

Experimental Unicode mathematical typesetting: The unicode-math package

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

2015/09/09 v0.8b

Abstract

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

The package provides support for both $X_{\text{E}}\text{TeX}$ and LuaTeX . The different engines provide differing levels of support for Unicode maths. Please let us know of any troubles.

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

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

Part I

User documentation

Table of Contents

1	Introduction	3
2	Acknowledgements	3
3	Getting started	3
3.1	New commands	4
3.2	Package options	4
4	Unicode maths font setup	5
4.1	Using multiple fonts	5
4.2	Script and scriptscript fonts/features	7
4.3	Maths ‘versions’	7
4.4	Legacy maths ‘alphabet’ commands	8
5	Maths input	9
5.1	Math ‘style’	9
5.2	Bold style	10
5.3	Sans serif style	11
5.4	All (the rest) of the mathematical styles	12
5.5	Miscellanea	13
6	Advanced	19
6.1	Warning messages	19
6.2	Programmer’s interface	19
A	stix table data extraction	20
B	Documenting maths support in the NFSS	20
C	Legacy T_EX font dimensions	22
D	X_YT_EX math font dimensions	22

1 Introduction

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

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

2 Acknowledgements

Many thanks to: Microsoft for developing the mathematics extension to OpenType as part of Microsoft Office 2007; Jonathan Kew for implementing Unicode math support in X_YTeX; Taco Hoekwater for implementing Unicode math support in LuaTeX; Barbara Beeton for her prodigious effort compiling the definitive list of Unicode math glyphs and their L^ATeX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions; Philipp Stephani for extending the package to support LuaTeX. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use TeX in the future. Apostolos Syropoulos, Joel Salomon, Khaled Hosny, and Mariusz Wodzicki have been fantastic beta testers.

3 Getting started

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

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

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

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

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

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

3.1 New commands

New v0.8: unicode-math provides the following commands to select specific ‘alphabets’ within the unicode maths font: (usage, e.g.: $\text{\textbackslash symbfsf{g}}$ \rightarrow **g**)

```
\symnormal \symliteral \symup \symbfup \symbfup \symbfup \symsfup \symsfit
\symsfsup \symsfsfit \symsfsf \symsbb \symsbbit \symscr \symsfscr
\symcal \symbcalf \symfrak \symbffrak \symup \symsf \symbf \symtt
\symit
```

Many of these are also defined with ‘familiar’ synonyms:

```
\mathnormal \mathbb \mathbbbit \mathscr \mathbfscr \mathcal \mathbfcal
\mathfrak \mathbffrak \mathbfup \mathbfup \mathsfup \mathsfup \mathbfsfup
\mathbfsfup \mathbfsfup
```

So what about $\text{\textbackslash mathup}$, $\text{\textbackslash mathit}$, $\text{\textbackslash mathbf}$, $\text{\textbackslash mathsf}$, and $\text{\textbackslash mathtt}$? (N.B.: $\text{\textbackslash mathrm}$ is defined as a synonym for $\text{\textbackslash mathup}$, but the latter is preferred as it is a script-agnostic term.) These commands have ‘overloaded’ meanings in \LaTeX , and it’s important to consider the subtle differences between, e.g., $\text{\textbackslash symbf}$ and $\text{\textbackslash mathbf}$. The former switches to single-letter mathematical symbols, whereas the second switches to a text font that behaves correctly in mathematics but should be used for multi-letter identifiers. These four commands (and $\text{\textbackslash mathrm}$) are defined in the traditional \LaTeX manner. Further details are discussed in section §4.4.

Additional similar commands can be defined using

```
\setmathfontface\mathfoo{...}
```

3.2 Package options

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

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

Note, however, that some package options affect how maths is initialised and changing an option such as $\text{\textbackslash math-style}$ 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 $\text{\textbackslash setmathfont}$ command. Therefore, the following two examples are equivalent:

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

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

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

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

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

implements this for every symbol and alphabetic variant. That means x to x , ξ to ξ , \leq to \leq , etc., `\symscr{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 `stix` font comes to mind as a possible exception. It will therefore be necessary to delegate specific Unicode ranges of glyphs to separate fonts:

`\setmathfont{}[range=<unicode range>,]`
 where *<unicode range>* is a comma-separated list of Unicode slot numbers and ranges such as {"27D0-"27EB,"27FF,"295B-"297F}. Note that T_EX's syntax for accessing the slot number of a character, such as `\+`, will also work here.

You may also use the macro for accessing the glyph, such as `\int`, or whole collection of symbols with the same math type, such as `\mathopen`, or complete math styles such as `\sympb`. (Only numerical slots, however, can be used in ranged declarations.)

4.1.1 Control over alphabet ranges

As discussed earlier, Unicode mathematics consists of a number of 'alphabet styles' within a single font. In unicode-math, these ranges are indicated with the following (hopefully self-explanatory) labels:

`up`, `it`, `tt`, `bfup`, `bfit`, `bb`, `bbit`, `scr`, `bfscr`, `cal`, `bfc`,
`frak`, `bfrac`, `sfup`, `sfit`, `bfsfup`, `bfsfit`, `bfsf`

Fonts can be selected for specified ranges only using the following syntax, in which case all other maths font setup remains untouched:

- `[range=bb]` to use the font for 'bb' letters only.
- `[range=bfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (also with `latin`, `Latin`, `num` as possible options for Latin lower-/upper-case and numbers, resp.).
- `[range=up->sfup]` to map to different output styles.

Note that 'meta-styles' such as 'bf' and 'sf' are not included here since they are context dependent. Use `[range=bfup]` and `[range=bfit]` to effect changes to the particular ranges selected by 'bf' (and similarly for 'sf').

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

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

and because the math plane fractur glyphs will be missing, unicode-math will know to use the ASCII ones instead. If necessary this behaviour can be forced with `[range=frak->up]`, since the 'up' range corresponds to ASCII letters.

If you wanted to swap the maths symbols with sans serif forms, it would be possible to write `[range={up->sfup,it->sfit}]`. Note, however, that at present Unicode does not encode glyphs for sans serif Greek (table 6).

Users of the impressive Minion Math fonts (commercial) may use remapping to access the bold glyphs using:

```
\setmathfont{MinionMath-Regular.otf}
\setmathfont{MinionMath-Bold.otf}[range={bfup->up,bfit->it}]
```

To set up the complete range of optical sizes for these fonts, a font declaration such as the following may be used: (adjust may be desired according to the font size of the document)

```
\setmathfont{Minion Math}[
  SizeFeatures = {
    {Size = -6.01, Font = MinionMath-Tiny},
    {Size = 6.01-8.41, Font = MinionMath-Capt},
    {Size = 8.41-13.01, Font = MinionMath-Regular},
    {Size = 13.01-19.91, Font = MinionMath-Subh},
    {Size = 19.91-, Font = MinionMath-Disp}
  }
]

\setmathfont{Minion Math}[range = {bfup->up,bfit->it},
  SizeFeatures = {
    {Size = -6.01, Font = MinionMath-BoldTiny},
    {Size = 6.01-8.41, Font = MinionMath-BoldCapt},
    {Size = 8.41-13.01, Font = MinionMath-Bold},
    {Size = 13.01-19.91, Font = MinionMath-BoldSubh},
    {Size = 19.91-, Font = MinionMath-BoldDisp}
  }
]
```

v0.8: Note that in previous versions of unicode-math, these features were labelled `[range=\mathbb]` and so on. This old syntax is still supported for backwards compatibility, but is now discouraged.

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_{BC}). Other typefaces (such as Minion Math) may use entirely separate font files.

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

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

4.3 Maths ‘versions’

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

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

`\setmathfont{}[version=<version name>,]`
 and to switch between maths versions mid-document use the standard L^AT_EX command `\mathversion{<version name>}`.

4.4 Legacy maths ‘alphabet’ commands

L^AT_EX traditionally uses `\DeclareMathAlphabet` and `\SetMathAlphabet` to define document commands such as `\mathit`, `\mathbf`, and so on. While these commands can still be used, unicode-math defines a wrapper command to assist with the creation of new such maths alphabet commands. This command is known as `\setmathface` in symmetry with fontspec’s `\newfontface` command; it takes syntax:

```
\setmathfontface<command>{<font name>}[<font features>]
\setmathfontface<command>{<font name>}[version=<version name>, <font features>]
```

For example, if you want to define a new legacy maths alphabet font `\mathittt`:

```
\setmathfontface\mathittt{texgyrecursor-italic.otf}
...
$\mathittt{foo} = \mathittt{a} + \mathittt{b}$
```

4.4.1 Default ‘text math’ fonts

The five ‘text math’ fonts, discussed above, are: `\mathrm`, `\mathbf`, `\mathit`, `\mathsf`, and `\mathtt`. These commands are also defined with their original definition under synonyms `\mathtextrm`, `\mathtextbf`, and so on.

When selecting document fonts using fontspec commands such as `\setmainfont`, unicode-math inserts some additional that keeps the current default fonts ‘in sync’ with their corresponding `\mathrm` commands, etc.

For example, in standard L^AT_EX, `\mathsf` doesn’t change even if the main document font is changed using `\renewcommand\sfdefault{...}`. With unicode-math loaded, after writing `\setsansfont{Helvetica}`, `\mathsf` will now be set in Helvetica.

If the `\mathsf` font is set explicitly at any time in the preamble, this ‘auto-following’ does not occur. The legacy math font switches can be defined either with commands defined by fontspec (`\setmathrm`, `\setmathsf`, etc.) or using the more general `\setmathfontface\mathsf` interface defined by unicode-math.

4.4.2 Replacing ‘text math’ fonts by symbols

For certain types of documents that use legacy input syntax (say you’re typesetting a new version of a book written in the 1990s), it would be preferable to use `\ymbf` rather than `\mathbf en masse`. For example, if bold maths is used only for vectors and matrices, a dedicated symbol font will produce better spacing and will better match the main math font.

Alternatively, you may have used an old version of unicode-math (pre-v0.8), when the `\symXYZ` commands were not defined and `\mathbf` behaved like `\ymbf`

Table 3: Maths text font configuration options. Note that `\mathup` and `\mathrm` are aliases of each other and cannot be configured separately.

Defaults (from ‘text’ font)	From ‘maths symbols’
<code>\mathrm=text</code>	<code>\mathrm=sym</code>
<code>\mathup=text*</code>	<code>\mathup=sym*</code>
<code>\mathit=text</code>	<code>\mathit=sym</code>
<code>\mathsf=text</code>	<code>\mathsf=sym</code>
<code>\mathbf=text</code>	<code>\mathbf=sym</code>
<code>\mathtt=text</code>	<code>\mathtt=sym</code>

does now. A series of package options (table 3) are provided to facilitate switching the definition of `\mathXYZ` for the five legacy text math font definitions.

A ‘smart’ macro is intended for a future version of unicode-math that can automatically distinguish between single- and multi-letter arguments to `\mathbf` and use either the maths symbol or the ‘text math’ font as appropriate.

4.4.3 Operator font

L^AT_EX defines an internal command `\operator@font` for typesetting elements such as `\sin` and `\cos`. This font is selected from the legacy operators NFSS ‘MathAlphabet’, which is no longer relevant in the context of unicode-math. By default, the `\operator@font` command is defined to switch to the `\mathrm` font. You may now change these using the command:

```
\setoperatorfont\mathit
```

Or, to select a unicode-math range:

```
\setoperatorfont\symscr
```

For example, after the latter above, `\sin x` will produce ‘*sin x*’.

5 Maths input

X_YL^AT_EX’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, in various historical contexts, often associated with French typesetting, it was common to use upright

Table 4: 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)	$(\alpha, \beta, \Gamma, \Xi)$

uppercase *Latin* letters as well as upright upper- and lowercase Greek, but italic lowercase latin. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

The unicode-math package accommodates these possibilities with the option `math-style` that takes one of four (case sensitive) arguments: `TeX`, `ISO`, `french`, or `upright`.¹ The `math-style` options' effects are shown in brief in table 4.

The philosophy behind the interface to the mathematical symbols lies in L^AT_EX'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: `\symup{g}`. Maths style commands such as `\symup` are detailed later.

'Literal' interface Some 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 `\symliteral{<syms>}` command can also be used, regardless of package setting, to force the style to match the literal input characters. This is a 'mirror' to `\symnormal{<syms>}` (also alias `\mathnormal`) which 'resets' the character mapping in its argument to that originally set up through package options.

5.2 Bold style

Similar as in the previous section, ISO standards differ somewhat to T_EX'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

¹Interface inspired by Walter Schmidt's `lucimatx` package.

Table 5: Effects of the bold-style package option.

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

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 by some that *italic* bold symbols should be used nowadays instead, but this practise is certainly not widespread.

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

In `unicode-math`, the `\symbf` command works directly with both Greek and Latin maths characters and depending on package option either switches to upright for Latin letters (`bold-style=TeX`) as well or keeps them italic (`bold-style=ISO`). To match the package options for non-bold characters, with option `bold-style=upright` all bold characters are upright, and `bold-style=literal` does not change the upright/italic shape of the letter. The bold-style options’ effects are shown in brief in table 5.

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 matching defaults are chosen based on the latter.

5.3 Sans serif style

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

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

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

N.B.: there is no medium-weight sans serif Greek range in Unicode. Therefore, `\symsf{\alpha}` does not make sense (it produces 'α'), while `\ymbfsf{\alpha}` gives 'α' or 'α' according to the `sans-style`.

5.4 All (the rest) of the mathematical styles

Unicode contains separate codepoints for most if not all variations of style shape one may wish to use in mathematical notation. The complete list is shown in table 6. Some of these have been covered in the previous sections.

The math font switching commands do not nest; therefore if you want sans serif bold, you must write `\ymbfsf{...}` rather than `\symsf{\ymbf{...}}`. This may change in the future.

5.4.1 Double-struck

The double-struck style (also known as 'blackboard bold') consists of upright Latin letters $\{0-z, \mathbb{A}, \mathbb{Z}\}$, numerals $0-9$, summation symbol $\mathbb{\Sigma}$, and four Greek letters only: $\{\mathbb{0}, \mathbb{1}, \mathbb{2}, \mathbb{3}\}$.

While `\ymbb{\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: $\mathbb{D}, \mathbb{d}, \mathbb{E}, \mathbb{e}, \mathbb{J}$. These can be accessed (if not with their literal characters or control sequences) with the `\mathbbi` style 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 a style for 'Script' letters, and while by default `\mathcal` and `\mathscr` are synonyms, there are some situations when a separate 'Caligraphic' style is needed as well.

If a font contains alternate glyphs for a separate caligraphic style, they can be selected explicitly as shown below. This feature is currently only supported by the

Table 6: Mathematical styles defined in Unicode. Black dots indicate an style exists in the font specified; blue dots indicate shapes that should always be taken from the upright font even in the italic style. See main text for description of `\mathbbi t`.

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>\mathbfssf</code>	•	•	•
Typewriter	Upright	Normal	<code>\mathtt</code>	•		•
Double-struck	Upright	Normal	<code>\mathbb</code>	•		•
	Italic	Normal	<code>\mathbbi</code>	•		
Script	Upright	Normal	<code>\mathscr</code>	•		
		Bold	<code>\matbfscr</code>	•		
Fraktur	Upright	Normal	<code>\mathfrak</code>	•		
		Bold	<code>\mathbffrac</code>	•		

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{xits-math.otf}[range={cal,bfcal},StylisticSet=1]
```

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 Miscellaneous

5.5.1 Nabla

The symbol ∇ comes in the six forms shown in table 7. We want an individual option to specify whether we want upright or italic nabla by default (when either upright or italic nabla is used in the source). \TeX classically uses an upright nabla, and iso standards agree with this convention. The package options `nabla=upright` and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. This is then inherited through `\sympb`; `\symit` and `\symup` can be used to force one way or the other.

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

Table 7: The various forms of nabla.

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

Table 8: The partial differential.

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

5.5.2 Partial

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

At time of writing, both the Cambria Math and STIX fonts display these two glyphs in the same italic style, but this is hopefully a bug that will be corrected in the future — the ‘plain’ partial differential should really have an upright shape.

Use the `partial=upright` or `partial=italic` package options to specify which one you would like, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests and argues otherwise) `partial=italic`.² `partial=literal` is activated following `math-style=literal`.

See table 8 for the variations on the partial differential symbol.

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

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

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

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

5.5.4 Primes

Primes (x') may be input in several ways. You may use any combination the ASCII straight quote (') or the Unicode prime `u+2032` ('); when multiple primes occur

²A 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 n h j r w y Z

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

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

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

next to each other, they chain together to form double, triple, or quadruple primes if the font contains pre-drawn glyphs. The individual prime glyphs are accessed, as usual, with the `\prime` command, and the double-, triple-, and quadruple-prime glyphs are available with `\dprime`, `\trprime`, and `\qprime`, respectively.

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

```
\ExplSyntaxOn
\muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }
\ExplSyntaxOff
```

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

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

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

5.5.5 Unicode subscripts and superscripts

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

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

Table 9: Slashes and backslashes.

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

5.5.6 Colon

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

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

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

To preserve input compatibility, we remap the ASCII input character ‘:’ to U+2236. Typing a literal U+2236 char will result in the same output. If `amsmath` is loaded, then the definition of `\colon` is inherited from there (it looks like a 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 9.

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: $A \setminus B$.³ 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’: $\pi \approx 7 \setminus 22$.⁴ 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`:

- `\frac{}`;
- `\slash`; and,
- `\backslash` (the only reverse slash).

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

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

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

5.5.8 Growing and non-growing accents

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

Older versions of $\text{\X}\text{\TeX}$ and $\text{\Lua}\text{\TeX}$ did not support this distinction, however, and *all* accents there were growing automatically. (I.e., `\hat` and `\widehat` are

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

⁴This 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>\vysmblkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smblkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmblkcircle</code>	●	◦	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdblkcircle</code>	●	◯	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgblkcircle</code>	●	◯	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	◯	<code>\lgwhtcircle</code>	U+25EF

Table 10: Filled and hollow Unicode circles.

equivalent.) As of Lua \TeX v0.65 and Xe \TeX v0.9998, these wide/non-wide commands will again behave in their expected manner.

5.5.9 Pre-drawn fraction characters

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

$\frac{1}{4}$ $\frac{1}{2}$ $\frac{3}{4}$ $\frac{2}{3}$ $\frac{1}{7}$ $\frac{1}{9}$ $\frac{1}{10}$ $\frac{1}{3}$ $\frac{2}{3}$ $\frac{1}{5}$ $\frac{2}{5}$ $\frac{3}{5}$ $\frac{4}{5}$ $\frac{1}{6}$ $\frac{5}{6}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{5}{8}$ $\frac{7}{8}$

For example, instead of writing ‘`\tfrac{12}{x}`’, you may consider it more readable to have ‘`\frac{x}{2}`’ in the source instead.

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

5.5.10 Circles

Unicode defines a large number of different types of circles for a variety of mathematical purposes. There are thirteen alone just considering the all white and all black ones, shown in table 10.

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

5.5.11 Triangles

While there aren’t as many different sizes of triangle as there are circle, there’s some important distinctions to make between a few similar characters. See table 11 for the full summary.

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 11: Different upwards pointing triangles.

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

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

Finally, given that \triangle 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.

6 Advanced

6.1 Warning messages

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

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

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

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

6.2 Programmer’s interface

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

A *STIX table data extraction*

The source for the $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ names for the very large number of mathematical glyphs are provided via Barbara Beeton’s table file for the `stix` project (ams.org/STIX). A version is located at <http://www.ams.org/STIX/bnb/stix-tbl.asc> but check <http://www.ams.org/STIX/> for more up-to-date info.

This table is converted into a form suitable for reading by $\mathrm{T}_{\mathrm{E}}\mathrm{X}$. 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!)

B *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: $\propto, \leq, \rightarrow$

`\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 $ABC-xyz, \mathfrak{A}\mathfrak{B}\mathfrak{C}-\mathcal{X}\mathcal{Y}\mathcal{Z}$, etc.

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

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

`\DeclareSymbolFontAlphabet{<cmd>}\{<name>\}`

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

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

`\SetSymbolFont{<name>}\{<maths version>\}\langle NFSS decl.\rangle`

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

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

Delimiters and radicals use wrappers around $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ’s `\delimiter/\radical` primitives, which are re-designed in $\mathrm{X}_{\mathrm{Y}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$. The syntax used in $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ’s NFSS is therefore not so relevant here.

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

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

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

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

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@}}}
```

C Legacy T_EX font dimensions

Text fonts		Maths font, \fam2		Maths font, \fam3	
ϕ_1	slant per pt	σ_5	x height	ζ_8	default rule thickness
ϕ_2	interword space	σ_6	quad	ζ_9	big op spacing1
ϕ_3	interword stretch	σ_8	num1	ζ_{10}	big op spacing2
ϕ_4	interword shrink	σ_9	num2	ζ_{11}	big op spacing3
ϕ_5	x-height	σ_{10}	num3	ζ_{12}	big op spacing4
ϕ_6	quad width	σ_{11}	denom1	ζ_{13}	big op spacing5
ϕ_7	extra space	σ_{12}	denom2		
ϕ_8	cap height (X _Y T _E X only)	σ_{13}	sup1		
		σ_{14}	sup2		
		σ_{15}	sup3		
		σ_{16}	sub1		
		σ_{17}	sub2		
		σ_{18}	sup drop		
		σ_{19}	sub drop		
		σ_{20}	delim1		
		σ_{21}	delim2		
		σ_{22}	axis height		

D X_YT_EX math font dimensions

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

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

\fontdimen	Dimension name	Description
14	MATHLEADING	White space to be left between math formulas to ensure proper line spacing. For example, for applications that treat line gap as a part of line ascender, formulas with ink going above (os2.sTypoAscender + os2.sTypoLineGap – MathLeading) or with ink going below os2.sTypoDescender will result in increasing line height.
15	AxisHEIGHT	Axis height of the font.
16	ACCENTBASEHEIGHT	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (os2.sxHeight) plus any possible overshots.
17	FLATTENEDACCENTBASE-HEIGHT	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font (os2.sCapHeight).
18	SUBSCRIPTSHIFTDOWN	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: os2.ySubscriptYOffset.
19	SUBSCRIPTTOPMAX	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: $\frac{1}{5}$ x-height.
20	SUBSCRIPTBASELINEDROPMIN	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.
21	SUPERSCRIPSHIFTUP	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.
22	SUPERSCRIPSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPBOTTOMMIN	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height.

\fontdimen	Dimension name	Description
24	SUPERSCRIPTBASELINEDROP-MAX	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SUBSUPERSCRIPGAPMIN	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	SUPERSCRIPBTOMMAX-WITHSUBSCRIPT	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: /5 x-height.
27	SPACEAFTERSCRIPT	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	UPPERLIMITGAPMIN	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	UPPERLIMITBASELINERISEMIN	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	LOWERLIMITGAPMIN	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LOWERLIMITBASELINEDROP-MIN	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	STACKTOPSHIFTUP	Standard shift up applied to the top element of a stack.
33	STACKTOPDISPLAYSTYLESHIFT-UP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
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.

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

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

\fontdimen	Dimension name	Description
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%.

