

Experimental Unicode mathematical typesetting: The unicode-math package

Will Robertson
will.robertson@latex-project.org

2010/07/14 v0.5a

Abstract

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

(But don't take the warning too seriously, either. I hope the package is now ready to use.)

This is the first release of the unicode-math package, which is intended to be a complete implementation of Unicode maths for \LaTeX using the \XeTeX and \LuaTeX typesetting engines. With this package, changing maths fonts will be as easy as changing text fonts — not that there are many Unicode maths fonts yet. Maths input can also be simplified with Unicode since literal glyphs may be entered instead of control sequences in your document source.

The package is fully tested under \XeTeX , but \LuaTeX support is not yet complete. User beware, but let me 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.tex'. It also comes with a separate document, 'unimath-symbols.pdf', containing a complete listing of mathematical symbols defined by unicode-math.

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.

Contents

1	Introduction	4	7.5	Radicals	46
2	Acknowledgements	4	7.6	Delimiters	46
3	Getting started	4	7.7	Maths accents	47
3.1	Package options	4	8	Font features	47
3.2	Known issues	5	8.1	OpenType maths font features	47
4	Unicode maths font setup	6	8.2	Script and scriptscript font options	47
4.1	Using multiple fonts	6	8.3	Range processing	47
4.2	Script and scriptscript fonts/features	7	8.4	Resolving Greek symbol name control sequences	51
5	Maths input	7	9	Maths alphabets mapping definitions	52
5.1	Math ‘style’	7	9.1	Initialising math styles	52
5.2	Bold style	8	9.2	Defining the math style macros	53
5.3	Sans serif style	9	9.3	Defining the math alphabets per style	54
5.4	All (the rest) of the mathematical alphabets	10	9.4	Mapping ‘naked’ math characters	56
5.5	Miscellanea	12	9.5	Mapping chars inside a math style	58
I	The unicode-math package	18	9.6	Alphabets	61
6	Things we need	18	10	A token list to contain the data of the math table	75
6.1	Extras	20	11	Definitions of the active math characters	75
6.2	Compatibility with LuaTeX	20	12	Epilogue	77
6.3	Alphabet Unicode positions	21	12.1	Primes	77
6.4	STIX fonts	27	12.2	Unicode radicals	82
6.5	Package options	31	12.3	Unicode sub- and super-scripts	84
6.6	Overcoming \@onlypreamble	36	12.4	Synonyms and all the rest	88
7	Fundamentals	37	12.5	Compatibility	89
7.1	Enlarging the number of maths families	37	13	Error messages	91
7.2	Setting math chars, math codes, etc.	37			
7.3	The main \setmathfont macro	40			
7.4	(Big) operators	45			

14	STIX table data extraction	92		the NFSS	92
A	Documenting maths support in		B	X_YTEX math font dimensions	94

1 Introduction

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

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

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 \XeTeX ; Barbara Beeton for her prodigious effort compiling the definitive list of Unicode math glyphs and their \LaTeX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use \TeX in the future. Apostolos Syropoulos, Joel Salomon, 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{Cambria Math}
```

3.1 Package options

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

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

Table 1: Package options.

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

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

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

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

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

3.2 Known issues

In some cases, \XTeX ’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 underbraces, overbraces, and arrows that can grow longer if necessary. Behaviour with such symbols is not necessarily going to be consistent; please report problem areas to me.

\LaTeX ’s concept of math ‘versions’ is not yet supported. The only way to get bold maths is to add markup for it all. This is still an area that requires investigation.

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.

Table 2: Maths font options.

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

4 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

$$\setmathfont[\langle font\ features \rangle]{\langle font\ name \rangle}$$

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

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

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

4.1 Using multiple fonts

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

$$\setmathfont[range=\langle unicode\ range \rangle, \langle font\ features \rangle]{\langle font\ name \rangle}$$

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

4.1.1 Control over maths alphabets

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

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

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

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

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

4.2 Script and scriptscript fonts/features

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

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

5 Maths input

X_YTeX’s Unicode support allows maths input through two methods. Like classical T_EX, 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, T_EX uses italic lowercase Greek letters and *upright* uppercase Greek letters for variables in mathematics. This is contrary to the ISO standards of using italic forms for both upper- and lowercase. Furthermore, the French (contrary again, *quelle surprise*) have been known to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

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

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

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

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

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

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.

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

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

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{bf}` in the former (`'\mathbf{M}'`), and `\bm` (or `\boldsymbol{symbol}`, deprecated) in the latter (`'\boldsymbol{\xi}'`).

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

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

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

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

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

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical alphabet characters. These may be explicitly selected with the `\mathsfup`, `\mathsfif`, `\mathbf{sfup}`, and `\mathbf{sfif}` 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 the seemingly-useless upright sans serif, including Greek; the `italic` style switches to using italic in both Latin and Greek alphabets. In other words, this option simply changes the meaning of `\mathsf` to either `\mathsfup` or `\mathsfif`, 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 `\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.

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.

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

5.4.1 Double-struck

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

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.

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

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

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

Table 6: The various forms of nabla.

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

when a separate ‘Caligraphic’ style is needed as well.

If a font contains alternate glyphs for a separate 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: $\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{X}\mathscr{Y}\mathscr{Z}$

The Caligraphic style (`\mathcal`) in XITS Math is: $\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{X}\mathcal{Y}\mathcal{Z}$

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). T_EX 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`.

5.5.2 Partial

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

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	∂

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

We have a package option to control this behaviour. With `vargreek-shape=TeX`, `\phi` and `\epsilonpsilon` produce ϕ and ε and `\varphi` and `\varepsilonpsilon` 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`.

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

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

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

5.5.4 Primes

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

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

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

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

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

5.5.5 Unicode subscripts and superscripts

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

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

5.5.6 Colon

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

$\text{A}_0 \text{ }_1 \text{ }_2 \text{ }_3 \text{ }_4 \text{ }_5 \text{ }_6 \text{ }_7 \text{ }_8 \text{ }_9 \text{ }_+ \text{ }_- \text{ }_ = \text{ }_ (\text{ }_) \text{ }_ a \text{ }_ e \text{ }_ i \text{ }_ o \text{ }_ r \text{ }_ u \text{ }_ v \text{ }_ x \text{ }_ \beta \text{ }_ \gamma \text{ }_ \rho \text{ }_ \varphi \text{ }_ \chi \text{ }_ \text{Z}$

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

colon with punctuation spacing: ‘ $a:b$ ’.

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

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

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

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

5.5.7 Slashes and backslashes

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

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

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

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

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

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

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

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

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 Σ).

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

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

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

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

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

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

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

²§4.4.5.11 <http://www.w3.org/TR/MathML3/>

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

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdotp</code>	.			
U+22C5	<code>\cdot</code>	.			
U+2219	<code>\vysmbllkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smbllkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbllkcircle</code>	●	◦	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdblkcircle</code>	●	○	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgbllkcircle</code>	●	○	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	○	<code>\lgwhtcircle</code>	U+25EF

Table 9: Filled and hollow Unicode circles.

For example: as mentioned above, Cambria Math’s stretchy slash is U+2044 fraction slash. When using Cambria Math, then unicode-math should be loaded with the `slash-delimiter=frac` option. (This should be a font option rather than a package option, but it will change soon.)

5.5.8 Pre-drawn fraction characters

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

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

For example, instead of writing ‘`\tfrac{12}{x}`’, it’s more readable to have ‘`\frac{x}{12}`’ in the source instead. (There are four missing glyphs above for 0/3, 1/7, 1/9, and 1/10; I don’t have a font that contains them.)

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

5.5.9 Circles

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

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

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

Table 10: Different upwards pointing triangles.

5.5.10 Triangles

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

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

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

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

File I

The unicode-math package

<*preamble>

6 Things we need

```

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

```

Packages

```

8 \RequirePackage{expl3}[2009/08/12]
9 \RequirePackage{xparse}[2009/08/31]
10 \RequirePackage{l3keys2e}
11 \RequirePackage{fontspec}[2010/05/18]
12 \RequirePackage{catchfile}

    Start using LATEX3 — finally!
13 \ExplSyntaxOn
14 \@ifclassloaded{memoir}{
15   \cs_set_eq:NN \um_after_pkg:nn \AtEndPackage
16 }{
17   \RequirePackage{scrfile}
18   \cs_set_eq:NN \um_after_pkg:nn \AfterPackage
19 }

```

Extra expl3 variants

```

20 \cs_generate_variant:Nn \tl_put_right:Nn {cx}
21 \cs_generate_variant:Nn \seq_if_in:NnTF {NV}
22 \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
23 \cs_generate_variant:Nn \prop_get:NnN {cxN}
24 \cs_generate_variant:Nn \prop_if_in:NnTF {cx}

25 \cs_new:Npn \exp_args:NNcc #1#2#3#4 {
26   \exp_after:wN #1 \exp_after:wN #2
27   \cs:w #3 \exp_after:wN \cs_end:
28   \cs:w #4 \cs_end:
29 }

```

Conditionals

```

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

```

For math-style:

```

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

```

For bold-style:

```

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

```

For sans-style:

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

For assorted package options:

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

Variables

```
53 \int_new:N \g_um_fam_int

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

6.1 Extras

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

```
60 \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
61   \etex_iffontchar:D \l_um_font #1 \scan_stop:
62   \prg_return_true:
63   \else:
64   \prg_return_false:
65   \fi:
66 }
67 \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
68 \cs_generate_variant:Nn \um_glyph_if_exist_nTF {c}
69 \cs_generate_variant:Nn \um_glyph_if_exist_nT {c}
70 \cs_generate_variant:Nn \um_glyph_if_exist_nF {c}
```

6.2 Compatibility with LuaT_EX

```
71 \xetex_or luatex:nnn { \cs_new:Npn \um_cs_compat:n #1 }
72 { \cs_set_eq:cc {U#1} {XeTeX#1} }
73 { \cs_set_eq:cc {U#1} {luatexU#1} }
74 \um_cs_compat:n {mathcode}
75 \um_cs_compat:n {delcode}
```

```

76 \um_cs_compat:n {mathcodenum}
77 \um_cs_compat:n {mathcharnum}
78 \um_cs_compat:n {mathchardef}
79 \um_cs_compat:n {radical}
80 \um_cs_compat:n {mathaccent}
81 \um_cs_compat:n {delimiter}

```

6.2.1 Function variants

```

82 \cs_generate_variant:Nn \fontspec_select:nn {x}
83 </preamble>

```

(Error messages and warning definitions go here from the msg chunk defined in section §13 on page 91.)

```

84 <*package>

```

6.3 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.

```

85 \cs_new:Npn \usv_set:nnn #1#2#3 {
86   \tl_set:cn { \um_to_usv:nn {#1}{#2} } {#3}
87 }
88 \cs_new:Npn \um_to_usv:nn #1#2 { g_um_#1_#2_usv }

```

Alphabets

```

89 \usv_set:nnn {up}{num}{48}
90 \usv_set:nnn {up}{Latin}{65}
91 \usv_set:nnn {up}{latin}{97}
92 \usv_set:nnn {up}{Greek}{391}
93 \usv_set:nnn {up}{greek}{381}
94 \usv_set:nnn {it}{Latin}{1D434}
95 \usv_set:nnn {it}{latin}{1D44E}
96 \usv_set:nnn {it}{Greek}{1D6E2}
97 \usv_set:nnn {it}{greek}{1D6FC}
98 \usv_set:nnn {bb}{num}{1D7D8}
99 \usv_set:nnn {bb}{Latin}{1D538}
100 \usv_set:nnn {bb}{latin}{1D552}
101 \usv_set:nnn {scr}{Latin}{1D49C}
102 \usv_set:nnn {cal}{Latin}{1D49C}
103 \usv_set:nnn {scr}{latin}{1D4B6}
104 \usv_set:nnn {frak}{Latin}{1D504}
105 \usv_set:nnn {frak}{latin}{1D51E}
106 \usv_set:nnn {sf}{num}{1D7E2}
107 \usv_set:nnn {sfup}{num}{1D7E2}

```

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

```

108 \usv_set:nnn {sfit}{num}{1D7E2}
109 \usv_set:nnn {sfup}{Latin}{1D5A0}
110 \usv_set:nnn {sf}{Latin}{1D5A0}
111 \usv_set:nnn {sfup}{latin}{1D5BA}
112 \usv_set:nnn {sf}{latin}{1D5BA}
113 \usv_set:nnn {sfit}{Latin}{1D608}
114 \usv_set:nnn {sfit}{latin}{1D622}
115 \usv_set:nnn {tt}{num}{1D7F6}
116 \usv_set:nnn {tt}{Latin}{1D670}
117 \usv_set:nnn {tt}{latin}{1D68A}

```

Bold:

```

118 \usv_set:nnn {bf}{num}{1D7CE}
119 \usv_set:nnn {bfup}{num}{1D7CE}
120 \usv_set:nnn {bfit}{num}{1D7CE}
121 \usv_set:nnn {bfup}{Latin}{1D400}
122 \usv_set:nnn {bfup}{latin}{1D41A}
123 \usv_set:nnn {bfup}{Greek}{1D6A8}
124 \usv_set:nnn {bfup}{greek}{1D6C2}
125 \usv_set:nnn {bfit}{Latin}{1D468}
126 \usv_set:nnn {bfit}{latin}{1D482}
127 \usv_set:nnn {bfit}{Greek}{1D71C}
128 \usv_set:nnn {bfit}{greek}{1D736}
129 \usv_set:nnn {bffrak}{Latin}{1D56C}
130 \usv_set:nnn {bffrak}{latin}{1D586}
131 \usv_set:nnn {bfscr}{Latin}{1D4D0}
132 \usv_set:nnn {bfcal}{Latin}{1D4D0}
133 \usv_set:nnn {bfscr}{latin}{1D4EA}
134 \usv_set:nnn {bfsf}{num}{1D7EC}
135 \usv_set:nnn {bfsfup}{num}{1D7EC}
136 \usv_set:nnn {bfsfit}{num}{1D7EC}
137 \usv_set:nnn {bfsfup}{Latin}{1D5D4}
138 \usv_set:nnn {bfsfup}{latin}{1D5EE}
139 \usv_set:nnn {bfsfup}{Greek}{1D756}
140 \usv_set:nnn {bfsfup}{greek}{1D770}
141 \usv_set:nnn {bfsfit}{Latin}{1D63C}
142 \usv_set:nnn {bfsfit}{latin}{1D656}
143 \usv_set:nnn {bfsfit}{Greek}{1D790}
144 \usv_set:nnn {bfsfit}{greek}{1D7AA}

145 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsfit_
146 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsfit_
147 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsfit_
148 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsfit_
149 \usv_set:nnn {bf}{Latin}{ \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_Lati
150 \usv_set:nnn {bf}{latin}{ \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_lati
151 \usv_set:nnn {bf}{Greek}{ \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_Gree
152 \usv_set:nnn {bf}{greek}{ \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_gree

```

Greek variants:

```
153 \usv_set:nnn {up}{varTheta}{ "3F4}
154 \usv_set:nnn {up}{Digamma}{ "3DC}
155 \usv_set:nnn {up}{varepsilon}{ "3F5}
156 \usv_set:nnn {up}{vartheta}{ "3D1}
157 \usv_set:nnn {up}{varkappa}{ "3F0}
158 \usv_set:nnn {up}{varphi}{ "3D5}
159 \usv_set:nnn {up}{varrho}{ "3F1}
160 \usv_set:nnn {up}{varpi}{ "3D6}
161 \usv_set:nnn {up}{digamma}{ "3DD}
```

Bold:

```
162 \usv_set:nnn {bfup}{varTheta}{ "1D6B9}
163 \usv_set:nnn {bfup}{Digamma}{ "1D7CA}
164 \usv_set:nnn {bfup}{varepsilon}{ "1D6DC}
165 \usv_set:nnn {bfup}{vartheta}{ "1D6DD}
166 \usv_set:nnn {bfup}{varkappa}{ "1D6DE}
167 \usv_set:nnn {bfup}{varphi}{ "1D6DF}
168 \usv_set:nnn {bfup}{varrho}{ "1D6E0}
169 \usv_set:nnn {bfup}{varpi}{ "1D6E1}
170 \usv_set:nnn {bfup}{digamma}{ "1D7CB}
```

Italic Greek variants:

```
171 \usv_set:nnn {it}{varTheta}{ "1D6F3}
172 \usv_set:nnn {it}{varepsilon}{ "1D716}
173 \usv_set:nnn {it}{vartheta}{ "1D717}
174 \usv_set:nnn {it}{varkappa}{ "1D718}
175 \usv_set:nnn {it}{varphi}{ "1D719}
176 \usv_set:nnn {it}{varrho}{ "1D71A}
177 \usv_set:nnn {it}{varpi}{ "1D71B}
```

Bold italic:

```
178 \usv_set:nnn {bfitalic}{varTheta}{ "1D72D}
179 \usv_set:nnn {bfitalic}{varepsilon}{ "1D750}
180 \usv_set:nnn {bfitalic}{vartheta}{ "1D751}
181 \usv_set:nnn {bfitalic}{varkappa}{ "1D752}
182 \usv_set:nnn {bfitalic}{varphi}{ "1D753}
183 \usv_set:nnn {bfitalic}{varrho}{ "1D754}
184 \usv_set:nnn {bfitalic}{varpi}{ "1D755}
```

Bold sans:

```
185 \usv_set:nnn {bfsfup}{varTheta}{ "1D767}
186 \usv_set:nnn {bfsfup}{varepsilon}{ "1D78A}
187 \usv_set:nnn {bfsfup}{vartheta}{ "1D78B}
188 \usv_set:nnn {bfsfup}{varkappa}{ "1D78C}
189 \usv_set:nnn {bfsfup}{varphi}{ "1D78D}
190 \usv_set:nnn {bfsfup}{varrho}{ "1D78E}
191 \usv_set:nnn {bfsfup}{varpi}{ "1D78F}
```

Bold sans italic:

```

192 \usv_set:nnn {bfsfit}{varTheta} {"1D7A1}
193 \usv_set:nnn {bfsfit}{varepsilon} {"1D7C4}
194 \usv_set:nnn {bfsfit}{vartheta} {"1D7C5}
195 \usv_set:nnn {bfsfit}{varkappa} {"1D7C6}
196 \usv_set:nnn {bfsfit}{varphi} {"1D7C7}
197 \usv_set:nnn {bfsfit}{varrho} {"1D7C8}
198 \usv_set:nnn {bfsfit}{varpi} {"1D7C9}

