders \vrule \@height #1 \@depth \z@ \hfill
            \downbracketend{#1}{#2}%
        }
\def\upbracketfill#1#2{%
            \edef\l_MT_bracketheight_fdim{.27ex}%
            \upbracketend{#1}{#2}
            \leaders \vrule \@height \z@ \@depth #1 \hfill
            \upbracketend{#1}{#2}%
        }
\let\Uoverbracket =\overbracket
\let\Uunderbracket=\underbracket
        \let\overbracket  =\MToverbracket
        \let\underbracket =\MTunderbracket
    }
}
```

\dblcolon \coloneqq \Coloneqq \eqqcolon

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

```
  \msg_warning:nn { unicode-math } { mathtools-colon }
  \NewDocumentCommand \dblcolon { } { \Colon }
  \NewDocumentCommand \coloneqq { } { \coloneq }
  \NewDocumentCommand \Coloneqq { } { \Coloneq }
  \NewDocumentCommand \eqqcolon { } { \eqcolon }
}
```

colonequals

\ratio \coloncolon \minuscolon \colonequals \equalscolon \coloncolonequals

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

```
\AtEndOfPackageFile * { colonequals } {
  \msg_warning:nn { unicode-math } { colonequals }
  \RenewDocumentCommand \ratio { } { \mathratio }
  \RenewDocumentCommand \coloncolon { } { \Colon }
  \RenewDocumentCommand \minuscolon { } { \dashcolon }
  \RenewDocumentCommand \colonequals { } { \coloneq }
  \RenewDocumentCommand \equalscolon { } { \eqcolon }
  \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
}
```

```
\ExplSyntaxOff
⟨/package&(XE|LU)⟩
```

17 Error messages

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

```
⟨*msg⟩
```

Wrapper functions:

```
\cs_new:Npn \um_warning:n { \msg_warning:nn {unicode-math} }
\cs_new:Npn \um_log:n   { \msg_log:nn   {unicode-math} }
\cs_new:Npn \um_log:nx  { \msg_log:nnx  {unicode-math} }
```

```
\msg_new:nnn {unicode-math} {no-tfrac}
{
  Small~ fraction~ command~ \protect\tfrac\ not~ defined.\\
  Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
}
\msg_new:nnn {unicode-math} {default-math-font}
{
  Defining~ the~ default~ maths~ font~ as~ '\l_um_fontname_tl'.
}
\msg_new:nnn {unicode-math} {setup-implicit}
{
  Setup~ alphabets:~ implicit~ mode.
}
\msg_new:nnn {unicode-math} {setup-explicit}
{
  Setup~ alphabets:~ explicit~ mode.
}
\msg_new:nnn {unicode-math} {alph-initialise}
{
  Initialising~ \@backslashchar math#1.
}
\msg_new:nnn {unicode-math} {setup-alph}
{
  Setup~ alphabet:~ #1.
}
\msg_new:nnn { unicode-math } { missing-alphabets }
  {
    Missing~math~alphabets~in~font~ "\fontname\l_um_font" \\ \\
    \seq_map_function:NN \l_um_missing_alph_seq \um_print_indent:n
  }
```