Part II

Package implementation

Table of Contents

E Header code	28
E.1 Extras	31
E.2 Alphabet Unicode positions	32
E.3 Package options	32
E.4 Programmers' interface	37
F Bifurcation	38
F.1 Engine differences	38
F.2 Overcoming <code>\onlypreamble</code>	38
G Fundamentals	38
G.1 Setting math chars, math codes, etc.	38
G.2 <code>\setmathalphabet</code>	42
G.3 Hooks into <code>fontspec</code>	43
G.4 The main <code>\setmathfont</code> macro	45
G.5 (Big) operators	53
G.6 Radicals	53
G.7 Maths accents	54
G.8 Common interface for font parameters	54
H Font features	59
H.1 Math version	59
H.2 Script and <code>scriptscript</code> font options	59
H.3 Range processing	59
H.4 Resolving Greek symbol name control sequences	63
I Maths alphabets	63
I.1 Hooks into $\LaTeX 2_{\epsilon}$	64
I.2 Setting styles	64
I.3 Defining the math style macros	65
I.4 Definition of alphabets and styles	66
I.5 Defining the math alphabets per style	70
I.6 Mapping 'naked' math characters	72

I.7	Mapping chars inside a math style	74
J	A token list to contain the data of the math table	77
K	Definitions of the active math characters	77
L	Fall-back font	78
M	Epilogue	79
M.1	Primes	79
M.2	Unicode radicals	85
M.3	Unicode sub- and super-scripts	87
M.4	Synonyms and all the rest	92
N	Error messages	94
N.1	Alphabet Unicode positions	96
N.2	STIX fonts	102
N.3	Alphabets	106
N.4	Compatibility	124

The prefix for unicode-math is um:

```
1 <@=um>
```

E Header code

We (later on) bifurcate the package based on the engine being used. These separate package files are indicated with the Docstrip flags LU and XE, respectively. Shared code executed before loading the engine-specific code is indicated with the flag preamble.

```
2 <(*load)
3 \luatex_if_engine:T { \RequirePackage{unicode-math-luatex} }
4 \xetex_if_engine:T { \RequirePackage{unicode-math-xetex} }
5 </load>
```

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

```
6 <(*preamble&!XE&!LU)

    Bail early if using pdfTeX.
7 \usepackage{ifxetex,ifluatex}
8 \ifxetex
9   \ifdim\number\XeTeXversion\XeTeXrevision in<0.9998in%
10    \PackageError{unicode-math}{%
11      Cannot run with this version of XeTeX!\MessageBreak
12      You need XeTeX 0.9998 or newer.%
13    }\@ehd
14  \fi
15 \else\ifluatex
16   \ifnum\luatexversion<64%
```

```

17 \PackageError{unicode-math}{%
18     Cannot run with this version of LuaTeX!\MessageBreak
19     You need LuaTeX 0.64 or newer.%
20 } \@ehd
21 \fi
22 \else
23 \PackageError{unicode-math}{%
24     Cannot be run with pdfLaTeX!\MessageBreak
25     Use XeLaTeX or LuaLaTeX instead.%
26 } \@ehd
27 \fi\fi

```

Packages

```

28 \RequirePackage{expl3}[2015/03/01]
29 \RequirePackage{ucharcat}
30 \RequirePackage{xparse}
31 \RequirePackage{l3keys2e}
32 \RequirePackage{fontspec}[2015/03/14]
33 \RequirePackage{catchfile}
34 \RequirePackage{fix-cm} % avoid some warnings
35 \RequirePackage{filehook}

36 \ExplSyntaxOn

    Variants needed from expl3:

37 \cs_set_protected_nopar:Npn \exp_last_unbraced:NNx { \::N \::x_unbraced \::: }

    For fontspec:

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

```

Conditionals

```

40 \bool_new:N \l_@@_ot_math_bool
41 \bool_new:N \l_@@_init_bool
42 \bool_new:N \l_@@_implicit_alph_bool
43 \bool_new:N \g_@@_mainfont_already_set_bool

```

For math-style:

```

44 \bool_new:N \g_@@_literal_bool
45 \bool_new:N \g_@@_upLatin_bool
46 \bool_new:N \g_@@_uplatin_bool
47 \bool_new:N \g_@@_upGreek_bool
48 \bool_new:N \g_@@_upgreek_bool

```

For bold-style:

```

49 \bool_new:N \g_@@_bfliteral_bool
50 \bool_new:N \g_@@_bfupLatin_bool
51 \bool_new:N \g_@@_bfuplatin_bool
52 \bool_new:N \g_@@_bfupGreek_bool
53 \bool_new:N \g_@@_bfupgreek_bool

```

For sans-style:

```
54 \bool_new:N \g_@@_upsans_bool
55 \bool_new:N \g_@@_sfliteral_bool
```

For assorted package options:

```
56 \bool_new:N \g_@@_upNabla_bool
57 \bool_new:N \g_@@_uppartial_bool
58 \bool_new:N \g_@@_literal_Nabla_bool
59 \bool_new:N \g_@@_literal_partial_bool
60 \bool_new:N \g_@@_texgreek_bool
61 \bool_set_true:N \g_@@_texgreek_bool
62 \bool_new:N \l_@@_smallfrac_bool
63 \bool_new:N \g_@@_literal_colon_bool
64 \bool_new:N \g_@@_mathrm_text_bool
65 \bool_new:N \g_@@_mathit_text_bool
66 \bool_new:N \g_@@_mathbf_text_bool
67 \bool_new:N \g_@@_mathsf_text_bool
68 \bool_new:N \g_@@_mathtt_text_bool
```

Variables

```
69 \int_new:N \g_@@_fam_int
```

For displaying in warning messages, etc.:

```
70 \tl_const:Nn \c_@@_math_alphabet_name_latin_tl {Latin,~lowercase}
71 \tl_const:Nn \c_@@_math_alphabet_name_Latin_tl {Latin,~uppercase}
72 \tl_const:Nn \c_@@_math_alphabet_name_greek_tl {Greek,~lowercase}
73 \tl_const:Nn \c_@@_math_alphabet_name_Greek_tl {Greek,~uppercase}
74 \tl_const:Nn \c_@@_math_alphabet_name_num_tl {Numerals}
75 \tl_const:Nn \c_@@_math_alphabet_name_misc_tl {Misc.}
76 \tl_new:N \l_@@_mathstyle_tl
```

Used to store the font switch for the \operator@font.

```
77 \tl_new:N \g_@@_operator_mathfont_tl
```

Variables:

```
78 \seq_new:N \l_@@_missing_alph_seq
79 \seq_new:N \l_@@_mathalph_seq
80 \seq_new:N \l_@@_char_range_seq
81 \seq_new:N \l_@@_mclass_range_seq
82 \seq_new:N \l_@@_cmd_range_seq
```

\g_@@_mathclasses_seq Every math class.

```
83 \seq_new:N \g_@@_mathclasses_seq
84 \seq_set_from_clist:Nn \g_@@_mathclasses_seq
85 {
86   \mathord,\mathalpha,\mathbin,\mathrel,\mathpunct,
87   \mathop,
88   \mathopen,\mathclose,
89   \mathfence,\mathover,\mathunder,
90   \mathaccent,\mathbotaccent,\mathaccentwide,\mathbotaccentwide
91 }
```

`\g_@@_default_mathalph_seq` This sequence stores the alphabets in each math style.

```
92 \seq_new:N \g_@@_default_mathalph_seq
```

`\g_@@_mathstyles_seq` This is every ‘named range’ and every ‘math style’ known to unicode-math. A named range is such as “bfit” and “sfit”, which are also math styles (with `\symbfit` and `\symsfit`). ‘Mathstyles’ are a superset of named ranges and also include commands such as `\symbf` and `\symsf`.

N.B. for parsing purposes ‘named ranges’ are defined as strings!

```
93 \seq_new:N \g_@@_named_ranges_seq
```

```
94 \seq_new:N \g_@@_mathstyles_seq
```

```
95 \muskip_new:N \g_@@_primekern_muskip
```

```
96 \muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }% arbitrary
```

```
97 \int_new:N \l_@@_primecount_int
```

```
98 \prop_new:N \g_@@_supers_prop
```

```
99 \prop_new:N \g_@@_subs_prop
```

```
100 \tl_new:N \l_not_token_name_tl
```

E.1 Extras

What might end up being provided by the kernel.

`\@@_glyph_if_exist:nTF` : TODO: Generalise for arbitrary fonts! `\l_@@_font` is not always the one used for a specific glyph!!

```
101 \prg_new_conditional:Nnn \@@_glyph_if_exist:n {p,TF,T,F}
102 {
103   \etex_iffontchar:D \l_@@_font #1 \scan_stop:
104   \prg_return_true:
105   \else:
106     \prg_return_false:
107   \fi:
108 }
```

`\@@_set_mathcode:nnnn` These are all wrappers for the primitive commands that take numerical input only.

```
\@@_set_mathcode:nnn 109 \cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4 {
\@@_set_mathchar:NNnn 110   \Umathcode \int_eval:n {#1} =
\@@_set_mathchar:cNnn 111   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
\@@_set_delcode:nnn 112 }
\@@_radical:nn 113 \cs_set:Npn \@@_set_mathcode:nnn #1#2#3 {
\@@_delimiter:Nnn 114   \Umathcode \int_eval:n {#1} =
\@@_accent:nnn 115   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
\@@_accent_keyword: 116 }
117 \cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4 {
118   \Umathchardef #1 =
119   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
120 }
121 \cs_new:Nn \@@_set_delcode:nnn {
122   \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
123 }
```



```

124 \cs_new:Nn \@@_radical:nn {
125   \Uradical \csname sym#1\endcsname #2 \scan_stop:
126 }
127 \cs_new:Nn \@@_delimiter:Nnn {
128   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
129 }
130 \cs_new:Nn \@@_accent:nnn {
131   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
132 }
133 \cs_generate_variant:Nn \@@_set_mathchar:NNnn {c}

```

\@@_char_gmake_mathactive:N

\@@_char_gmake_mathactive:n

```

134 \cs_new:Nn \@@_char_gmake_mathactive:N
135 {
136   \global\mathcode `#1 = "8000 \scan_stop:
137 }
138 \cs_new:Nn \@@_char_gmake_mathactive:n
139 {
140   \global\mathcode #1 = "8000 \scan_stop:
141 }

```

E.2 Alphabet Unicode positions

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.⁵

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

```

142 \cs_new:Nn \usv_set:nnn
143 { \tl_set:cn { g_@@_#1_#2_usv } {#3} }
144 \cs_new:Nn \@@_to_usv:nn
145 { \use:c { g_@@_#1_#2_usv } }
146 \prg_new_conditional:Nnn \@@_usv_if_exist:nn {T,F,TF}
147 {
148   \cs_if_exist:cTF { g_@@_#1_#2_usv }
149   \prg_return_true: \prg_return_false:
150 }

```

E.3 Package options

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

```

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

```

\@@_keys_choices:nn To simplify the creation of option keys, let's iterate in pairs rather than worry about equals signs and commas.

```

153 \cs_new:Nn \@@_keys_choices:nn

```

⁵'u.s.v.' stands for 'Unicode scalar value'.

```

154 {
155   \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
156   \use:x
157   {
158     \exp_not:N \keys_define:nn {unicode-math}
159     {
160       #1 .choice: ,
161       \@@_tl_map_dbl:nN {#2} \@@_keys_choices_fn:nn
162     }
163   }
164 }
165 \cs_new:Nn \@@_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
166
167 \cs_new:Nn \@@_tl_map_dbl:nN
168 {
169   \__@@_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
170 }
171 \cs_new:Nn \__@@_tl_map_dbl:Nnn
172 {
173   \quark_if_recursion_tail_stop:n {#2}
174   \quark_if_recursion_tail_stop:n {#3}
175   #1 {#2} {#3}
176   \__@@_tl_map_dbl:Nnn #1
177 }

```

Compatibility

```

178 \@@_keys_choices:nn {mathup}
179 {
180   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
181   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
182 }
183 \@@_keys_choices:nn {mathrm}
184 {
185   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
186   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
187 }
188 \@@_keys_choices:nn {mathit}
189 {
190   {sym} { \bool_set_false:N \g_@@_mathit_text_bool }
191   {text} { \bool_set_true:N \g_@@_mathit_text_bool }
192 }
193 \@@_keys_choices:nn {mathbf}
194 {
195   {sym} { \bool_set_false:N \g_@@_mathbf_text_bool }
196   {text} { \bool_set_true:N \g_@@_mathbf_text_bool }
197 }
198 \@@_keys_choices:nn {mathsf}
199 {
200   {sym} { \bool_set_false:N \g_@@_mathsf_text_bool }

```

```

201 {text} { \bool_set_true:N \g_@@_mathsf_text_bool }
202 }
203 \@@_keys_choices:nn {mathtt}
204 {
205 {sym} { \bool_set_false:N \g_@@_mathtt_text_bool }
206 {text} { \bool_set_true:N \g_@@_mathtt_text_bool }
207 }

```

math-style

```

208 \@@_keys_choices:nn {normal-style}
209 {
210 {ISO} {
211 \bool_set_false:N \g_@@_literal_bool
212 \bool_set_false:N \g_@@_upGreek_bool
213 \bool_set_false:N \g_@@_upgreek_bool
214 \bool_set_false:N \g_@@_upLatin_bool
215 \bool_set_false:N \g_@@_uplatin_bool
216 }
217 {TeX} {
218 \bool_set_false:N \g_@@_literal_bool
219 \bool_set_true:N \g_@@_upGreek_bool
220 \bool_set_false:N \g_@@_upgreek_bool
221 \bool_set_false:N \g_@@_upLatin_bool
222 \bool_set_false:N \g_@@_uplatin_bool
223 }
224 {french} {
225 \bool_set_false:N \g_@@_literal_bool
226 \bool_set_true:N \g_@@_upGreek_bool
227 \bool_set_true:N \g_@@_upgreek_bool
228 \bool_set_true:N \g_@@_upLatin_bool
229 \bool_set_false:N \g_@@_uplatin_bool
230 }
231 {upright} {
232 \bool_set_false:N \g_@@_literal_bool
233 \bool_set_true:N \g_@@_upGreek_bool
234 \bool_set_true:N \g_@@_upgreek_bool
235 \bool_set_true:N \g_@@_upLatin_bool
236 \bool_set_true:N \g_@@_uplatin_bool
237 }
238 {literal} {
239 \bool_set_true:N \g_@@_literal_bool
240 }
241 }
242 \@@_keys_choices:nn {math-style}
243 {
244 {ISO} {
245 \unimathsetup { nabla=upright, partial=italic,
246 normal-style=ISO, bold-style=ISO, sans-style=italic }

```

```

247     }
248     {TeX} {
249         \unimathsetup { nabra=upright, partial=italic,
250             normal-style=TeX, bold-style=TeX, sans-style=upright }
251     }
252     {french} {
253         \unimathsetup { nabra=upright, partial=upright,
254             normal-style=french, bold-style=upright, sans-style=upright }
255     }
256     {upright} {
257         \unimathsetup { nabra=upright, partial=upright,
258             normal-style=upright, bold-style=upright, sans-style=upright }
259     }
260     {literal} {
261         \unimathsetup { colon=literal, nabra=literal, partial=literal,
262             normal-style=literal, bold-style=literal, sans-style=literal }
263     }
264 }

```

bold-style

```

265 \@@_keys_choices:nn {bold-style}
266 {
267     {ISO} {
268         \bool_set_false:N \g_@@_bfliteral_bool
269         \bool_set_false:N \g_@@_bfupGreek_bool
270         \bool_set_false:N \g_@@_bfupgreek_bool
271         \bool_set_false:N \g_@@_bfupLatin_bool
272         \bool_set_false:N \g_@@_bfuplatin_bool
273     }
274     {TeX} {
275         \bool_set_false:N \g_@@_bfliteral_bool
276         \bool_set_true:N \g_@@_bfupGreek_bool
277         \bool_set_false:N \g_@@_bfupgreek_bool
278         \bool_set_true:N \g_@@_bfupLatin_bool
279         \bool_set_true:N \g_@@_bfuplatin_bool
280     }
281     {upright} {
282         \bool_set_false:N \g_@@_bfliteral_bool
283         \bool_set_true:N \g_@@_bfupGreek_bool
284         \bool_set_true:N \g_@@_bfupgreek_bool
285         \bool_set_true:N \g_@@_bfupLatin_bool
286         \bool_set_true:N \g_@@_bfuplatin_bool
287     }
288     {literal} {
289         \bool_set_true:N \g_@@_bfliteral_bool
290     }
291 }

```

sans-style

```
292 \@@_keys_choices:nn {sans-style}
293 {
294   {italic} { \bool_set_false:N \g_@@_upsans_bool }
295   {upright} { \bool_set_true:N \g_@@_upsans_bool }
296   {literal} { \bool_set_true:N \g_@@_sfliteral_bool }
297 }
```

Nabla and partial

```
298 \@@_keys_choices:nn {nabla}
299 {
300   {upright} {
301             \bool_set_false:N \g_@@_literal_Nabla_bool
302             \bool_set_true:N \g_@@_upNabla_bool
303           }
304   {italic} {
305             \bool_set_false:N \g_@@_literal_Nabla_bool
306             \bool_set_false:N \g_@@_upNabla_bool
307           }
308   {literal} { \bool_set_true:N \g_@@_literal_Nabla_bool }
309 }
310 \@@_keys_choices:nn {partial}
311 {
312   {upright} {
313             \bool_set_false:N \g_@@_literal_partial_bool
314             \bool_set_true:N \g_@@_uppartial_bool
315           }
316   {italic} {
317             \bool_set_false:N \g_@@_literal_partial_bool
318             \bool_set_false:N \g_@@_uppartial_bool
319           }
320   {literal} { \bool_set_true:N \g_@@_literal_partial_bool }
321 }
```

Epsilon and phi shapes

```
322 \@@_keys_choices:nn {vargreek-shape}
323 {
324   {unicode} { \bool_set_false:N \g_@@_texgreek_bool }
325   {TeX} { \bool_set_true:N \g_@@_texgreek_bool }
326 }
```

Colon style

```
327 \@@_keys_choices:nn {colon}
328 {
329   {literal} { \bool_set_true:N \g_@@_literal_colon_bool }
330   {TeX} { \bool_set_false:N \g_@@_literal_colon_bool }
331 }
```

Slash delimiter style

```
332 \@@_keys_choices:nn {slash-delimiter}
333 {
334   {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
335   {frac} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
336   {div} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
337 }
```

Active fraction style

```
338 \@@_keys_choices:nn {active-frac}
339 {
340   {small}
341   {
342     \cs_if_exist:NTF \tfrac
343     { \bool_set_true:N \l_@@_smallfrac_bool }
344     {
345       \@@_warning:n {no-tfrac}
346       \bool_set_false:N \l_@@_smallfrac_bool
347     }
348     \use:c {@@_setup_active_frac:}
349   }
350
351   {normalsize}
352   {
353     \bool_set_false:N \l_@@_smallfrac_bool
354     \use:c {@@_setup_active_frac:}
355   }
356 }
```

Debug/tracing

```
357 \keys_define:nn {unicode-math}
358 {
359   warnings-off .code:n =
360   {
361     \clist_map_inline:nn {#1}
362     { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
363   }
364 }
365 \@@_keys_choices:nn {trace}
366 {
367   {on} {} % default
368   {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
369   {off} { \msg_redirect_module:nnn { unicode-math } { log } { none } }
370 }
371 \unimathsetup {math-style=TeX}
372 \unimathsetup {slash-delimiter=ascii}
373 \unimathsetup {trace=off}
```

```

374 \unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}
375 \cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
376 \ProcessKeysOptions {unicode-math}

```

E.4 Programmers' interface

`\unimath_get_mathstyle:` This command expands to the currently math style.

```

377 \cs_new:Nn \unimath_get_mathstyle:
378 {
379   \tl_use:N \l_@@_mathstyle_tl
380 }

```

End of preamble code.

```

381 </preamble&!XE&!LU>

```

(Error messages and warning definitions go here from the `msg` chunk defined in section §N on page 94.)

F Bifurcation

And here the split begins. Most of the code is still shared, but code for Lua \TeX uses the ‘LU’ flag and code for Xe \TeX uses ‘XE’.

```

382 <*package&(XE|LU)>
383 \ExplSyntaxOn

```

F.1 Engine differences

Xe \TeX before version 0.9999 did not support `\U` prefix for extended math primitives, and while Lua \TeX had it from the start, prior 0.75.0 the \LaTeX format did not provide them without the `\luatex` prefix. We assume that users of `unicode-math` are using up-to-date engines however.

```

384 <*LU>
385 \RequirePackage{luaotfload} [2014/05/18]
386 \RequirePackage{lualatex-math}[2011/08/07]
387 </LU>

```

F.2 Overcoming `\@onlypreamble`

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

```

388 \tl_map_inline:nn
389 {
390   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
391   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
392   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
393   \version@list\version@elt\alpha@list\alpha@elt
394   \restore@mathversion\init@restore@version\dorestore@version\process@table

```

```

395 \new@mathversion\DeclareSymbolFont\group@list\group@elt
396 \new@symbolfont\SetSymbolFont\SetSymbolFont@get@cdp
397 \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
398 \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
399 \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
400 \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
401 \set@@mathdelimiter\DeclareMathRadical\mathchar@type
402 \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
403 }
404 {
405 \tl_remove_once:Nn \@preamblecmds {\do#1}
406 }

```

G Fundamentals

G.1 Setting math chars, math codes, etc.