```

Nabla:

```

199 \usv_set:nnn {up} {Nabla} {"02207}
200 \usv_set:nnn {it} {Nabla} {"1D6FB}
201 \usv_set:nnn {bfup} {Nabla} {"1D6C1}
202 \usv_set:nnn {bfit} {Nabla} {"1D735}
203 \usv_set:nnn {bfsfup}{Nabla} {"1D76F}
204 \usv_set:nnn {bfsfit}{Nabla} {"1D7A9}

```

Partial:

```

205 \usv_set:nnn {up} {partial} {"02202}
206 \usv_set:nnn {it} {partial} {"1D715}
207 \usv_set:nnn {bfup} {partial} {"1D6DB}
208 \usv_set:nnn {bfit} {partial} {"1D74F}
209 \usv_set:nnn {bfsfup}{partial} {"1D789}
210 \usv_set:nnn {bfsfit}{partial} {"1D7C3}

```

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

```

211 \usv_set:nnn {up}{B}{`\B}
212 \usv_set:nnn {up}{C}{`\C}
213 \usv_set:nnn {up}{D}{`\D}
214 \usv_set:nnn {up}{E}{`\E}
215 \usv_set:nnn {up}{F}{`\F}
216 \usv_set:nnn {up}{H}{`\H}
217 \usv_set:nnn {up}{I}{`\I}
218 \usv_set:nnn {up}{L}{`\L}
219 \usv_set:nnn {up}{M}{`\M}
220 \usv_set:nnn {up}{N}{`\N}
221 \usv_set:nnn {up}{P}{`\P}
222 \usv_set:nnn {up}{Q}{`\Q}
223 \usv_set:nnn {up}{R}{`\R}
224 \usv_set:nnn {up}{Z}{`\Z}

225 \usv_set:nnn {it}{B} {"1D435}
226 \usv_set:nnn {it}{C} {"1D436}
227 \usv_set:nnn {it}{D} {"1D437}
228 \usv_set:nnn {it}{E} {"1D438}
229 \usv_set:nnn {it}{F} {"1D439}

```



```

230 \usv_set:nnn {it}{H}{1D43B}
231 \usv_set:nnn {it}{I}{1D43C}
232 \usv_set:nnn {it}{L}{1D43F}
233 \usv_set:nnn {it}{M}{1D440}
234 \usv_set:nnn {it}{N}{1D441}
235 \usv_set:nnn {it}{P}{1D443}
236 \usv_set:nnn {it}{Q}{1D444}
237 \usv_set:nnn {it}{R}{1D445}
238 \usv_set:nnn {it}{Z}{1D44D}

239 \usv_set:nnn {up}{d}{`d}
240 \usv_set:nnn {up}{e}{`e}
241 \usv_set:nnn {up}{g}{`g}
242 \usv_set:nnn {up}{h}{`h}
243 \usv_set:nnn {up}{i}{`i}
244 \usv_set:nnn {up}{j}{`j}
245 \usv_set:nnn {up}{o}{`o}

246 \usv_set:nnn {it}{d}{1D451}
247 \usv_set:nnn {it}{e}{1D452}
248 \usv_set:nnn {it}{g}{1D454}
249 \usv_set:nnn {it}{h}{0210E}
250 \usv_set:nnn {it}{i}{1D456}
251 \usv_set:nnn {it}{j}{1D457}
252 \usv_set:nnn {it}{o}{1D45C}

```

Latin ‘h’:

```

253 \usv_set:nnn {bb} {h}{1D559}
254 \usv_set:nnn {tt} {h}{1D691}
255 \usv_set:nnn {scr} {h}{1D4BD}
256 \usv_set:nnn {frak} {h}{1D525}
257 \usv_set:nnn {bfup} {h}{1D421}
258 \usv_set:nnn {bfit} {h}{1D489}
259 \usv_set:nnn {sfup} {h}{1D5C1}
260 \usv_set:nnn {sfit} {h}{1D629}
261 \usv_set:nnn {bffrak}{h}{1D58D}
262 \usv_set:nnn {bfscr} {h}{1D4F1}
263 \usv_set:nnn {bfsfup}{h}{1D5F5}
264 \usv_set:nnn {bfsfit}{h}{1D65D}

```

Dotless ‘i’ and ‘j’:

```

265 \usv_set:nnn {up}{dotlessi}{00131}
266 \usv_set:nnn {up}{dotlessj}{00237}
267 \usv_set:nnn {it}{dotlessi}{1D6A4}
268 \usv_set:nnn {it}{dotlessj}{1D6A5}

```

Blackboard:

```

269 \usv_set:nnn {bb}{C}{2102}
270 \usv_set:nnn {bb}{H}{210D}

```

```

271 \usv_set:nnn {bb}{N}{ "2115}
272 \usv_set:nnn {bb}{P}{ "2119}
273 \usv_set:nnn {bb}{Q}{ "211A}
274 \usv_set:nnn {bb}{R}{ "211D}
275 \usv_set:nnn {bb}{Z}{ "2124}
276 \usv_set:nnn {up}{Pi}      {"003A0}
277 \usv_set:nnn {up}{pi}      {"003C0}
278 \usv_set:nnn {up}{Gamma}    {"00393}
279 \usv_set:nnn {up}{gamma}    {"003B3}
280 \usv_set:nnn {up}{summation}{ "02211}
281 \usv_set:nnn {it}{Pi}      {"1D6F1}
282 \usv_set:nnn {it}{pi}      {"1D70B}
283 \usv_set:nnn {it}{Gamma}    {"1D6E4}
284 \usv_set:nnn {it}{gamma}    {"1D6FE}
285 \usv_set:nnn {bb}{Pi}      {"0213F}
286 \usv_set:nnn {bb}{pi}      {"0213C}
287 \usv_set:nnn {bb}{Gamma}    {"0213E}
288 \usv_set:nnn {bb}{gamma}    {"0213D}
289 \usv_set:nnn {bb}{summation}{ "02140}

```

Italic blackboard:

```

290 \usv_set:nnn {bbit}{D}{ "2145}
291 \usv_set:nnn {bbit}{d}{ "2146}
292 \usv_set:nnn {bbit}{e}{ "2147}
293 \usv_set:nnn {bbit}{i}{ "2148}
294 \usv_set:nnn {bbit}{j}{ "2149}

```

Script exceptions:

```

295 \usv_set:nnn {scr}{B}{ "212C}
296 \usv_set:nnn {scr}{E}{ "2130}
297 \usv_set:nnn {scr}{F}{ "2131}
298 \usv_set:nnn {scr}{H}{ "210B}
299 \usv_set:nnn {scr}{I}{ "2110}
300 \usv_set:nnn {scr}{L}{ "2112}
301 \usv_set:nnn {scr}{M}{ "2133}
302 \usv_set:nnn {scr}{R}{ "211B}
303 \usv_set:nnn {scr}{e}{ "212F}
304 \usv_set:nnn {scr}{g}{ "210A}
305 \usv_set:nnn {scr}{o}{ "2134}

306 \usv_set:nnn {cal}{B}{ "212C}
307 \usv_set:nnn {cal}{E}{ "2130}
308 \usv_set:nnn {cal}{F}{ "2131}
309 \usv_set:nnn {cal}{H}{ "210B}
310 \usv_set:nnn {cal}{I}{ "2110}
311 \usv_set:nnn {cal}{L}{ "2112}
312 \usv_set:nnn {cal}{M}{ "2133}
313 \usv_set:nnn {cal}{R}{ "211B}

```

Fraktur exceptions:

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

6.4 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.

```
319 </package>
320 <*stix>
```

Upright

```
321 \usv_set:nnn {stixsfup}{partial}{ "E17C}
322 \usv_set:nnn {stixsfup}{Greek}{ "E17D}
323 \usv_set:nnn {stixsfup}{greek}{ "E196}
324 \usv_set:nnn {stixsfup}{varTheta}{ "E18E}
325 \usv_set:nnn {stixsfup}{varepsilon}{ "E1AF}
326 \usv_set:nnn {stixsfup}{vartheta}{ "E1B0}
327 \usv_set:nnn {stixsfup}{varkappa}{0000} % ???
328 \usv_set:nnn {stixsfup}{varphi}{ "E1B1}
329 \usv_set:nnn {stixsfup}{varrho}{ "E1B2}
330 \usv_set:nnn {stixsfup}{varpi}{ "E1B3}
331 \usv_set:nnn {stixupslash}{Greek}{ "E2FC}
```

Italic

```
332 \usv_set:nnn {stixbbit}{A}{ "E154}
333 \usv_set:nnn {stixbbit}{B}{ "E155}
334 \usv_set:nnn {stixbbit}{E}{ "E156}
335 \usv_set:nnn {stixbbit}{F}{ "E157}
336 \usv_set:nnn {stixbbit}{G}{ "E158}
337 \usv_set:nnn {stixbbit}{I}{ "E159}
338 \usv_set:nnn {stixbbit}{J}{ "E15A}
339 \usv_set:nnn {stixbbit}{K}{ "E15B}
340 \usv_set:nnn {stixbbit}{L}{ "E15C}
341 \usv_set:nnn {stixbbit}{M}{ "E15D}
342 \usv_set:nnn {stixbbit}{O}{ "E15E}
343 \usv_set:nnn {stixbbit}{S}{ "E15F}
344 \usv_set:nnn {stixbbit}{T}{ "E160}
345 \usv_set:nnn {stixbbit}{U}{ "E161}
```

```

346 \usv_set:nnn {stixbbit}{V}{E162}
347 \usv_set:nnn {stixbbit}{W}{E163}
348 \usv_set:nnn {stixbbit}{X}{E164}
349 \usv_set:nnn {stixbbit}{Y}{E165}

350 \usv_set:nnn {stixbbit}{a}{E166}
351 \usv_set:nnn {stixbbit}{b}{E167}
352 \usv_set:nnn {stixbbit}{c}{E168}
353 \usv_set:nnn {stixbbit}{f}{E169}
354 \usv_set:nnn {stixbbit}{g}{E16A}
355 \usv_set:nnn {stixbbit}{h}{E16B}
356 \usv_set:nnn {stixbbit}{k}{E16C}
357 \usv_set:nnn {stixbbit}{l}{E16D}
358 \usv_set:nnn {stixbbit}{m}{E16E}
359 \usv_set:nnn {stixbbit}{n}{E16F}
360 \usv_set:nnn {stixbbit}{o}{E170}
361 \usv_set:nnn {stixbbit}{p}{E171}
362 \usv_set:nnn {stixbbit}{q}{E172}
363 \usv_set:nnn {stixbbit}{r}{E173}
364 \usv_set:nnn {stixbbit}{s}{E174}
365 \usv_set:nnn {stixbbit}{t}{E175}
366 \usv_set:nnn {stixbbit}{u}{E176}
367 \usv_set:nnn {stixbbit}{v}{E177}
368 \usv_set:nnn {stixbbit}{w}{E178}
369 \usv_set:nnn {stixbbit}{x}{E179}
370 \usv_set:nnn {stixbbit}{y}{E17A}
371 \usv_set:nnn {stixbbit}{z}{E17B}

372 \usv_set:nnn {stixsfit}{Numerals}{E1B4}
373 \usv_set:nnn {stixsfit}{partial}{E1BE}
374 \usv_set:nnn {stixsfit}{Greek}{E1BF}
375 \usv_set:nnn {stixsfit}{greek}{E1D8}
376 \usv_set:nnn {stixsfit}{varTheta}{E1D0}
377 \usv_set:nnn {stixsfit}{varepsilon}{E1F1}
378 \usv_set:nnn {stixsfit}{vartheta}{E1F2}
379 \usv_set:nnn {stixsfit}{varkappa}{0000} % ???
380 \usv_set:nnn {stixsfit}{varphi}{E1F3}
381 \usv_set:nnn {stixsfit}{varrho}{E1F4}
382 \usv_set:nnn {stixsfit}{varpi}{E1F5}

383 \usv_set:nnn {stixcal}{Latin}{E22D}
384 \usv_set:nnn {stixcal}{num}{E262}
385 \usv_set:nnn {scr}{num}{48}
386 \usv_set:nnn {it}{num}{48}

387 \usv_set:nnn {stixsfitslash}{Latin}{E294}
388 \usv_set:nnn {stixsfitslash}{latin}{E2C8}
389 \usv_set:nnn {stixsfitslash}{greek}{E32C}
390 \usv_set:nnn {stixsfitslash}{varepsilon}{E37A}

```

```

391 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35E}
392 \usv_set:nnn {stixsfitslash}{varkappa}{ "E374}
393 \usv_set:nnn {stixsfitslash}{varphi}{ "E360}
394 \usv_set:nnn {stixsfitslash}{varrho}{ "E376}
395 \usv_set:nnn {stixsfitslash}{varpi}{ "E362}
396 \usv_set:nnn {stixsfitslash}{digamma}{ "E36A}

```

Bold

```

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

399 \usv_set:nnn {stixbfbb}{A}{ "E38A}
400 \usv_set:nnn {stixbfbb}{B}{ "E38B}
401 \usv_set:nnn {stixbfbb}{E}{ "E38D}
402 \usv_set:nnn {stixbfbb}{F}{ "E38E}
403 \usv_set:nnn {stixbfbb}{G}{ "E38F}
404 \usv_set:nnn {stixbfbb}{I}{ "E390}
405 \usv_set:nnn {stixbfbb}{J}{ "E391}
406 \usv_set:nnn {stixbfbb}{K}{ "E392}
407 \usv_set:nnn {stixbfbb}{L}{ "E393}
408 \usv_set:nnn {stixbfbb}{M}{ "E394}
409 \usv_set:nnn {stixbfbb}{O}{ "E395}
410 \usv_set:nnn {stixbfbb}{S}{ "E396}
411 \usv_set:nnn {stixbfbb}{T}{ "E397}
412 \usv_set:nnn {stixbfbb}{U}{ "E398}
413 \usv_set:nnn {stixbfbb}{V}{ "E399}
414 \usv_set:nnn {stixbfbb}{W}{ "E39A}
415 \usv_set:nnn {stixbfbb}{X}{ "E39B}
416 \usv_set:nnn {stixbfbb}{Y}{ "E39C}

417 \usv_set:nnn {stixbfbb}{a}{ "E39D}
418 \usv_set:nnn {stixbfbb}{b}{ "E39E}
419 \usv_set:nnn {stixbfbb}{c}{ "E39F}
420 \usv_set:nnn {stixbfbb}{f}{ "E3A2}
421 \usv_set:nnn {stixbfbb}{g}{ "E3A3}
422 \usv_set:nnn {stixbfbb}{h}{ "E3A4}
423 \usv_set:nnn {stixbfbb}{k}{ "E3A7}
424 \usv_set:nnn {stixbfbb}{l}{ "E3A8}
425 \usv_set:nnn {stixbfbb}{m}{ "E3A9}
426 \usv_set:nnn {stixbfbb}{n}{ "E3AA}
427 \usv_set:nnn {stixbfbb}{o}{ "E3AB}
428 \usv_set:nnn {stixbfbb}{p}{ "E3AC}
429 \usv_set:nnn {stixbfbb}{q}{ "E3AD}
430 \usv_set:nnn {stixbfbb}{r}{ "E3AE}
431 \usv_set:nnn {stixbfbb}{s}{ "E3AF}
432 \usv_set:nnn {stixbfbb}{t}{ "E3B0}
433 \usv_set:nnn {stixbfbb}{u}{ "E3B1}

```

```

434 \usv_set:nnn {stixbfbb}{v}{ "E3B2}
435 \usv_set:nnn {stixbfbb}{w}{ "E3B3}
436 \usv_set:nnn {stixbfbb}{x}{ "E3B4}
437 \usv_set:nnn {stixbfbb}{y}{ "E3B5}
438 \usv_set:nnn {stixbfbb}{z}{ "E3B6}
439 \usv_set:nnn {stixbfsfup}{Numerals}{ "E3B7}

```

Bold Italic

```

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

441 \usv_set:nnn {stixbfbbbit}{A}{ "E200}
442 \usv_set:nnn {stixbfbbbit}{B}{ "E201}
443 \usv_set:nnn {stixbfbbbit}{E}{ "E203}
444 \usv_set:nnn {stixbfbbbit}{F}{ "E204}
445 \usv_set:nnn {stixbfbbbit}{G}{ "E205}
446 \usv_set:nnn {stixbfbbbit}{I}{ "E206}
447 \usv_set:nnn {stixbfbbbit}{J}{ "E207}
448 \usv_set:nnn {stixbfbbbit}{K}{ "E208}
449 \usv_set:nnn {stixbfbbbit}{L}{ "E209}
450 \usv_set:nnn {stixbfbbbit}{M}{ "E20A}
451 \usv_set:nnn {stixbfbbbit}{O}{ "E20B}
452 \usv_set:nnn {stixbfbbbit}{S}{ "E20C}
453 \usv_set:nnn {stixbfbbbit}{T}{ "E20D}
454 \usv_set:nnn {stixbfbbbit}{U}{ "E20E}
455 \usv_set:nnn {stixbfbbbit}{V}{ "E20F}
456 \usv_set:nnn {stixbfbbbit}{W}{ "E210}
457 \usv_set:nnn {stixbfbbbit}{X}{ "E211}
458 \usv_set:nnn {stixbfbbbit}{Y}{ "E212}

459 \usv_set:nnn {stixbfbbbit}{a}{ "E213}
460 \usv_set:nnn {stixbfbbbit}{b}{ "E214}
461 \usv_set:nnn {stixbfbbbit}{c}{ "E215}
462 \usv_set:nnn {stixbfbbbit}{e}{ "E217}
463 \usv_set:nnn {stixbfbbbit}{f}{ "E218}
464 \usv_set:nnn {stixbfbbbit}{g}{ "E219}
465 \usv_set:nnn {stixbfbbbit}{h}{ "E21A}
466 \usv_set:nnn {stixbfbbbit}{k}{ "E21D}
467 \usv_set:nnn {stixbfbbbit}{l}{ "E21E}
468 \usv_set:nnn {stixbfbbbit}{m}{ "E21F}
469 \usv_set:nnn {stixbfbbbit}{n}{ "E220}
470 \usv_set:nnn {stixbfbbbit}{o}{ "E221}
471 \usv_set:nnn {stixbfbbbit}{p}{ "E222}
472 \usv_set:nnn {stixbfbbbit}{q}{ "E223}
473 \usv_set:nnn {stixbfbbbit}{r}{ "E224}
474 \usv_set:nnn {stixbfbbbit}{s}{ "E225}
475 \usv_set:nnn {stixbfbbbit}{t}{ "E226}
476 \usv_set:nnn {stixbfbbbit}{u}{ "E227}