```
\cs_new:Nn \um_print_indent:n { \space\space\space\space #1 \\ }
\msg_new:nnn {unicode-math} {macro-expected}
{
  I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
}
\msg_new:nnn {unicode-math} {wrong-meaning}
{
  I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ mean-
ing~ #2.
}
\msg_new:nnn {unicode-math} {patch-macro}
{
  I'm~ going~ to~ patch~ macro~ #1.
}
\msg_new:nnn { unicode-math } { mathtools-overbracket } {
  Using~ \token_to_str:N \overbracket\ and~
         \token_to_str:N \underbracke\ from~
 `mathtools'~ package.\\
  \\
  Use~ \token_to_str:N \Uoverbracket\ and~
       \token_to_str:N \Uunderbracke\ for~
       original~ `unicode-math'~ definition.
}
\msg_new:nnn { unicode-math } { mathtools-colon } {
  I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
  the~ `mathtools'~ package: \\ \\
  \ \ \ \ \token_to_str:N \dblcolon,~
  \token_to_str:N \coloneqq,~
  \token_to_str:N \Coloneqq,~
  \token_to_str:N \eqqcolon. \\ \\
  Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
  commands,~ using~ them~ will~ lead~ to~ inconsistencies.
}
\msg_new:nnn { unicode-math } { colonequals } {
  I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
  the~ `colonequals'~ package: \\ \\
  \ \ \ \ \token_to_str:N \ratio,~
          \token_to_str:N \coloncolon,~
          \token_to_str:N \minuscolon, \\
  \ \ \ \ \token_to_str:N \colonequals,~
          \token_to_str:N \equalscolon,~
          \token_to_str:N \coloncolonequals. \\ \\
  Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
  commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
  Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
  or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
  any~ effect~ on~ the~ re-defined~ commands.
}
⟨/msg⟩
```

The end.

18 STIX table data extraction

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

This table is converted into a form suitable for reading by XeTEX. 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!)

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

A Documenting maths support in the NFSS

In the following, ⟨*NFSS decl.*⟩ stands for something like `{T1}{lmr}{m}{n}`.

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

`\DeclareSymbolFont{`⟨*name*⟩`}`⟨*NFSS decl.*⟩
Declares a named maths font such as `operators` from which symbols are defined with `\DeclareMathSymbol`.

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

`\DeclareMathAlphabet{`⟨*cmd*⟩`}`⟨*NFSS decl.*⟩

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

`\DeclareSymbolFontAlphabet{`⟨*cmd*⟩`}{`⟨*name*⟩`}`

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

Maths 'versions' Different maths weights can be defined with the following, switched in text with the `\mathversion{`⟨*maths version*⟩`}` command.

`\SetSymbolFont{`⟨*name*⟩`}{`⟨*maths version*⟩`}`⟨*NFSS decl.*⟩
`\SetMathAlphabet{`⟨*cmd*⟩`}{`⟨*maths version*⟩`}`⟨*NFSS decl.*⟩

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

Delimiters and radicals use wrappers around TEX's `\delimiter`/`\radical` primitives, which are re-designed in XeTEX. The syntax used in LATEX's NFSS is therefore not so relevant here.

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

`\DeclareMathDelimiter{`⟨*symbol*⟩`}{`⟨*type*⟩`}{`⟨*sym. font*⟩`}{`⟨*slot*⟩`}{`⟨*sym. font*⟩`}{`⟨*slot*⟩`}`

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

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 XƎTEX.

Accents are not included yet.

Summary For symbols, something like:

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

For characters, something like:

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

B Legacy TEX font dimensions

Text fonts	
ϕ_1	slant per pt
ϕ_2	interword space
ϕ_3	interword stretch
ϕ_4	interword shrink
ϕ_5	x-height
ϕ_6	quad width
ϕ_7	extra space
ϕ_8	cap height (XƎTEX only)

Maths font, `\fam2`	
σ_5	x height
σ_6	quad
σ_8	num1
σ_9	num2
σ_{10}	num3
σ_{11}	denom1
σ_{12}	denom2
σ_{13}	sup1
σ_{14}	sup2
σ_{15}	sup3
σ_{16}	sub1
σ_{17}	sub2
σ_{18}	sup drop
σ_{19}	sub drop
σ_{20}	delim1
σ_{21}	delim2
σ_{22}	axis height

Maths font, `\fam3`	
ξ_8	default rule thickness
ξ_9	big op spacing1
ξ_{10}	big op spacing2
ξ_{11}	big op spacing3
ξ_{12}	big op spacing4
ξ_{13}	big op spacing5

C XƎTEX math font dimensions

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

`\fontdimen`	Dimension name	Description
10	ScriptPercentScaleDown	Percentage of scaling down for script level 1. Suggested value: 80%.
11	ScriptScriptPercentScale-Down	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	DelimitedSubFormulaMin-Height	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height × 1.5.
13	DisplayOperatorMinHeight	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.

`\fontdimen`	Dimension name	Description
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	SuperscriptShiftUp	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.
22	SuperscriptShiftUpCramped	Standard shift of superscripts relative to the base, in cramped style.
23	SuperscriptBottomMin	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: ¼ 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.

`\fontdimen`	Dimension name	Description
25	SubSuperscriptGapMin	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	SuperscriptBottomMaxWithSubscript	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: /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	LowerLimitBaselineDropMin	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	StackTopShiftUp	Standard shift up applied to the top element of a stack.
33	StackTopDisplayStyleShiftUp	Standard shift up applied to the top element of a stack in display style.
34	StackBottomShiftDown	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	StackBottomDisplayStyleShiftDown	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.

`\fontdimen`	Dimension name	Description
39	StretchStackBottomShiftDown	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	FractionNumeratorDisplayStyleShiftUp	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FractionDenominatorShiftDown	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FractionDenominatorDisplayStyleShiftDown	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FractionNumeratorGapMin	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	FractionNumDisplayStyleGapMin	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	FractionDenominatorGapMin	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FractionDenomDisplayStyleGapMin	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.

`\fontdimen`	Dimension name	Description
51	SkewedFractionHorizontalGap	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SkewedFractionVerticalGap	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OverbarVerticalGap	Distance between the overbar and the (ink) top of he base. Suggested: 3×default rule thickness.
54	OverbarRuleThickness	Thickness of overbar. Suggested: default rule thickness.
55	OverbarExtraAscender	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UnderbarVerticalGap	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UnderbarRuleThickness	Thickness of underbar. Suggested: default rule thickness.
58	UnderbarExtraDescender	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RadicalVerticalGap	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RadicalDisplayStyleVerticalGap	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
61	RadicalRuleThickness	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RadicalExtraAscender	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.
63	RadicalKernBeforeDegree	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RadicalKernAfterDegree	Negative kern after the degree of a radical, if such is present. Suggested: −10/18 of em.
65	RadicalDegreeBottomRaisePercent	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.