`\@@_set_mathsymbol:nNNn` #1 : A L^AT_EX symbol font, e.g., operators
 #2 : Symbol macro, e.g., `\alpha`
 #3 : Type, e.g., `\mathalpha`
 #4 : Slot, e.g., "221E

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

The catcode setting is to work around (strange?) behaviour in LuaTeX in which catcode 11 characters don't have italic correction for maths. We don't adjust ascii chars, however, because certain punctuation should not have their catcodes changed.

```

407 \cs_set:Nn \@@_set_mathsymbol:nNNn
408 {
409   \bool_if:nT
410   {
411     \int_compare_p:nNn {#4} > {127} &&
412     \int_compare_p:nNn { \char_value_catcode:n {#4} } = {11}
413   }
414   { \char_set_catcode_other:n {#4} }
415
416   \tl_case:Nn #3
417   {
418     \mathord { \@@_set_mathcode:nnn {#4} {#3} {#1} }
419     \mathalpha { \@@_set_mathcode:nnn {#4} {#3} {#1} }
420     \mathbin { \@@_set_mathcode:nnn {#4} {#3} {#1} }
421     \mathrel { \@@_set_mathcode:nnn {#4} {#3} {#1} }
422     \mathpunct { \@@_set_mathcode:nnn {#4} {#3} {#1} }
423     \mathop { \@@_set_big_operator:nnn {#1} {#2} {#4} }
424     \mathopen { \@@_set_math_open:nnn {#1} {#2} {#4} }
425     \mathclose { \@@_set_math_close:nnn {#1} {#2} {#4} }
426     \mathfence { \@@_set_math_fence:nnnn {#1} {#2} {#3} {#4} }

```



```

427 \mathaccent
428 { \@@_set_math_accent:Nnnn #2 {fixed} {#1} {#4} }
429 \mathbotaccent
430 { \@@_set_math_accent:Nnnn #2 {bottom~ fixed} {#1} {#4} }
431 \mathaccentwide
432 { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
433 \mathbotaccentwide
434 { \@@_set_math_accent:Nnnn #2 {bottom} {#1} {#4} }
435 \mathover
436 { \@@_set_math_overunder:Nnnn #2 {} {#1} {#4} }
437 \mathunder
438 { \@@_set_math_overunder:Nnnn #2 {bottom} {#1} {#4} }
439 }
440 }

441 \edef\mathfence{\string\mathfence}
442 \edef\mathover{\string\mathover}
443 \edef\mathunder{\string\mathunder}
444 \edef\mathbotaccent{\string\mathbotaccent}
445 \edef\mathaccentwide{\string\mathaccentwide}
446 \edef\mathbotaccentwide{\string\mathbotaccentwide}

```

\@@_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 *L^AT_EX*/amsmath.)
- Define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

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

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

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

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

```

447 \cs_new:Nn \@@_set_big_operator:nnn
448 {
449   \group_begin:
450   \char_set_catcode_active:n {#3}
451   \@@_char_gmake_mathactive:n {#3}

```

```

452 \@@_active_char_set:wc #3 \q_nil { \cs_to_str:N #2 _sym }
453 \group_end:
454
455 \@@_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
456
457 \cs_gset:cpx { \cs_to_str:N #2 _sym }
458 {
459 \exp_not:c { \cs_to_str:N #2 op }
460 \exp_not:n { \tl_if_in:NnT \l_@@_nolimits_tl {#2} \nolimits }
461 }
462 }

```

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

```

463 \cs_new:Nn \@@_set_math_open:nnn
464 {
465 \tl_if_in:NnTF \l_@@_radicals_tl {#2}
466 {
467 \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
468 { \@@_radical:nn {#1} {#3} }
469 \tl_set:cn {l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
470 }
471 {
472 \@@_set_delcode:nnn {#1} {#3} {#3}
473 \@@_set_mathcode:nnn {#3} \mathopen {#1}
474 \cs_gset_protected_nopar:Npx #2
475 { \@@_delimiter:Nnn \mathopen {#1} {#3} }
476 }
477 }

```

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

```

478 \cs_new:Nn \@@_set_math_close:nnn
479 {
480 \@@_set_delcode:nnn {#1} {#3} {#3}
481 \@@_set_mathcode:nnn {#3} \mathclose {#1}
482 \cs_gset_protected_nopar:Npx #2
483 { \@@_delimiter:Nnn \mathclose {#1} {#3} }
484 }

```

`\@@_set_math_fence:nnnn` #1 : Symbol font name
 #2 : Macro to assign
 #3 : Type, *e.g.*, `\mathalpha`
 #4 : Glyph slot

```

485 \cs_new:Nn \@@_set_math_fence:nnnn
486 {
487 \@@_set_mathcode:nnn {#4} {#3} {#1}

```

```

488 \@@_set_delcode:nnn {#1} {#4} {#4}
489 \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
490 { \@@_delimiter:Nnn \mathopen {#1} {#4} }
491 \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
492 { \@@_delimiter:Nnn \mathclose {#1} {#4} }
493 }

```

`\@@_set_math_accent:Nnnn` #1 : Accend command
 #2 : Accent type (string)
 #3 : Symbol font name
 #4 : Glyph slot

```

494 \cs_new:Nn \@@_set_math_accent:Nnnn
495 {
496   \cs_gset_protected_nopar:Npx #1
497   { \@@_accent:nnn {#2} {#3} {#4} }
498 }

```

`\@@_set_math_overunder:Nnnn` #1 : Accend command
 #2 : Accent type (string)
 #3 : Symbol font name
 #4 : Glyph slot

```

499 \cs_new:Nn \@@_set_math_overunder:Nnnn
500 {
501   \cs_gset_protected_nopar:Npx #1 ##1
502   {
503     \mathop
504     { \@@_accent:nnn {#2} {#3} {#4} {##1} }
505     \limits
506   }
507 }

```

G.2 \setmathalphabet

`\setmathalphabet`

```

508 \keys_define:nn {@@_mathface}
509 {
510   version .code:n =
511   { \tl_set:Nn \l_@@_mversion_tl {#1} }
512 }
513
514 \DeclareDocumentCommand \setmathfontface { m O{} m O{} }
515 {
516   \tl_clear:N \l_@@_mversion_tl
517
518   \keys_set_known:nnN {@@_mathface} {#2,#4} \l_@@_keyval_clist
519   \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
520   { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}
521

```

```

522 \tl_if_empty:NT \l_@@_mversion_tl
523 {
524   \tl_set:Nn \l_@@_mversion_tl {normal}
525   \DeclareMathAlphabet #1 {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\mdde-
fault} {\updefault}
526 }
527 \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\md-
default} {\updefault}
528
529 % integrate with fontspec's \setmathrm etc:
530 \tl_case:Nn #1
531 {
532   \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
533   \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
534   \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
535 }
536 }
537
538 \onlypreamble \setmathfontface

```

Note that L^AT_EX's SetMathAlphabet simply doesn't work to "reset" a maths alphabet font after `\begin{document}`, so unlike most of the other maths commands around we still restrict this one to the preamble.

`\setoperatorfont` TODO: add check?

```

539 \DeclareDocumentCommand \setoperatorfont {m}
540 { \tl_set:Nn \g_@@_operator_mathfont_tl {#1} }
541 \setoperatorfont{\mathrm}

```

G.3 Hooks into fontspec

Historically, `\mathrm` and so on were completely overwritten by `unicode-math`, and `fontspec`'s methods for setting these fonts in the classical manner were bypassed.

While we could now re-activate the way that `fontspec` does the following, because we can now change maths fonts whenever it's better to define new commands in `unicode-math` to define the `\mathXYZ` fonts.

G.3.1 Text font

```

542 \cs_generate_variant:Nn \tl_if_eq:nnT {o}
543 \cs_set:Nn \__fontspec_setmainfont:nn
544 {
545   \fontspec_set_family:Nnn \rmdefault {#1}{#2}
546   \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault}
547   {
548     (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
549     (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
550     \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
551     \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
552     \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault

```

```

553 }
554 \normalfont
555 \ignorespaces
556 }
557
558 \cs_set:Nn \__fontspec_setsansfont:nn
559 {
560   \fontspec_set_family:Nnn \sfdefault {#1}{#2}
561   \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault}
562   {
563     (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
564     (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
565     \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
566     \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
567   }
568   \normalfont
569   \ignorespaces
570 }
571
572 \cs_set:Nn \__fontspec_setmonofont:nn
573 {
574   \fontspec_set_family:Nnn \ttdefault {#1}{#2}
575   \tl_if_eq:onT {\g__fontspec_mathtt_tl} {\ttdefault}
576   {
577     (XE) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
578     (LU) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
579     \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
580     \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
581   }
582   \normalfont
583   \ignorespaces
584 }

```

G.3.2 *Maths font*

If the maths fonts are set explicitly, then the text commands above will not execute their branches to set the maths font alphabets.

```

585 \cs_set:Nn \__fontspec_setmathrm:nn
586 {
587   (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
588   (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
589   \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
590   \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
591   \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
592 }
593 \cs_set:Nn \__fontspec_setboldmathrm:nn
594 {
595   (XE) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {#1} {#2}
596   (LU) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {Renderer=Basic,#1} {#2}
597   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\updefault

```

```

598 \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
599 \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\itdefault
600 }
601 \cs_set:Nn \__fontspec_setmathsf:nn
602 {
603 (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
604 (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
605 \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
606 \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
607 }
608 \cs_set:Nn \__fontspec_setmathtt:nn
609 {
610 (XE) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
611 (LU) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
612 \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
613 \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
614 }

```

G.4 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 (first optional argument retained for backwards compatibility)
 #2 : font name
 [#3]: font features

```

615 \DeclareDocumentCommand \setmathfont { O{} m O{} }
616 {
617 \tl_set:Nn \l_@@_fontname_tl {#2}
618 \@@_init:

```

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

```

619 \cs_if_exist:cf { S@ \f@size } { \calculate@math@sizes }
620 \csname S@\f@size\endcsname

```

Parse options and tell people what's going on:

```

621 \keys_set_known:nnN {unicode-math} {#1,#3} \l_@@_unknown_keys_clist
622 \bool_if:NT \l_@@_init_bool { \@@_log:n {default-math-font} }

```

Use fontspec to select a font to use. After loading the font, we detect what sizes it recommends for scriptsize and scriptscriptsize, so after setting those values appropriately, we reload the font to take these into account.

```

623
624 (debug) \csname TIC\endcsname
625 \@@_fontspec_select_font:
626 (debug) \csname TOC\endcsname
627 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }

```

```

628 {
629   \@@_declare_math_sizes:
630   \@@_fontspec_select_font:
631 }

```

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

```

632 \cs_if_exist:cF { sym \@@_symfont_tl }
633 {
634   \DeclareSymbolFont{\@@_symfont_tl}
635     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
636 }
637 \SetSymbolFont{\@@_symfont_tl}{\l_@@_mversion_tl}
638   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}

```

Set the bold math version.

```

639 \tl_set:Nn \l_@@_tmpa_tl {normal}
640 \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
641 {
642   \SetSymbolFont{\@@_symfont_tl}{bold}
643     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
644 }

```

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

```

645 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
646 {
647   \bool_set_true:N \g_@@_mainfont_already_set_bool
648   \@@_setup_legacy_fam_two:
649   \@@_setup_legacy_fam_three:
650 }

```

And now we input every single maths char.

```

651 (debug) \csname TIC\endcsname
652 \@@_input_math_symbol_table:
653 (debug) \csname TOC\endcsname

```

Finally,

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

```

654 \@@_remap_symbols:
655 \@@_setup_mathactives:
656 \@@_setup_delcodes:
657 (debug) \csname TIC\endcsname
658 \@@_setup_alphabets:
659 (debug) \csname TOC\endcsname
660 \@@_setup_negations:

```

Prevent spaces, and that's it:

```
661 \ignorespaces
662 }
```

Backward compatibility alias.

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

`\@@_init:`

```
664 \cs_new:Nn \@@_init:
665 {
```

- Initially assume we're using a proper OpenType font with unicode maths.

```
666 \bool_set_true:N \l_@@_ot_math_bool
```

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

```
667 \cs_set_eq:NN \glb@currsiz \scan_stop:
```

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

```
668 \bool_set_true:N \l_@@_init_bool
669 \seq_clear:N \l_@@_char_range_seq
670 \clist_clear:N \l_@@_char_nrange_clist
671 \seq_clear:N \l_@@_mathalph_seq
672 \seq_clear:N \l_@@_missing_alph_seq
```

- By default use the 'normal' math version.

```
673 \tl_set:Nn \l_@@_mversion_tl {normal}
```

- Other range initialisations.

```
674 \tl_set:Nn \@@_symfont_tl {operators}
675 \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_noparse:nnn
676 \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
677 \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_noparse:nnn
678 \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
679 \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
680 \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_noparse:nn
681 \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_noparse:nNN
```

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

```
682 \tl_set:Nn \l_@@_script_features_tl {Style=MathScript}
683 \tl_set:Nn \l_@@_sscript_features_tl {Style=MathScriptScript}
684 \tl_set_eq:NN \l_@@_script_font_tl \l_@@_fontname_tl
685 \tl_set_eq:NN \l_@@_sscript_font_tl \l_@@_fontname_tl
```



```
686 }
```

`\@@declare_math_sizes:` Set the math sizes according to the recommended font parameters:

```
687 \cs_new:Nn \@@declare_math_sizes:
688 {
689   \dim_compare:nF { \fontdimen 10 \l_@@_font == 0pt }
690   {
691     \DeclareMathSizes { \f@size } { \f@size }
692     { \@@_fontdimen_to_scale:nn {10} {\l_@@_font} }
693     { \@@_fontdimen_to_scale:nn {11} {\l_@@_font} }
694   }
695 }
```

`\@@setup_legacy_fam_two:` T_EX won't load the same font twice at the same scale, so we need to magnify this one by an imperceptible amount.

```
696 \cs_new:Nn \@@setup_legacy_fam_two:
697 {
698   \fontspec_set_family:Nxn \l_@@_family_tl
699   {
700     \l_@@_font_keyval_tl,
701     Scale=1.00001,
702     FontAdjustment =
703     {
704       \fontdimen8\font= \@@_get_fontparam:nn {43} {FractionNumeratorDis-
705         playStyleShiftUp}\relax
706       \fontdimen9\font= \@@_get_fontparam:nn {42} {FractionNumerator-
707         ShiftUp}\relax
708       \fontdimen10\font=\@@_get_fontparam:nn {32} {StackTopShiftUp}\relax
709       \fontdimen11\font=\@@_get_fontparam:nn {45} {FractionDenominatorDis-
710         playStyleShiftDown}\relax
711       \fontdimen12\font=\@@_get_fontparam:nn {44} {FractionDenominatorShift-
712         Down}\relax
713       \fontdimen13\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
714       \fontdimen14\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
715       \fontdimen15\font=\@@_get_fontparam:nn {22} {SuperscriptShif-
716         tUpCramped}\relax
717       \fontdimen16\font=\@@_get_fontparam:nn {18} {SubscriptShiftDown}\relax
718       \fontdimen17\font=\@@_get_fontparam:nn {18} {SubscriptShiftDownWith-
719         Superscript}\relax
720       \fontdimen18\font=\@@_get_fontparam:nn {24} {SuperscriptBaselineDrop-
721         Max}\relax
722       \fontdimen19\font=\@@_get_fontparam:nn {20} {SubscriptBaselineDrop-
723         Min}\relax
724       \fontdimen20\font=0pt\relax % delim1 = FractionDelimiterDisplaySize
725       \fontdimen21\font=0pt\relax % delim2 = FractionDelimiterSize
726       \fontdimen22\font=\@@_get_fontparam:nn {15} {AxisHeight}\relax
727     }
728   } { \l_@@_fontname_tl }
729   \SetSymbolFont{symbols}{\l_@@_mversion_tl}
```

```

722     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
723
724     \tl_set:Nn \l_@@_tmpa_tl {normal}
725     \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
726     {
727         \SetSymbolFont{symbols}{bold}
728         {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
729     }
730 }

```

\@@_setup_legacy_fam_three: Similarly, this font is shrunk by an imperceptable amount for T_EX to load it again.

```

731 \cs_new:Nn \@@_setup_legacy_fam_three:
732 {
733     \fontspec_set_family:Nxn \l_@@_family_tl
734     {
735         \l_@@_font_keyval_tl,
736         Scale=0.99999,
737         FontAdjustment={
738             \fontdimen8\font= \@@_get_fontparam:nn {48} {FractionRuleThick-
ness}\relax
739             \fontdimen9\font= \@@_get_fontparam:nn {28} {UpperLimitGapMin}\relax
740             \fontdimen10\font=\@@_get_fontparam:nn {30} {LowerLimitGapMin}\relax
741             \fontdimen11\font=\@@_get_fontparam:nn {29} {UpperLimitBaselineR-
iseMin}\relax
742             \fontdimen12\font=\@@_get_fontparam:nn {31} {LowerLimitBaselineDrop-
Min}\relax
743             \fontdimen13\font=0pt\relax
744         }
745     } {\l_@@_fontname_tl}
746     \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
747     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
748
749     \tl_set:Nn \l_@@_tmpa_tl {normal}
750     \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
751     {
752         \SetSymbolFont{largesymbols}{bold}
753         {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
754     }
755 }

756 \cs_new:Nn \@@_get_fontparam:nn
757 (XE) { \the\fontdimen#1\l_@@_font\relax }
758 (LU) { \directlua{fontspec.mathfontdimen("\l_@@_font","#2")}}

```

\@@_fontspec_select_font: Select the font with \fontspec and define \l_@@_font from it.

```

759 \cs_new:Nn \@@_fontspec_select_font:
760 {
761     \tl_set:Nx \l_@@_font_keyval_tl {
762 (LU)         Renderer = Basic,
763         BoldItalicFont = {}, ItalicFont = {},

```

```

764     Script = Math,
765     SizeFeatures =
766     {
767     {
768         Size = \tf@size-
769     } ,
770     {
771         Size = \sf@size-\tf@size ,
772         Font = \l_@@_script_font_tl ,
773         \l_@@_script_features_tl
774     } ,
775     {
776         Size = -\sf@size ,
777         Font = \l_@@_sscript_font_tl ,
778         \l_@@_sscript_features_tl
779     }
780     } ,
781     \l_@@_unknown_keys_clist
782 }
783 \fontspec_set_fontface:NNxn \l_@@_font \l_@@_family_tl
784 {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

```

Check whether we're using a real maths font:

```

785 \group_begin:
786     \fontfamily{\l_@@_family_tl}\selectfont
787     \fontspec_if_script:nF {math} {\bool_gset_false:N \l_@@_ot_math_bool}
788 \group_end:
789 }

```

G.4.1 Functions for setting up symbols with mathcodes

\@@_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 §H.3 for the code that enables this.

```

790 \cs_set:Nn \@@_process_symbol_noparse:nnn
791 {
792     \@@_set_mathsymbol:nNNn {\@@_symfont_tl} #2 #3 {#1}
793 }
794 \cs_set:Nn \@@_process_symbol_parse:nnn
795 {
796     \@@_if_char_spec:nNNT {#1} {#2} {#3}
797     {
798         \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
799     }
800 }

```

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

```

\@@_remap_symbol_noparse:nnn
\@@_remap_symbol_parse:nnn
801 \cs_new:Npn \@@_remap_symbols:
802 {

```

```

803 \@@_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
804 \@@_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "centred as-
    terisk"
805 \bool_if:NF \g_@@_literal_colon_bool
806 {
807   \@@_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
808 }
809 }

```

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

```

810 \cs_new:Nn \@@_remap_symbol_parse:nnn
811 {
812   \@@_if_char_spec:nNTT {#3} {\@nil} {#2}
813   { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
814 }
815 \cs_new:Nn \@@_remap_symbol_noparse:nnn
816 {
817   \clist_map_inline:nn {#1}
818   { \@@_set_mathcode:nnnn {##1} {#2} {\@@_symfont_tl} {#3} }
819 }

```

G.4.2 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.

`\@@_setup_mathactives:`

```

820 \cs_new:Npn \@@_setup_mathactives:
821 {
822   \@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
823   \@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
824   \@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
825   \@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar \mathord
826   \@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
827   \@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
828   \@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
829   \@@_make_mathactive:nNN {'\'} \mathstraightquote \mathord
830   \@@_make_mathactive:nNN {'\'} \mathbacktick \mathord
831 }

```

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

```

832 \cs_new:Nn \@@_make_mathactive_parse:nNN
833 {
834   \@@_if_char_spec:nNTT {#1} #2 #3
835   { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
836 }
837 \cs_new:Nn \@@_make_mathactive_noparse:nNN
838 {

```

```

839 \@@_set_mathchar:NNnn #2 #3 {\@@_symfont_tl} {#1}
840 \@@_char_gmake_mathactive:n {#1}
841 }

```

G.4.3 Delimiter codes

\@@_assign_delcode:nn

```

842 \cs_new:Nn \@@_assign_delcode_noparse:nn
843 {
844   \@@_set_delcode:nnn \@@_symfont_tl {#1} {#2}
845 }
846 \cs_new:Nn \@@_assign_delcode_parse:nn
847 {
848   \@@_if_char_spec:nNNT {#2} {\@nil} {\@nil}
849   {
850     \@@_assign_delcode_noparse:nn {#1} {#2}
851   }
852 }

```

\@@_assign_delcode:n Shorthand.

```

853 \cs_new:Nn \@@_assign_delcode:n { \@@_assign_delcode:nn {#1} {#1} }

```

\@@_setup_delcodes: 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.

```

854 \cs_new:Npn \@@_setup_delcodes:
855 {
856   % ensure \left. and \right. work:
857   \@@_set_delcode:nnn \@@_symfont_tl {'\.'} {\c_zero}
858   % this is forcefully done to fix a bug -- indicates a larger problem!
859
860   \@@_assign_delcode:nn {'\'} {\g_@@_slash_delimiter_usv}
861   \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
862   \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
863   \@@_assign_delcode:n {"005C} % backslash
864   \@@_assign_delcode:nn {'\<} {"27E8} % angle brackets with ascii notation
865   \@@_assign_delcode:nn {'\>} {"27E9} % angle brackets with ascii notation
866   \@@_assign_delcode:n {"2191} % up arrow
867   \@@_assign_delcode:n {"2193} % down arrow
868   \@@_assign_delcode:n {"2195} % updown arrow
869   \@@_assign_delcode:n {"219F} % up arrow twohead
870   \@@_assign_delcode:n {"21A1} % down arrow twohead
871   \@@_assign_delcode:n {"21A5} % up arrow from bar
872   \@@_assign_delcode:n {"21A7} % down arrow from bar
873   \@@_assign_delcode:n {"21A8} % updown arrow from bar
874   \@@_assign_delcode:n {"21BE} % up harpoon right
875   \@@_assign_delcode:n {"21BF} % up harpoon left
876   \@@_assign_delcode:n {"21C2} % down harpoon right

```

```

877 \@@_assign_delcode:n {"21C3} % down harpoon left
878 \@@_assign_delcode:n {"21C5} % arrows up down
879 \@@_assign_delcode:n {"21F5} % arrows down up
880 \@@_assign_delcode:n {"21C8} % arrows up up
881 \@@_assign_delcode:n {"21CA} % arrows down down
882 \@@_assign_delcode:n {"21D1} % double up arrow
883 \@@_assign_delcode:n {"21D3} % double down arrow
884 \@@_assign_delcode:n {"21D5} % double updown arrow
885 \@@_assign_delcode:n {"21DE} % up arrow double stroke
886 \@@_assign_delcode:n {"21DF} % down arrow double stroke
887 \@@_assign_delcode:n {"21E1} % up arrow dashed
888 \@@_assign_delcode:n {"21E3} % down arrow dashed
889 \@@_assign_delcode:n {"21E7} % up white arrow
890 \@@_assign_delcode:n {"21E9} % down white arrow
891 \@@_assign_delcode:n {"21EA} % up white arrow from bar
892 \@@_assign_delcode:n {"21F3} % updown white arrow
893 }

```

G.5 (Big) operators

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

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

l@@_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 $\text{\@@_set_mathsymbol:nNNn}$). 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 $\int\int\int\int$, but that might be a matter of preference.

```

894 \tl_new:N \l_@@_nolimits_tl
895 \tl_set:Nn \l_@@_nolimits_tl
896 {
897   \int\iint\iiint\iiiiint\oint\oiint\oiint
898   \intclockwise\varointclockwise\ointctrclockwise\sumint
899   \intbar\intBar\oint\cirfnint\awint\rrppolint
900   \scpolint\ncpolint\pointint\sqint\intlarhk\intx
901   \intcap\intcup\upint\lowint
902 }

```

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

```

903 \DeclareDocumentCommand \addnolimits {m}
904 {
905   \tl_put_right:Nn \l_@@_nolimits_tl {#1}

```

```
906 }
```

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

```
907 \DeclareDocumentCommand \removenolimits {m}
908 {
909   \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
910 }
```

G.6 Radicals

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

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

`\l_@@_radicals_tl` We organise radicals in the same way as nolimits-operators.

```
911 \tl_new:N \l_@@_radicals_tl
912 \tl_set:Nn \l_@@_radicals_tl {\sqrt \longdivision}
```

G.7 Maths accents

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

G.8 Common interface for font parameters

X_YTeX and LuaTeX have different interfaces for math font parameters. We use LuaTeX's interface because it's much better, but rename the primitives to be more L^AT_EX3-like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the LuaTeX names, with underscores inserted between words. For every parameter `\Umath⟨LuaTeX name⟩`, we define an expandable getter command `\@@_⟨LATEX3 name⟩:N` and a protected setter command `\@@_set_⟨LATEX3 name⟩:Nn`. The getter command takes one of the style primitives (`\displaystyle` etc.) and expands to the font parameter, which is a *⟨dimension⟩*. The setter command takes a style primitive and a dimension expression, which is parsed with `\dim_eval:n`.