```

```

477 \usv_set:nnn {stixbfbbbit}{v}{ "E228}
478 \usv_set:nnn {stixbfbbbit}{w}{ "E229}
479 \usv_set:nnn {stixbfbbbit}{x}{ "E22A}
480 \usv_set:nnn {stixbfbbbit}{y}{ "E22B}
481 \usv_set:nnn {stixbfbbbit}{z}{ "E22C}

482 \usv_set:nnn {stixbfcal}{Latin}{ "E247}

483 \usv_set:nnn {stixbfitslash}{Latin}{ "E295}
484 \usv_set:nnn {stixbfitslash}{latin}{ "E2C9}
485 \usv_set:nnn {stixbfitslash}{greek}{ "E32D}
486 \usv_set:nnn {stixsfitslash}{varepsilon}{ "E37B}
487 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35F}
488 \usv_set:nnn {stixsfitslash}{varkappa}{ "E375}
489 \usv_set:nnn {stixsfitslash}{varphi}{ "E361}
490 \usv_set:nnn {stixsfitslash}{varrho}{ "E377}
491 \usv_set:nnn {stixsfitslash}{varpi}{ "E363}
492 \usv_set:nnn {stixsfitslash}{digamma}{ "E36B}

493 </stix>
494 <*package>

```

6.5 Package options

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

```

495 \DeclareDocumentCommand \unimathsetup {m} {
496   \clist_clear:N \l_um_unknown_keys_clist
497   \keys_set:nn {unicode-math} {#1}
498 }

```

math-style

```

499 \keys_define:nn {unicode-math} {
500   normal-style .choice_code:n =
501   {
502     \bool_set_false:N \g_um_literal_bool
503     \ifcase \l_keys_choice_int
504       \bool_set_false:N \g_um_upGreek_bool
505       \bool_set_false:N \g_um_upgreek_bool
506       \bool_set_false:N \g_um_upLatin_bool
507       \bool_set_false:N \g_um_uplatin_bool
508     \or
509       \bool_set_true:N \g_um_upGreek_bool
510       \bool_set_false:N \g_um_upgreek_bool
511       \bool_set_false:N \g_um_upLatin_bool
512       \bool_set_false:N \g_um_uplatin_bool
513     \or

```

```

514     \bool_set_true:N \g_um_upGreek_bool
515     \bool_set_true:N \g_um_upgreek_bool
516     \bool_set_true:N \g_um_upLatin_bool
517     \bool_set_false:N \g_um_uplatin_bool
518   \or
519     \bool_set_true:N \g_um_upGreek_bool
520     \bool_set_true:N \g_um_upgreek_bool
521     \bool_set_true:N \g_um_upLatin_bool
522     \bool_set_true:N \g_um_uplatin_bool
523   \or
524     \bool_set_true:N \g_um_literal_bool
525   \fi
526 } ,
527 normal-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
528 }

529 \keys_define:nn {unicode-math} {
530   math-style .choice_code:n =
531   {
532     \ifcase \l_keys_choice_int
533       \unimathsetup {
534         normal-style=ISO,
535         bold-style=ISO,
536         sans-style=italic,
537         nabla=upright,
538         partial=italic,
539       }
540     \or
541       \unimathsetup {
542         normal-style=TeX,
543         bold-style=TeX,
544         sans-style=upright,
545         nabla=upright,
546         partial=italic,
547       }
548     \or
549       \unimathsetup {
550         normal-style=french,
551         bold-style=upright,
552         sans-style=upright,
553         nabla=upright,
554         partial=upright,
555       }
556     \or
557       \unimathsetup {
558         normal-style=upright,
559         bold-style=upright,

```



```

560     sans-style=upright,
561     nabla=upright,
562     partial=upright,
563   }
564   \or
565   \unimathsetup {
566     normal-style=literal,
567     bold-style=literal,
568     sans-style=literal,
569     colon=literal,
570     nabla=literal,
571     partial=literal,
572   }
573   \fi
574 } ,
575 math-style .generate_choices:n = {ISO,TeX,french,upright,literal} ,
576 }

```

bold-style

```

577 \keys_define:nn {unicode-math} {
578   bold-style .choice_code:n = {
579     \bool_set_false:N \g_um_bfliteral_bool
580     \ifcase \l_keys_choice_int
581       \bool_set_false:N \g_um_bfupgreek_bool
582       \bool_set_false:N \g_um_bfupgreek_bool
583       \bool_set_false:N \g_um_bfuplatin_bool
584       \bool_set_false:N \g_um_bfuplatin_bool
585     \or
586       \bool_set_true:N \g_um_bfupgreek_bool
587       \bool_set_false:N \g_um_bfupgreek_bool
588       \bool_set_true:N \g_um_bfuplatin_bool
589       \bool_set_true:N \g_um_bfuplatin_bool
590     \or
591       \bool_set_true:N \g_um_bfupgreek_bool
592       \bool_set_true:N \g_um_bfupgreek_bool
593       \bool_set_true:N \g_um_bfuplatin_bool
594       \bool_set_true:N \g_um_bfuplatin_bool
595     \or
596       \bool_set_true:N \g_um_bfliteral_bool
597     \fi
598   } ,
599   bold-style .generate_choices:n = {ISO,TeX,upright,literal} ,
600 }

```

sans-style

```

601 \keys_define:nn {unicode-math} {
602   sans-style .choice_code:n = {
603     \ifcase \l_keys_choice_int
604       \bool_set_false:N \g_um_upsans_bool
605     \or
606       \bool_set_true:N \g_um_upsans_bool
607     \or
608       \bool_set_true:N \g_um_sfliteral_bool
609     \fi
610   } ,
611   sans-style .generate_choices:n = {italic,upright,literal} ,
612 }

```

Nabla and partial

```

613 \keys_define:nn {unicode-math} {
614   nabla .choice_code:n = {
615     \bool_set_false:N \g_um_literal_Nabla_bool
616     \ifcase \l_keys_choice_int
617       \bool_set_true:N \g_um_upNabla_bool
618     \or
619       \bool_set_false:N \g_um_upNabla_bool
620     \or
621       \bool_set_true:N \g_um_literal_Nabla_bool
622     \fi
623   } ,
624   nabla .generate_choices:n = {upright,italic,literal} ,
625 }

626 \keys_define:nn {unicode-math} {
627   partial .choice_code:n = {
628     \bool_set_false:N \g_um_literal_partial_bool
629     \ifcase \l_keys_choice_int
630       \bool_set_true:N \g_um_uppartial_bool
631     \or
632       \bool_set_false:N \g_um_uppartial_bool
633     \or
634       \bool_set_true:N \g_um_literal_partial_bool
635     \fi
636   } ,
637   partial .generate_choices:n = {upright,italic,literal} ,
638 }

```

Epsilon and phi shapes

```

639 \keys_define:nn {unicode-math} {
640   vargreek-shape .choice: ,
641   vargreek-shape / unicode .code:n = {

```

```

642     \bool_set_false:N \g_um_texgreek_bool
643   } ,
644   vargreek-shape / TeX .code:n = {
645     \bool_set_true:N \g_um_texgreek_bool
646   }
647 }

```

Colon style

```

648 \keys_define:nn {unicode-math} {
649   colon .choice: ,
650   colon / literal .code:n = {
651     \bool_set_true:N \g_um_literal_colon_bool
652   } ,
653   colon / TeX .code:n = {
654     \bool_set_false:N \g_um_literal_colon_bool
655   }
656 }

```

Slash delimiter style

```

657 \keys_define:nn {unicode-math} {
658   slash-delimiter .choice: ,
659   slash-delimiter / ascii .code:n = {
660     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
661   } ,
662   slash-delimiter / frac .code:n = {
663     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
664   } ,
665   slash-delimiter / div .code:n = {
666     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
667   }
668 }

```

Active fraction style

```

669 \keys_define:nn {unicode-math} {
670   active-frac .choice: ,
671   active-frac / small .code:n = {
672     \cs_if_exist:NTF \tfrac {
673       \bool_set_true:N \l_um_smallfrac_bool
674     }{
675       \um_warning:n {no-tfrac}
676       \bool_set_false:N \l_um_smallfrac_bool
677     }
678     \use:c{um_setup_active_frac:}
679   } ,
680   active-frac / normalsize .code:n = {

```

```

681 \bool_set_false:N \l_um_smallfrac_bool
682 \use:c{um_setup_active_frac:}
683 }
684 }

```

Debug/tracing

```

685 \keys_define:nn {unicode-math} {
686   trace .choice: ,
687   trace / debug .code:n = {
688     \msg_redirect_module:nnn { unicode-math } { trace } { warning }
689   } ,
690   trace / on .code:n = {
691     \msg_redirect_module:nnn { unicode-math } { trace } { trace }
692   } ,
693   trace / off .code:n = {
694     \msg_redirect_module:nnn { unicode-math } { trace } { none }
695   } ,
696 }

697 \clist_new:N \l_um_unknown_keys_clist
698 \keys_define:nn {unicode-math} {
699   unknown .code:n = {
700     \clist_put_right:No \l_um_unknown_keys_clist {
701       \l_keys_key_tl = {#1}
702     }
703   }
704 }

705 \unimathsetup {math-style=TeX}
706 \unimathsetup {slash-delimiter=ascii}
707 \unimathsetup {trace=off}
708 \cs_if_exist:NT \tfrac {
709   \unimathsetup {active-frac=small}
710 }
711 \ProcessKeysOptions {unicode-math}

```

6.6 Overcoming \@onlypreamble

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

```

712 \tl_map_inline:nn {
713   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
714   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
715   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
716   \version@list\version@elt\alpha@list\alpha@elt
717   \restore@mathversion\init@restore@version\dorestore@version\process@table
718   \new@mathversion\DeclareSymbolFont\group@list\group@elt

```

```

719 \new@symbolfont\SetSymbolFont\SetSymbolFont@get@cdp
720 \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
721 \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
722 \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
723 \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
724 \set@mathdelimiter\DeclareMathRadical\mathchar@type
725 \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
726 }{
727 \tl_remove_in:Nn \@preamblecmds {\do#1}
728 }

```

7 Fundamentals

7.1 Enlarging the number of maths families

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

```

729 \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
730 \let\newfam\new@mathgroup

```

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

7.2 Setting math chars, math codes, etc.

```

\um_set_mathsymbol:nNNn #1 : A LATEX symbol font, e.g., operators
#2 : Symbol macro, e.g., \alpha
#3 : Type, e.g., \mathalpha
#4 : Slot, e.g., "221E

```

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

```

731 \cs_set:Npn \um_set_mathsymbol:nNNn #1#2#3#4 {
732   \prg_case_tl:Nnn #3 {
733     \mathop {
734       \um_set_big_operator:nnn {#1} {#2} {#4}
735     }
736     \mathopen {
737       \tl_if_in:NnTF \l_um_radicals_tl {#2} {
738         \cs_gset:cpx {\cs_to_str:N #2 sign} { \um_radical:nn {#1} {#4} }
739         \tl_set:cn {l_um_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #4}
740       }{
741         \um_set_delcode:nnn {#1} {#4} {#4}
742         \um_set_mathcode:nnn {#4} \mathopen {#1}
743         \cs_gset:Npx #2 { \um_delimiter:Nnn \mathopen {#1} {#4} }
744       }

```

```

745 }
746 \mathclose {
747   \um_set_delcode:nnn {#1} {#4} {#4}
748   \um_set_mathcode:nnn {#4} \mathclose {#1}
749   \cs_gset:Npx #2 { \um_delimiter:Nnn \mathclose {#1} {#4} }
750 }
751 \mathaccent {
752   \cs_gset:Npx #2 { \um_accent:Nnn #3 {#1} {#4} }
753 }
754 \mathfence {
755   \um_set_mathcode:nnn {#4} {#3} {#1}
756   \um_set_delcode:nnn {#1} {#4} {#4}
757   \cs_gset:cpx {l \cs_to_str:N #2} { \um_delimiter:Nnn \math-
open {#1} {#4} }
758   \cs_gset:cpx {r \cs_to_str:N #2} { \um_delimiter:Nnn \math-
close {#1} {#4} }
759 }
760 \mathover { % LuaTeX only
761   \cs_set:Npn #2 ##1 { \mathop { \um_overbrace:nnn {#1} {#4} {##1} } \lim-
its }
762 }
763 \mathunder { % LuaTeX only
764   \cs_set:Npn #2 ##1 { \mathop { \um_underbrace:nnn {#1} {#4} {##1} } \lim-
its }
765 }
766 }{
767   \um_set_mathcode:nnn {#4} {#3} {#1}
768 }
769 }

```

`\um_set_big_operator:nnn` #1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot

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

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

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

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

$(\sum \rightarrow) \rightarrow \sum \rightarrow \sum_{\text{sym}} \rightarrow \sum_{\text{op}} \rightarrow \sum_{\text{nolimits}}$
 $(\int \rightarrow) \rightarrow \int \rightarrow \int_{\text{sym}} \rightarrow \int_{\text{top}}$

```

770 \cs_new:Npn \um_set_big_operator:nnn #1#2#3 {
771   \group_begin:
772     \char_make_active:n {#3}
773     \char_gmake_mathactive:n {#3}
774     \um@scanactivedef #3 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }
775   \group_end:
776   \um_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
777   \cs_gset:cpx { \cs_to_str:N #2 _sym } {
778     \exp_not:c { \cs_to_str:N #2 op }
779     \exp_not:n { \tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits }
780   }
781 }

```

$\backslash\text{um_set_mathcode:nnnn}$ These are all wrappers for the primitive commands that take numerical input only.

```

\um_set_mathcode:nnnn 782 \cs_set:Npn \um_set_mathcode:nnnn #1#2#3#4 {
\um_set_mathchar:NNnn 783   \Umathcode \intexpr_eval:n {#1} =
\um_set_mathchar:cNnn 784   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
\um_set_delcode:nnnn 785 }
\um_radical:nn 786 \cs_set:Npn \um_set_mathcode:nnn #1#2#3 {
\um_delimiter:Nnn 787   \Umathcode \intexpr_eval:n {#1} =
\um_accent:Nnn 788   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#1} \scan_stop:
789 }
790 \cs_set:Npn \um_set_mathchar:NNnn #1#2#3#4 {
791   \Umathchardef #1 =
792   \mathchar@type#2 \csname sym#3\endcsname \intexpr_eval:n {#4} \scan_stop:
793 }
794 \cs_new:Npn \um_set_delcode:nnn #1#2#3 {
795   \Udelcode#2 = \csname sym#1\endcsname #3
796 }
797 \cs_new:Npn \um_radical:nn #1#2 {
798   \Uradical \csname sym#1\endcsname #2 \scan_stop:
799 }
800 \cs_new:Npn \um_delimiter:Nnn #1#2#3 {
801   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
802 }
803 \cs_new:Npn \um_accent:Nnn #1#2#3 {
804   \Umathaccent \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
805 }
806 \cs_generate_variant:Nn \um_set_mathchar:NNnn {c}

```

$\backslash\text{um_overbrace:nnn}$ LuaTeX functions for defining over/under-braces
 $\backslash\text{um_underbrace:nnn}$

```

807 \cs_set:Npn \um_overbrace:nnn #1#2#3 {
808   \luatexUdelimiterover \csname sym#1\endcsname #2 {#3}
809 }
810 \cs_set:Npn \um_underbrace:nnn #1#2#3 {
811   \luatexUdelimiterunder \csname sym#1\endcsname #2 {#3}
812 }

\char_gmake_mathactive:N
\char_gmake_mathactive:n 813 \cs_new:Npn \char_gmake_mathactive:N #1 {
814   \global\mathcode `#1 = "8000 \scan_stop:
815 }
816 \cs_new:Npn \char_gmake_mathactive:n #1 {
817   \global\mathcode #1 = "8000 \scan_stop:
818 }

```

7.3 The main `\setmathfont` macro

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

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

```

819 \cs_new:Npn \um_init: {

```

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

```

820   \let\glb@currsizel\relax

```

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

```

821   \bool_set_true:N \l_um_init_bool
822   \seq_clear:N \l_um_char_range_seq
823   \clist_clear:N \l_um_char_num_range_clist
824   \seq_clear:N \l_um_mathalph_seq
825   \clist_clear:N \l_um_unknown_keys_clist
826   \seq_clear:N \l_um_missing_alph_seq
827 }
828 \DeclareDocumentCommand \setmathfont { 0{ } m } {
829   \um_init:

```

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

```

830   \csname S@\f@size\endcsname

```


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

```
831 \tl_set:Nn \l_um_mversion_tf {normal}
832 \DeclareMathVersion{\l_um_mversion_tf}
```

Define default font features for the script and scriptscript font.

```
833 \tl_set:Nn \l_um_script_features_tl {ScriptStyle}
834 \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
835 \tl_set:Nn \l_um_script_font_tl      {#2}
836 \tl_set:Nn \l_um_sscript_font_tl    {#2}
```

Use fontspec to select a font to use. The macro `\S@{size}` contains the definitions of the sizes used for maths letters, subscripts and subsubscripts in `\tf@size`, `\sf@size`, and `\ssf@size`, respectively.

```
837 \keys_set:nn {unicode-math} {#1}
838 \um_fontspec_select_font:n {#2}
```

Check for the correct number of `\fontdimens`:

```
839 %% \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
840 %% \bool_set_true:N \l_um_ot_math_bool
841 %% \else
842 %% \bool_set_false:N \l_um_ot_math_bool
843 %% \PackageWarningNoLine{unicode-math}{
844 %% The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
845 %% Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
846 %% in~ a~ substandard~ manner
847 %% }
848 %% \fi
```