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

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

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

`\@@_new_cramped_style:N` #1 : command

Define $\langle command \rangle$ as a new cramped style switch. For Lua \TeX , simply rename the corresponding primitive if it is not already defined. For Xe \TeX , define $\langle command \rangle$ as a new quark.

```

913 \cs_new_protected_nopar:Nn \@@_new_cramped_style:N
914 (XE) { \quark_new:N #1 }
915 (LU) {
916 (LU)   \cs_if_exist:NF #1
917 (LU)   { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
918 (LU) }

```

`\crampeddisplaystyle` The cramped style commands.

```

\crampedtextstyle 919 \@@_new_cramped_style:N \crampeddisplaystyle
\crampedscriptstyle 920 \@@_new_cramped_style:N \crampedtextstyle
\crampedscriptscriptstyle 921 \@@_new_cramped_style:N \crampedscriptstyle
922 \@@_new_cramped_style:N \crampedscriptscriptstyle

```

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

`\@@_font_dimen:Nnnnn` #1 : style token

#2 : font dimen for display style

#3 : font dimen for cramped display style

#4 : font dimen for non-display styles

#5 : font dimen for cramped non-display styles

Map math style to Xe \TeX math font dimension. $\langle style token \rangle$ must be one of the style switches (`\displaystyle`, `\crampeddisplaystyle`, ...). The other parameters are integer constants referring to font dimension numbers. The macro expands to a dimension which contains the appropriate font dimension.

```

923 (*XE)
924 \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
925   \fontdimen
926   \cs_if_eq:NNTF #1 \displaystyle {
927     #2 \textfont
928   } {
929     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
930       #3 \textfont

```



```

931 } {
932   \cs_if_eq:NNTF #1 \textstyle {
933     #4 \textfont
934   } {
935     \cs_if_eq:NNTF #1 \crampedtextstyle {
936       #5 \textfont
937     } {
938       \cs_if_eq:NNTF #1 \scriptstyle {
939         #4 \scriptfont
940       } {
941         \cs_if_eq:NNTF #1 \crampedscriptstyle {
942           #5 \scriptfont
943         } {
944           \cs_if_eq:NNTF #1 \scriptscriptstyle {
945             #4 \scriptscriptfont
946           } {

```

Should we check here if the style is invalid?

```

947       #5 \scriptscriptfont
948     }
949   }
950 }
951 }
952 }
953 }
954 }

```

Which family to use?

```

955   \c_two
956 }
957 </XE>

```

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

```

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

```

This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The Lua \TeX font parameter name is produced by removing all underscores and prefixing the result with `luatexUmath`. The X \TeX font dimension numbers must be integer constants.

```

958 \cs_new_protected_nopar:Nn \@@_font_param:nnnnn
959 < *XE>
960 {
961   \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
962   { #2 } { #3 } { #4 } { #5 }

```

```

963 }
964 </XE>
965 <*LU>
966 {
967   \tl_set:Nn \l_@@_tmpa_tl { #1 }
968   \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
969   \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
970   { luatexUmath \l_@@_tmpa_tl }
971 }
972 </LU>

```

`\@@_font_param:nnn` #1 : name

#2 : font dimension for display style

#3 : font dimension for non-display styles

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

```

973 \cs_new_protected_nopar:Nn \@@_font_param:nnn
974 {
975   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
976 }

```

`\@@_font_param:nn` #1 : name

#2 : font dimension

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

```

977 \cs_new_protected_nopar:Nn \@@_font_param:nn
978 {
979   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
980 }

```

`\@@_font_param:n` #1 : name

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

```

981 \cs_new_protected_nopar:Nn \@@_font_param:n
982 <XE> { }
983 <LU> { \@@_font_param:nnnnn { #1 } { 0 } { 0 } { 0 } { 0 } }

```

`\@@_font_param_aux:NNnnnn` Auxiliary macros for generating font parameter accessor macros.

`\@@_font_param_aux:NNN`

```

984 <*XE>
985 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNnnnn
986 {
987   \cs_new_nopar:Npn #1 ##1
988   {

```

```

989 \@@_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
990 }
991 \cs_new_protected_nopar:Npn #2 ##1 ##2
992 {
993   #1 ##1 \dim_eval:n { ##2 }
994 }
995 }
996 \cs_generate_variant:Nn \@@_font_param_aux:NNnnnn { cc }
997 </XE>
998 <*LU>
999 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNN
1000 {
1001   \cs_new_nopar:Npn #1 ##1
1002   {
1003     #3 ##1
1004   }
1005   \cs_new_protected_nopar:Npn #2 ##1 ##2
1006   {
1007     #3 ##1 \dim_eval:n { ##2 }
1008   }
1009 }
1010 \cs_generate_variant:Nn \@@_font_param_aux:NNN { ccc }
1011 </LU>

```

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

```

1012 \@@_font_param:nn { axis } { 15 }
1013 \@@_font_param:nn { operator_size } { 13 }
1014 \@@_font_param:n { fraction_del_size }
1015 \@@_font_param:nnn { fraction_denom_down } { 45 } { 44 }
1016 \@@_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
1017 \@@_font_param:nnn { fraction_num_up } { 43 } { 42 }
1018 \@@_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
1019 \@@_font_param:nn { fraction_rule } { 48 }
1020 \@@_font_param:nn { limit_above_bgap } { 29 }
1021 \@@_font_param:n { limit_above_kern }
1022 \@@_font_param:nn { limit_above_vgap } { 28 }
1023 \@@_font_param:nn { limit_below_bgap } { 31 }
1024 \@@_font_param:n { limit_below_kern }
1025 \@@_font_param:nn { limit_below_vgap } { 30 }
1026 \@@_font_param:nn { over_delimiter_vgap } { 41 }
1027 \@@_font_param:nn { over_delimiter_bgap } { 38 }
1028 \@@_font_param:nn { under_delimiter_vgap } { 40 }
1029 \@@_font_param:nn { under_delimiter_bgap } { 39 }
1030 \@@_font_param:nn { overbar_kern } { 55 }
1031 \@@_font_param:nn { overbar_rule } { 54 }
1032 \@@_font_param:nn { overbar_vgap } { 53 }
1033 \@@_font_param:n { quad }
1034 \@@_font_param:nn { radical_kern } { 62 }
1035 \@@_font_param:nn { radical_rule } { 61 }

```

```

1036 \@@_font_param:nnn { radical_vgap } { 60 } { 59 }
1037 \@@_font_param:nn { radical_degree_before } { 63 }
1038 \@@_font_param:nn { radical_degree_after } { 64 }
1039 \@@_font_param:nn { radical_degree_raise } { 65 }
1040 \@@_font_param:nn { space_after_script } { 27 }
1041 \@@_font_param:nnn { stack_denom_down } { 35 } { 34 }
1042 \@@_font_param:nnn { stack_num_up } { 33 } { 32 }
1043 \@@_font_param:nnn { stack_vgap } { 37 } { 36 }
1044 \@@_font_param:nn { sub_shift_down } { 18 }
1045 \@@_font_param:nn { sub_shift_drop } { 20 }
1046 \@@_font_param:n { subsup_shift_down }
1047 \@@_font_param:nn { sub_top_max } { 19 }
1048 \@@_font_param:nn { subsup_vgap } { 25 }
1049 \@@_font_param:nn { sup_bottom_min } { 23 }
1050 \@@_font_param:nn { sup_shift_drop } { 24 }
1051 \@@_font_param:nnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
1052 \@@_font_param:nn { supsub_bottom_max } { 26 }
1053 \@@_font_param:nn { underbar_kern } { 58 }
1054 \@@_font_param:nn { underbar_rule } { 57 }
1055 \@@_font_param:nn { underbar_vgap } { 56 }
1056 \@@_font_param:n { connector_overlap_min }

```

H Font features

H.1 Math version

```

1057 \keys_define:nn {unicode-math}
1058 {
1059   version .code:n =
1060   {
1061     \tl_set:Nn \l_@@_mversion_tl {#1}
1062     \DeclareMathVersion {\l_@@_mversion_tl}
1063   }
1064 }

```

H.2 Script and scriptscript font options

```

1065 \keys_define:nn {unicode-math}
1066 {
1067   script-features .tl_set:N = \l_@@_script_features_tl ,
1068   sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
1069   script-font .tl_set:N = \l_@@_script_font_tl ,
1070   sscript-font .tl_set:N = \l_@@_sscript_font_tl ,
1071 }

```

H.3 Range processing

```

1072 \keys_define:nn {unicode-math}
1073 {
1074   range .code:n =

```

```

1075 {
1076   \bool_set_false:N \l_@@_init_bool

```

Set processing functions if we’re not defining the full Unicode math repertoire. Math symbols are defined with `_@@_sym:nnn`; see section §G.4.1 for the individual definitions

```

1077   \int_incr:N \g_@@_fam_int
1078   \tl_set:Nx \@@_symfont_tl {@@_fam\int_use:N\g_@@_fam_int}
1079   \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
1080   \cs_set_eq:NN \@@_set_mathalphabet_char:Nnn \@@_mathmap_parse:Nnn
1081   \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
1082   \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
1083   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
1084   \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
1085   \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_parse:nNN

```

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

```

1086   \seq_clear:N \l_@@_char_range_seq
1087   \seq_clear:N \l_@@_mclass_range_seq
1088   \seq_clear:N \l_@@_cmd_range_seq
1089   \seq_clear:N \l_@@_mathalph_seq
1090
1091   \clist_map_inline:nn {#1}
1092   {
1093     \@@_if_mathalph_decl:nTF {##1}
1094     {
1095       \seq_put_right:Nx \l_@@_mathalph_seq
1096       {
1097         { \exp_not:V \l_@@_tmpa_tl }
1098         { \exp_not:V \l_@@_tmpb_tl }
1099         { \exp_not:V \l_@@_tmpc_tl }
1100       }
1101     }
1102   }

```

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

```

1103     \seq_if_in:NnTF \g_@@_mathclasses_seq {##1}
1104     { \seq_put_right:Nn \l_@@_mclass_range_seq {##1} }
1105     {
1106       \bool_if:nTF { \tl_if_single_p:n {##1} && \token_if_cs_p:N ##1 }
1107       { \seq_put_right:Nn \l_@@_cmd_range_seq {##1} }
1108       { \seq_put_right:Nn \l_@@_char_range_seq {##1} }
1109     }
1110   }
1111 }
1112 }
1113 }

```

`\@@_if_mathalph_decl:nTF` Possible forms of input:
`\mathscr`

$\backslash\mathscr\rightarrow\backslash\mathup$
 $\backslash\mathscr\{\text{Latin}\}$
 $\backslash\mathscr\{\text{Latin}\}\rightarrow\backslash\mathup$

Outputs:

tmpa: math style (e.g., $\backslash\mathscr$)

tmpb: alphabets (e.g., Latin)

tmpc: remap style (e.g., $\backslash\mathup$). Defaults to tmpa.

The remap style can also be $\backslash\mathcal\rightarrow\text{stixcal}$, which I marginally prefer in the general case.

```

1114 \prg_new_conditional:Nnn \@@_if_mathalph_decl:n {TF}
1115 {
1116   \tl_set:Nn \l_@@_tmpa_tl {#1}
1117   \tl_clear:N \l_@@_tmpb_tl
1118   \tl_clear:N \l_@@_tmpc_tl
1119
1120   \tl_if_in:NnT \l_@@_tmpa_tl {->}
1121   { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }
1122
1123   \tl_if_in:NnT \l_@@_tmpa_tl {/}
1124   { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }
1125
1126   \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
1127   \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
1128   \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
1129   \tl_trim_spaces:N \l_@@_tmpa_tl
1130
1131   \tl_if_empty:NT \l_@@_tmpc_tl
1132   { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }
1133
1134   \seq_if_in:NVTF \g_@@_named_ranges_seq \l_@@_tmpa_tl
1135   { \prg_return_true: } { \prg_return_false: }
1136 }
1137 \cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
1138 {
1139   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1140   \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
1141 }
1142 \cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
1143 {
1144   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1145   \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
1146 }

```

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

$\backslash\@@_if_char_spec:nNNT$ #1 : Unicode character slot
 #2 : control sequence (character macro)

#3 : control sequence (math class)

#4 : code to execute

This macro expands to #4 if any of its arguments are contained in \l_@@_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 \@@_if_char_spec:nNNT, which accepts input in the form shown in table 13.

Table 13: Ranges accepted by \@@_if_char_spec:nNNT.

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

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

```

1147 \cs_new:Nn \@@_if_char_spec:nNNT
1148 {
1149   % math class:
1150   \seq_if_in:NnT \l_@@_mclass_range_seq {#3}
1151   { \use_none_delimit_by_q_nil:w }
1152
1153   % command name:
1154   \seq_if_in:NnT \l_@@_cmd_range_seq {#2}
1155   { \use_none_delimit_by_q_nil:w }
1156
1157   % character slot:
1158   \seq_map_inline:Nn \l_@@_char_range_seq
1159   {
1160     \@@_int_if_slot_in_range:nnT {#1} {##1}
1161     { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
1162   }
1163
1164   % the following expands to nil if no match was found:
1165   \use_none:nnn
1166   \q_nil
1167   \use:n
1168   {
1169     \clist_put_right:Nx \l_@@_char_nrange_clist { \int_eval:n {#1} }
1170     #4
1171   }
1172 }
```

\@@_int_if_slot_in_range:nnT A ‘numrange’ is like -2,5-8,12,17- (can be unsorted).

Four cases, four argument types:

```

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

1173 \cs_new:Nn \@@_int_if_slot_in_range:nnT
1174 { \@@_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
1175 \cs_set:Npn \@@_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
1176 {
1177   \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
1178   {
1179     \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
1180     {
1181       \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
1182       {
1183         \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
1184       } } }
1185 }

```

H.4 Resolving Greek symbol name control sequences

`\@@_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.

```

1186 \AtBeginDocument{\@@_resolve_greek:}
1187 \cs_new:Npn \@@_resolve_greek:
1188 {
1189   \clist_map_inline:nn
1190   {
1191     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1192     alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
1193     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1194     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,  chi,psi,omega,
1195     varTheta,
1196     varsigma,vartheta,varkappa,varrho,varpi
1197   }
1198   {
1199     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
1200     \tl_set:cx {up ##1} { \exp_not:N \symup \exp_not:c { ##1 } }
1201     \tl_set:cx {it ##1} { \exp_not:N \symit \exp_not:c { ##1 } }
1202   }
1203   \tl_set:Nn \epsilon
1204   { \bool_if:NTF \g_@@_texgreek_bool \mitvarepsilon \mitepsilon }
1205   \tl_set:Nn \phi
1206   { \bool_if:NTF \g_@@_texgreek_bool \mitvarphi \mitphi }
1207   \tl_set:Nn \varepsilon
1208   { \bool_if:NTF \g_@@_texgreek_bool \mitepsilon \mitvarepsilon }

```



```

1209 \tl_set:Nn \varphi
1210 { \bool_if:NTF \g_@@_texgreek_bool \mitphi \mitvarphi }
1211 }

```

I Maths alphabets

Defining commands like `\mathrm` is not as simple with Unicode fonts. In traditional \TeX maths font setups, you simply switch between different ‘families’ (`\fam`), which is analogous to changing from one font to another—a symbol such as ‘a’ will be upright in one font, bold in another, and so on.

In `pkgunicode-math`, a different mechanism is used to switch between styles. For every letter (start with `ascii a-zA-Z` and numbers to keep things simple for now), they are assigned a ‘mathcode’ with `\Umathcode` that maps from input letter to output font glyph slot. This is done with the equivalent of

```

% \Umathcode'\a = 7 1 "1D44E\relax
% \Umathcode'\b = 7 1 "1D44F\relax
% \Umathcode'\c = 7 1 "1D450\relax
% ...

```

When switching from regular letters to, say, `\mathrm`, we now need to execute a new mapping:

```

% \Umathcode'\a = 7 1 '\a\relax
% \Umathcode'\b = 7 1 '\b\relax
% \Umathcode'\c = 7 1 '\c\relax
% ...

```

This is fairly straightforward to perform when we’re defining our own commands such as `\sympf` and so on. However, this means that ‘classical’ \TeX font setups will break, because with the original mapping still in place, the engine will be attempting to insert unicode maths glyphs from a standard font.

I.1 Hooks into $\text{\LaTeX} 2_{\epsilon}$

To overcome this, we patch `\use@mathgroup`. (An alternative is to patch `\extract@alph@from@version`, which constructs the `\mathXYZ` commands, but this method fails if the command has been defined using `\DeclareSymbolFontAlpha`.) As far as I can tell, this is only used inside of commands such as `\mathXYZ`, so this shouldn’t have any major side-effects.

```

1212 \cs_set:Npn \use@mathgroup #1 #2
1213 {
1214   \mode_if_math:T % <- not sure if this is really necessary since we’ve just checked for mmode and raised
    ror if not!
1215   {
1216     \math@bgroup
1217     \cs_if_eq:cNF {M@\f@encoding} #1 {#1}
1218     \@@_switchto_literal:

```

```

1219     \mathgroup #2 \relax
1220     \math@egroup
1221   }
1222 }

```

I.2 *Setting styles*

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 glyph slots instead.

I.3 *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.

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

The flag `\l_@@_mathstyle_tl` is for other applications to query the current math style.

```

1223 \cs_new:Nn \@@_prepare_mathstyle:n
1224 {
1225   \seq_put_right:Nn \g_@@_mathstyles_seq {#1}
1226   \@@_init_alphabet:n {#1}
1227   \cs_set:cpn {_@@_sym_#1_aux:n}

```

```

1228 { \use:c {@@_switchto_#1:} \math@egroup }
1229 \cs_set_protected:cpx {sym#1}
1230 {
1231   \exp_not:n
1232   {
1233     \math@bgroup
1234     \mode_if_math:F
1235     {
1236       \egroup\expandafter
1237       \non@alpherr\expandafter{\csname sym#1\endcsname\space}
1238     }
1239     \tl_set:Nn \l_@@_mathstyle_tl {#1}
1240   }
1241   \exp_not:c {_@@_sym_#1_aux:n}
1242 }
1243 }

```

`\@@_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 when the math alphabet macro is first defined, but then used later when redefining a particular maths alphabet.

```

1244 \cs_set:Nn \@@_init_alphabet:n
1245 {
1246   \@@_log:nx {alph-initialise} {#1}
1247   \cs_set_eq:cN {@@_switchto_#1:} \prg_do_nothing:
1248 }

```

I.4 Definition of alphabets and styles

First of all, we break up unicode into ‘named ranges’, such as up, bb, sfup, and so on, which refer to specific blocks of unicode that contain various symbols (usually alphabetical symbols).

```

1249 \cs_new:Nn \@@_new_named_range:n
1250 {
1251   \prop_new:c {g_@@_named_range_#1_prop}
1252 }
1253 \clist_set:Nn \g_@@_named_ranges_clist
1254 {
1255   up, it, tt, bfup, bfit, bb , bbit, scr, bfscr, cal, bfcalf,
1256   frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
1257 }
1258 \clist_map_inline:Nn \g_@@_named_ranges_clist
1259 { \@@_new_named_range:n {#1} }

```

Each of these styles usually contains one or more ‘alphabets’, which are currently latin, Latin, greek, Greek, num, and misc, although there’s an implicit potential for more. misc is not included in the official list to avoid checking code.

```

1260 \clist_new:N \g_@@_alphabets_seq
1261 \clist_set:Nn \g_@@_alphabets_seq { latin, Latin, greek, Greek, num }

```

Each alphabet style needs to be configured. This happens in the `unicode-math-alphabets.dtx` file.

```

1262 \cs_new:Nn \@@_new_alphabet_config:nnn
1263 {
1264   \prop_if_exist:cF {g_@@_named_range_#1_prop}
1265   { \@@_warning:nnn {no-named-range} {#1} {#2} }
1266
1267   \prop_gput:cnn {g_@@_named_range_#1_prop} { alpha_tl }
1268   {
1269     \prop_item:cn {g_@@_named_range_#1_prop} { alpha_tl }
1270     {#2}
1271   }
1272   % Q: do I need to bother removing duplicates?
1273
1274   \cs_new:cn { @@_config_#1_#2:n } {#3}
1275 }
1276
1277 \cs_new:Nn \@@_alphabet_config:nnn
1278 {
1279   \use:c {@@_config_#1_#2:n} {#3}
1280 }
1281
1282 \prg_new_conditional:Nnn \@@_if_alphabet_exists:nn {T,TF}
1283 {
1284   \cs_if_exist:cTF {@@_config_#1_#2:n}
1285   \prg_return_true: \prg_return_false:
1286 }

```

The linking between named ranges and symbol style commands happens here. It's currently not using all of the machinery we're in the process of setting up above. Baby steps.

```

1285 \cs_new:Nn \@@_default_mathalph:nnn
1286 {
1287   \seq_put_right:Nx \g_@@_named_ranges_seq { \tl_to_str:n {#1} }
1288   \seq_put_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
1289   \prop_gput:cnn { g_@@_named_range_#1_prop } { default-alpha } {#2}
1290 }
1291
1292 \@@_default_mathalph:nnn {up} {latin,Latin,greek,Greek,num,misc} {up}
1293 \@@_default_mathalph:nnn {it} {latin,Latin,greek,Greek,misc} {it}
1294 \@@_default_mathalph:nnn {bb} {latin,Latin,num,misc} {bb}
1295 \@@_default_mathalph:nnn {bbit} {misc} {bbit}
1296 \@@_default_mathalph:nnn {scr} {latin,Latin} {scr}
1297 \@@_default_mathalph:nnn {cal} {Latin} {cal}
1298 \@@_default_mathalph:nnn {bfcal} {Latin} {bfcal}
1299 \@@_default_mathalph:nnn {frak} {latin,Latin} {frak}
1300 \@@_default_mathalph:nnn {tt} {latin,Latin,num} {tt}
1301 \@@_default_mathalph:nnn {sfup} {latin,Latin,num} {sfup}
1302 \@@_default_mathalph:nnn {sfit} {latin,Latin} {sfit}
1303 \@@_default_mathalph:nnn {bfup} {latin,Latin,greek,Greek,num,misc} {bfup}
1304 \@@_default_mathalph:nnn {bfit} {latin,Latin,greek,Greek,misc} {bfit}

```

```

1304 \@@_default_mathalph:nnn {bfscr } {latin, Latin} {bfscr }
1305 \@@_default_mathalph:nnn {bffrak} {latin, Latin} {bffrak}
1306 \@@_default_mathalph:nnn {bfsfup} {latin, Latin, greek, Greek, num, misc} {bfsfup}
1307 \@@_default_mathalph:nnn {bfsfit} {latin, Latin, greek, Greek, misc} {bfsfit}

```

I.4.1 Define symbol style commands

Finally, all of the ‘symbol styles’ commands are set up, which are the commands to access each of the named alphabet styles. There is not a one-to-one mapping between symbol style commands and named style ranges!

```

1308 \clist_map_inline:nn
1309 {
1310   up, it, bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf,
1311   tt, bb, bbit, scr, bfscr, cal, bfcalf, frak, bffrak,
1312   normal, literal, sf, bf,
1313 }
1314 { \@@_prepare_mathstyle:n {#1} }

```

I.4.2 New names for legacy textmath alphabet selection

In case a package option overwrites, say, `\mathbf` with `\sympb`.

```

1315 \clist_map_inline:nn
1316 { rm, it, bf, sf, tt }
1317 { \cs_set_eq:cc { mathtext #1 } { math #1 } }

```

Perhaps these should actually be defined using a hypothetical unicode-math interface to creating new such styles. To come.

I.4.3 Replacing legacy pure-maths alphabets

The following are alphabets which do not have a math/text ambiguity.

```

1318 \clist_map_inline:nn
1319 {
1320   normal, bb , bbit, scr, bfscr, cal, bfcalf, frak, bffrak, tt,
1321   bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
1322 }
1323 {
1324   \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
1325 }

```

I.4.4 New commands for ambiguous alphabets

```

1326 \AtBeginDocument{
1327   \clist_map_inline:nn
1328   { rm, it, bf, sf, tt }
1329   {
1330     \cs_set_protected:cpx { math #1 }
1331     {
1332       \exp_not:n { \bool_if:NTF } \exp_not:c { g_@@_ math #1 _text_bool }
1333       { \exp_not:c { mathtext #1 } }

```

```

1334     { \exp_not:c { sym #1 } }
1335   }
1336 }}

```

Alias `\mathrm` as legacy name for `\mathup`

```

1337 \cs_set_protected:Npn \mathup { \mathrm }
1338 \cs_set_protected:Npn \symrm { \symup }

```

1.4.5 Fixing up `\operator@font`

In LaTeX maths, the command `\operator@font` is defined that switches to the operator mathgroup. The classic example is the `\sin` in `sin{x}`; essentially we’re using `\mathrm` to typeset the upright symbols, but the syntax is `{\operator@font sin}`.

It turns out that hooking into `\operator@font` is hard because all other maths font selection in 2e uses `\mathrm{...}` style.

Then reading source2e a little more I stumbled upon: (in the definition of `\select@group`)

```

We surround \select@group with braces so that functions using it can
be used directly after _ or ^. However, if we use oldstyle syntax where
the math alphabet doesn’t have arguments (ie if \math@bgroup is not
\bgroup) we need to get rid of the extra group.

```

So there’s a trick we can use. Because it’s late and I’m tired, I went for the first thing that jumped out at me:

```

% \documentclass{article}
% \DeclareMathAlphabet\mathfoo{OT1}{lmdh}{m}{n}
% \begin{document}
% \makeatletter
% ${\operator@font Mod}\, x$
%
% \def\operator@font{%
%   \let \math@bgroup \relax
%   \def \math@egroup {\let \math@bgroup \@math@bgroup
%                       \let \math@egroup \@math@egroup}%
%   \mathfoo}
% ${\operator@font Mod}\, x$
% \end{document}

```

We define a new math alphabet `\mathfoo` to select the Latin Modern Dunhill font, and then locally redefine `\math@bgroup` to allow `\mathfoo` to be used without an argument temporarily.

Now that I’ve written this whole thing out, another solution pops to mind:

```

% \documentclass{article}
% \DeclareSymbolFont{foo}{OT1}{lmdh}{m}{n}
% \DeclareSymbolFontAlphabet\mathfoo{foo}
% \begin{document}
% \makeatletter

```

```

%    ${\operator@font Mod}\, x$
%
%    \def\operator@font{\mathgroup\symfoo}
%    ${\operator@font Mod}\, x$
%    \end{document}

```

I guess that's the better approach!!

Or perhaps I should just use `\fontswitch` to do the first solution with a nicer wrapper. I really should read things more carefully:

`\operator@font`

```

1339 \cs_set:Npn \operator@font
1340 {
1341   \@@_switchto_literal:
1342   \fontswitch {} { \g_@@_operator_mathfont_tl }
1343 }

```

1.5 Defining the math alphabets per style

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

```

1344 \cs_new:Npn \@@_setup_alphabets:
1345 {

```

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

```

1346   \seq_if_empty:NTF \l_@@_mathalph_seq
1347   {
1348     \@@_log:n {setup-implicit}
1349     \seq_set_eq:NN \l_@@_mathalph_seq \g_@@_default_mathalph_seq
1350     \bool_set_true:N \l_@@_implicit_alph_bool
1351     \@@_maybe_init_alphabet:n {sf}
1352     \@@_maybe_init_alphabet:n {bf}
1353     \@@_maybe_init_alphabet:n {bfsf}
1354   }

```

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

```

1355   {
1356     \@@_log:n {setup-explicit}
1357     \bool_set_false:N \l_@@_implicit_alph_bool
1358     \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
1359     \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
1360   }
1361
1362   % Now perform the mapping:
1363   \seq_map_inline:Nn \l_@@_mathalph_seq
1364   {
1365     \tl_set:No \l_@@_style_tl { \use_i:nnn ##1 }
1366     \clist_set:No \l_@@_alphabet_clist { \use_ii:nnn ##1 }
1367     \tl_set:No \l_@@_remap_style_tl { \use_iii:nnn ##1 }

```

```

1368
1369 % If no set of alphabets is defined:
1370 \clist_if_empty:NT \l_@@_alphabet_clist
1371 {
1372   \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
1373   \prop_get:cnN { g_@@_named_range_ \l_@@_style_tl _prop }
1374   { default-alpha } \l_@@_alphabet_clist
1375 }
1376
1377 \@@_setup_math_alphabet:
1378 }
1379 \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
1380 }

```

\@@_setup_math_alphabet:

```

1381 \cs_new:Nn \@@_setup_math_alphabet:
1382 {
First check that at least one of the alphabets for the font shape is defined (this
process is fast) ...
1383 \clist_map_inline:Nn \l_@@_alphabet_clist
1384 {
1385   \tl_set:Nn \l_@@_alphabet_tl {##1}
1386   \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
1387   {
1388     \str_if_eq_x:nnTF {\l_@@_alphabet_tl} {misc}
1389     {
1390       \@@_maybe_init_alphabet:n \l_@@_style_tl
1391       \clist_map_break:
1392     }
1393     {
1394       \@@_glyph_if_exist:nT { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
1395       {
1396         \@@_maybe_init_alphabet:n \l_@@_style_tl
1397         \clist_map_break:
1398       }
1399     }
1400   }
1401   { \msg_warning:nnx {unicode-math} {no-alphabet} { \l_@@_style_tl / \l_@@_alphabet_tl } }
1402 }

```

...and then loop through them defining the individual ranges: (currently this process is slow)

```

1403 (debug) \csname TIC\endcsname
1404 \clist_map_inline:Nn \l_@@_alphabet_clist
1405 {
1406   \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
1407   \cs_if_exist:cT {@@_config_ \l_@@_style_tl _ \l_@@_alphabet_tl :n}
1408   {
1409     \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}

```



```

1410     {
1411       \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1412       \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
1413     }
1414     {
1415       \@@_glyph_if_exist:NTF { \@@_to_usv:nn {\l_@@_remap_style_tl} {\l_@@_alphabet_tl} }
1416       {
1417         \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1418         \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
1419       }
1420       {
1421         \bool_if:NTF \l_@@_implicit_alph_bool
1422         {
1423           \seq_put_right:Nx \l_@@_missing_alph_seq
1424           {
1425             \@backslashchar sym \l_@@_style_tl \space
1426             (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
1427           }
1428         }
1429         {
1430           \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {up}
1431         }
1432       }
1433     }
1434   }
1435 }
1436 (debug) \csname TOC\endcsname
1437 }

```

I.6 Mapping ‘naked’ math characters

Before we show the definitions of the alphabet mappings using the functions `\@@_alphabet_config:nnn \l_@@_style_tl {##1} {...}`, we first want to define some functions to be used inside them to actually perform the character mapping.

I.6.1 Functions

`\@@_map_char_single:nn` Wrapper for `\@@_map_char_noparse:nn` or `\@@_map_char_parse:nn` depending on the context.

```

\@@_map_char_noparse:nn
\@@_map_char_parse:nn
1438 \cs_new:Nn \@@_map_char_noparse:nn
1439 { \@@_set_mathcode:nnnn {#1}{\mathalpha}{\@@_symfont_tl}{#2} }
1440 \cs_new:Nn \@@_map_char_parse:nn
1441 {
1442   \@@_if_char_spec:nNNT {#1} {\@nil} {\mathalpha}
1443   { \@@_map_char_noparse:nn {#1}{#2} }
1444 }

```

```

\@@_map_char_single:nnn #1 : char name ('dotlessi')
                        #2 : from alphabet(s)
                        #3 : to alphabet
                        Logical interface to \@@_map_char_single:nn.

1445 \cs_new:Nn \@@_map_char_single:nnn
1446 {
1447   \@@_map_char_single:nn { \@@_to_usv:nn {#1}{#3} }
1448                           { \@@_to_usv:nn {#2}{#3} }
1449 }

```

```

\@@_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:

1450 \cs_set:Nn \@@_map_chars_range:nnn
1451 {
1452   \int_step_inline:nnnn {0}{1}{#1-1}
1453   { \@@_map_char_single:nn {#2+##1}{#3+##1} }
1454 }

```

And the wrapper with names:

```

1455 \cs_new:Nn \@@_map_chars_range:nnnn
1456 {
1457   \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2}{#4} }
1458                           { \@@_to_usv:nn {#3}{#4} }
1459 }

```

1.6.2 Functions for 'normal' alphabet symbols

```

\@@_set_normal_char:nnn

1460 \cs_set:Nn \@@_set_normal_char:nnn
1461 {
1462   \@@_usv_if_exist:nnT {#3} {#1}
1463   {
1464     \clist_map_inline:nn {#2}
1465     {
1466       \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
1467       \@@_map_char_single:nnn {##1} {#3} {#1}
1468     }
1469   }
1470 }

1471 \cs_new:Nn \@@_set_normal_Latin:nn
1472 {
1473   \clist_map_inline:nn {#1}
1474   {
1475     \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
1476     \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}

```

```

1477     }
1478 }

1479 \cs_new:Nn \@@_set_normal_latin:nn
1480 {
1481   \clist_map_inline:nn {#1}
1482   {
1483     \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
1484     \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
1485   }
1486 }

1487 \cs_new:Nn \@@_set_normal_greek:nn
1488 {
1489   \clist_map_inline:nn {#1}
1490   {
1491     \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
1492     \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
1493     \@@_map_char_single:nnn {##1} {#2} {\varepsilon}
1494     \@@_map_char_single:nnn {##1} {#2} {\vartheta}
1495     \@@_map_char_single:nnn {##1} {#2} {\varkappa}
1496     \@@_map_char_single:nnn {##1} {#2} {\varphi}
1497     \@@_map_char_single:nnn {##1} {#2} {\varrho}
1498     \@@_map_char_single:nnn {##1} {#2} {\varpi}
1499     \@@_set_mathalphabet_pos:nnnn {normal} {\varepsilon} {##1} {#2}
1500     \@@_set_mathalphabet_pos:nnnn {normal} {\vartheta} {##1} {#2}
1501     \@@_set_mathalphabet_pos:nnnn {normal} {\varkappa} {##1} {#2}
1502     \@@_set_mathalphabet_pos:nnnn {normal} {\varphi} {##1} {#2}
1503     \@@_set_mathalphabet_pos:nnnn {normal} {\varrho} {##1} {#2}
1504     \@@_set_mathalphabet_pos:nnnn {normal} {\varpi} {##1} {#2}
1505   }
1506 }

1507 \cs_new:Nn \@@_set_normal_Greek:nn
1508 {
1509   \clist_map_inline:nn {#1}
1510   {
1511     \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
1512     \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
1513     \@@_map_char_single:nnn {##1} {#2} {\varTheta}
1514     \@@_set_mathalphabet_pos:nnnn {normal} {\varTheta} {##1} {#2}
1515   }
1516 }

1517 \cs_new:Nn \@@_set_normal_numbers:nn
1518 {
1519   \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
1520   \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
1521 }

```

I.7 Mapping chars inside a math style

I.7.1 Functions for setting up the maths alphabets

`\@@_set_mathalphabet_char:Nnn` This is a wrapper for either `\@@_mathmap_noparse:nnn` or `\@@_mathmap_parse:Nnn`, depending on the context.

`\@@_mathmap_noparse:nnn` #1 : Maths alphabet, e.g., ‘bb’
 #2 : Input slot(s), e.g., the slot for ‘A’ (comma separated)
 #3 : Output slot, e.g., the slot for ‘A’
 Adds `\@@_set_mathcode:nnnn` declarations to the specified maths alphabet’s definition.

```

1522 \cs_new:Nn \@@_mathmap_noparse:nnn
1523 {
1524   \clist_map_inline:nn {#2}
1525   {
1526     \tl_put_right:cx {@_switchto_#1:}
1527     {
1528       \@@_set_mathcode:nnnn {##1} {\mathalpha} {\@@_symfont_tl} {#3}
1529     }
1530   }
1531 }
```

`\@@_mathmap_parse:nnn` #1 : Maths alphabet, e.g., ‘bb’
 #2 : Input slot(s), e.g., the slot for ‘A’ (comma separated)
 #3 : Output slot, e.g., the slot for ‘A’
 When `\@@_if_char_spec:nNT` is executed, it populates the `\l_@@_char_nrange_clist` macro with slot numbers corresponding to the specified range. This range is used to conditionally add `\@@_set_mathcode:nnnn` declarations to the maths alphabet definition.

```

1532 \cs_new:Nn \@@_mathmap_parse:nnn
1533 {
1534   \clist_if_in:NnT \l_@@_char_nrange_clist {#3}
1535   {
1536     \@@_mathmap_noparse:nnn {#1}{#2}{#3}
1537   }
1538 }
```

`\@@_set_mathalphabet_char:nnnn` #1 : math style command
 #2 : input math alphabet name
 #3 : output math alphabet name
 #4 : char name to map

```

1539 \cs_new:Nn \@@_set_mathalphabet_char:nnnn
1540 {
1541   \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
1542   { \@@_to_usv:nn {#3} {#4} }
1543 }
```

\@@_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:

```
1544 \cs_new:Nn \@@_set_mathalph_range:nnnn
1545 {
1546   \int_step_inline:nnnn {0} {1} {#1-1}
1547   { \@@_set_mathalphabet_char:nnn {#2} { ##1 + #3 } { ##1 + #4 } }
1548 }
```

Then the wrapper version that uses names:

```
1549 \cs_new:Nn \@@_set_mathalph_range:nnnnn
1550 {
1551   \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
1552   { \@@_to_usv:nn {#4} {#5} }
1553 }
```

1.7.2 Individual mapping functions for different alphabets

```
1554 \cs_new:Nn \@@_set_mathalphabet_pos:nnnn
1555 {
1556   \@@_usv_if_exist:nnT {#4} {#2}
1557   {
1558     \clist_map_inline:nn {#3}
1559     { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }
1560   }
1561 }

1562 \cs_new:Nn \@@_set_mathalphabet_numbers:nnn
1563 {
1564   \clist_map_inline:nn {#2}
1565   { \@@_set_mathalph_range:nnnnn {10} {#1} {##1} {#3} {num} }
1566 }

1567 \cs_new:Nn \@@_set_mathalphabet_Latin:nnn
1568 {
1569   \clist_map_inline:nn {#2}
1570   { \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {Latin} }
1571 }

1572 \cs_new:Nn \@@_set_mathalphabet_latin:nnn
1573 {
1574   \clist_map_inline:nn {#2}
1575   {
1576     \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {latin}
1577     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {h}
1578   }
1579 }

1580 \cs_new:Nn \@@_set_mathalphabet_Greek:nnn
```

```

1581 {
1582 \clist_map_inline:nn {#2}
1583 {
1584 \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {Greek}
1585 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varTheta}
1586 }
1587 }

1588 \cs_new:Nn \@@_set_mathalphabet_greek:nnn
1589 {
1590 \clist_map_inline:nn {#2}
1591 {
1592 \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {greek}
1593 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varepsilon}
1594 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {vartheta}
1595 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varkappa}
1596 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varphi}
1597 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varrho}
1598 \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varpi}
1599 }
1600 }

```

J 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.

```

1601 \cs_new:Npn \@@_symbol_setup:
1602 {
1603 \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
1604 {
1605 \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
1606 }
1607 }

1608 \CatchFileEdef \g_@@_mathtable_tl {unicode-math-table.tex} {\@@_symbol_setup:}

```

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

```

1609 \cs_new:Nn \@@_input_math_symbol_table: {\g_@@_mathtable_tl}

```

K Definitions of the active math characters

`\@@_cs_set_eq_active_char:Nw` This macro is used to give each Unicode math codepoint an equivalent macro name. The two are equivalent, in a `\let\xyz=^^^1234` kind of way. (For now!!)

```

1610 \cs_set:Npn \@@_cs_set_eq_active_char:Nw #1 = "#2 \q_nil

```

```

1611 {
1612   \exp_last_unbraced:NNx \cs_gset_eq:NN #1 { \Ucharcat "#2 ~ 12 ~ }
1613 }

```

`\@@_active_char_set:wc` We need to do some trickery to transform the `_@@_sym:nnn` argument "ABCDEF 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 `^` into a regular ‘other’ character and define the macro to perform the lowercasing and `\let`. `\scantokens` changes the carets back into their original meaning after the group has ended and `^`’s catcode returns to normal.

```

1614 \group_begin:
1615   \char_set_catcode_other:N \^
1616   \cs_new:Npn \@@_active_char_set:wc "#1 \q_nil #2
1617   {
1618     \tex_lowercase:D
1619     {
1620       \tl_rescan:nn { \ExplSyntaxOn }
1621       { \cs_gset_protected_nopar:Npx ^^^^^#1 { \exp_not:c {#2} } }
1622     }
1623   }
1624 \group_end:

```

Could be defined simply as the following if `Ucharcar` allowed active chars!

```
% \exp_last_unbraced:Nx \cs_gset_protected_nopar:Npx { \Ucharcat "#2 ~ 13 ~ } { \exp_not:c {#2} }
```

Now give `_@@_sym:nnn` a definition in terms of `\@@_cs_set_eq_active_char:Nw` and we’re good to go.

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

```

1625 \AtBeginDocument{\@@_define_math_chars:}
1626 \cs_new:Nn \@@_define_math_chars:
1627 {
1628   \group_begin:
1629     \char_set_catcode_math_superscript:N \^
1630     \cs_set:Npn \_@@_sym:nnn ##1##2##3
1631     {
1632       \tl_if_in:nnT
1633       { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
1634       {##3}
1635       {
1636         \@@_cs_set_eq_active_char:Nw ##2 = ##1 \q_nil \ignorespaces
1637       }
1638     }
1639     \char_set_catcode_other:N \#
1640     \@@_input_math_symbol_table:
1641   \group_end:
1642 }

```

L *Fall-back font*

Want to load Latin Modern Math if nothing else. Reset the ‘font already loaded’ boolean so that a new font being set will do the right thing. TODO: need a better way to do this for the general case.

```
1643 \AtBeginDocument { \@@_load_lm_if_necessary: }
1644 \cs_new:Nn \@@_load_lm_if_necessary:
1645 {
1646   \cs_if_exist:NF \l_@@_fontname_tl
1647   {
1648     % TODO: update this when lmmath-bold.otf is released
1649     \setmathfont{latinmodern-math.otf}[BoldFont={latinmodern-math.otf}]
1650     \bool_set_false:N \g_@@_mainfont_already_set_bool
1651   }
1652 }
```

M *Epilogue*

Lots of little things to tidy up.

M.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.

```

1653 \cs_new:Nn \@@_arg_i_before_egroup:n {#1\egroup}
1654 \cs_new:Nn \@@_superscript:n
1655 {
1656   ^\bgroup #1
1657   \peek_meaning_remove:NTF ^ \@@_arg_i_before_egroup:n \egroup
1658 }

1659 \cs_new:Nn \@@_nprimes:Nn
1660 {
1661   \@@_superscript:n
1662   {
1663     #1
1664     \prg_replicate:nn {#2-1} { \mskip \g_@@_primekern_muskip #1 }
1665   }
1666 }

1667
1668 \cs_new:Nn \@@_nprimes_select:nn
1669 {
1670   \int_case:nnF {#2}
1671   {
1672     {1} { \@@_superscript:n {#1} }
1673     {2} {
1674       \@@_glyph_if_exist:nTF {"2033}
1675       { \@@_superscript:n {\@@_prime_double_mchar} }
1676       { \@@_nprimes:Nn #1 {#2} }
1677     }
1678     {3} {
1679       \@@_glyph_if_exist:nTF {"2034}
1680       { \@@_superscript:n {\@@_prime_triple_mchar} }
1681       { \@@_nprimes:Nn #1 {#2} }
1682     }
1683     {4} {
1684       \@@_glyph_if_exist:nTF {"2057}
1685       { \@@_superscript:n {\@@_prime_quad_mchar} }

```

```

1686         { \@@_nprimes:Nn #1 {#2} }
1687     }
1688 }
1689 {
1690     \@@_nprimes:Nn #1 {#2}
1691 }
1692 }
1693 \cs_new:Nn \@@_nbackprimes_select:nn
1694 {
1695     \int_case:nnF {#2}
1696     {
1697         {1} { \@@_superscript:n {#1} }
1698         {2} {
1699             \@@_glyph_if_exist:nTF {"2036}
1700             { \@@_superscript:n {\@@_backprime_double_mchar} }
1701             { \@@_nprimes:Nn #1 {#2} }
1702         }
1703         {3} {
1704             \@@_glyph_if_exist:nTF {"2037}
1705             { \@@_superscript:n {\@@_backprime_triple_mchar} }
1706             { \@@_nprimes:Nn #1 {#2} }
1707         }
1708     }
1709     {
1710         \@@_nprimes:Nn #1 {#2}
1711     }
1712 }

```

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

```

1713 \cs_new:Npn \@@_scan_prime:
1714 {
1715     \cs_set_eq:NN \@@_superscript:n \use:n
1716     \int_zero:N \l_@@_primecount_int
1717     \@@_scanprime_collect:N \@@_prime_single_mchar
1718 }
1719 \cs_new:Npn \@@_scan_dprime:
1720 {
1721     \cs_set_eq:NN \@@_superscript:n \use:n
1722     \int_set:Nn \l_@@_primecount_int {1}
1723     \@@_scanprime_collect:N \@@_prime_single_mchar
1724 }
1725 \cs_new:Npn \@@_scan_trprime:
1726 {
1727     \cs_set_eq:NN \@@_superscript:n \use:n
1728     \int_set:Nn \l_@@_primecount_int {2}
1729     \@@_scanprime_collect:N \@@_prime_single_mchar
1730 }
1731 \cs_new:Npn \@@_scan_qprime:
1732 {
1733     \cs_set_eq:NN \@@_superscript:n \use:n

```

```

1734 \int_set:Nn \l_@@_primecount_int {3}
1735 \@@_scanprime_collect:N \@@_prime_single_mchar
1736 }
1737 \cs_new:Npn \@@_scan_sup_prime:
1738 {
1739   \int_zero:N \l_@@_primecount_int
1740   \@@_scanprime_collect:N \@@_prime_single_mchar
1741 }
1742 \cs_new:Npn \@@_scan_sup_dprime:
1743 {
1744   \int_set:Nn \l_@@_primecount_int {1}
1745   \@@_scanprime_collect:N \@@_prime_single_mchar
1746 }
1747 \cs_new:Npn \@@_scan_sup_trprime:
1748 {
1749   \int_set:Nn \l_@@_primecount_int {2}
1750   \@@_scanprime_collect:N \@@_prime_single_mchar
1751 }
1752 \cs_new:Npn \@@_scan_sup_qprime:
1753 {
1754   \int_set:Nn \l_@@_primecount_int {3}
1755   \@@_scanprime_collect:N \@@_prime_single_mchar
1756 }
1757 \cs_new:Nn \@@_scanprime_collect:N
1758 {
1759   \int_incr:N \l_@@_primecount_int
1760   \peek_meaning_remove:NTF '
1761   { \@@_scanprime_collect:N #1 }
1762   {
1763     \peek_meaning_remove:NTF \@@_scan_prime:
1764     { \@@_scanprime_collect:N #1 }
1765     {
1766       \peek_meaning_remove:NTF ^^^^2032
1767       { \@@_scanprime_collect:N #1 }
1768       {
1769         \peek_meaning_remove:NTF \@@_scan_dprime:
1770         {
1771           \int_incr:N \l_@@_primecount_int
1772           \@@_scanprime_collect:N #1
1773         }
1774         {
1775           \peek_meaning_remove:NTF ^^^^2033
1776           {
1777             \int_incr:N \l_@@_primecount_int
1778             \@@_scanprime_collect:N #1
1779           }
1780           {
1781             \peek_meaning_remove:NTF \@@_scan_trprime:
1782             {