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

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

```
849 \bool_if:NTF \l_um_init_bool {
850 \tl_set:Nn \um_symfont_tl {um_allsym}
851 \msg_trace:nnx {unicode-math} {default-math-font} {#2}
852 \cs_set_eq:NN \um_sym:nnn \um_process_symbol_noparse:nnn
853 \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
854 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
855 \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
856 \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
857 \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_noparse:nn
858 }{
859 \int_incr:N \g_um_fam_int
860 \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
861 \cs_set_eq:NN \um_sym:nnn \um_process_symbol_parse:nnn
```

```

862 \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_parse:Nnn
863 \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
864 \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
865 \cs_set_eq:NN \um_map_char_single:nn \um_map_char_parse:nn
866 \cs_set_eq:NN \um_assign_delcode:nn \um_assign_delcode_parse:nn
867 }

```

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

```

868 \DeclareSymbolFont{\um_symfont_t1}
869 {\encodingdefault}{\zf@family}{\mddefault}{\updefault}

```

And now we input every single maths char.

```

870 \um_input_math_symbol_table:

```

Finally,

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

```

871 \um_remap_symbols:
872 \um_setup_mathactives:
873 \um_setup_delcodes:
874 \um_setup_alphabets:

```

Prevent spaces:

```

875 \ignorespaces
876 }

```

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

```

877 \cs_new:Npn \um_fontspec_select_font:n #1 {
878 \bool_set_true:N \l_um_fontspec_feature_bool
879 \fontspec_select:xn
880 {
881 \luatex_if_engine:T { Renderer = Basic, }
882 BoldFont = {}, ItalicFont = {},
883 Script = Math,
884 SizeFeatures = {
885 {Size = \tf@size-} ,
886 {Size = \sf@size-\tf@size ,
887 Font = \l_um_script_font_t1 ,
888 \l_um_script_features_t1
889 } ,
890 {Size = -\sf@size ,
891 Font = \l_um_sscript_font_t1 ,

```

```

892         \l_um_sscript_features_tl
893     },
894     \l_um_unknown_keys_clist
895 }
896 {#1}
897 \tl_set_eq:NN \l_um_font \zf@basefont
898 \bool_set_false:N \l_um_fontspec_feature_bool
899 }
900 }

```

7.3.1 Functions for setting up symbols with mathcodes

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

`\um_process_symbol_parse:nnn`

```

901 \cs_set:Npn \um_process_symbol_noparse:nnn #1#2#3 {
902   \um_set_mathsymbol:nNNn {\um_symfont_tl} #2#3{#1}
903 }
904 \cs_set:Npn \um_process_symbol_parse:nnn #1#2#3 {
905   \um@parse@term{#1}{#2}{#3}{
906     \um_process_symbol_noparse:nnn {#1}{#2}{#3}
907   }
908 }

```

`\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.

```

909 \cs_new:Npn \um_remap_symbols: {
910   \um_remap_symbol:nnn{`-}{\mathbin}{"02212}% hyphen to minus
911   \um_remap_symbol:nnn{`*}{\mathbin}{"02217}% text asterisk to "cen-
    tred asterisk"
912   \bool_if:NF \g_um_literal_colon_bool {
913     \um_remap_symbol:nnn{`:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
914   }
915 }

```

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

```

916 \cs_new:Npn \um_remap_symbol_parse:nnn #1#2#3 {
917   \um@parse@term {#3} {\@nil} {#2} {
918     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
919   }
920 }
921 \cs_new:Npn \um_remap_symbol_noparse:nnn #1#2#3 {
922   \clist_map_inline:nn {#1} {
923     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
924   }
925 }

```

7.3.2 Active math characters

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

`\um_setup_mathactives:`

```
926 \cs_new:Npn \um_setup_mathactives: {  
927   \um_make_mathactive:nNN {"2032} \um_prime_single_mchar \mathord  
928   \um_make_mathactive:nNN {"2033} \um_prime_double_mchar \mathord  
929   \um_make_mathactive:nNN {"2034} \um_prime_triple_mchar \mathord  
930   \um_make_mathactive:nNN {"2057} \um_prime_quad_mchar \mathord  
931   \um_make_mathactive:nNN {"2035} \um_backprime_single_mchar \mathord  
932   \um_make_mathactive:nNN {"2036} \um_backprime_double_mchar \mathord  
933   \um_make_mathactive:nNN {"2037} \um_backprime_triple_mchar \mathord  
934   \um_make_mathactive:nNN {\`}` \mathstraightquote \mathord  
935   \um_make_mathactive:nNN {\`\`}` \mathbacktick \mathord  
936 }
```

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

```
937 \cs_new:Npn \um_make_mathactive:nNN #1#2#3 {  
938   \um_set_mathchar:NNnn #2 #3 {\um_symfont_t1} {#1}  
939   \char_gmake_mathactive:n {#1}  
940 }
```

7.3.3 Delimiter codes

`\um_assign_delcode:nn` : TODO : hook csnames into range feature

```
941 \cs_new:Npn \um_assign_delcode_noparse:nn #1#2 {  
942   \um_set_delcode:nnn \um_symfont_t1 {#1} {#2}  
943 }  
944 \cs_new:Npn \um_assign_delcode_parse:nn #1#2 {  
945   \um@parse@term {#2}{\@nil}{\@nil} {  
946     \um_assign_delcode_noparse:nn {#1} {#2}  
947   }  
948 }
```

`\um_assign_delcode:n` Shorthand.

```
949 \cs_new:Npn \um_assign_delcode:n #1 {  
950   \um_assign_delcode:nn {#1} {#1}  
951 }
```

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:`

```
952 \cs_new:Npn \um_setup_delcodes: {  
953   \um_assign_delcode:nn {\`\/} {\g_um_slash_delimiter_usv}  
954   \um_assign_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash  
955   \um_assign_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash  
956   \um_assign_delcode:n {"005C} % backslash  
957   \um_assign_delcode:nn {\`<} {"27E8} % angle brackets with ascii notation  
958   \um_assign_delcode:nn {\`>} {"27E9} % angle brackets with ascii notation  
959   \um_assign_delcode:n {"2191} % up arrow  
960   \um_assign_delcode:n {"2193} % down arrow  
961   \um_assign_delcode:n {"2195} % updown arrow  
962   \um_assign_delcode:n {"219F} % up arrow twohead  
963   \um_assign_delcode:n {"21A1} % down arrow twohead  
964   \um_assign_delcode:n {"21A5} % up arrow from bar  
965   \um_assign_delcode:n {"21A7} % down arrow from bar  
966   \um_assign_delcode:n {"21A8} % updown arrow from bar  
967   \um_assign_delcode:n {"21BE} % up harpoon right  
968   \um_assign_delcode:n {"21BF} % up harpoon left  
969   \um_assign_delcode:n {"21C2} % down harpoon right  
970   \um_assign_delcode:n {"21C3} % down harpoon left  
971   \um_assign_delcode:n {"21C5} % arrows up down  
972   \um_assign_delcode:n {"21F5} % arrows down up  
973   \um_assign_delcode:n {"21C8} % arrows up up  
974   \um_assign_delcode:n {"21CA} % arrows down down  
975   \um_assign_delcode:n {"21D1} % double up arrow  
976   \um_assign_delcode:n {"21D3} % double down arrow  
977   \um_assign_delcode:n {"21D5} % double updown arrow  
978   \um_assign_delcode:n {"21DE} % up arrow double stroke  
979   \um_assign_delcode:n {"21DF} % down arrow double stroke  
980   \um_assign_delcode:n {"21E1} % up arrow dashed  
981   \um_assign_delcode:n {"21E3} % down arrow dashed  
982   \um_assign_delcode:n {"21E7} % up white arrow  
983   \um_assign_delcode:n {"21E9} % down white arrow  
984   \um_assign_delcode:n {"21EA} % up white arrow from bar  
985   \um_assign_delcode:n {"21F3} % updown white arrow  
986 }
```

7.4 (Big) operators

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

However, the limits aren't set automatically; that is, we want to define, a la Plain \TeX *etc.*, `\def\int{\intop\nolimits}`, so there needs to be a transformation from `\int` to `\intop` during the expansion of `_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 $\int\!\!\!\int$, but that might be a matter of preference.

```

987 \tl_new:Nn \l_um_nolimits_tl {
988   \int\iint\iiint\iiiiint\oint\oiint\oiint
989   \intclockwise\varointclockwise\ointctrclockwise\sumint
990   \intbar\intBar\find\cirfnint\awint\rppoint
991   \scpoint\ncpoint\pointint\sqint\intlarhk\intx
992   \intcap\intcup\upoint\lowint
993 }

```

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

```

994 \DeclareDocumentCommand \addnolimits {m} {
995   \tl_put_right:Nn \l_um_nolimits_tl {#1}
996 }

```

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

```

997 \DeclareDocumentCommand \removenolimits {m} {
998   \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
999 }

```

7.5 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?

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

```

1000 \tl_new:Nn \l_um_radicals_tl {\sqrt}

```

7.6 Delimiters

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

<http://www.latex-project.org/cgi-bin/ltbugs2html?pr=latex/3853&prlatex/3754>

```

1001 \let\left@primitive\left
1002 \def\left{\mathopen{}}\left@primitive}

```

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

7.7 Maths accents

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

8 Font features

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

```
1003 \newcommand\um@zf@feature[2]{
1004   \define@key[zf]{options}{#1}[]{
1005     \bool_if:NTF \l_um_fontspec_feature_bool {
1006       #2
1007     }{
1008       \um_warning:n {maths-feature-only}
1009     }
1010   }
1011 }
```

8.1 OpenType maths font features

```
1012 \xetex_or luatex:nnn { \um@zf@feature {ScriptStyle} }
1013 { \zf@update@ff{+ssty=0} }
1014 { \zf@update@ff{+ssty=1} }
1015 \xetex_or luatex:nnn { \um@zf@feature {ScriptScriptStyle} }
1016 { \zf@update@ff{+ssty=1} }
1017 { \zf@update@ff{+ssty=2} }
```

8.2 Script and scriptscript font options

```
1018 \keys_define:nn {unicode-math}
1019 {
1020   script-features .tl_set:N = \l_um_script_features_tl ,
1021   sscript-features .tl_set:N = \l_um_sscript_features_tl ,
1022   script-font .tl_set:N = \l_um_script_font_tl ,
1023   sscript-font .tl_set:N = \l_um_sscript_font_tl ,
1024 }
```

8.3 Range processing

```
1025 \seq_new:N \l_um_mathalph_seq
1026 \seq_new:N \l_um_char_range_seq
1027 \keys_define:nn {unicode-math} {
1028   range .code:n = {
1029     \bool_set_false:N \l_um_init_bool
1030     \seq_clear:N \l_um_char_range_seq
1031     \seq_clear:N \l_um_mathalph_seq
```

```

1032 \clist_map_inline:nn {#1} {
1033   \um_if_mathalph_decl:nTF {##1} {
1034     \seq_put_right:Nx \l_um_mathalph_seq {
1035       { \exp_not:V \l_um_tmpa_tl }
1036       { \exp_not:V \l_um_tmpb_tl }
1037       { \exp_not:V \l_um_tmpc_tl }
1038     }
1039   }{
1040     \seq_put_right:Nn \l_um_char_range_seq {##1}
1041   }
1042 }
1043 }
1044 }

```

\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.

```

1045 \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
1046   \KV_remove_surrounding_spaces:nw {\tl_set:Nf\l_um_tmpa_tl} #1 \q_nil
1047   \tl_clear:N \l_um_tmpb_tl
1048   \tl_clear:N \l_um_tmpc_tl
1049   \tl_if_in:NnT \l_um_tmpa_tl {->} {
1050     \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
1051   }
1052   \tl_if_in:NnT \l_um_tmpa_tl {/} {
1053     \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
1054   }
1055   \tl_if_empty:NT \l_um_tmpc_tl { \tl_set_eq:NN \l_um_tmpc_tl \l_um_tmpa_tl }
1056   \seq_if_in:NVTF \g_um_mathstyles_seq \l_um_tmpa_tl {
1057     \prg_return_true:
1058   }{
1059     \prg_return_false:
1060   }
1061 }
1062 \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
1063   \tl_set:Nn \l_um_tmpa_tl {#1}
1064   \tl_if_single:nTF {#2}
1065     { \tl_set:Nn \l_um_tmpc_tl {#2} }

```



```

1066 { \exp_args:Nnc \tl_set:Nn \l_um_tmpe_tl {math#2} }
1067 }
1068 \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
1069   \tl_set:Nn \l_um_tmpe_tl {#1}
1070   \tl_set:Nn \l_um_tmpe_tl {#2}
1071 }

```

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

`\um@parse@term` #1 : Unicode character slot
 #2 : control sequence (character macro)
 #3 : control sequence (math type)
 #4 : code to execute

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

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

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

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

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

```

1072 \newcommand\um@parse@term[4]{
1073   \seq_map_variable:NNn \l_um_char_range_seq \@ii {
1074     \unless\ifx\@ii\@empty
1075       \@tempswafalse

```

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

```

1076   \expandafter\um@firstchar\expandafter{\@ii}
1077   \ifx\@tempa\um@backslash
1078     \expandafter\ifx\@ii#2\relax
1079     \@tempswatrue
1080   \else
1081     \expandafter\ifx\@ii#3\relax
1082     \@tempswatrue
1083   \fi
1084 \fi

```

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

```

1085     \else
1086     \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
1087     \fi

```

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

```

1088     \if@tempswa
1089     \clist_put_right:Nx \l_um_char_num_range_clist { \int-
1090     expr_eval:n {#1} }
1091     #4
1092     \fi
1093     \fi
1094 }
1095 \def\um@firstof#1#2\@nil{#1}
1096 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
1097 \def\um@firstchar#1{\edef\@tempa{\expandafter\um@firstof\string#1\@nil}}

```

`\um@parse@range` Weird syntax. As shown previously in table 11, this macro can be passed four different input types via `\um@parse@term`.

```

1098 \def\um@parse@range#1-#2-#3\@nil#4\@nil{
1099   \def\@tempa{#1}
1100   \def\@tempb{#2}

```

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

```

1101   \expandafter\ifx\expandafter\@marker\@tempb\relax
1102   \intexpr_compare:nT {#4=#1} \@tempswatrue
1103   \else

```

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

```

1104   \ifx\@empty\@tempb
1105   \intexpr_compare:nT {#4>#1-1} \@tempswatrue
1106   \else

```

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

```

1107   \ifx\@empty\@tempa
1108   \intexpr_compare:nT {#4<#2+1} \@tempswatrue

```

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

```

1109     \else
1110         \intexpr_compare:nT {#4>#1-1} {
1111             \intexpr_compare:nT {#4<#2+1} \@tempswatrue
1112         }
1113     \fi
1114 \fi
1115 \fi
1116 }

```

8.4 Resolving Greek symbol name control sequences

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

```

1117 \AtBeginDocument{\um_resolve_greek:}
1118 \cs_new:Npn \um_resolve_greek: {
1119     \clist_map_inline:nn {
1120         Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1121         alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
1122         Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1123         mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,    chi,psi,omega,
1124         varTheta,
1125         varsigma,vartheta,varkappa,varrho,varpi
1126     }{
1127         \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
1128     }
1129     \tl_set:Nn \epsilon {
1130         \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitempsilon
1131     }
1132     \tl_set:Nn \phi {
1133         \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
1134     }
1135     \tl_set:Nn \varepsilon {
1136         \bool_if:NTF \g_um_texgreek_bool \mitempsilon \mitvarepsilon
1137     }
1138     \tl_set:Nn \varphi {
1139         \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
1140     }
1141 }

```

9 Maths alphabets mapping definitions

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

Implicit mode:

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

Explicit mode:

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

9.1 Initialising math styles

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

```

1142 \cs_new:Npn \um_new_mathstyle:N #1 {
1143   \um_prepare_mathstyle:f {\exp_after:wN \use_none:nnnnn \token_to_str:N #1}
1144   \seq_put_right:Nn \g_um_mathstyles_seq {#1}
1145 }
```

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

```

1146 \seq_new:N \g_um_default_mathalph_seq
```

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

```

1147 \seq_new:N \g_um_mathstyles_seq

1148 \AtEndOfPackage{
1149   \clist_map_inline:nn {
1150     {\mathup    } {\latin, Latin, greek, Greek, num, misc} {\mathup    } ,
1151     {\mathit    } {\latin, Latin, greek, Greek, misc}     {\mathit    } ,
1152     {\mathbb    } {\latin, Latin, num, misc}               {\mathbb    } ,
1153     {\mathbbbit } {\misc}                                 {\mathbbbit } ,
```

```

1154 {\mathscr } {latin, Latin} {\mathscr } ,
1155 {\mathcal } {Latin} {\mathscr } ,
1156 {\mathbfcal } {Latin} {\mathbfscr } ,
1157 {\mathfrak } {latin, Latin} {\mathfrak } ,
1158 {\mathtt } {latin, Latin, num} {\mathtt } ,
1159 {\mathsfup } {latin, Latin, num} {\mathsfup } ,
1160 {\mathsfit } {latin, Latin} {\mathsfit } ,
1161 {\mathbfup } {latin, Latin, greek, Greek, num, misc} {\mathbfup } ,
1162 {\mathbfit } {latin, Latin, greek, Greek, misc} {\mathbfit } ,
1163 {\mathbfscr } {latin, Latin} {\mathbfscr } ,
1164 {\mathbffrac } {latin, Latin} {\mathbffrac } ,
1165 {\mathbfsfup } {latin, Latin, greek, Greek, num, misc} {\mathbfsfup } ,
1166 {\mathbfsfit } {latin, Latin, greek, Greek, misc} {\mathbfsfit }
1167 }{
1168 \seq_put_right:Nn \g_um_default_mathalph_seq {#1}
1169 \exp_after:wN \um_new_mathstyle:N \use_i:nnn #1
1170 }

```

These are ‘false’ mathstyles that inherit other definitions:

```

1171 \um_new_mathstyle:N \mathsf
1172 \um_new_mathstyle:N \mathbf
1173 \um_new_mathstyle:N \mathbfsf
1174 }

```

9.2 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.

```

1175 \cs_new:Npn \um_prepare_mathstyle:n #1 {
1176   \um_init_alphabet:x {#1}
1177   \cs_set:cpn {_um_math#1_aux:n} ##1 {
1178     \use:c {um_switchto_math#1:} ##1 \egroup
1179   }
1180   \cs_set_protected:cpn {math#1} {
1181     \exp_not:n{
1182       \bgroup
1183       \mode_if_math:F {
1184         \egroup\expandafter
1185         \non@alpherr\expandafter{\csname math#1\endcsname\space}
1186       }

```

```

1187     }
1188     \exp_not:c {_um_math#1_aux:n}
1189   }
1190 }
1191 \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.

```

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

```

Variants

```

1197 \cs_new:Npn \um_maybe_init_alphabet:V {
1198   \exp_args:NV \um_maybe_init_alphabet:n
1199 }

```

9.3 Defining the math alphabets per style

Variables:

```

1200 \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.

```

1201 \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:

```

1202   \seq_if_empty:NTF \l_um_mathalph_seq {
1203     \um_trace:n {setup-implicit}
1204     \seq_set_eq:NN \l_um_mathalph_seq \g_um_default_mathalph_seq
1205     \bool_set_true:N \l_um_implicit_alph_bool
1206     \um_maybe_init_alphabet:n {sf}
1207     \um_maybe_init_alphabet:n {bf}
1208     \um_maybe_init_alphabet:n {bfsf}
1209   }

```

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

```

1210   {
1211     \um_trace:n {setup-explicit}
1212     \bool_set_false:N \l_um_implicit_alph_bool
1213     \cs_set_eq:NN \um_set_mathalphabet_char:Nnn \um_mathmap_noparse:Nnn
1214     \cs_set_eq:NN \um_map_char_single:nn \um_map_char_noparse:nn
1215   }

```

Now perform the mapping:

```

1216 \seq_map_inline:Nn \l_um_mathalph_seq {
1217   \tl_set:No \l_um_tmpa_tl { \use_i:nnn ##1 }
1218   \tl_set:No \l_um_tmppb_tl { \use_ii:nnn ##1 }
1219   \tl_set:No \l_um_remap_style_tl { \use_iii:nnn ##1 }
1220   \tl_set:Nx \l_um_remap_style_tl {
1221     \exp_after:wN \exp_after:wN \exp_after:wN \use_none:nnnnn
1222     \exp_after:wN \token_to_str:N \l_um_remap_style_tl
1223   }
1224   \tl_if_empty:NT \l_um_tmppb_tl {
1225     \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
1226     \tl_set:Nn \l_um_tmppb_tl { latin, Latin, greek, Greek, num, misc }
1227   }
1228   \um_setup_math_alphabet:VVV
1229   \l_um_tmpa_tl \l_um_tmppb_tl \l_um_remap_style_tl
1230 }
1231 \um_warn_missing_alphabets:
1232 }

1233 \cs_new:Npn \um_warn_missing_alphabets: {
1234   \seq_if_empty:NF \l_um_missing_alph_seq {
1235     \typeout{
1236       Package~unicode-math~Warning:~
1237       missing~math~alphabets~in~font~ \fontname\l_um_font
1238     }
1239     \seq_map_inline:Nn \l_um_missing_alph_seq {
1240       \typeout{\space\space\space\space##1}
1241     }
1242   }
1243 }

```

`\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`)

```

1244 \cs_new:Npn \um_setup_math_alphabet:Nnn #1#2#3 {
1245   \tl_set:Nx \l_um_style_tl {
1246     \exp_after:wN \use_none:nnnnn \token_to_str:N #1
1247   }

```

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

```

1248 \clist_map_inline:nn {#2} {
1249   \cs_if_exist:cT {um_config_ \l_um_style_tl _##1:n} {
1250     \tl_if_eq:nnTF {##1}{misc} {
1251       \um_maybe_init_alphabet:V \l_um_style_tl
1252       \clist_map_break:
1253     }{
1254       \um_glyph_if_exist:cT { \um_to_usv:nn {#3}{##1} }{

```

```

1255         \um_maybe_init_alphabet:V \l_um_style_tl
1256         \clist_map_break:
1257     }
1258 }
1259 }
1260 }

```

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

```

1261 \clist_map_inline:nn {#2} {
1262     \cs_if_exist:cT {um_config_ \l_um_style_tl _##1:n} {
1263         \tl_if_eq:nnTF {##1}{misc} {
1264             \um_trace:nx {setup-alph} {math \l_um_style_tl~(##1)}
1265             \use:c {um_config_ \l_um_style_tl _##1:n} {#3}
1266         }{
1267             \um_glyph_if_exist:cTF { \um_to_usv:nn {#3}{##1} } {
1268                 \um_trace:nx {setup-alph} {math \l_um_style_tl~(##1)}
1269                 \use:c {um_config_ \l_um_style_tl _##1:n} {#3}
1270             }{
1271                 \bool_if:NTF \l_um_implicit_alph_bool {
1272                     \seq_put_right:Nx \l_um_missing_alph_seq {
1273                         \@backslashchar math \l_um_style_tl \space
1274                         (\tl_use:c{g_um_math_alphabet_name_##1_tl})
1275                     }
1276                 }{
1277                     \use:c {um_config_ \l_um_style_tl _##1:n} {up}
1278                 }
1279             }
1280         }
1281     }
1282 }
1283 }
1284 \cs_generate_variant:Nn \um_setup_math_alphabet:Nnn {VVV}

```

9.4 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.

9.4.1 Functions

`\um_map_char_single:nn` Wrapper for `\um_map_char_noparse:nn` or `\um_map_char_parse:nn` depending on the context.

```

1285 \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`


```

1286 \cs_new:Npn \um_map_char_noparse:nn #1#2 {
1287   \um_set_mathcode:nnnn {#1}{\mathalpha}{\um_symfont_t1}{#2}
1288 }
1289 \cs_new:Npn \um_map_char_parse:nn #1#2 {
1290   \um@parse@term {#1} {\@nil} {\mathalpha} {
1291     \um_map_char_noparse:nn {#1}{#2}
1292   }
1293 }

```

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

```

1294 \cs_new:Npn \um_map_char_single:nnn #1#2#3 {
1295   \um_map_char_single:cc { \um_to_usv:nn {#1}{#3} }
1296   { \um_to_usv:nn {#2}{#3} }
1297 }
1298 \cs_set:Npn \um_map_single:nnn #1#2#3 {
1299   \cs_if_exist:cT { \um_to_usv:nn {#3} {#1} }
1300   {
1301     \clist_map_inline:nn {#2} {
1302       \um_map_char_single:nnn {##1} {#3} {#1}
1303     }
1304   }
1305 }

```

\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:

```

1306 \cs_set:Npn \um_map_chars_range:nnn #1#2#3 {
1307   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {
1308     \um_map_char_single:nn {#2+##1}{#3+##1}
1309   }
1310 }
1311 \cs_generate_variant:Nn \um_map_chars_range:nnn {ncc}

```

And the wrapper with names:

```

1312 \cs_new:Npn \um_map_chars_range:nnnn #1#2#3#4 {
1313   \um_map_chars_range:ncc {#1} { \um_to_usv:nn {#2}{#4} }
1314   { \um_to_usv:nn {#3}{#4} }
1315 }

```

9.4.2 Functions for alphabets

```

1316 \cs_set:Npn \um_map_chars_Latin:nn #1#2 {

```

```

1317 \clist_map_inline:nn {#1} {
1318   \um_map_chars_range:nnnn {26} {##1} {#2} {Latin}
1319 }
1320 }

1321 \cs_set:Npn \um_map_chars_latin:nn #1#2 {
1322   \clist_map_inline:nn {#1} {
1323     \um_map_chars_range:nnnn {26} {##1} {#2} {latin}
1324   }
1325 }

1326 \cs_set:Npn \um_map_chars_greek:nn #1#2 {
1327   \clist_map_inline:nn {#1} {
1328     \um_map_chars_range:nnnn {25} {##1} {#2} {greek}
1329     \um_map_char_single:nnn {##1} {#2} {\varepsilon}
1330     \um_map_char_single:nnn {##1} {#2} {\vartheta}
1331     \um_map_char_single:nnn {##1} {#2} {\varkappa}
1332     \um_map_char_single:nnn {##1} {#2} {\varphi}
1333     \um_map_char_single:nnn {##1} {#2} {\varrho}
1334     \um_map_char_single:nnn {##1} {#2} {\varpi}
1335   }
1336 }

1337 \cs_set:Npn \um_map_chars_Greek:nn #1#2 {
1338   \clist_map_inline:nn {#1} {
1339     \um_map_chars_range:nnnn {25} {##1} {#2} {Greek}
1340     \um_map_char_single:nnn {##1} {#2} {\varTheta}
1341   }
1342 }

1343 \cs_set:Npn \um_map_chars_numbers:nn #1#2 {
1344   \um_map_chars_range:nnnn {10} {#1} {#2} {num}
1345 }

```

9.5 Mapping chars inside a math style

9.5.1 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.

```

1346 \cs_new:Npn \um_set_mathalphabet_char:Ncc {
1347   \exp_args:NNcc \um_set_mathalphabet_char:Nnn
1348 }

```

`\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.

```

1349 \cs_set:Npn \um_mathmap_noparse:Nnn #1#2#3 {
1350   \clist_map_inline:nn {#2} {
1351     \tl_put_right:cx {um_switchto_\cs_to_str:N #1:} {
1352       \um_set_mathcode:nnnn{##1}{\mathalpha}{\um_symfont_tl}{#3}
1353     }
1354   }
1355 }

```

`\um_mathmap_parse:Nnn` #1 : Maths alphabet, *e.g.*, `\mathbb`
 #2 : Input slot(s), *e.g.*, the slot for ‘A’ (comma separated)
 #3 : Output slot, *e.g.*, the slot for ‘A’

When `\um@parse@term` is executed, it populates the `\l_um_char_num_range_clist` macro with slot numbers corresponding to the specified range. This range is used to conditionally add `\um_set_mathcode:nnnn` declarations to the maths alphabet definition.

```

1356 \cs_set:Npn \um_mathmap_parse:Nnn #1#2#3 {
1357   \clist_if_in:NnT \l_um_char_num_range_clist {#3} {
1358     \um_mathmap_noparse:Nnn {#1}{#2}{#3}
1359   }
1360 }

```

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

```

1361 \cs_new:Npn \um_set_mathalphabet_char:Nnnn #1#2#3#4 {
1362   \um_set_mathalphabet_char:Ncc #1 { \um_to_usv:nn {#2} {#4} }
1363   { \um_to_usv:nn {#3} {#4} }
1364 }

```

`\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:

```

1365 \cs_new:Npn \um_set_mathalph_range:nNnn #1#2#3#4 {
1366   \prg_stepwise_inline:nnnn {0}{1}{#1-1} {
1367     \um_set_mathalphabet_char:Nnn {#2} { ##1 + #3 } { ##1 + #4 }
1368   }
1369 }
1370 \cs_generate_variant:Nn \um_set_mathalph_range:nNnn {nNcc}

```

Then the wrapper version that uses names:

```

1371 \cs_new:Npn \um_set_mathalph_range:nNnnn #1#2#3#4#5 {
1372   \um_set_mathalph_range:Ncc {#1} #2 { \um_to_usv:nn {#3} {#5} }

```

```

1373 { \um_to_usv:nn {#4} {#5} }
1374 }

```

9.5.2 Individual mapping functions for different alphabets

```

1375 \cs_new:Npn \um_set_mathalphabet_pos:Nnnn #1#2#3#4 {
1376   \cs_if_exist:cT { \um_to_usv:nn {#4}{#2} } {
1377     \clist_map_inline:nn {#3} {
1378       \um_set_mathalphabet_char:Nnnn #1 {##1} {#4} {#2}
1379     }
1380   }
1381 }

1382 \cs_new:Npn \um_set_mathalphabet_numbers:Nnn #1#2#3 {
1383   \clist_map_inline:nn {#2} {
1384     \um_set_mathalph_range:nNnnn {10} #1 {##1} {#3} {num}
1385   }
1386 }

1387 \cs_new:Npn \um_set_mathalphabet_Latin:Nnn #1#2#3 {
1388   \clist_map_inline:nn {#2} {
1389     \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {Latin}
1390   }
1391 }

1392 \cs_new:Npn \um_set_mathalphabet_latin:Nnn #1#2#3 {
1393   \clist_map_inline:nn {#2} {
1394     \um_set_mathalph_range:nNnnn {26} #1 {##1} {#3} {latin}
1395     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {h}
1396   }
1397 }

1398 \cs_new:Npn \um_set_mathalphabet_Greek:Nnn #1#2#3 {
1399   \clist_map_inline:nn {#2} {
1400     \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {Greek}
1401     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varTheta}
1402   }
1403 }

1404 \cs_new:Npn \um_set_mathalphabet_greek:Nnn #1#2#3 {
1405   \clist_map_inline:nn {#2} {
1406     \um_set_mathalph_range:nNnnn {25} #1 {##1} {#3} {greek}
1407     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varepsilon}
1408     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {vartheta}
1409     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varkappa}
1410     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varphi}
1411     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varrho}
1412     \um_set_mathalphabet_char:Nnnn #1 {##1} {#3} {varpi}
1413   }
1414 }

```

9.6 Alphabets

9.6.1 Upright: \mathup

```
1415 \cs_new:Npn \um_config_up_num:n #1 {
1416   \um_map_chars_numbers:nn {up}{#1}
1417   \um_set_mathalphabet_numbers:Nnn \mathup {up}{#1}
1418 }
1419 \cs_new:Npn \um_config_up_Latin:n #1 {
1420   \bool_if:NTF \g_um_literal_bool {
1421     \um_map_chars_Latin:nn {up} {#1}
1422   }{
1423     \bool_if:NT \g_um_upLatin_bool {
1424       \um_map_chars_Latin:nn {up,it} {#1}
1425     }
1426   }
1427   \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{#1}
1428 }
1429 \cs_new:Npn \um_config_up_latin:n #1 {
1430   \bool_if:NTF \g_um_literal_bool {
1431     \um_map_chars_latin:nn {up} {#1}
1432   }{
1433     \bool_if:NT \g_um_uplatin_bool {
1434       \um_map_chars_latin:nn {up,it} {#1}
1435       \um_map_single:nnn {h} {up,it} {#1}
1436       \um_map_single:nnn {dotlessi} {up,it} {#1}
1437       \um_map_single:nnn {dotlessj} {up,it} {#1}
1438     }
1439   }
1440   \um_set_mathalphabet_latin:Nnn \mathup {up,it}{#1}
1441 }
1442 \cs_new:Npn \um_config_up_Greek:n #1 {
1443   \bool_if:NTF \g_um_literal_bool {
1444     \um_map_chars_Greek:nn {up}{#1}
1445   }{
1446     \bool_if:NT \g_um_upGreek_bool {
1447       \um_map_chars_Greek:nn {up,it}{#1}
1448     }
1449   }
1450   \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{#1}
1451 }
1452 \cs_new:Npn \um_config_up_greek:n #1 {
1453   \bool_if:NTF \g_um_literal_bool {
1454     \um_map_chars_greek:nn {up} {#1}
1455   }{
1456     \bool_if:NT \g_um_upgreek_bool {
1457       \um_map_chars_greek:nn {up,it} {#1}
```

```

1458     }
1459   }
1460   \um_set_mathalphabet_greek:Nnn \mathup {up,it} {#1}
1461 }
1462 \cs_new:Npn \um_config_up_misc:n #1 {
1463   \bool_if:NTF \g_um_literal_Nabla_bool {
1464     \um_map_single:nnn {Nabla}{up}{up}
1465   }{
1466     \bool_if:NT \g_um_upNabla_bool {
1467       \um_map_single:nnn {Nabla}{up,it}{up}
1468     }
1469   }
1470   \bool_if:NTF \g_um_literal_partial_bool {
1471     \um_map_single:nnn {partial}{up}{up}
1472   }{
1473     \bool_if:NT \g_um_uppartial_bool {
1474       \um_map_single:nnn {partial}{up,it}{up}
1475     }
1476   }
1477   \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it} {#1}
1478   \um_set_mathalphabet_pos:Nnnn \mathup {Nabla} {up,it} {#1}
1479   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessi} {up,it} {#1}
1480   \um_set_mathalphabet_pos:Nnnn \mathup {dotlessj} {up,it} {#1}
1481 }

```

9.6.2 Italic: `\mathit`

```

1482 \cs_new:Npn \um_config_it_Latin:n #1 {
1483   \bool_if:NTF \g_um_literal_bool {
1484     \um_map_chars_Latin:nn {it} {#1}
1485   }{
1486     \bool_if:NF \g_um_upLatin_bool {
1487       \um_map_chars_Latin:nn {up,it} {#1}
1488     }
1489   }
1490   \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{#1}
1491 }
1492 \cs_new:Npn \um_config_it_latin:n #1 {
1493   \bool_if:NTF \g_um_literal_bool {
1494     \um_map_chars_latin:nn {it} {#1}
1495     \um_map_single:nnn {h}{it}{#1}
1496   }{
1497     \bool_if:NF \g_um_uplatin_bool {
1498       \um_map_chars_latin:nn {up,it} {#1}
1499       \um_map_single:nnn {h}{up,it}{#1}
1500       \um_map_single:nnn {dotlessi}{up,it}{#1}
1501       \um_map_single:nnn {dotlessj}{up,it}{#1}

```

```

1502     }
1503   }
1504   \um_set_mathalphabet_latin:Nnn \mathit          {up,it} {#1}
1505   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessi} {up,it} {#1}
1506   \um_set_mathalphabet_pos:Nnnn \mathit {dotlessj} {up,it} {#1}
1507 }
1508 \cs_new:Npn \um_config_it_Greek:n #1 {
1509   \bool_if:NTF \g_um_literal_bool {
1510     \um_map_chars_Greek:nn {it}{#1}
1511   }{
1512     \bool_if:NF \g_um_upGreek_bool {
1513       \um_map_chars_Greek:nn {up,it}{#1}
1514     }
1515   }
1516   \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{#1}
1517 }
1518 \cs_new:Npn \um_config_it_greek:n #1 {
1519   \bool_if:NTF \g_um_literal_bool {
1520     \um_map_chars_greek:nn {it} {#1}
1521   }{
1522     \bool_if:NF \g_um_upgreek_bool {
1523       \um_map_chars_greek:nn {it,up} {#1}
1524     }
1525   }
1526   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {#1}
1527 }
1528 \cs_new:Npn \um_config_it_misc:n #1 {
1529   \bool_if:NTF \g_um_literal_Nabla_bool {
1530     \um_map_single:nnn {Nabla}{it}{it}
1531   }{
1532     \bool_if:NF \g_um_upNabla_bool {
1533       \um_map_single:nnn {Nabla}{up,it}{it}
1534     }
1535   }
1536   \bool_if:NTF \g_um_literal_partial_bool {
1537     \um_map_single:nnn {partial}{it}{it}
1538   }{
1539     \bool_if:NF \g_um_uppartial_bool {
1540       \um_map_single:nnn {partial}{up,it}{it}
1541     }
1542   }
1543   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{#1}
1544   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla} {up,it}{#1}
1545 }