```

```

1783         \int_add:Nn \l_@@_primecount_int {2}
1784         \@@_scanprime_collect:N #1
1785     }
1786     {
1787         \peek_meaning_remove:NTF ^^^^2034
1788         {
1789             \int_add:Nn \l_@@_primecount_int {2}
1790             \@@_scanprime_collect:N #1
1791         }
1792         {
1793             \peek_meaning_remove:NTF \@@_scan_qprime:
1794             {
1795                 \int_add:Nn \l_@@_primecount_int {3}
1796                 \@@_scanprime_collect:N #1
1797             }
1798             {
1799                 \peek_meaning_remove:NTF ^^^^2057
1800                 {
1801                     \int_add:Nn \l_@@_primecount_int {3}
1802                     \@@_scanprime_collect:N #1
1803                 }
1804                 {
1805                     \@@_nprimes_select:nn {#1} {\l_@@_primecount_int}
1806                 }
1807             }
1808         }
1809     }
1810 }
1811 }
1812 }
1813 }
1814 }
1815 }
1816 \cs_new:Npn \@@_scan_backprime:
1817 {
1818     \cs_set_eq:NN \@@_superscript:n \use:n
1819     \int_zero:N \l_@@_primecount_int
1820     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1821 }
1822 \cs_new:Npn \@@_scan_backdprime:
1823 {
1824     \cs_set_eq:NN \@@_superscript:n \use:n
1825     \int_set:Nn \l_@@_primecount_int {1}
1826     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1827 }
1828 \cs_new:Npn \@@_scan_backtrprime:
1829 {
1830     \cs_set_eq:NN \@@_superscript:n \use:n
1831     \int_set:Nn \l_@@_primecount_int {2}

```

```

1832 \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1833 }
1834 \cs_new:Npn \@@_scan_sup_backprime:
1835 {
1836   \int_zero:N \l_@@_primecount_int
1837   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1838 }
1839 \cs_new:Npn \@@_scan_sup_backdprime:
1840 {
1841   \int_set:Nn \l_@@_primecount_int {1}
1842   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1843 }
1844 \cs_new:Npn \@@_scan_sup_backtrprime:
1845 {
1846   \int_set:Nn \l_@@_primecount_int {2}
1847   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1848 }
1849 \cs_new:Nn \@@_scanbackprime_collect:N
1850 {
1851   \int_incr:N \l_@@_primecount_int
1852   \peek_meaning_remove:NTF `
1853   {
1854     \@@_scanbackprime_collect:N #1
1855   }
1856   {
1857     \peek_meaning_remove:NTF \@@_scan_backprime:
1858     {
1859       \@@_scanbackprime_collect:N #1
1860     }
1861     {
1862       \peek_meaning_remove:NTF ^^^^2035
1863       {
1864         \@@_scanbackprime_collect:N #1
1865       }
1866       {
1867         \peek_meaning_remove:NTF \@@_scan_backdprime:
1868         {
1869           \int_incr:N \l_@@_primecount_int
1870           \@@_scanbackprime_collect:N #1
1871         }
1872         {
1873           \peek_meaning_remove:NTF ^^^^2036
1874           {
1875             \int_incr:N \l_@@_primecount_int
1876             \@@_scanbackprime_collect:N #1
1877           }
1878           {
1879             \peek_meaning_remove:NTF \@@_scan_backtrprime:
1880             {

```

```

1881         \int_add:Nn \l_@@_primecount_int {2}
1882     \@@_scanbackprime_collect:N #1
1883 }
1884 {
1885     \peek_meaning_remove:NTF ^^^^2037
1886     {
1887         \int_add:Nn \l_@@_primecount_int {2}
1888         \@@_scanbackprime_collect:N #1
1889     }
1890     {
1891         \@@_nbackprimes_select:nn {#1} {\l_@@_primecount_int}
1892     }
1893 }
1894 }
1895 }
1896 }
1897 }
1898 }
1899 }

1900 \AtBeginDocument{\@@_define_prime_commands: \@@_define_prime_chars:}
1901 \cs_new:Nn \@@_define_prime_commands:
1902 {
1903     \cs_set_eq:NN \prime \@@_prime_single_mchar
1904     \cs_set_eq:NN \dprime \@@_prime_double_mchar
1905     \cs_set_eq:NN \trprime \@@_prime_triple_mchar
1906     \cs_set_eq:NN \qprime \@@_prime_quad_mchar
1907     \cs_set_eq:NN \backprime \@@_backprime_single_mchar
1908     \cs_set_eq:NN \backdprime \@@_backprime_double_mchar
1909     \cs_set_eq:NN \backtrprime \@@_backprime_triple_mchar
1910 }
1911 \group_begin:
1912     \char_set_catcode_active:N \'
1913     \char_set_catcode_active:N `
1914     \char_set_catcode_active:n {"2032}
1915     \char_set_catcode_active:n {"2033}
1916     \char_set_catcode_active:n {"2034}
1917     \char_set_catcode_active:n {"2057}
1918     \char_set_catcode_active:n {"2035}
1919     \char_set_catcode_active:n {"2036}
1920     \char_set_catcode_active:n {"2037}
1921     \cs_gset:Nn \@@_define_prime_chars:
1922     {
1923         \cs_set_eq:NN ' \@@_scan_sup_prime:
1924         \cs_set_eq:NN ^^^^2032 \@@_scan_sup_prime:
1925         \cs_set_eq:NN ^^^^2033 \@@_scan_sup_dprime:
1926         \cs_set_eq:NN ^^^^2034 \@@_scan_sup_trprime:
1927         \cs_set_eq:NN ^^^^2057 \@@_scan_sup_qprime:
1928         \cs_set_eq:NN ` \@@_scan_sup_backprime:
1929         \cs_set_eq:NN ^^^^2035 \@@_scan_sup_backprime:

```

```

1930 \cs_set_eq:NN ^^^^2036 \@@_scan_sup_backdprime:
1931 \cs_set_eq:NN ^^^^2037 \@@_scan_sup_backtrprime:
1932 }
1933 \group_end:

```

M.2 Unicode radicals

Make sure `\Uroot` is defined in the case where the \LaTeX kernel doesn't make it available with its native name.

```

1934 <(*LU)
1935 \cs_if_exist:NF \Uroot
1936 { \cs_new_eq:NN \Uroot \luatexUroot }
1937 </(*LU)

1938 \AtBeginDocument{\@@_redefine_radical:}
1939 \cs_new:Nn \@@_redefine_radical:
1940 <(*XE)
1941 {
1942   \ifpackageloaded { amsmath } { { }
1943   {

```

`\r@@t` #1 : A mathstyle (for `\mathpalette`)

#2 : Leading superscript for the sqrt sign

A re-implementation of \LaTeX 's hard-coded n-root sign using the appropriate `\fontdimens`.

```

1944 \cs_set_nopar:Npn \r@@@t ##1 ##2
1945 {
1946   \hbox_set:Nn \l_tmpa_box
1947   {
1948     \c_math_toggle_token
1949     \m@th
1950     ##1
1951     \sqrtsign { ##2 }
1952     \c_math_toggle_token
1953   }
1954   \@@_mathstyle_scale:Nnn ##1 { \kern }
1955   { \fontdimen 63 \l_@@_font }
1956   \box_move_up:nn
1957   {
1958     (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
1959     * \number \fontdimen 65 \l_@@_font / 100
1960   }
1961   { \box_use:N \rootbox }
1962   \@@_mathstyle_scale:Nnn ##1 { \kern }
1963   { \fontdimen 64 \l_@@_font }
1964   \box_use_clear:N \l_tmpa_box
1965 }

1966 }
1967 }

```

```

1968 </XE>
1969 <*LU>
1970 {
1971   \ifpackageloaded { amsmath } { { }
1972   {

```

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

```

1973   \cs_set:Npn \root ##1 \of ##2
1974   {
1975     \Uroot \l_@@_radical_sqrt_tl { ##1 } { ##2 }
1976   }
1977 }
1978 }
1979 </LU>

```

`\@@_fontdimen_to_percent:nn` #1 : Font dimen number

`\@@_fontdimen_to_scale:nn` #2 : Font ‘variable’

`\fontdimens` 10, 11, and 65 aren’t actually dimensions, they’re percentage values given in units of sp. `\@@_fontdimen_to_percent:nn` takes a font dimension number and outputs the decimal value of the associated parameter. `\@@_fontdimen_to_scale:nn` returns a dimension correspond to the current font size relative proportion based on that percentage.

```

1980 \cs_new:Nn \@@_fontdimen_to_percent:nn
1981 {
1982   \fp_eval:n { \dim_to_decimal:n { \fontdimen #1 #2 } * 65536 / 100 }
1983 }
1984 \cs_new:Nn \@@_fontdimen_to_scale:nn
1985 {
1986   \fp_eval:n { \@@_fontdimen_to_percent:nn {#1} {#2} * \f@size } pt
1987 }

```

`\@@_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.

```

1988 \cs_new:Nn \@@_mathstyle_scale:Nnn
1989 {
1990   \ifx#1\scriptstyle
1991     #2 \@@_fontdimen_to_percent:nn {10} \l_@@_font #3
1992   \else
1993     \ifx#1\scriptscriptstyle
1994       #2 \@@_fontdimen_to_percent:nn {11} \l_@@_font #3
1995     \else
1996       #2 #3
1997     \fi
1998   \fi
1999 }

```


M.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 X_YTeX 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?

```
2000 \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.

```
2001 \cs_new:Nn \@@_setup_active_superscript:nn
2002 {
2003   \prop_gput:Nn \g_@@_supers_prop   {\meaning #1} {#2}
2004   \char_set_catcode_active:N #1
2005   \@@_char_gmake_mathactive:N #1
2006   \scantokens
2007   {
2008     \cs_gset:Npn #1
2009     {
2010       \tl_set:Nn \l_@@_ss_chain_tl {#2}
2011       \cs_set_eq:NN \@@_sub_or_super:n \sp
2012       \tl_set:Nn \l_@@_tmpa_tl {supers}
2013       \@@_scan_sscript:
2014     }
2015   }
2016 }
```

Bam:

```
2017 \@@_setup_active_superscript:nn {^^^2070} {0}
2018 \@@_setup_active_superscript:nn {^^^00b9} {1}
2019 \@@_setup_active_superscript:nn {^^^00b2} {2}
2020 \@@_setup_active_superscript:nn {^^^00b3} {3}
2021 \@@_setup_active_superscript:nn {^^^2074} {4}
2022 \@@_setup_active_superscript:nn {^^^2075} {5}
2023 \@@_setup_active_superscript:nn {^^^2076} {6}
2024 \@@_setup_active_superscript:nn {^^^2077} {7}
2025 \@@_setup_active_superscript:nn {^^^2078} {8}
2026 \@@_setup_active_superscript:nn {^^^2079} {9}
2027 \@@_setup_active_superscript:nn {^^^207a} {+}
2028 \@@_setup_active_superscript:nn {^^^207b} {-}
```

```

2029 \@@_setup_active_superscript:nn {^^^^207c} {=}
2030 \@@_setup_active_superscript:nn {^^^^207d} {(}
2031 \@@_setup_active_superscript:nn {^^^^207e} {)}
2032 \@@_setup_active_superscript:nn {^^^^2071} {i}
2033 \@@_setup_active_superscript:nn {^^^^207f} {n}
2034 \@@_setup_active_superscript:nn {^^^^02b0} {h}
2035 \@@_setup_active_superscript:nn {^^^^02b2} {j}
2036 \@@_setup_active_superscript:nn {^^^^02b3} {r}
2037 \@@_setup_active_superscript:nn {^^^^02b7} {w}
2038 \@@_setup_active_superscript:nn {^^^^02b8} {y}

```

Subscripts Ditto above.

```

2039 \cs_new:Nn \@@_setup_active_subscript:nn
2040 {
2041   \prop_gput:Non \g_@@_subs_prop {\meaning #1} {#2}
2042   \char_set_catcode_active:N #1
2043   \@@_char_gmake_mathactive:N #1
2044   \scantokens
2045   {
2046     \cs_gset:Npn #1
2047     {
2048       \tl_set:Nn \l_@@_ss_chain_tl {#2}
2049       \cs_set_eq:NN \@@_sub_or_super:n \sb
2050       \tl_set:Nn \l_@@_tmpa_tl {subs}
2051       \@@_scan_sscript:
2052     }
2053   }
2054 }

```

A few more subscripts than superscripts:

```

2055 \@@_setup_active_subscript:nn {^^^^2080} {0}
2056 \@@_setup_active_subscript:nn {^^^^2081} {1}
2057 \@@_setup_active_subscript:nn {^^^^2082} {2}
2058 \@@_setup_active_subscript:nn {^^^^2083} {3}
2059 \@@_setup_active_subscript:nn {^^^^2084} {4}
2060 \@@_setup_active_subscript:nn {^^^^2085} {5}
2061 \@@_setup_active_subscript:nn {^^^^2086} {6}
2062 \@@_setup_active_subscript:nn {^^^^2087} {7}
2063 \@@_setup_active_subscript:nn {^^^^2088} {8}
2064 \@@_setup_active_subscript:nn {^^^^2089} {9}
2065 \@@_setup_active_subscript:nn {^^^^208a} {+}
2066 \@@_setup_active_subscript:nn {^^^^208b} {-}
2067 \@@_setup_active_subscript:nn {^^^^208c} {=}
2068 \@@_setup_active_subscript:nn {^^^^208d} {(}
2069 \@@_setup_active_subscript:nn {^^^^208e} {)}
2070 \@@_setup_active_subscript:nn {^^^^2090} {a}
2071 \@@_setup_active_subscript:nn {^^^^2091} {e}
2072 \@@_setup_active_subscript:nn {^^^^1d62} {i}
2073 \@@_setup_active_subscript:nn {^^^^2092} {o}

```

```

2074 \@@_setup_active_subscript:nn {^^^1d63} {r}
2075 \@@_setup_active_subscript:nn {^^^1d64} {u}
2076 \@@_setup_active_subscript:nn {^^^1d65} {v}
2077 \@@_setup_active_subscript:nn {^^^2093} {x}
2078 \@@_setup_active_subscript:nn {^^^1d66} {\beta}
2079 \@@_setup_active_subscript:nn {^^^1d67} {\gamma}
2080 \@@_setup_active_subscript:nn {^^^1d68} {\rho}
2081 \@@_setup_active_subscript:nn {^^^1d69} {\phi}
2082 \@@_setup_active_subscript:nn {^^^1d6a} {\chi}
2083 \group_end:

```

The scanning command, evident in its purpose:

```

2084 \cs_new:Npn \@@_scan_sscript:
2085 {
2086   \@@_scan_sscript:TF
2087   {
2088     \@@_scan_sscript:
2089   }
2090   {
2091     \@@_sub_or_super:n {\l_@@_ss_chain_tl}
2092   }
2093 }

```

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

```

2094 \cs_new:Npn \@@_scan_sscript:TF #1#2
2095 {
2096   \tl_set:Nx \__peek_true_aux:w { \exp_not:n{ #1 } }
2097   \tl_set_eq:NN \__peek_true:w \__peek_true_remove:w
2098   \tl_set:Nx \__peek_false:w { \exp_not:n { \group_align_safe_end: #2 } }
2099   \group_align_safe_begin:
2100   \peek_after:Nw \@@_peek_execute_branches_ss:
2101 }

```

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

```

2102 \cs_new:Npn \@@_peek_execute_branches_ss:
2103 {
2104   \bool_if:nTF
2105   {
2106     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2107     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2108     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2109   }
2110   { \__peek_false:w }
2111   { \@@_peek_execute_branches_ss_aux: }
2112 }

```

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.)

```

2113 \cs_new:Npn \@@_peek_execute_branches_ss_aux:
2114 {
2115   \prop_if_in:coTF
2116     {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token}
2117   {
2118     \prop_get:coN
2119       {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token} \l_@@_tmpb_tl
2120     \tl_put_right:NV \l_@@_ss_chain_tl \l_@@_tmpb_tl
2121     \__peek_true:w
2122   }
2123   { \__peek_false:w }
2124 }

```

M.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 L^AT_EX fraction declaration.

```

2125 \cs_new:Npn \@@_define_active_frac:Nw #1 #2/#3
2126 {
2127   \char_set_catcode_active:N #1
2128   \@@_char_gmake_mathactive:N #1
2129   \tl_rescan:nn
2130   {
2131     \catcode`\_ =11\relax
2132     \catcode`\: =11\relax
2133   }
2134   {
2135     \cs_gset:Npx #1
2136     {
2137       \bool_if:NTF \l_@@_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2138       {#2} {#3}
2139     }
2140   }
2141 }

```

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

```

2142 \cs_new:Npn \@@_setup_active_frac:
2143 {
2144   \group_begin:
2145   \@@_define_active_frac:Nw ^^^^2189 0/3
2146   \@@_define_active_frac:Nw ^^^^2152 1/{10}
2147   \@@_define_active_frac:Nw ^^^^2151 1/9
2148   \@@_define_active_frac:Nw ^^^^215b 1/8

```

```

2149 \@@_define_active_frac:Nw ^^^^2150 1/7
2150 \@@_define_active_frac:Nw ^^^^2159 1/6
2151 \@@_define_active_frac:Nw ^^^^2155 1/5
2152 \@@_define_active_frac:Nw ^^^^00bc 1/4
2153 \@@_define_active_frac:Nw ^^^^2153 1/3
2154 \@@_define_active_frac:Nw ^^^^215c 3/8
2155 \@@_define_active_frac:Nw ^^^^2156 2/5
2156 \@@_define_active_frac:Nw ^^^^00bd 1/2
2157 \@@_define_active_frac:Nw ^^^^2157 3/5
2158 \@@_define_active_frac:Nw ^^^^215d 5/8
2159 \@@_define_active_frac:Nw ^^^^2154 2/3
2160 \@@_define_active_frac:Nw ^^^^00be 3/4
2161 \@@_define_active_frac:Nw ^^^^2158 4/5
2162 \@@_define_active_frac:Nw ^^^^215a 5/6
2163 \@@_define_active_frac:Nw ^^^^215e 7/8
2164 \group_end:
2165 }
2166 \@@_setup_active_frac:

```

M.4 *Synonyms and all the rest*

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

```

2167 \def\to{\rightarrow}
2168 \def\le{\leq}
2169 \def\ge{\geq}
2170 \def\neq{\neq}
2171 \def\triangle{\mathord{\bigtriangleup}}
2172 \def\bigcirc{\mdlgwhtcircle}
2173 \def\circ{\vysmwhtcircle}
2174 \def\bullet{\smbllkcircle}
2175 \def\mathyen{\yen}
2176 \def\mathsterling{\sterling}
2177 \def\diamond{\smwhtdiamond}
2178 \def\emptyset{\varnothing}
2179 \def\hbar{\hslash}
2180 \def\land{\wedge}
2181 \def\lor{\vee}
2182 \def\owns{\ni}
2183 \def\gets{\leftarrow}
2184 \def\mathring{\ocirc}
2185 \def\lnot{\neg}
2186 \def\longdivision{\longdivisionsign}

```

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

```

2187 \def\backepsilon{\upbackepsilon}
2188 \def\eth{\matheth}

```

These are names that are ‘frozen’ in HTML but have dumb names:

```

2189 \def\dbkarow {\dbkarrow}
2190 \def\drbkarow{\drbkarow}
2191 \def\hksearrow{\hksearrow}
2192 \def\hkswarrow{\hkswarrow}

```

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

```

2193 \def\smallint{\mathop{\textstyle\int}\limits}

```

\underbar

```

2194 \cs_set_eq:NN \latexe_underbar:n \underbar
2195 \renewcommand\underbar
2196 {
2197   \mode_if_math:TF \mathunderbar \latexe_underbar:n
2198 }

```

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

```

2199 \@ifpackageloaded{amsmath}
2200 {
2201   % define their own colon, perhaps I should just steal it. (It does look much bet-
2202   % ter.)
2203 }
2204 {
2205   \cs_set_protected:Npn \colon
2206   {
2207     \bool_if:NTF \g_@@_literal_colon_bool {::} { \mathpunct{:} }
2208   }
2209 }

```

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

```

\Digamma 2209 \def\digamma{\updigamma}
2210 \def\Digamma{\upDigamma}

```

Symbols

```

2211 \cs_set:Npn \l {\Vert}

\mathinner items:

2212 \cs_set:Npn \matheellipsis {\mathinner{\unicodeellipsis}}
2213 \cs_set:Npn \cdots {\mathinner{\unicodedots}}

2214 \cs_set_eq:NN \@@_text_slash: \slash
2215 \cs_set_protected:Npn \slash
2216 {
2217   \mode_if_math:TF {\mathslash} {\@@_text_slash:}
2218 }

```

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

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

This code is based on an answer to a TeX – Stack Exchange question by Enrico Gregorio⁶.

```

2219 \cs_new:Npn \@@_newnot:N #1
2220 {
2221   \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
2222   \exp_args:Nx \tl_if_empty:nF { \tl_tail:V \l_not_token_name_tl }
2223   {
2224     \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }
2225   }
2226   \cs_if_exist:cTF { n \l_not_token_name_tl }
2227   {
2228     \use:c { n \l_not_token_name_tl }
2229   }
2230   {
2231     \cs_if_exist:cTF { not \l_not_token_name_tl }
2232     {
2233       \use:c { not \l_not_token_name_tl }
2234     }
2235     {
2236       \@@_oldnot: #1
2237     }
2238   }
2239 }

2240 \cs_set_eq:NN \@@_oldnot: \not
2241 \AtBeginDocument{\cs_set_eq:NN \not \@@_newnot:N}

2242 \cs_new_protected_nopar:Nn \@@_setup_negations:
2243 {
2244   \cs_gset:cpn { not= } { \neq }
2245   \cs_gset:cpn { not< } { \nless }
2246   \cs_gset:cpn { not> } { \ngtr }
2247   \cs_gset:Npn \ngets { \leftarrow }
2248   \cs_gset:Npn \nsimeq { \simeq }
2249   \cs_gset:Npn \nequal { \ne }
2250   \cs_gset:Npn \nle { \nleq }
2251   \cs_gset:Npn \nge { \ngeq }
2252   \cs_gset:Npn \ngreater { \ngtr }
2253   \cs_gset:Npn \nforksnot { \forks }
2254 }

```

⁶<http://tex.stackexchange.com/a/47260/729>

2255 </package&(XE|LU)>

N *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.

2256 <*msg>

Wrapper functions:

```

2257 \cs_new:Npn \@@_error:n { \msg_error:nn {unicode-math} }
2258 \cs_new:Npn \@@_warning:n { \msg_warning:nn {unicode-math} }
2259 \cs_new:Npn \@@_warning:nnn { \msg_warning:nnxx {unicode-math} }
2260 \cs_new:Npn \@@_log:n { \msg_log:nn {unicode-math} }
2261 \cs_new:Npn \@@_log:nx { \msg_log:nnx {unicode-math} }

2262 \msg_new:nnn {unicode-math} {no-tfrac}
2263 {
2264   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\.
2265   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
2266 }
2267 \msg_new:nnn {unicode-math} {default-math-font}
2268 {
2269   Defining~ the~ default~ maths~ font~ as~ '\l_@@_fontname_tl'.
2270 }
2271 \msg_new:nnn {unicode-math} {setup-implicit}
2272 {
2273   Setup~ alphabets:~ implicit~ mode.
2274 }
2275 \msg_new:nnn {unicode-math} {setup-explicit}
2276 {
2277   Setup~ alphabets:~ explicit~ mode.
2278 }
2279 \msg_new:nnn {unicode-math} {alph-initialise}
2280 {
2281   Initialising~ \@backslashchar math#1.
2282 }
2283 \msg_new:nnn {unicode-math} {setup-alph}
2284 {
2285   Setup~ alphabet:~ #1.
2286 }
2287 \msg_new:nnn {unicode-math} {no-alphabet}
2288 {
2289   I~ am~ trying~ to~ set~ up~ alphabet~"#1"~ but~ there~ are~ no~ configura-
        tion~ settings~ for~ it.~
2290   (See~ source~ file~ "unicode-math-alphabets.dtx"~ to~ debug.)
2291 }
2292 \msg_new:nnn { unicode-math } { no-named-range }
2293 {

```