```

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

```

1546 \cs_new:Npn \um_config_bb_latin:n #1 {
1547   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{#1}
1548 }
1549 \cs_new:Npn \um_config_bb_Latin:n #1 {
1550   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{#1}
1551   \um_set_mathalphabet_pos:Nnnn \mathbb {C} {up,it} {#1}
1552   \um_set_mathalphabet_pos:Nnnn \mathbb {H} {up,it} {#1}
1553   \um_set_mathalphabet_pos:Nnnn \mathbb {N} {up,it} {#1}
1554   \um_set_mathalphabet_pos:Nnnn \mathbb {P} {up,it} {#1}
1555   \um_set_mathalphabet_pos:Nnnn \mathbb {Q} {up,it} {#1}
1556   \um_set_mathalphabet_pos:Nnnn \mathbb {R} {up,it} {#1}
1557   \um_set_mathalphabet_pos:Nnnn \mathbb {Z} {up,it} {#1}
1558 }
1559 \cs_new:Npn \um_config_bb_num:n #1 {
1560   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{#1}
1561 }
1562 \cs_new:Npn \um_config_bb_misc:n #1 {
1563   \um_set_mathalphabet_pos:Nnnn \mathbb {Pi} {up,it} {#1}
1564   \um_set_mathalphabet_pos:Nnnn \mathbb {pi} {up,it} {#1}
1565   \um_set_mathalphabet_pos:Nnnn \mathbb {\Gamma} {up,it} {#1}
1566   \um_set_mathalphabet_pos:Nnnn \mathbb {\gamma} {up,it} {#1}
1567   \um_set_mathalphabet_pos:Nnnn \mathbb {summation} {up} {#1}
1568 }
1569 \cs_new:Npn \um_config_bbit_misc:n #1 {
1570   \um_set_mathalphabet_pos:Nnnn \mathbbit {D} {up,it} {#1}
1571   \um_set_mathalphabet_pos:Nnnn \mathbbit {d} {up,it} {#1}
1572   \um_set_mathalphabet_pos:Nnnn \mathbbit {e} {up,it} {#1}
1573   \um_set_mathalphabet_pos:Nnnn \mathbbit {i} {up,it} {#1}
1574   \um_set_mathalphabet_pos:Nnnn \mathbbit {j} {up,it} {#1}
1575 }

```

9.6.4 Script and caligraphic: `\mathscr` and `\mathcal`

```

1576 \cs_new:Npn \um_config_scr_Latin:n #1 {
1577   \um_set_mathalphabet_Latin:Nnn \mathscr {up,it}{#1}
1578   \um_set_mathalphabet_pos:Nnnn \mathscr {B}{up,it}{#1}
1579   \um_set_mathalphabet_pos:Nnnn \mathscr {E}{up,it}{#1}
1580   \um_set_mathalphabet_pos:Nnnn \mathscr {F}{up,it}{#1}
1581   \um_set_mathalphabet_pos:Nnnn \mathscr {H}{up,it}{#1}
1582   \um_set_mathalphabet_pos:Nnnn \mathscr {I}{up,it}{#1}
1583   \um_set_mathalphabet_pos:Nnnn \mathscr {L}{up,it}{#1}
1584   \um_set_mathalphabet_pos:Nnnn \mathscr {M}{up,it}{#1}
1585   \um_set_mathalphabet_pos:Nnnn \mathscr {R}{up,it}{#1}
1586 }
1587 \cs_new:Npn \um_config_scr_latin:n #1 {
1588   \um_set_mathalphabet_latin:Nnn \mathscr {up,it}{#1}
1589   \um_set_mathalphabet_pos:Nnnn \mathscr {e}{up,it}{#1}

```



```

1590 \um_set_mathalphabet_pos:Nnnn \mathscr {g}{up,it}{#1}
1591 \um_set_mathalphabet_pos:Nnnn \mathscr {o}{up,it}{#1}
1592 }

```

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

```

1593 \cs_new:Npn \um_config_cal_Latin:n #1 {
1594   \um_set_mathalphabet_Latin:Nnn \mathcal {up,it}{#1}
1595   \um_set_mathalphabet_pos:Nnnn \mathcal {B}{up,it}{#1}
1596   \um_set_mathalphabet_pos:Nnnn \mathcal {E}{up,it}{#1}
1597   \um_set_mathalphabet_pos:Nnnn \mathcal {F}{up,it}{#1}
1598   \um_set_mathalphabet_pos:Nnnn \mathcal {H}{up,it}{#1}
1599   \um_set_mathalphabet_pos:Nnnn \mathcal {I}{up,it}{#1}
1600   \um_set_mathalphabet_pos:Nnnn \mathcal {L}{up,it}{#1}
1601   \um_set_mathalphabet_pos:Nnnn \mathcal {M}{up,it}{#1}
1602   \um_set_mathalphabet_pos:Nnnn \mathcal {R}{up,it}{#1}
1603 }

```

9.6.5 Fraktur or fraktur or blackletter: `\mathfrak`

```

1604 \cs_new:Npn \um_config_frak_Latin:n #1 {
1605   \um_set_mathalphabet_Latin:Nnn \mathfrak {up,it}{#1}
1606   \um_set_mathalphabet_pos:Nnnn \mathfrak {C}{up,it}{#1}
1607   \um_set_mathalphabet_pos:Nnnn \mathfrak {H}{up,it}{#1}
1608   \um_set_mathalphabet_pos:Nnnn \mathfrak {I}{up,it}{#1}
1609   \um_set_mathalphabet_pos:Nnnn \mathfrak {R}{up,it}{#1}
1610   \um_set_mathalphabet_pos:Nnnn \mathfrak {Z}{up,it}{#1}
1611 }
1612 \cs_new:Npn \um_config_frak_latin:n #1 {
1613   \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{#1}
1614 }

```

9.6.6 Sans serif upright: `\mathsfup`

```

1615 \cs_new:Npn \um_config_sfup_num:n #1 {
1616   \um_set_mathalphabet_numbers:Nnn \mathsf {up}{#1}
1617   \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{#1}
1618 }
1619 \cs_new:Npn \um_config_sfup_Latin:n #1 {
1620   \bool_if:NTF \g_um_sfliteral_bool {
1621     \um_map_chars_Latin:nn {sfup} {#1}
1622     \um_set_mathalphabet_Latin:Nnn \mathsf {up}{#1}
1623   }{
1624     \bool_if:NT \g_um_upsans_bool {
1625       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1626       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1627     }
1628   }
1629   \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{#1}

```

```

1630 }
1631 \cs_new:Npn \um_config_sfup_latin:n #1 {
1632   \bool_if:NTF \g_um_sfliteral_bool {
1633     \um_map_chars_latin:nn {sfup} {#1}
1634     \um_set_mathalphabet_latin:Nnn \mathsf {up}{#1}
1635   }{
1636     \bool_if:NT \g_um_upsans_bool {
1637       \um_map_chars_latin:nn {sfup,sfit} {#1}
1638       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1639     }
1640   }
1641   \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{#1}
1642 }

```

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

```

1643 \cs_new:Npn \um_config_sfit_Latin:n #1 {
1644   \bool_if:NTF \g_um_sfliteral_bool {
1645     \um_map_chars_Latin:nn {sfit} {#1}
1646     \um_set_mathalphabet_Latin:Nnn \mathsf {it}{#1}
1647   }{
1648     \bool_if:NF \g_um_upsans_bool {
1649       \um_map_chars_Latin:nn {sfup,sfit} {#1}
1650       \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{#1}
1651     }
1652   }
1653   \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{#1}
1654 }
1655 \cs_new:Npn \um_config_sfit_latin:n #1 {
1656   \bool_if:NTF \g_um_sfliteral_bool {
1657     \um_map_chars_latin:nn {sfit} {#1}
1658     \um_set_mathalphabet_latin:Nnn \mathsf {it}{#1}
1659   }{
1660     \bool_if:NF \g_um_upsans_bool {
1661       \um_map_chars_latin:nn {sfup,sfit} {#1}
1662       \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{#1}
1663     }
1664   }
1665   \um_set_mathalphabet_latin:Nnn \mathsfit {up,it}{#1}
1666 }

```

9.6.8 Typewriter or monospaced: `\mathtt`

```

1667 \cs_new:Npn \um_config_tt_num:n #1 {
1668   \um_set_mathalphabet_numbers:Nnn \mathtt {up}{#1}
1669 }
1670 \cs_new:Npn \um_config_tt_Latin:n #1 {
1671   \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{#1}
1672 }

```

```

1673 \cs_new:Npn \um_config_tt_latin:n #1 {
1674   \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{#1}
1675 }

```

9.6.9 Bold Italic: `\mathbfit`

```

1676 \cs_new:Npn \um_config_bfit_Latin:n #1 {
1677   \bool_if:NF \g_um_bfupLatin_bool {
1678     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1679   }
1680   \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{#1}
1681   \bool_if:NTF \g_um_bfliteral_bool {
1682     \um_map_chars_Latin:nn {bfit} {#1}
1683     \um_set_mathalphabet_Latin:Nnn \mathbf {it}{#1}
1684   }{
1685     \bool_if:NF \g_um_bfupLatin_bool {
1686       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1687       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1688     }
1689   }
1690 }
1691 \cs_new:Npn \um_config_bfit_latin:n #1 {
1692   \bool_if:NF \g_um_bfuplatin_bool {
1693     \um_map_chars_latin:nn {bfup,bfit} {#1}
1694   }
1695   \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{#1}
1696   \bool_if:NTF \g_um_bfliteral_bool {
1697     \um_map_chars_latin:nn {bfit} {#1}
1698     \um_set_mathalphabet_latin:Nnn \mathbf {it}{#1}
1699   }{
1700     \bool_if:NF \g_um_bfuplatin_bool {
1701       \um_map_chars_latin:nn {bfup,bfit} {#1}
1702       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1703     }
1704   }
1705 }
1706 \cs_new:Npn \um_config_bfit_Greek:n #1 {
1707   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{#1}
1708   \bool_if:NTF \g_um_bfliteral_bool {
1709     \um_map_chars_Greek:nn {bfit}{#1}
1710     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{#1}
1711   }{
1712     \bool_if:NF \g_um_bfupGreek_bool {
1713       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1714       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1715     }
1716   }

```

```

1717 }
1718 \cs_new:Npn \um_config_bfit_greek:n #1 {
1719   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {#1}
1720   \bool_if:NTF \g_um_bfliteral_bool {
1721     \um_map_chars_greek:nn {bfit} {#1}
1722     \um_set_mathalphabet_greek:Nnn \mathbf {it} {#1}
1723   }{
1724     \bool_if:NF \g_um_bfupgreek_bool {
1725       \um_map_chars_greek:nn {bfit,bfup} {#1}
1726       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1727     }
1728   }
1729 }
1730 \cs_new:Npn \um_config_bfit_misc:n #1 {
1731   \bool_if:NTF \g_um_literal_Nabla_bool {
1732     \um_map_single:nnn {Nabla}{bfit}{#1}
1733   }{
1734     \bool_if:NF \g_um_upNabla_bool {
1735       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1736     }
1737   }
1738   \bool_if:NTF \g_um_literal_partial_bool {
1739     \um_map_single:nnn {partial}{bfit}{#1}
1740   }{
1741     \bool_if:NF \g_um_uppartial_bool {
1742       \um_map_single:nnn {partial}{bfup,bfit}{#1}
1743     }
1744   }
1745   \um_set_mathalphabet_pos:Nnnn \mathbfit {partial} {up,it}{#1}
1746   \um_set_mathalphabet_pos:Nnnn \mathbfit {Nabla} {up,it}{#1}
1747   \bool_if:NTF \g_um_literal_partial_bool {
1748     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {it}{#1}
1749   }{
1750     \bool_if:NF \g_um_uppartial_bool {
1751       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1752     }
1753   }
1754   \bool_if:NTF \g_um_literal_Nabla_bool {
1755     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {it}{#1}
1756   }{
1757     \bool_if:NF \g_um_upNabla_bool {
1758       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}
1759     }
1760   }
1761 }

```

9.6.10 Bold Upright: `\mathbfup`

```

1762 \cs_new:Npn \um_config_bfup_num:n #1 {
1763   \um_set_mathalphabet_numbers:Nnn \mathbf {up}{#1}
1764   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{#1}
1765 }
1766 \cs_new:Npn \um_config_bfup_Latin:n #1 {
1767   \bool_if:NT \g_um_bfupLatin_bool {
1768     \um_map_chars_Latin:nn {bfup,bfit} {#1}
1769   }
1770   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{#1}
1771   \bool_if:NTF \g_um_bfliteral_bool {
1772     \um_map_chars_Latin:nn {bfup} {#1}
1773     \um_set_mathalphabet_Latin:Nnn \mathbf {up}{#1}
1774   }{
1775     \bool_if:NT \g_um_bfupLatin_bool {
1776       \um_map_chars_Latin:nn {bfup,bfit} {#1}
1777       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{#1}
1778     }
1779   }
1780 }
1781 \cs_new:Npn \um_config_bfup_latin:n #1 {
1782   \bool_if:NT \g_um_bfuplatin_bool {
1783     \um_map_chars_latin:nn {bfup,bfit} {#1}
1784   }
1785   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{#1}
1786   \bool_if:NTF \g_um_bfliteral_bool {
1787     \um_map_chars_latin:nn {bfup} {#1}
1788     \um_set_mathalphabet_latin:Nnn \mathbf {up}{#1}
1789   }{
1790     \bool_if:NT \g_um_bfuplatin_bool {
1791       \um_map_chars_latin:nn {bfup,bfit} {#1}
1792       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{#1}
1793     }
1794   }
1795 }
1796 \cs_new:Npn \um_config_bfup_Greek:n #1 {
1797   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{#1}
1798   \bool_if:NTF \g_um_bfliteral_bool {
1799     \um_map_chars_Greek:nn {bfup}{#1}
1800     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{#1}
1801   }{
1802     \bool_if:NT \g_um_bfupGreek_bool {
1803       \um_map_chars_Greek:nn {bfup,bfit}{#1}
1804       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{#1}
1805     }
1806   }

```

```

1807 }
1808 \cs_new:Npn \um_config_bfup_greek:n #1 {
1809   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {#1}
1810   \bool_if:NTF \g_um_bfliteral_bool {
1811     \um_map_chars_greek:nn {bfup} {#1}
1812     \um_set_mathalphabet_greek:Nnn \mathbf {up} {#1}
1813   }{
1814     \bool_if:NT \g_um_bfupgreek_bool {
1815       \um_map_chars_greek:nn {bfup,bfit} {#1}
1816       \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {#1}
1817     }
1818   }
1819 }
1820 \cs_new:Npn \um_config_bfup_misc:n #1 {
1821   \bool_if:NTF \g_um_literal_Nabla_bool {
1822     \um_map_single:nnn {Nabla}{bfup}{#1}
1823   }{
1824     \bool_if:NT \g_um_upNabla_bool {
1825       \um_map_single:nnn {Nabla}{bfup,bfit}{#1}
1826     }
1827   }
1828   \bool_if:NTF \g_um_literal_partial_bool {
1829     \um_map_single:nnn {partial}{bfup}{#1}
1830   }{
1831     \bool_if:NT \g_um_uppartial_bool {
1832       \um_map_single:nnn {partial}{bfup,bfit}{#1}
1833     }
1834   }
1835   \um_set_mathalphabet_pos:Nnnn \mathbfup {partial} {up,it}{#1}
1836   \um_set_mathalphabet_pos:Nnnn \mathbfup {Nabla} {up,it}{#1}
1837   \um_set_mathalphabet_pos:Nnnn \mathbfup {digamma} {up}{#1}
1838   \um_set_mathalphabet_pos:Nnnn \mathbfup {Digamma} {up}{#1}
1839   \um_set_mathalphabet_pos:Nnnn \mathbf {digamma} {up}{#1}
1840   \um_set_mathalphabet_pos:Nnnn \mathbf {Digamma} {up}{#1}
1841   \bool_if:NTF \g_um_literal_partial_bool {
1842     \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up}{#1}
1843   }{
1844     \bool_if:NT \g_um_uppartial_bool {
1845       \um_set_mathalphabet_pos:Nnnn \mathbf {partial} {up,it}{#1}
1846     }
1847   }
1848   \bool_if:NTF \g_um_literal_Nabla_bool {
1849     \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up}{#1}
1850   }{
1851     \bool_if:NT \g_um_upNabla_bool {
1852       \um_set_mathalphabet_pos:Nnnn \mathbf {Nabla} {up,it}{#1}

```

```

1853     }
1854   }
1855 }

```

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

```

1856 \cs_new:Npn \um_config_bffrak_Latin:n #1 {
1857   \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{#1}
1858 }
1859 \cs_new:Npn \um_config_bffrak_latin:n #1 {
1860   \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{#1}
1861 }

```

9.6.12 Bold script or calligraphic: `\mathbfscr`

```

1862 \cs_new:Npn \um_config_bfscr_Latin:n #1 {
1863   \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{#1}
1864 }
1865 \cs_new:Npn \um_config_bfscr_latin:n #1 {
1866   \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{#1}
1867 }
1868 \cs_new:Npn \um_config_bfcal_Latin:n #1 {
1869   \um_set_mathalphabet_Latin:Nnn \mathbfcal {up,it}{#1}
1870 }

```

9.6.13 Bold upright sans serif: `\mathbfsfup`

```

1871 \cs_new:Npn \um_config_bfsfup_num:n #1 {
1872   \um_set_mathalphabet_numbers:Nnn \mathbfsf {up}{#1}
1873   \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{#1}
1874 }
1875 \cs_new:Npn \um_config_bfsfup_Latin:n #1 {
1876   \bool_if:NTF \g_um_sfliteral_bool {
1877     \um_map_chars_Latin:nn {bfsfup} {#1}
1878     \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{#1}
1879   }{
1880     \bool_if:NT \g_um_upsans_bool {
1881       \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1882       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1883     }
1884   }
1885   \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{#1}
1886 }
1887 \cs_new:Npn \um_config_bfsfup_latin:n #1 {
1888   \bool_if:NTF \g_um_sfliteral_bool {
1889     \um_map_chars_latin:nn {bfsfup} {#1}
1890     \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{#1}
1891   }{
1892     \bool_if:NT \g_um_upsans_bool {