```

2294 I~ am~ trying~ to~ define~ new~ alphabet~ "#2"~ in~ range~ "#1",~ but~ range~ "#1"~ hasn't~ been~ de-
      fined~ yet.
2295 }
2296 \msg_new:nnn { unicode-math } { missing-alphabets }
2297 {
2298   Missing~math~alphabets~in~font~ "\fontname\l_@@_font" \ \ \
2299   \seq_map_function:NN \l_@@_missing_alph_seq \@@_print_indent:n
2300 }
2301 \cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \ \ }
2302 \msg_new:nnn { unicode-math } { macro-expected }
2303 {
2304   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
2305 }
2306 \msg_new:nnn { unicode-math } { wrong-meaning }
2307 {
2308   I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ mean-
      ing~ #2.
2309 }
2310 \msg_new:nnn { unicode-math } { patch-macro }
2311 {
2312   I'm~ going~ to~ patch~ macro~ #1.
2313 }
2314 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
2315   Using~ \token_to_str:N \overbracket\ and~
2316   \token_to_str:N \underbracket\ from~
2317   'mathtools'~ package.\ \
2318   \ \
2319   Use~ \token_to_str:N \Uoverbracket\ and~
2320   \token_to_str:N \Uunderbracket\ for~
2321   original~ 'unicode-math'~ definition.
2322 }
2323 \msg_new:nnn { unicode-math } { mathtools-colon } {
2324   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
2325   the~ 'mathtools'~ package: \ \ \
2326   \ \ \ \ \token_to_str:N \dblcolon,~
2327   \token_to_str:N \coloneqq,~
2328   \token_to_str:N \Coloneqq,~
2329   \token_to_str:N \eqqcolon. \ \ \
2330   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
2331   commands,~ using~ them~ will~ lead~ to~ inconsistencies.
2332 }
2333 \msg_new:nnn { unicode-math } { colonequals } {
2334   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
2335   the~ 'colonequals'~ package: \ \ \
2336   \ \ \ \ \token_to_str:N \ratio,~
2337   \token_to_str:N \coloncolon,~
2338   \token_to_str:N \minuscolon, \ \
2339   \ \ \ \ \token_to_str:N \colonequals,~
2340   \token_to_str:N \equalscolon,~

```

```

2341         \token_to_str:N \coloncolonequals. \ \ \
2342 Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
2343 commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
2344 Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
2345 or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
2346 any~ effect~ on~ the~ re-defined~ commands.
2347 }
2348 </msg>

```

N.1 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.⁷

```

2349 <*usv>

```

Alphabets

```

2350 \usv_set:nnn {normal} {num} {48}
2351 \usv_set:nnn {normal} {Latin}{1D434}
2352 \usv_set:nnn {normal} {latin}{1D44E}
2353 \usv_set:nnn {normal} {Greek}{1D6E2}
2354 \usv_set:nnn {normal} {greek}{1D6FC}
2355 \usv_set:nnn {normal}{varTheta} {1D6F3}
2356 \usv_set:nnn {normal}{varepsilon}{1D716}
2357 \usv_set:nnn {normal}{vartheta} {1D717}
2358 \usv_set:nnn {normal}{varkappa} {1D718}
2359 \usv_set:nnn {normal}{varphi} {1D719}
2360 \usv_set:nnn {normal}{varrho} {1D71A}
2361 \usv_set:nnn {normal}{varpi} {1D71B}
2362 \usv_set:nnn {normal} {Nabla}{1D6FB}
2363 \usv_set:nnn {normal} {partial}{1D715}
2364
2365 \usv_set:nnn {up} {num} {48}
2366 \usv_set:nnn {up} {Latin}{65}
2367 \usv_set:nnn {up} {latin}{97}
2368 \usv_set:nnn {up} {Greek}{391}
2369 \usv_set:nnn {up} {greek}{3B1}
2370 \usv_set:nnn {it} {Latin}{1D434}
2371 \usv_set:nnn {it} {latin}{1D44E}
2372 \usv_set:nnn {it} {Greek}{1D6E2}
2373 \usv_set:nnn {it} {greek}{1D6FC}
2374 \usv_set:nnn {bb} {num} {1D7D8}
2375 \usv_set:nnn {bb} {Latin}{1D538}
2376 \usv_set:nnn {bb} {latin}{1D552}
2377 \usv_set:nnn {scr} {Latin}{1D49C}
2378 \usv_set:nnn {cal} {Latin}{1D49C}
2379 \usv_set:nnn {scr} {latin}{1D4B6}

```

⁷'u.s.v.' stands for 'Unicode scalar value'.

2380 \usv_set:nnn {frak}{Latin}{ "1D504}
 2381 \usv_set:nnn {frak}{latin}{ "1D51E}
 2382 \usv_set:nnn {sf} {num} { "1D7E2}
 2383 \usv_set:nnn {sfup}{num} { "1D7E2}
 2384 \usv_set:nnn {sfrit}{num} { "1D7E2}
 2385 \usv_set:nnn {sfup}{Latin}{ "1D5A0}
 2386 \usv_set:nnn {sf} {Latin}{ "1D5A0}
 2387 \usv_set:nnn {sfup}{latin}{ "1D5BA}
 2388 \usv_set:nnn {sf} {latin}{ "1D5BA}
 2389 \usv_set:nnn {sfrit}{Latin}{ "1D608}
 2390 \usv_set:nnn {sfrit}{latin}{ "1D622}
 2391 \usv_set:nnn {tt} {num} { "1D7F6}
 2392 \usv_set:nnn {tt} {Latin}{ "1D670}
 2393 \usv_set:nnn {tt} {latin}{ "1D68A}

Bold:

2394 \usv_set:nnn {bf} {num} { "1D7CE}
 2395 \usv_set:nnn {bfup} {num} { "1D7CE}
 2396 \usv_set:nnn {bfit} {num} { "1D7CE}
 2397 \usv_set:nnn {bfup} {Latin}{ "1D400}
 2398 \usv_set:nnn {bfup} {latin}{ "1D41A}
 2399 \usv_set:nnn {bfup} {Greek}{ "1D6A8}
 2400 \usv_set:nnn {bfup} {greek}{ "1D6C2}
 2401 \usv_set:nnn {bfit} {Latin}{ "1D468}
 2402 \usv_set:nnn {bfit} {latin}{ "1D482}
 2403 \usv_set:nnn {bfit} {Greek}{ "1D71C}
 2404 \usv_set:nnn {bfit} {greek}{ "1D736}
 2405 \usv_set:nnn {bffrak}{Latin}{ "1D56C}
 2406 \usv_set:nnn {bffrak}{latin}{ "1D586}
 2407 \usv_set:nnn {bfscr} {Latin}{ "1D4D0}
 2408 \usv_set:nnn {bfcal} {Latin}{ "1D4D0}
 2409 \usv_set:nnn {bfscr} {latin}{ "1D4EA}
 2410 \usv_set:nnn {bfsf} {num} { "1D7EC}
 2411 \usv_set:nnn {bfsfup}{num} { "1D7EC}
 2412 \usv_set:nnn {bfsfit}{num} { "1D7EC}
 2413 \usv_set:nnn {bfsfup}{Latin}{ "1D5D4}
 2414 \usv_set:nnn {bfsfup}{latin}{ "1D5EE}
 2415 \usv_set:nnn {bfsfup}{Greek}{ "1D756}
 2416 \usv_set:nnn {bfsfup}{greek}{ "1D770}
 2417 \usv_set:nnn {bfsfit}{Latin}{ "1D63C}
 2418 \usv_set:nnn {bfsfit}{latin}{ "1D656}
 2419 \usv_set:nnn {bfsfit}{Greek}{ "1D790}
 2420 \usv_set:nnn {bfsfit}{greek}{ "1D7AA}

2421 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_@@_upLatin_bool \g_@@_bfsfup_Latin_usv \g_@@_bfsfit_Lat
 2422 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_@@_uplatin_bool \g_@@_bfsfup_latin_usv \g_@@_bfsfit_lat
 2423 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_@@_upGreek_bool \g_@@_bfsfup_Greek_usv \g_@@_bfsfit_Gre
 2424 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_@@_upgreek_bool \g_@@_bfsfup_greek_usv \g_@@_bfsfit_gre
 2425 \usv_set:nnn {bf} {Latin}{ \bool_if:NTF \g_@@_bfupLatin_bool \g_@@_bfup_Latin_usv \g_@@_bfit_Latin
 2426 \usv_set:nnn {bf} {latin}{ \bool_if:NTF \g_@@_bfuplatin_bool \g_@@_bfup_latin_usv \g_@@_bfit_latin

2427 \usv_set:nnn {bf} {Greek}{ \bool_if:NTF \g_@@_bfupGreek_bool \g_@@_bfup_Greek_usv \g_@@_bfit_Greek_

2428 \usv_set:nnn {bf} {greek}{ \bool_if:NTF \g_@@_bfupgreek_bool \g_@@_bfup_greek_usv \g_@@_bfit_greek_

Greek variants:

2429 \usv_set:nnn {up}{varTheta} {"3F4}

2430 \usv_set:nnn {up}{Digamma} {"3DC}

2431 \usv_set:nnn {up}{varepsilon}{ "3F5}

2432 \usv_set:nnn {up}{vartheta} {"3D1}

2433 \usv_set:nnn {up}{varkappa} {"3F0}

2434 \usv_set:nnn {up}{varphi} {"3D5}

2435 \usv_set:nnn {up}{varrho} {"3F1}

2436 \usv_set:nnn {up}{varpi} {"3D6}

2437 \usv_set:nnn {up}{digamma} {"3DD}

Bold:

2438 \usv_set:nnn {bfup}{varTheta} {"1D6B9}

2439 \usv_set:nnn {bfup}{Digamma} {"1D7CA}

2440 \usv_set:nnn {bfup}{varepsilon}{ "1D6DC}

2441 \usv_set:nnn {bfup}{vartheta} {"1D6DD}

2442 \usv_set:nnn {bfup}{varkappa} {"1D6DE}

2443 \usv_set:nnn {bfup}{varphi} {"1D6DF}

2444 \usv_set:nnn {bfup}{varrho} {"1D6E0}

2445 \usv_set:nnn {bfup}{varpi} {"1D6E1}

2446 \usv_set:nnn {bfup}{digamma} {"1D7CB}

Italic Greek variants:

2447 \usv_set:nnn {it}{varTheta} {"1D6F3}

2448 \usv_set:nnn {it}{varepsilon}{ "1D716}

2449 \usv_set:nnn {it}{vartheta} {"1D717}

2450 \usv_set:nnn {it}{varkappa} {"1D718}

2451 \usv_set:nnn {it}{varphi} {"1D719}

2452 \usv_set:nnn {it}{varrho} {"1D71A}

2453 \usv_set:nnn {it}{varpi} {"1D71B}

Bold italic:

2454 \usv_set:nnn {bfit}{varTheta} {"1D72D}

2455 \usv_set:nnn {bfit}{varepsilon}{ "1D750}

2456 \usv_set:nnn {bfit}{vartheta} {"1D751}

2457 \usv_set:nnn {bfit}{varkappa} {"1D752}

2458 \usv_set:nnn {bfit}{varphi} {"1D753}

2459 \usv_set:nnn {bfit}{varrho} {"1D754}

2460 \usv_set:nnn {bfit}{varpi} {"1D755}

Bold sans:

2461 \usv_set:nnn {bfsfup}{varTheta} {"1D767}

2462 \usv_set:nnn {bfsfup}{varepsilon}{ "1D78A}

2463 \usv_set:nnn {bfsfup}{vartheta} {"1D78B}

2464 \usv_set:nnn {bfsfup}{varkappa} {"1D78C}

2465 \usv_set:nnn {bfsfup}{varphi} {"1D78D}

2466 \usv_set:nnn {bfsfup}{varrho} {"1D78E}

2467 \usv_set:nnn {bfsfup}{varpi} {"1D78F}

Bold sans italic:

```
2468 \usv_set:nnn {bfsfit}{varTheta} {"1D7A1}
2469 \usv_set:nnn {bfsfit}{varepsilon} {"1D7C4}
2470 \usv_set:nnn {bfsfit}{vartheta} {"1D7C5}
2471 \usv_set:nnn {bfsfit}{varkappa} {"1D7C6}
2472 \usv_set:nnn {bfsfit}{varphi} {"1D7C7}
2473 \usv_set:nnn {bfsfit}{varrho} {"1D7C8}
2474 \usv_set:nnn {bfsfit}{varpi} {"1D7C9}
```

Nabla:

```
2475 \usv_set:nnn {up} {Nabla} {"02207}
2476 \usv_set:nnn {it} {Nabla} {"1D6FB}
2477 \usv_set:nnn {bfup} {Nabla} {"1D6C1}
2478 \usv_set:nnn {bfit} {Nabla} {"1D735}
2479 \usv_set:nnn {bfsfup}{Nabla} {"1D76F}
2480 \usv_set:nnn {bfsfit}{Nabla} {"1D7A9}
```

Partial:

```
2481 \usv_set:nnn {up} {partial} {"02202}
2482 \usv_set:nnn {it} {partial} {"1D715}
2483 \usv_set:nnn {bfup} {partial} {"1D6DB}
2484 \usv_set:nnn {bfit} {partial} {"1D74F}
2485 \usv_set:nnn {bfsfup}{partial} {"1D789}
2486 \usv_set:nnn {bfsfit}{partial} {"1D7C3}
```

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

```
2487 \usv_set:nnn {up}{B}{'\B}
2488 \usv_set:nnn {up}{C}{'\C}
2489 \usv_set:nnn {up}{D}{'\D}
2490 \usv_set:nnn {up}{E}{'\E}
2491 \usv_set:nnn {up}{F}{'\F}
2492 \usv_set:nnn {up}{H}{'\H}
2493 \usv_set:nnn {up}{I}{'\I}
2494 \usv_set:nnn {up}{L}{'\L}
2495 \usv_set:nnn {up}{M}{'\M}
2496 \usv_set:nnn {up}{N}{'\N}
2497 \usv_set:nnn {up}{P}{'\P}
2498 \usv_set:nnn {up}{Q}{'\Q}
2499 \usv_set:nnn {up}{R}{'\R}
2500 \usv_set:nnn {up}{Z}{'\Z}

2501 \usv_set:nnn {it}{B} {"1D435}
2502 \usv_set:nnn {it}{C} {"1D436}
2503 \usv_set:nnn {it}{D} {"1D437}
2504 \usv_set:nnn {it}{E} {"1D438}
2505 \usv_set:nnn {it}{F} {"1D439}
2506 \usv_set:nnn {it}{H} {"1D43B}
2507 \usv_set:nnn {it}{I} {"1D43C}
2508 \usv_set:nnn {it}{L} {"1D43F}
```

2509 \usv_set:nnn {it}{M}{1D440}
 2510 \usv_set:nnn {it}{N}{1D441}
 2511 \usv_set:nnn {it}{P}{1D443}
 2512 \usv_set:nnn {it}{Q}{1D444}
 2513 \usv_set:nnn {it}{R}{1D445}
 2514 \usv_set:nnn {it}{Z}{1D44D}

 2515 \usv_set:nnn {up}{d}{\d}
 2516 \usv_set:nnn {up}{e}{\e}
 2517 \usv_set:nnn {up}{g}{\g}
 2518 \usv_set:nnn {up}{h}{\h}
 2519 \usv_set:nnn {up}{i}{\i}
 2520 \usv_set:nnn {up}{j}{\j}
 2521 \usv_set:nnn {up}{o}{\o}

 2522 \usv_set:nnn {it}{d}{1D451}
 2523 \usv_set:nnn {it}{e}{1D452}
 2524 \usv_set:nnn {it}{g}{1D454}
 2525 \usv_set:nnn {it}{h}{0210E}
 2526 \usv_set:nnn {it}{i}{1D456}
 2527 \usv_set:nnn {it}{j}{1D457}
 2528 \usv_set:nnn {it}{o}{1D45C}

Latin ‘h’:

2529 \usv_set:nnn {bb} {h}{1D559}
 2530 \usv_set:nnn {tt} {h}{1D691}
 2531 \usv_set:nnn {scr} {h}{1D4BD}
 2532 \usv_set:nnn {frak} {h}{1D525}
 2533 \usv_set:nnn {bfup} {h}{1D421}
 2534 \usv_set:nnn {bfit} {h}{1D489}
 2535 \usv_set:nnn {sfup} {h}{1D5C1}
 2536 \usv_set:nnn {sfit} {h}{1D629}
 2537 \usv_set:nnn {bffrak}{h}{1D58D}
 2538 \usv_set:nnn {bfscr} {h}{1D4F1}
 2539 \usv_set:nnn {bfsfup}{h}{1D5F5}
 2540 \usv_set:nnn {bfsfit}{h}{1D65D}

Dotless ‘i’ and ‘j’:

2541 \usv_set:nnn {up}{dotlessi}{00131}
 2542 \usv_set:nnn {up}{dotlessj}{00237}
 2543 \usv_set:nnn {it}{dotlessi}{1D6A4}
 2544 \usv_set:nnn {it}{dotlessj}{1D6A5}

Blackboard:

2545 \usv_set:nnn {bb}{C}{2102}
 2546 \usv_set:nnn {bb}{H}{210D}
 2547 \usv_set:nnn {bb}{N}{2115}
 2548 \usv_set:nnn {bb}{P}{2119}
 2549 \usv_set:nnn {bb}{Q}{211A}
 2550 \usv_set:nnn {bb}{R}{211D}
 2551 \usv_set:nnn {bb}{Z}{2124}
 2552 \usv_set:nnn {up}{Pi} {003A0}

2553 \usv_set:nnn {up}{pi} {"003C0}
 2554 \usv_set:nnn {up}{Gamma} {"00393}
 2555 \usv_set:nnn {up}{gamma} {"003B3}
 2556 \usv_set:nnn {up}{summation}{"02211}
 2557 \usv_set:nnn {it}{Pi} {"1D6F1}
 2558 \usv_set:nnn {it}{pi} {"1D70B}
 2559 \usv_set:nnn {it}{Gamma} {"1D6E4}
 2560 \usv_set:nnn {it}{gamma} {"1D6FE}
 2561 \usv_set:nnn {bb}{Pi} {"0213F}
 2562 \usv_set:nnn {bb}{pi} {"0213C}
 2563 \usv_set:nnn {bb}{Gamma} {"0213E}
 2564 \usv_set:nnn {bb}{gamma} {"0213D}
 2565 \usv_set:nnn {bb}{summation}{"02140}

Italic blackboard:

2566 \usv_set:nnn {bbit}{D}{"2145}
 2567 \usv_set:nnn {bbit}{d}{"2146}
 2568 \usv_set:nnn {bbit}{e}{"2147}
 2569 \usv_set:nnn {bbit}{i}{"2148}
 2570 \usv_set:nnn {bbit}{j}{"2149}

Script exceptions:

2571 \usv_set:nnn {scr}{B}{"212C}
 2572 \usv_set:nnn {scr}{E}{"2130}
 2573 \usv_set:nnn {scr}{F}{"2131}
 2574 \usv_set:nnn {scr}{H}{"210B}
 2575 \usv_set:nnn {scr}{I}{"2110}
 2576 \usv_set:nnn {scr}{L}{"2112}
 2577 \usv_set:nnn {scr}{M}{"2133}
 2578 \usv_set:nnn {scr}{R}{"211B}
 2579 \usv_set:nnn {scr}{e}{"212F}
 2580 \usv_set:nnn {scr}{g}{"210A}
 2581 \usv_set:nnn {scr}{o}{"2134}
 2582 \usv_set:nnn {cal}{B}{"212C}
 2583 \usv_set:nnn {cal}{E}{"2130}
 2584 \usv_set:nnn {cal}{F}{"2131}
 2585 \usv_set:nnn {cal}{H}{"210B}
 2586 \usv_set:nnn {cal}{I}{"2110}
 2587 \usv_set:nnn {cal}{L}{"2112}
 2588 \usv_set:nnn {cal}{M}{"2133}
 2589 \usv_set:nnn {cal}{R}{"211B}

Fraktur exceptions:

2590 \usv_set:nnn {frak}{C}{"212D}
 2591 \usv_set:nnn {frak}{H}{"210C}
 2592 \usv_set:nnn {frak}{I}{"2111}
 2593 \usv_set:nnn {frak}{R}{"211C}
 2594 \usv_set:nnn {frak}{Z}{"2128}

2595 **<*usv>**

N.2 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.

2596 *(***stix**)*

Upright

```
2597 \usv_set:nnn {stixsfup}{partial}{ "E17C}
2598 \usv_set:nnn {stixsfup}{Greek}{ "E17D}
2599 \usv_set:nnn {stixsfup}{greek}{ "E196}
2600 \usv_set:nnn {stixsfup}{varTheta}{ "E18E}
2601 \usv_set:nnn {stixsfup}{varepsilon}{ "E1AF}
2602 \usv_set:nnn {stixsfup}{vartheta}{ "E1B0}
2603 \usv_set:nnn {stixsfup}{varkappa}{0000} % ???
2604 \usv_set:nnn {stixsfup}{varphi}{ "E1B1}
2605 \usv_set:nnn {stixsfup}{varrho}{ "E1B2}
2606 \usv_set:nnn {stixsfup}{varpi}{ "E1B3}
2607 \usv_set:nnn {stixupslash}{Greek}{ "E2FC}
```

Italic

```
2608 \usv_set:nnn {stixbbit}{A}{ "E154}
2609 \usv_set:nnn {stixbbit}{B}{ "E155}
2610 \usv_set:nnn {stixbbit}{E}{ "E156}
2611 \usv_set:nnn {stixbbit}{F}{ "E157}
2612 \usv_set:nnn {stixbbit}{G}{ "E158}
2613 \usv_set:nnn {stixbbit}{I}{ "E159}
2614 \usv_set:nnn {stixbbit}{J}{ "E15A}
2615 \usv_set:nnn {stixbbit}{K}{ "E15B}
2616 \usv_set:nnn {stixbbit}{L}{ "E15C}
2617 \usv_set:nnn {stixbbit}{M}{ "E15D}
2618 \usv_set:nnn {stixbbit}{O}{ "E15E}
2619 \usv_set:nnn {stixbbit}{S}{ "E15F}
2620 \usv_set:nnn {stixbbit}{T}{ "E160}
2621 \usv_set:nnn {stixbbit}{U}{ "E161}
2622 \usv_set:nnn {stixbbit}{V}{ "E162}
2623 \usv_set:nnn {stixbbit}{W}{ "E163}
2624 \usv_set:nnn {stixbbit}{X}{ "E164}
2625 \usv_set:nnn {stixbbit}{Y}{ "E165}