```

```

1893     \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1894     \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1895   }
1896 }
1897 \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{#1}
1898 }
1899 \cs_new:Npn \um_config_bfsfup_greek:n #1 {
1900   \bool_if:NTF \g_um_sfliteral_bool {
1901     \um_map_chars_greek:nn {bfsfup}{#1}
1902     \um_set_mathalphabet_greek:Nnn \mathbfsf {up}{#1}
1903   }{
1904     \bool_if:NT \g_um_upsans_bool {
1905       \um_map_chars_greek:nn {bfsfup,bfsfit}{#1}
1906       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it}{#1}
1907     }
1908   }
1909   \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it}{#1}
1910 }
1911 \cs_new:Npn \um_config_bfsfup_greek:n #1 {
1912   \bool_if:NTF \g_um_sfliteral_bool {
1913     \um_map_chars_greek:nn {bfsfup} {#1}
1914     \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {#1}
1915   }{
1916     \bool_if:NT \g_um_upsans_bool {
1917       \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1918       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1919     }
1920   }
1921   \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {#1}
1922 }
1923 \cs_new:Npn \um_config_bfsfup_misc:n #1 {
1924   \bool_if:NTF \g_um_literal_Nabla_bool {
1925     \um_map_single:nnn {Nabla}{bfsfup}{#1}
1926   }{
1927     \bool_if:NT \g_um_upNabla_bool {
1928       \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
1929     }
1930   }
1931   \bool_if:NTF \g_um_literal_partial_bool {
1932     \um_map_single:nnn {partial}{bfsfup}{#1}
1933   }{
1934     \bool_if:NT \g_um_uppartial_bool {
1935       \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
1936     }
1937   }
1938   \um_set_mathalphabet_pos:Nnnn \mathbfsfup {partial} {up,it}{#1}

```



```

1939 \um_set_mathalphabet_pos:Nnnn \mathbfsfup {Nabla} {up,it}{#1}
1940 \bool_if:NTF \g_um_literal_partial_bool {
1941 \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up}{#1}
1942 }{
1943 \bool_if:NT \g_um_uppartial_bool {
1944 \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
1945 }
1946 }
1947 \bool_if:NTF \g_um_literal_Nabla_bool {
1948 \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up}{#1}
1949 }{
1950 \bool_if:NT \g_um_upNabla_bool {
1951 \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
1952 }
1953 }
1954 }

```

9.6.14 Bold italic sans serif: `\mathbfsfit`

```

1955 \cs_new:Npn \um_config_bfsfit_Latin:n #1 {
1956 \bool_if:NTF \g_um_sfliteral_bool {
1957 \um_map_chars_Latin:nn {bfsfit} {#1}
1958 \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{#1}
1959 }{
1960 \bool_if:NF \g_um_upsans_bool {
1961 \um_map_chars_Latin:nn {bfsfup,bfsfit} {#1}
1962 \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{#1}
1963 }
1964 }
1965 \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{#1}
1966 }
1967 \cs_new:Npn \um_config_bfsfit_latin:n #1 {
1968 \bool_if:NTF \g_um_sfliteral_bool {
1969 \um_map_chars_latin:nn {bfsfit} {#1}
1970 \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{#1}
1971 }{
1972 \bool_if:NF \g_um_upsans_bool {
1973 \um_map_chars_latin:nn {bfsfup,bfsfit} {#1}
1974 \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{#1}
1975 }
1976 }
1977 \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{#1}
1978 }
1979 \cs_new:Npn \um_config_bfsfit_Greek:n #1 {
1980 \bool_if:NTF \g_um_sfliteral_bool {
1981 \um_map_chars_Greek:nn {bfsfit}{#1}
1982 \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{#1}

```

```

1983   }{
1984     \bool_if:NF \g_um_upsans_bool {
1985       \um_map_chars_Greek:nn {bfsfup,bfsfit}{#1}
1986       \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{#1}
1987     }
1988   }
1989   \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{#1}
1990 }
1991 \cs_new:Npn \um_config_bfsfit_greek:n #1 {
1992   \bool_if:NTF \g_um_sfliteral_bool {
1993     \um_map_chars_greek:nn {bfsfit} {#1}
1994     \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {#1}
1995   }{
1996     \bool_if:NF \g_um_upsans_bool {
1997       \um_map_chars_greek:nn {bfsfup,bfsfit} {#1}
1998       \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {#1}
1999     }
2000   }
2001   \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {#1}
2002 }
2003 \cs_new:Npn \um_config_bfsfit_misc:n #1 {
2004   \bool_if:NTF \g_um_literal_Nabla_bool {
2005     \um_map_single:nnn {Nabla}{bfsfit}{#1}
2006   }{
2007     \bool_if:NF \g_um_upNabla_bool {
2008       \um_map_single:nnn {Nabla}{bfsfup,bfsfit}{#1}
2009     }
2010   }
2011   \bool_if:NTF \g_um_literal_partial_bool {
2012     \um_map_single:nnn {partial}{bfsfit}{#1}
2013   }{
2014     \bool_if:NF \g_um_uppartial_bool {
2015       \um_map_single:nnn {partial}{bfsfup,bfsfit}{#1}
2016     }
2017   }
2018   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {partial} {up,it}{#1}
2019   \um_set_mathalphabet_pos:Nnnn \mathbfsfit {Nabla} {up,it}{#1}
2020   \bool_if:NTF \g_um_literal_partial_bool {
2021     \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {it}{#1}
2022   }{
2023     \bool_if:NF \g_um_uppartial_bool {
2024       \um_set_mathalphabet_pos:Nnnn \mathbfsf {partial} {up,it}{#1}
2025     }
2026   }
2027   \bool_if:NTF \g_um_literal_Nabla_bool {
2028     \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {it}{#1}

```

```

2029   }{
2030     \bool_if:NF \g_um_upNabla_bool {
2031       \um_set_mathalphabet_pos:Nnnn \mathbfsf {Nabla} {up,it}{#1}
2032     }
2033   }
2034 }

```

10 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.

```

2035 \xetex_or luatex:nnn { \cs_set:Npn \um_symbol_setup: }
2036 {
2037   \def\mathfence{\mathfence}
2038   \def\mathover{\mathover}
2039   \def\mathunder{\mathunder}
2040   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4 {
2041     \prg_case_tl:Nnn ##3 { \mathover {} \mathunder {} }
2042     {
2043       \exp_not:n {\_um_sym:nnn{##1}{##2}{##3}}
2044     }
2045   }
2046 }
2047 {
2048   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4 {
2049     \exp_not:n {\_um_sym:nnn{##1}{##2}{##3}}
2050   }
2051 }
2052 \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.

```

2053 \cs_new:Npn \um_input_math_symbol_table: {\g_um_mathtable_tl}

```

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

To do this, turn \wedge 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 \wedge ’s catcode returns to normal.

```

2054 \begingroup
2055   \char_make_other:N \^
2056   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
2057     \lowercase{
2058       \tl_rescan:nn {
2059         \char_make_other:N \{
2060         \char_make_other:N \}
2061         \char_make_other:N \&
2062         \char_make_other:N \%
2063         \char_make_other:N \$
2064       }{
2065         \global\let#1=^^^^#2
2066       }
2067     }
2068   }

```

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

```

2069   \gdef\um@scanactivedef"#1\@nil#2{
2070     \lowercase{
2071       \tl_rescan:nn{
2072         \ExplSyntaxOn
2073         \char_make_math_superscript:N \^
2074       }{
2075         \global\def^^^^#1{#2}
2076       }
2077     }
2078   }
2079 \endgroup

```

Now give `_um_sym:nnn` a definition in terms of `\um@scancharlet` and we’re good to go.

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

```

2080 \AtBeginDocument{
2081   \group_begin:
2082     \char_make_math_superscript:N \^
2083     \cs_set:Npn \_um\_sym:nnn #1#2#3 {
2084       \bool_if:nF { \cs_if_eq_p:NN #3 \mathaccent ||
2085                     \cs_if_eq_p:NN #3 \mathopen  ||
2086                     \cs_if_eq_p:NN #3 \mathclose ||
2087                     \cs_if_eq_p:NN #3 \mathover  ||
2088                     \cs_if_eq_p:NN #3 \mathunder } {

```

```

2089         \um@scancharlet#2=#1\@nil\ignorespaces
2090     }
2091 }
2092 \char_make_other:N \#
2093 \um_input_math_symbol_table:
2094 \group_end:
2095 }

```

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

```

2096 \group_begin:
2097   \lccode`*=`\\
2098   \char_make_escape:N \
2099   \char_make_other:N \\
2100   |lowercase{
2101     |AtBeginDocument{
2102       |let|backslash=*
2103     }
2104   }
2105 |group_end:

```

Fix `\backslash`:

12 Epilogue

Lots of little things to tidy up.

12.1 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' x'' x''' x''''

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

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

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

Our algorithm is:

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

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

```

2106 \cs_new:Nn \um_arg_i_before_egroup:n {#1\egroup}
2107 \cs_new:Nn \um_superscript:n {
2108   ^\bgroup #1
2109   \peek_meaning_remove:NTF ^
2110     \um_arg_i_before_egroup:n \egroup
2111 }

2112 \muskip_new:N \g_um_primekern_muskip
2113 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
2114 \int_new:N \l_um_primecount_int

2115 \cs_new:Npn \um_nprimes:Nn #1#2 {
2116   \um_superscript:n {
2117     #1
2118     \prg_replicate:nn {#2-1} { \mskip \g_um_primekern_muskip #1 }
2119   }
2120 }

2121 \cs_new:Npn \um_nprimes_select:nn #1#2 {
2122   \prg_case_int:nnn {#2}{
2123     {1} { \um_superscript:n {#1} }
2124     {2} {
2125       \um_glyph_if_exist:NTF {"2033}
2126         { \um_superscript:n {\um_prime_double_mchar} }
2127         { \um_nprimes:Nn #1 {#2} }
2128     }

```

```

2129 {3} {
2130     \um_glyph_if_exist:nTF {"2034}
2131     { \um_superscript:n {\um_prime_triple_mchar} }
2132     { \um_nprimes:Nn #1 {#2} }
2133 }
2134 {4} {
2135     \um_glyph_if_exist:nTF {"2057}
2136     { \um_superscript:n {\um_prime_quad_mchar} }
2137     { \um_nprimes:Nn #1 {#2} }
2138 }
2139 }{
2140     \um_nprimes:Nn #1 {#2}
2141 }
2142 }
2143 \cs_new:Npn \um_nbackprimes_select:nn #1#2 {
2144     \prg_case_int:nnn {#2}{
2145         {1} { \um_superscript:n {#1} }
2146         {2} {
2147             \um_glyph_if_exist:nTF {"2036}
2148             { \um_superscript:n {\um_backprime_double_mchar} }
2149             { \um_nprimes:Nn #1 {#2} }
2150         }
2151         {3} {
2152             \um_glyph_if_exist:nTF {"2037}
2153             { \um_superscript:n {\um_backprime_triple_mchar} }
2154             { \um_nprimes:Nn #1 {#2} }
2155         }
2156     }{
2157         \um_nprimes:Nn #1 {#2}
2158     }
2159 }

```

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

```

2160 \cs_new:Npn \um_scan_prime: {
2161     \int_zero:N \l_um_primecount_int
2162     \um_scanprime_collect:N \um_prime_single_mchar
2163 }
2164 \cs_new:Npn \um_scan_dprime: {
2165     \int_set:Nn \l_um_primecount_int {1}
2166     \um_scanprime_collect:N \um_prime_single_mchar
2167 }
2168 \cs_new:Npn \um_scan_trprime: {
2169     \int_set:Nn \l_um_primecount_int {2}
2170     \um_scanprime_collect:N \um_prime_single_mchar
2171 }
2172 \cs_new:Npn \um_scan_qprime: {
2173     \int_set:Nn \l_um_primecount_int {3}

```

```

2174 \um_scanprime_collect:N \um_prime_single_mchar
2175 }
2176 \cs_new:Npn \um_scanprime_collect:N #1 {
2177   \int_incr:N \l_um_primecount_int
2178   \peek_meaning_remove:NTF ' {
2179     \um_scanprime_collect:N #1
2180   }{
2181     \peek_meaning_remove:NTF \um_scan_prime: {
2182       \um_scanprime_collect:N #1
2183     }{
2184       \peek_meaning_remove:NTF ^^^^2032 {
2185         \um_scanprime_collect:N #1
2186       }{
2187         \peek_meaning_remove:NTF \um_scan_dprime: {
2188           \int_incr:N \l_um_primecount_int
2189           \um_scanprime_collect:N #1
2190         }{
2191           \peek_meaning_remove:NTF ^^^^2033 {
2192             \int_incr:N \l_um_primecount_int
2193             \um_scanprime_collect:N #1
2194           }{
2195             \peek_meaning_remove:NTF \um_scan_trprime: {
2196               \int_add:Nn \l_um_primecount_int {2}
2197               \um_scanprime_collect:N #1
2198             }{
2199               \peek_meaning_remove:NTF ^^^^2034 {
2200                 \int_add:Nn \l_um_primecount_int {2}
2201                 \um_scanprime_collect:N #1
2202               }{
2203                 \peek_meaning_remove:NTF \um_scan_qprime: {
2204                   \int_add:Nn \l_um_primecount_int {3}
2205                   \um_scanprime_collect:N #1
2206                 }{
2207                   \peek_meaning_remove:NTF ^^^^2057 {
2208                     \int_add:Nn \l_um_primecount_int {3}
2209                     \um_scanprime_collect:N #1
2210                   }{
2211                     \um_nprimes_select:nn {#1} {\l_um_primecount_int}
2212                   }
2213                 }
2214               }
2215             }
2216           }
2217         }
2218       }
2219     }

```



```

2220     }
2221 }
2222 \cs_new:Npn \um_scan_backprime: {
2223   \int_zero:N \l_um_primecount_int
2224   \um_scanbackprime_collect:N \um_backprime_single_mchar
2225 }
2226 \cs_new:Npn \um_scan_backdprime: {
2227   \int_set:Nn \l_um_primecount_int {1}
2228   \um_scanbackprime_collect:N \um_backprime_single_mchar
2229 }
2230 \cs_new:Npn \um_scan_backtrprime: {
2231   \int_set:Nn \l_um_primecount_int {2}
2232   \um_scanbackprime_collect:N \um_backprime_single_mchar
2233 }
2234 \cs_new:Npn \um_scanbackprime_collect:N #1 {
2235   \int_incr:N \l_um_primecount_int
2236   \peek_meaning_remove:NTF ` {
2237     \um_scanbackprime_collect:N #1
2238   }{
2239     \peek_meaning_remove:NTF \um_scan_backprime: {
2240       \um_scanbackprime_collect:N #1
2241     }{
2242       \peek_meaning_remove:NTF ^^^^2035 {
2243         \um_scanbackprime_collect:N #1
2244       }{
2245         \peek_meaning_remove:NTF \um_scan_backdprime: {
2246           \int_incr:N \l_um_primecount_int
2247           \um_scanbackprime_collect:N #1
2248         }{
2249           \peek_meaning_remove:NTF ^^^^2036 {
2250             \int_incr:N \l_um_primecount_int
2251             \um_scanbackprime_collect:N #1
2252           }{
2253             \peek_meaning_remove:NTF \um_scan_backtrprime: {
2254               \int_add:Nn \l_um_primecount_int {2}
2255               \um_scanbackprime_collect:N #1
2256             }{
2257               \peek_meaning_remove:NTF ^^^^2037 {
2258                 \int_add:Nn \l_um_primecount_int {2}
2259                 \um_scanbackprime_collect:N #1
2260               }{
2261                 \um_nbackprimes_select:nn {#1} {\l_um_primecount_int}
2262               }
2263             }
2264           }
2265         }

```

```

2266     }
2267   }
2268 }
2269 }

2270 \AtBeginDocument {
2271   \cs_set_eq:NN \prime      \um_scan_prime:
2272   \cs_set_eq:NN \drime      \um_scan_dprime:
2273   \cs_set_eq:NN \trprime    \um_scan_trprime:
2274   \cs_set_eq:NN \qprime     \um_scan_qprime:
2275   \cs_set_eq:NN \backprime   \um_scan_backprime:
2276   \cs_set_eq:NN \backdprime \um_scan_backdprime:
2277   \cs_set_eq:NN \backtrprime \um_scan_backtrprime:
2278 }
2279 \group_begin:
2280   \char_make_active:N \'
2281   \char_make_active:N `
2282   \char_make_active:n {"2032}
2283   \char_make_active:n {"2033}
2284   \char_make_active:n {"2034}
2285   \char_make_active:n {"2057}
2286   \char_make_active:n {"2035}
2287   \char_make_active:n {"2036}
2288   \char_make_active:n {"2037}
2289   \AtBeginDocument{
2290     \cs_set_eq:NN '      \um_scan_prime:
2291     \cs_set_eq:NN ^^^^2032 \um_scan_prime:
2292     \cs_set_eq:NN ^^^^2033 \um_scan_dprime:
2293     \cs_set_eq:NN ^^^^2034 \um_scan_trprime:
2294     \cs_set_eq:NN ^^^^2057 \um_scan_qprime:
2295     \cs_set_eq:NN `      \um_scan_backprime:
2296     \cs_set_eq:NN ^^^^2035 \um_scan_backprime:
2297     \cs_set_eq:NN ^^^^2036 \um_scan_backdprime:
2298     \cs_set_eq:NN ^^^^2037 \um_scan_backtrprime:
2299   }
2300 \group_end:

```

12.2 Unicode radicals

`\sqrt` Redefine this macro for Lua_T_EX, which provides us a nice primitive to use.

```

2301 \luatex_if_engine:T {
2302   \DeclareDocumentCommand \sqrt { 0{ } m } {
2303     \luatexUroot \l_um_radical_sqrt_tl {#1} {#2}
2304   }
2305   \cs_set:Npn \root #1 \of #2 {
2306     \luatexUroot \l_um_radical_sqrt_tl {#1} {#2}
2307   }

```

2308 }

`\r@@t` #1 : A mathstyle (for `\mathpalette`)
 #2 : Leading superscript for the sqrt sign
 A re-implementation of L^AT_EX's hard-coded n-root sign using the appropriate `\fontdimens`.

```
2309 \cs_set_nopar:Npn \r@@t #1#2 {
2310   \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
2311   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen63\l_um_font}
2312   \raise \dimexpr(
2313     \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-
2314     \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
2315   )\relax
2316   \copy \rootbox
2317   \um_mathstyle_scale:Nnn{#1}{\kern}{\fontdimen64\l_um_font}
2318   \box \z@
2319 }
```

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

```
2320 \cs_new:Npn \um_fontdimen_to_percent:nn #1#2 {
2321   0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
2322 }
```

`\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.

```
2323 \cs_new:Npn \um_mathstyle_scale:Nnn #1#2#3 {
2324   \ifx#1\scriptstyle
2325     #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
2326   \else
2327     \ifx#1\scriptscriptstyle
2328       #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
2329     \else
2330       #2#3
2331     \fi
2332   \fi
2333 }
```

12.3 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?

```
2334 \prop_new:N \g_um_supers_prop
2335 \prop_new:N \g_um_subs_prop
2336 \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.

```
2337 \cs_set:Npn \um_setup_active_superscript:nn #1#2 {
2338   \prop_gput:Nxn \g_um_supers_prop {\meaning #1} {#2}
2339   \char_make_active:N #1
2340   \char_gmake_mathactive:N #1
2341   \scantokens{
2342     \cs_gset:Npn #1 {
2343       \tl_set:Nn \l_um_ss_chain_tl {#2}
2344       \cs_set_eq:NN \um_sub_or_super:n \sp
2345       \tl_set:Nn \l_um_tmpa_tl {supers}
2346       \um_scan_sscript:
2347     }
2348   }
2349 }
```

Bam:

```
2350 \um_setup_active_superscript:nn {^^^2070} {0}
2351 \um_setup_active_superscript:nn {^^^00b9} {1}
2352 \um_setup_active_superscript:nn {^^^00b2} {2}
2353 \um_setup_active_superscript:nn {^^^00b3} {3}
2354 \um_setup_active_superscript:nn {^^^2074} {4}
2355 \um_setup_active_superscript:nn {^^^2075} {5}
2356 \um_setup_active_superscript:nn {^^^2076} {6}
2357 \um_setup_active_superscript:nn {^^^2077} {7}
2358 \um_setup_active_superscript:nn {^^^2078} {8}
2359 \um_setup_active_superscript:nn {^^^2079} {9}
```

```

2360 \um_setup_active_superscript:nn {^^^^207a} {+}
2361 \um_setup_active_superscript:nn {^^^^207b} {-}
2362 \um_setup_active_superscript:nn {^^^^207c} {=}
2363 \um_setup_active_superscript:nn {^^^^207d} {(}
2364 \um_setup_active_superscript:nn {^^^^207e} {)}
2365 \um_setup_active_superscript:nn {^^^^2071} {i}
2366 \um_setup_active_superscript:nn {^^^^207f} {n}

```

Subscripts Ditto above.

```

2367 \cs_set:Npn \um_setup_active_subscript:nn #1#2 {
2368   \prop_gput:Nxn \g_um_subs_prop {\meaning #1} {#2}
2369   \char_make_active:N #1
2370   \char_gmake_mathactive:N #1
2371   \scantokens{
2372     \cs_gset:Npn #1 {
2373       \tl_set:Nn \l_um_ss_chain_tl {#2}
2374       \cs_set_eq:NN \um_sub_or_super:n \sb
2375       \tl_set:Nn \l_um_tmpa_tl {subs}
2376       \um_scan_sscript:
2377     }
2378   }
2379 }

```

A few more subscripts than superscripts:

```

2380 \um_setup_active_subscript:nn {^^^^2080} {0}
2381 \um_setup_active_subscript:nn {^^^^2081} {1}
2382 \um_setup_active_subscript:nn {^^^^2082} {2}
2383 \um_setup_active_subscript:nn {^^^^2083} {3}
2384 \um_setup_active_subscript:nn {^^^^2084} {4}
2385 \um_setup_active_subscript:nn {^^^^2085} {5}
2386 \um_setup_active_subscript:nn {^^^^2086} {6}
2387 \um_setup_active_subscript:nn {^^^^2087} {7}
2388 \um_setup_active_subscript:nn {^^^^2088} {8}
2389 \um_setup_active_subscript:nn {^^^^2089} {9}
2390 \um_setup_active_subscript:nn {^^^^208a} {+}
2391 \um_setup_active_subscript:nn {^^^^208b} {-}
2392 \um_setup_active_subscript:nn {^^^^208c} {=}
2393 \um_setup_active_subscript:nn {^^^^208d} {(}
2394 \um_setup_active_subscript:nn {^^^^208e} {)}
2395 \um_setup_active_subscript:nn {^^^^2090} {a}
2396 \um_setup_active_subscript:nn {^^^^2091} {e}
2397 \um_setup_active_subscript:nn {^^^^1d62} {i}
2398 \um_setup_active_subscript:nn {^^^^2092} {o}
2399 \um_setup_active_subscript:nn {^^^^1d63} {r}
2400 \um_setup_active_subscript:nn {^^^^1d64} {u}
2401 \um_setup_active_subscript:nn {^^^^1d65} {v}

```

```

2402 \um_setup_active_subscript:nn {^^^2093} {x}
2403 \um_setup_active_subscript:nn {^^^1d66} {\beta}
2404 \um_setup_active_subscript:nn {^^^1d67} {\gamma}
2405 \um_setup_active_subscript:nn {^^^1d68} {\rho}
2406 \um_setup_active_subscript:nn {^^^1d69} {\phi}
2407 \um_setup_active_subscript:nn {^^^1d6a} {\chi}
2408 \group_end:

```

The scanning command, evident in its purpose:

```

2409 \cs_new:Npn \um_scan_sscript: {
2410   \um_scan_sscript:TF {
2411     \um_scan_sscript:
2412   }{
2413     \um_sub_or_super:n {\l_um_ss_chain_tl}
2414   }
2415 }

```

The main theme here is stolen from the source to the various `\peek_` functions. Consider this function as simply boilerplate:

```

2416 \cs_new:Npn \um_scan_sscript:TF #1#2 {
2417   \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
2418   \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
2419   \tl_set:Nx \l_peek_false_tl { \exp_not:n{ \group_align_safe_end: #2 } }
2420   \group_align_safe_begin:
2421     \peek_after:NN \um_peek_execute_branches_ss:
2422 }

```

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

```

2423 \cs_new:Npn \um_peek_execute_branches_ss: {
2424   \bool_if:nTF {
2425     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2426     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2427     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2428   }
2429   { \l_peek_false_tl }
2430   { \um_peek_execute_branches_ss_aux: }
2431 }

```

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.)

```

2432 \cs_new:Npn \um_peek_execute_branches_ss_aux: {
2433   \prop_if_in:cxTF

```

```

2434 {g_um_\l_um_tmpa_tl _prop}
2435 {\meaning\l_peek_token}
2436 {
2437   \prop_get:cxN
2438   {g_um_\l_um_tmpa_tl _prop}
2439   {\meaning\l_peek_token}
2440   \l_um_tmpb_tl
2441   \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
2442   \l_peek_true_tl
2443 }
2444 {\l_peek_false_tl}
2445 }

```

12.3.1 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.

```

2446 \cs_new:Npn \um_define_active_frac:Nw #1 #2/#3 {
2447   \char_make_active:N #1
2448   \char_gmake_mathactive:N #1
2449   \tl_rescan:nn {
2450     \ExplSyntaxOn
2451   }{
2452     \cs_gset:Npx #1 {
2453       \bool_if:NTF \l_um_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2454       {#2} {#3}
2455     }
2456   }
2457 }

```

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

```

2458 \cs_new:Npn \um_setup_active_frac: {
2459   \group_begin:
2460   \um_define_active_frac:Nw ^^^^2189 0/3
2461   \um_define_active_frac:Nw ^^^^2152 1/{10}
2462   \um_define_active_frac:Nw ^^^^2151 1/9
2463   \um_define_active_frac:Nw ^^^^215b 1/8
2464   \um_define_active_frac:Nw ^^^^2150 1/7
2465   \um_define_active_frac:Nw ^^^^2159 1/6
2466   \um_define_active_frac:Nw ^^^^2155 1/5
2467   \um_define_active_frac:Nw ^^^^00bc 1/4
2468   \um_define_active_frac:Nw ^^^^2153 1/3
2469   \um_define_active_frac:Nw ^^^^215c 3/8
2470   \um_define_active_frac:Nw ^^^^2156 2/5

```

```

2471 \um_define_active_frac:Nw ^^^^00bd 1/2
2472 \um_define_active_frac:Nw ^^^^2157 3/5
2473 \um_define_active_frac:Nw ^^^^215d 5/8
2474 \um_define_active_frac:Nw ^^^^2154 2/3
2475 \um_define_active_frac:Nw ^^^^00be 3/4
2476 \um_define_active_frac:Nw ^^^^2158 4/5
2477 \um_define_active_frac:Nw ^^^^215a 5/6
2478 \um_define_active_frac:Nw ^^^^215e 7/8
2479 \group_end:
2480 }
2481 \um_setup_active_frac:

```

12.4 Synonyms and all the rest

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

```

2482 \def\to{\rightarrow}
2483 \def\overrightarrow{\vec}
2484 \def\le{\leq}
2485 \def\ge{\geq}
2486 \def\neq{\ne}
2487 \def\triangle{\mathord{\bigtriangleup}}
2488 \def\bigcirc{\mdlgwhtcircle}
2489 \def\circ{\vysmwhtcircle}
2490 \def\bullet{\smbkcircle}
2491 \def\mathyen{\yen}
2492 \def\mathsterling{\sterling}

```

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

```

2493 \@ifpackageloaded{amsmath}{
2494   % define their own colon, perhaps I should just steal it. (It does look much bet-
2495   % ter.)
2496   }{
2497   \cs_set_protected:Npn \colon {
2498     \bool_if:NTF \g_um_literal_colon_bool {:} { \mathpunct{:} }
2499   }

```

`\mathrm`

```

2500 \def\mathrm{\mathup}
2501 \let\mathfence\mathord

```

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

```

\Digamma 2502 \def\digamma{\updigamma}
2503 \def\Digamma{\upDigamma}

```


12.5 Compatibility

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

```
2504 \def\operator@font{\um_switchto_mathup:}

\um_patch_pkg:nn #1 : package
#2 : code
If <package> is loaded either already or later in the preamble, <code> is executed
(after the package is loaded in the latter case).

2505 \cs_new:Npn \um_patch_pkg:nn #1#2 {
2506   \ifpackageloaded {#1} {
2507     #2
2508   }{
2509     \um_after_pkg:nn {#1} {#2}
2510   }
2511 }
```

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.

```
2512 \um_patch_pkg:nn {url} {
2513   \tl_put_left:Nn \url@FormatString { \um_switchto_mathup: }
2514   \tl_put_right:Nn \url@Specials {
2515     \do\{\mathchar`\}
2516     \do\'\mathchar`\'
2517     \do\$\mathchar`\$
2518     \do\&\mathchar`\&
2519   }
2520 }
```

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:

```
2521 \um_patch_pkg:nn {amsmath} {
2522   \tl_remove_in:Nn \@begindocumenthook {
2523     \mathchardef\std@minus\mathcode`\-\relax
2524     \mathchardef\std@equal\mathcode`\=\relax
2525   }
2526   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
2527   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
```

```

2528 \cs_set:Npn \@cdots {\mathinner{\cdots}}
2529 \cs_set_eq:NN \dotsb@ \cdots

```

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

```

2530 \def \resetMathstrut@ {%
2531   \setbox\z@\hbox{$($}%
2532   \ht\Mathstrutbox@ \ht\z@ \dp\Mathstrutbox@ \dp\z@
2533 }
2534 }

```

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:

```

2535 \um_patch_pkg:n {amsopn} {
2536   \cs_set:Npn \newmcodes@ {
2537     \mathcode`\'39\scan_stop:
2538     \mathcode`\'*42\scan_stop:
2539     \mathcode`\'."613A\scan_stop:
2540     %% \ifnum\mathcode`\'=45 \else
2541     %%   \mathchardef\std@minus\mathcode`\'-\relax
2542     %% \fi
2543     \mathcode`\'-45\scan_stop:
2544     \mathcode`\'/47\scan_stop:
2545     \mathcode`\'."603A\scan_stop:
2546   }
2547 }

```

Symbols

```

2548 \cs_set:Npn \l {\Vert}

```

`\mathinner` items:

```

2549 \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2550 \cs_set:Npn \cdots {\mathinner{\unicodcdots}}

```

Accents

```

2551 \AtBeginDocument{
2552   \def\widehat{\hat}
2553   \def\widetilde{\tilde}
2554 }

```

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

```

2555 \AtEndOfPackage{
2556   \@ifclassloaded{beamer}{
2557     \ifbeamer@suppressreplacements\else
2558     \um_warning:n {disable-beamer}
2559     \beamer@suppressreplacementstrue

```

```

2560     \fi
2561   }{}
2562 }
2563 \ExplSyntaxOff
2564 \endpackage

```

13 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.

```

2565 <*msg>

    Wrapper functions:
2566 \cs_new:Npn \um_warning:n { \msg_warning:nn {unicode-math} }
2567 \cs_new:Npn \um_trace:n { \msg_trace:nn {unicode-math} }
2568 \cs_new:Npn \um_trace:nx { \msg_trace:nxx {unicode-math} }
2569 \msg_new:nnn {unicode-math} {maths-feature-only}
2570 {
2571   The~ '#1'~ font~ feature~ can~ only~ be~ used~ for~ maths~ fonts.
2572 }
2573 \msg_new:nnn {unicode-math} {disable-beamer}
2574 {
2575   Disabling~ beamer's~ math~ setup.\\
2576   Please~ load~ beamer~ with~ the~ [professionalfonts]~ class~ option.
2577 }
2578 \msg_new:nnn {unicode-math} {no-tfrac}
2579 {
2580   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\\
2581   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
2582 }
2583 \msg_new:nnn {unicode-math} {default-math-font}
2584 {
2585   Defining~ the~ default~ maths~ font~ as~ '#1'.
2586 }
2587 \msg_new:nnn {unicode-math} {setup-implicit}
2588 {
2589   Setup~ alphabets:~ implicit~ mode.
2590 }
2591 \msg_new:nnn {unicode-math} {setup-explicit}
2592 {
2593   Setup~ alphabets:~ explicit~ mode.
2594 }
2595 \msg_new:nnn {unicode-math} {alph-initialise}
2596 {

```

```

2597   Initialising~ \@backslashchar math#1.
2598 }
2599 \msg_new:nnn {unicode-math} {setup-alph}
2600 {
2601   Setup~ alphabet:~ #1.
2602 }
2603 </msg>

```

The end.

14 stix table data extraction

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

This table is converted into a form suitable for reading by \XeTeX . A single file is produced containing all (more than 3298) symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols. Performance for now seems to be acceptable without such measures.

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

```

2604 < See stix-extract.sh for now. >

```

A Documenting maths support in the NFSS

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

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

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

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

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

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

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

```
\DeclareSymbolFontAlphabet{\cmd}\{\name\}
```

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

Maths ‘versions’ Different maths weights can be defined with the following, switched in text with the `\mathversion{<maths version>}` command.

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

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

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

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

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

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

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

Accents are not included yet.

Summary For symbols, something like:

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

For characters, something like:

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

B X_YTeX math font dimensions

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

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

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

\fontdimen	Dimension name	Description
30	LOWERLIMITGAPMIN	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LOWERLIMITBASELINEDROP-MIN	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	STACKTOPSHIFTUP	Standard shift up applied to the top element of a stack.
33	STACKTOPDISPLAYSTYLESHIFT-UP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	STACKBOTTOMDISPLAYSTYLE-SHIFTDOWN	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	STACKGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.
37	STACKDISPLAYSTYLEGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	STRETCHSTACKTOPSHIFTUP	Standard shift up applied to the top element of the stretch stack.
39	STRETCHSTACKBOTTOMSHIFT-DOWN	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	STRETCHSTACKGAPABOVEMIN	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: UpperLimitGapMin
41	STRETCHSTACKGAPBELOWMIN	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: LowerLimitGapMin.
42	FRACTIONNUMERATORSHIFTUP	Standard shift up applied to the numerator.

\fontdimen	Dimension name	Description
43	FRACTIONNUMERATOR- DISPLAYSTYLESHIFTUP	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FRACTIONDENOMINATORSHIFT- DOWN	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FRACTIONDENOMINATOR- DISPLAYSTYLESHIFTDOWN	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FRACTIONNUMERATORGAP- MIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	FRACTIONNUMDISPLAYSTYLE- GAPMIN	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
48	FRACTIONRULETHICKNESS	Thickness of the fraction bar. Suggested: default rule thickness.
49	FRACTIONDENOMINATORGAP- MIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FRACTIONDENOMDISPLAY- STYLEGAPMIN	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
51	SKEWEDFRACTION- HORIZONTALGAP	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SKEWEDFRACTIONVERTICAL- GAP	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OVERBARVERTICALGAP	Distance between the overbar and the (ink) top of the base. Suggested: 3×default rule thickness.
54	OVERBARRULETHICKNESS	Thickness of overbar. Suggested: default rule thickness.

\fontdimen	Dimension name	Description
55	OVERBAREXTRAASCENDER	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UNDERBARVERTICALGAP	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UNDERBARRULETHICKNESS	Thickness of underbar. Suggested: default rule thickness.
58	UNDERBAREXTRADESCENDER	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RADICALVERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RADICALDISPLAYSTYLE- VERTICALGAP	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
61	RADICALRULETHICKNESS	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RADICALEXTRAASCENDER	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.
63	RADICALKERNBEFOREDEGREE	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RADICALKERNAFTERDEGREE	Negative kern after the degree of a radical, if such is present. Suggested: −10/18 of em.
65	RADICALDEGREEBOTTOM- RAISEPERCENT	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.