2626 \usv_set:nnn {stixbbit}{a}{ "E166}
2627 \usv_set:nnn {stixbbit}{b}{ "E167}
2628 \usv_set:nnn {stixbbit}{c}{ "E168}
2629 \usv_set:nnn {stixbbit}{f}{ "E169}
2630 \usv_set:nnn {stixbbit}{g}{ "E16A}
2631 \usv_set:nnn {stixbbit}{h}{ "E16B}
2632 \usv_set:nnn {stixbbit}{k}{ "E16C}
2633 \usv_set:nnn {stixbbit}{l}{ "E16D}
```


2634 \usv_set:nnn {stixbbit}{m}{ "E16E}
 2635 \usv_set:nnn {stixbbit}{n}{ "E16F}
 2636 \usv_set:nnn {stixbbit}{o}{ "E170}
 2637 \usv_set:nnn {stixbbit}{p}{ "E171}
 2638 \usv_set:nnn {stixbbit}{q}{ "E172}
 2639 \usv_set:nnn {stixbbit}{r}{ "E173}
 2640 \usv_set:nnn {stixbbit}{s}{ "E174}
 2641 \usv_set:nnn {stixbbit}{t}{ "E175}
 2642 \usv_set:nnn {stixbbit}{u}{ "E176}
 2643 \usv_set:nnn {stixbbit}{v}{ "E177}
 2644 \usv_set:nnn {stixbbit}{w}{ "E178}
 2645 \usv_set:nnn {stixbbit}{x}{ "E179}
 2646 \usv_set:nnn {stixbbit}{y}{ "E17A}
 2647 \usv_set:nnn {stixbbit}{z}{ "E17B}

 2648 \usv_set:nnn {stixsfit}{Numerals}{ "E1B4}
 2649 \usv_set:nnn {stixsfit}{partial}{ "E1BE}
 2650 \usv_set:nnn {stixsfit}{Greek}{ "E1BF}
 2651 \usv_set:nnn {stixsfit}{greek}{ "E1D8}
 2652 \usv_set:nnn {stixsfit}{varTheta}{ "E1D0}
 2653 \usv_set:nnn {stixsfit}{varepsilon}{ "E1F1}
 2654 \usv_set:nnn {stixsfit}{vartheta}{ "E1F2}
 2655 \usv_set:nnn {stixsfit}{varkappa}{0000} % ???
 2656 \usv_set:nnn {stixsfit}{varphi}{ "E1F3}
 2657 \usv_set:nnn {stixsfit}{varrho}{ "E1F4}
 2658 \usv_set:nnn {stixsfit}{varpi}{ "E1F5}

 2659 \usv_set:nnn {stixcal}{Latin}{ "E22D}
 2660 \usv_set:nnn {stixcal}{num}{ "E262}
 2661 \usv_set:nnn {scr}{num}{48}
 2662 \usv_set:nnn {it}{num}{48}

 2663 \usv_set:nnn {stixsfitslash}{Latin}{ "E294}
 2664 \usv_set:nnn {stixsfitslash}{latin}{ "E2C8}
 2665 \usv_set:nnn {stixsfitslash}{greek}{ "E32C}
 2666 \usv_set:nnn {stixsfitslash}{varepsilon}{ "E37A}
 2667 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35E}
 2668 \usv_set:nnn {stixsfitslash}{varkappa}{ "E374}
 2669 \usv_set:nnn {stixsfitslash}{varphi}{ "E360}
 2670 \usv_set:nnn {stixsfitslash}{varrho}{ "E376}
 2671 \usv_set:nnn {stixsfitslash}{varpi}{ "E362}
 2672 \usv_set:nnn {stixsfitslash}{digamma}{ "E36A}

Bold

2673 \usv_set:nnn {stixbfupslash}{Greek}{ "E2FD}
 2674 \usv_set:nnn {stixbfupslash}{Digamma}{ "E369}

 2675 \usv_set:nnn {stixbfbb}{A}{ "E38A}
 2676 \usv_set:nnn {stixbfbb}{B}{ "E38B}
 2677 \usv_set:nnn {stixbfbb}{E}{ "E38D}
 2678 \usv_set:nnn {stixbfbb}{F}{ "E38E}

2679 \usv_set:nnn {stixbfbb}{G}{ "E38F}
 2680 \usv_set:nnn {stixbfbb}{I}{ "E390}
 2681 \usv_set:nnn {stixbfbb}{J}{ "E391}
 2682 \usv_set:nnn {stixbfbb}{K}{ "E392}
 2683 \usv_set:nnn {stixbfbb}{L}{ "E393}
 2684 \usv_set:nnn {stixbfbb}{M}{ "E394}
 2685 \usv_set:nnn {stixbfbb}{O}{ "E395}
 2686 \usv_set:nnn {stixbfbb}{S}{ "E396}
 2687 \usv_set:nnn {stixbfbb}{T}{ "E397}
 2688 \usv_set:nnn {stixbfbb}{U}{ "E398}
 2689 \usv_set:nnn {stixbfbb}{V}{ "E399}
 2690 \usv_set:nnn {stixbfbb}{W}{ "E39A}
 2691 \usv_set:nnn {stixbfbb}{X}{ "E39B}
 2692 \usv_set:nnn {stixbfbb}{Y}{ "E39C}

 2693 \usv_set:nnn {stixbfbb}{a}{ "E39D}
 2694 \usv_set:nnn {stixbfbb}{b}{ "E39E}
 2695 \usv_set:nnn {stixbfbb}{c}{ "E39F}
 2696 \usv_set:nnn {stixbfbb}{f}{ "E3A2}
 2697 \usv_set:nnn {stixbfbb}{g}{ "E3A3}
 2698 \usv_set:nnn {stixbfbb}{h}{ "E3A4}
 2699 \usv_set:nnn {stixbfbb}{k}{ "E3A7}
 2700 \usv_set:nnn {stixbfbb}{l}{ "E3A8}
 2701 \usv_set:nnn {stixbfbb}{m}{ "E3A9}
 2702 \usv_set:nnn {stixbfbb}{n}{ "E3AA}
 2703 \usv_set:nnn {stixbfbb}{o}{ "E3AB}
 2704 \usv_set:nnn {stixbfbb}{p}{ "E3AC}
 2705 \usv_set:nnn {stixbfbb}{q}{ "E3AD}
 2706 \usv_set:nnn {stixbfbb}{r}{ "E3AE}
 2707 \usv_set:nnn {stixbfbb}{s}{ "E3AF}
 2708 \usv_set:nnn {stixbfbb}{t}{ "E3B0}
 2709 \usv_set:nnn {stixbfbb}{u}{ "E3B1}
 2710 \usv_set:nnn {stixbfbb}{v}{ "E3B2}
 2711 \usv_set:nnn {stixbfbb}{w}{ "E3B3}
 2712 \usv_set:nnn {stixbfbb}{x}{ "E3B4}
 2713 \usv_set:nnn {stixbfbb}{y}{ "E3B5}
 2714 \usv_set:nnn {stixbfbb}{z}{ "E3B6}

 2715 \usv_set:nnn {stixbfsfup}{Numerals}{ "E3B7}

Bold Italic

2716 \usv_set:nnn {stixbfsfit}{Numerals}{ "E1F6}

 2717 \usv_set:nnn {stixbfbbbit}{A}{ "E200}
 2718 \usv_set:nnn {stixbfbbbit}{B}{ "E201}
 2719 \usv_set:nnn {stixbfbbbit}{E}{ "E203}
 2720 \usv_set:nnn {stixbfbbbit}{F}{ "E204}
 2721 \usv_set:nnn {stixbfbbbit}{G}{ "E205}
 2722 \usv_set:nnn {stixbfbbbit}{I}{ "E206}
 2723 \usv_set:nnn {stixbfbbbit}{J}{ "E207}
 2724 \usv_set:nnn {stixbfbbbit}{K}{ "E208}

2725 \usv_set:nnn {stixbfbbbit}{L}{E209}
2726 \usv_set:nnn {stixbfbbbit}{M}{E20A}
2727 \usv_set:nnn {stixbfbbbit}{O}{E20B}
2728 \usv_set:nnn {stixbfbbbit}{S}{E20C}
2729 \usv_set:nnn {stixbfbbbit}{T}{E20D}
2730 \usv_set:nnn {stixbfbbbit}{U}{E20E}
2731 \usv_set:nnn {stixbfbbbit}{V}{E20F}
2732 \usv_set:nnn {stixbfbbbit}{W}{E210}
2733 \usv_set:nnn {stixbfbbbit}{X}{E211}
2734 \usv_set:nnn {stixbfbbbit}{Y}{E212}

2735 \usv_set:nnn {stixbfbbbit}{a}{E213}
2736 \usv_set:nnn {stixbfbbbit}{b}{E214}
2737 \usv_set:nnn {stixbfbbbit}{c}{E215}
2738 \usv_set:nnn {stixbfbbbit}{e}{E217}
2739 \usv_set:nnn {stixbfbbbit}{f}{E218}
2740 \usv_set:nnn {stixbfbbbit}{g}{E219}
2741 \usv_set:nnn {stixbfbbbit}{h}{E21A}
2742 \usv_set:nnn {stixbfbbbit}{k}{E21D}
2743 \usv_set:nnn {stixbfbbbit}{l}{E21E}
2744 \usv_set:nnn {stixbfbbbit}{m}{E21F}
2745 \usv_set:nnn {stixbfbbbit}{n}{E220}
2746 \usv_set:nnn {stixbfbbbit}{o}{E221}
2747 \usv_set:nnn {stixbfbbbit}{p}{E222}
2748 \usv_set:nnn {stixbfbbbit}{q}{E223}
2749 \usv_set:nnn {stixbfbbbit}{r}{E224}
2750 \usv_set:nnn {stixbfbbbit}{s}{E225}
2751 \usv_set:nnn {stixbfbbbit}{t}{E226}
2752 \usv_set:nnn {stixbfbbbit}{u}{E227}
2753 \usv_set:nnn {stixbfbbbit}{v}{E228}
2754 \usv_set:nnn {stixbfbbbit}{w}{E229}
2755 \usv_set:nnn {stixbfbbbit}{x}{E22A}
2756 \usv_set:nnn {stixbfbbbit}{y}{E22B}
2757 \usv_set:nnn {stixbfbbbit}{z}{E22C}

2758 \usv_set:nnn {stixbfcal}{Latin}{E247}

2759 \usv_set:nnn {stixbfitslash}{Latin}{E295}
2760 \usv_set:nnn {stixbfitslash}{latin}{E2C9}
2761 \usv_set:nnn {stixbfitslash}{greek}{E32D}
2762 \usv_set:nnn {stixsfitslash}{varepsilon}{E37B}
2763 \usv_set:nnn {stixsfitslash}{vartheta}{E35F}
2764 \usv_set:nnn {stixsfitslash}{varkappa}{E375}
2765 \usv_set:nnn {stixsfitslash}{varphi}{E361}
2766 \usv_set:nnn {stixsfitslash}{varrho}{E377}
2767 \usv_set:nnn {stixsfitslash}{varpi}{E363}
2768 \usv_set:nnn {stixsfitslash}{digamma}{E36B}

2769 </stix>

N.3 Alphabets

2770 *(*alphabets)*

N.3.1 Upright: up

```
2771 \@@_new_alphabet_config:nnn {up} {num}
2772 {
2773   \@@_set_normal_numbers:nn {up} {#1}
2774   \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
2775 }
2776
2777 \@@_new_alphabet_config:nnn {up} {Latin}
2778 {
2779   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
2780   {
2781     \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2782   }
2783   \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
2784   \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
2785   \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
2786 }
2787
2788 \@@_new_alphabet_config:nnn {up} {latin}
2789 {
2790   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
2791   {
2792     \bool_if:NT \g_@@_uplatin_bool
2793     {
2794       \@@_set_normal_latin:nn {up,it} {#1}
2795       \@@_set_normal_char:nnn {h} {up,it} {#1}
2796       \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
2797       \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
2798     }
2799   }
2800   \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
2801   \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
2802   \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
2803 }
2804
2805 \@@_new_alphabet_config:nnn {up} {Greek}
2806 {
2807   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
2808   {
2809     \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2810   }
2811   \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
2812   \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
2813   \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
2814 }
2815
2816 \@@_new_alphabet_config:nnn {up} {greek}
```

```

2817 {
2818 \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
2819 {
2820 \bool_if:NT \g_@@_upgreek_bool
2821 {
2822 \@@_set_normal_greek:nn {up,it} {#1}
2823 }
2824 }
2825 \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
2826 \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
2827 \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
2828 }
2829
2830 \@@_new_alphabet_config:nnn {up} {misc}
2831 {
2832 \bool_if:NTF \g_@@_literal_Nabla_bool
2833 {
2834 \@@_set_normal_char:nnn {Nabla}{up}{up}
2835 }
2836 {
2837 \bool_if:NT \g_@@_upNabla_bool
2838 {
2839 \@@_set_normal_char:nnn {Nabla}{up,it}{up}
2840 }
2841 }
2842 \bool_if:NTF \g_@@_literal_partial_bool
2843 {
2844 \@@_set_normal_char:nnn {partial}{up}{up}
2845 }
2846 {
2847 \bool_if:NT \g_@@_uppartial_bool
2848 {
2849 \@@_set_normal_char:nnn {partial}{up,it}{up}
2850 }
2851 }
2852 \@@_set_mathalphabet_pos:nnnn {up} {partial} {up,it} {#1}
2853 \@@_set_mathalphabet_pos:nnnn {up} {Nabla} {up,it} {#1}
2854 \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
2855 \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
2856 }

```

N.3.2 *Italic: it*

```

2857 \@@_new_alphabet_config:nnn {it} {Latin}
2858 {
2859 \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
2860 {
2861 \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2862 }
2863 \@@_set_mathalphabet_Latin:nnn {it}{up,it}{#1}

```

```

2864 }
2865
2866 \@@_new_alphabet_config:nnn {it} {latin}
2867 {
2868   \bool_if:NTF \g_@@_literal_bool
2869   {
2870     \@@_set_normal_latin:nn {it} {#1}
2871     \@@_set_normal_char:nnn {h}{it}{#1}
2872   }
2873   {
2874     \bool_if:NF \g_@@_uplatin_bool
2875     {
2876       \@@_set_normal_latin:nn {up,it} {#1}
2877       \@@_set_normal_char:nnn {h}{up,it}{#1}
2878       \@@_set_normal_char:nnn {dotlessi}{up,it}{#1}
2879       \@@_set_normal_char:nnn {dotlessj}{up,it}{#1}
2880     }
2881   }
2882   \@@_set_mathalphabet_latin:nnn {it} {up,it} {#1}
2883   \@@_set_mathalphabet_pos:nnnn {it} {dotlessi} {up,it} {#1}
2884   \@@_set_mathalphabet_pos:nnnn {it} {dotlessj} {up,it} {#1}
2885 }
2886
2887 \@@_new_alphabet_config:nnn {it} {Greek}
2888 {
2889   \bool_if:NTF \g_@@_literal_bool
2890   {
2891     \@@_set_normal_Greek:nn {it}{#1}
2892   }
2893   {
2894     \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2895   }
2896   \@@_set_mathalphabet_Greek:nnn {it} {up,it}{#1}
2897 }
2898
2899 \@@_new_alphabet_config:nnn {it} {greek}
2900 {
2901   \bool_if:NTF \g_@@_literal_bool
2902   {
2903     \@@_set_normal_greek:nn {it} {#1}
2904   }
2905   {
2906     \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
2907   }
2908   \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
2909 }
2910
2911 \@@_new_alphabet_config:nnn {it} {misc}
2912 {

```

```

2913 \bool_if:NTF \g_@@_literal_Nabla_bool
2914 {
2915   \@@_set_normal_char:nnn {Nabla}{it}{it}
2916 }
2917 {
2918   \bool_if:NF \g_@@_upNabla_bool
2919   {
2920     \@@_set_normal_char:nnn {Nabla}{up,it}{it}
2921   }
2922 }
2923 \bool_if:NTF \g_@@_literal_partial_bool
2924 {
2925   \@@_set_normal_char:nnn {partial}{it}{it}
2926 }
2927 {
2928   \bool_if:NF \g_@@_uppartial_bool
2929   {
2930     \@@_set_normal_char:nnn {partial}{up,it}{it}
2931   }
2932 }
2933 \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
2934 \@@_set_mathalphabet_pos:nnnn {it} {Nabla} {up,it}{#1}
2935 }

```

N.3.3 Blackboard or double-struck: *bb* and *bbit*

```

2936 \@@_new_alphabet_config:nnn {bb} {latin}
2937 {
2938   \@@_set_mathalphabet_latin:nnn {bb} {up,it}{#1}
2939 }
2940
2941 \@@_new_alphabet_config:nnn {bb} {Latin}
2942 {
2943   \@@_set_mathalphabet_Latin:nnn {bb} {up,it}{#1}
2944   \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
2945   \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
2946   \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
2947   \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
2948   \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
2949   \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
2950   \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
2951 }
2952
2953 \@@_new_alphabet_config:nnn {bb} {num}
2954 {
2955   \@@_set_mathalphabet_numbers:nnn {bb} {up}{#1}
2956 }
2957
2958 \@@_new_alphabet_config:nnn {bb} {misc}
2959 {

```

```

2960 \@@_set_mathalphabet_pos:nnnn {bb}      {Pi} {up,it} {#1}
2961 \@@_set_mathalphabet_pos:nnnn {bb}      {pi} {up,it} {#1}
2962 \@@_set_mathalphabet_pos:nnnn {bb}      {Gamma} {up,it} {#1}
2963 \@@_set_mathalphabet_pos:nnnn {bb}      {gamma} {up,it} {#1}
2964 \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up} {#1}
2965 }
2966
2967 \@@_new_alphabet_config:nnn {bbit} {misc}
2968 {
2969 \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
2970 \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
2971 \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
2972 \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
2973 \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
2974 }

```

N.3.4 *Script and caligraphic: scr and cal*

```

2975 \@@_new_alphabet_config:nnn {scr} {Latin}
2976 {
2977 \@@_set_mathalphabet_Latin:nnn {scr} {up,it}{#1}
2978 \@@_set_mathalphabet_pos:nnnn {scr} {B}{up,it}{#1}
2979 \@@_set_mathalphabet_pos:nnnn {scr} {E}{up,it}{#1}
2980 \@@_set_mathalphabet_pos:nnnn {scr} {F}{up,it}{#1}
2981 \@@_set_mathalphabet_pos:nnnn {scr} {H}{up,it}{#1}
2982 \@@_set_mathalphabet_pos:nnnn {scr} {I}{up,it}{#1}
2983 \@@_set_mathalphabet_pos:nnnn {scr} {L}{up,it}{#1}
2984 \@@_set_mathalphabet_pos:nnnn {scr} {M}{up,it}{#1}
2985 \@@_set_mathalphabet_pos:nnnn {scr} {R}{up,it}{#1}
2986 }
2987
2988 \@@_new_alphabet_config:nnn {scr} {latin}
2989 {
2990 \@@_set_mathalphabet_latin:nnn {scr} {up,it}{#1}
2991 \@@_set_mathalphabet_pos:nnnn {scr} {e}{up,it}{#1}
2992 \@@_set_mathalphabet_pos:nnnn {scr} {g}{up,it}{#1}
2993 \@@_set_mathalphabet_pos:nnnn {scr} {o}{up,it}{#1}
2994 }

```

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

```

2995 \@@_new_alphabet_config:nnn {cal} {Latin}
2996 {
2997 \@@_set_mathalphabet_Latin:nnn {cal} {up,it}{#1}
2998 \@@_set_mathalphabet_pos:nnnn {cal} {B}{up,it}{#1}
2999 \@@_set_mathalphabet_pos:nnnn {cal} {E}{up,it}{#1}
3000 \@@_set_mathalphabet_pos:nnnn {cal} {F}{up,it}{#1}
3001 \@@_set_mathalphabet_pos:nnnn {cal} {H}{up,it}{#1}
3002 \@@_set_mathalphabet_pos:nnnn {cal} {I}{up,it}{#1}
3003 \@@_set_mathalphabet_pos:nnnn {cal} {L}{up,it}{#1}
3004 \@@_set_mathalphabet_pos:nnnn {cal} {M}{up,it}{#1}

```



```

3005 \@@_set_mathalphabet_pos:nnnn {cal} {R}{up,it}{#1}
3006 }

```

N.3.5 *Fraktur or fraktur or blackletter: frak*

```

3007 \@@_new_alphabet_config:nnn {frak} {Latin}
3008 {
3009 \@@_set_mathalphabet_Latin:nnn {frak} {up,it}{#1}
3010 \@@_set_mathalphabet_pos:nnnn {frak} {C}{up,it}{#1}
3011 \@@_set_mathalphabet_pos:nnnn {frak} {H}{up,it}{#1}
3012 \@@_set_mathalphabet_pos:nnnn {frak} {I}{up,it}{#1}
3013 \@@_set_mathalphabet_pos:nnnn {frak} {R}{up,it}{#1}
3014 \@@_set_mathalphabet_pos:nnnn {frak} {Z}{up,it}{#1}
3015 }
3016 \@@_new_alphabet_config:nnn {frak} {latin}
3017 {
3018 \@@_set_mathalphabet_latin:nnn {frak} {up,it}{#1}
3019 }

```

N.3.6 *Sans serif upright: sfup*

```

3020 \@@_new_alphabet_config:nnn {sfup} {num}
3021 {
3022 \@@_set_mathalphabet_numbers:nnn {sf} {up}{#1}
3023 \@@_set_mathalphabet_numbers:nnn {sfup} {up}{#1}
3024 }
3025 \@@_new_alphabet_config:nnn {sfup} {Latin}
3026 {
3027 \bool_if:NTF \g_@@_sfliteral_bool
3028 {
3029 \@@_set_normal_Latin:nn {sfup} {#1}
3030 \@@_set_mathalphabet_Latin:nnn {sf} {up}{#1}
3031 }
3032 {
3033 \bool_if:NT \g_@@_upsans_bool
3034 {
3035 \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3036 \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3037 }
3038 }
3039 \@@_set_mathalphabet_Latin:nnn {sfup} {up,it}{#1}
3040 }
3041 \@@_new_alphabet_config:nnn {sfup} {latin}
3042 {
3043 \bool_if:NTF \g_@@_sfliteral_bool
3044 {
3045 \@@_set_normal_latin:nn {sfup} {#1}
3046 \@@_set_mathalphabet_latin:nnn {sf} {up}{#1}
3047 }
3048 {
3049 \bool_if:NT \g_@@_upsans_bool

```

```

3050     {
3051       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3052       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
3053     }
3054   }
3055   \@@_set_mathalphabet_latin:nnn {sfup} {up,it}{#1}
3056 }

```

N.3.7 *Sans serif italic: sfit*

```

3057 \@@_new_alphabet_config:nnn {sfit} {Latin}
3058 {
3059   \bool_if:NTF \g_@@_sfliteral_bool
3060   {
3061     \@@_set_normal_Latin:nn {sfit} {#1}
3062     \@@_set_mathalphabet_Latin:nnn {sf} {it}{#1}
3063   }
3064   {
3065     \bool_if:NF \g_@@_upsans_bool
3066     {
3067       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3068       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3069     }
3070   }
3071   \@@_set_mathalphabet_Latin:nnn {sfit} {up,it}{#1}
3072 }
3073 \@@_new_alphabet_config:nnn {sfit} {latin}
3074 {
3075   \bool_if:NTF \g_@@_sfliteral_bool
3076   {
3077     \@@_set_normal_latin:nn {sfit} {#1}
3078     \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
3079   }
3080   {
3081     \bool_if:NF \g_@@_upsans_bool
3082     {
3083       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3084       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
3085     }
3086   }
3087   \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
3088 }

```

N.3.8 *Typewriter or monospaced: tt*

```

3089 \@@_new_alphabet_config:nnn {tt} {num}
3090 {
3091   \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
3092 }
3093 \@@_new_alphabet_config:nnn {tt} {Latin}
3094 {

```

```

3095 \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
3096 }
3097 \@@_new_alphabet_config:nnn {tt} {latin}
3098 {
3099   \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
3100 }

```

N.3.9 *Bold Italic: bfit*

```

3101 \@@_new_alphabet_config:nnn {bfit} {Latin}
3102 {
3103   \bool_if:NF \g_@@_bfupLatin_bool
3104   {
3105     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3106   }
3107   \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
3108   \bool_if:NTF \g_@@_bfliteral_bool
3109   {
3110     \@@_set_normal_Latin:nn {bfit} {#1}
3111     \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
3112   }
3113   {
3114     \bool_if:NF \g_@@_bfupLatin_bool
3115     {
3116       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3117       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3118     }
3119   }
3120 }
3121
3122 \@@_new_alphabet_config:nnn {bfit} {latin}
3123 {
3124   \bool_if:NF \g_@@_bfuplatin_bool
3125   {
3126     \@@_set_normal_latin:nn {bfup,bfit} {#1}
3127   }
3128   \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
3129   \bool_if:NTF \g_@@_bfliteral_bool
3130   {
3131     \@@_set_normal_latin:nn {bfit} {#1}
3132     \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
3133   }
3134   {
3135     \bool_if:NF \g_@@_bfuplatin_bool
3136     {
3137       \@@_set_normal_latin:nn {bfup,bfit} {#1}
3138       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3139     }
3140   }
3141 }

```

```

3142
3143 \@@_new_alphabet_config:nnn {bfit} {Greek}
3144 {
3145   \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
3146   \bool_if:NTF \g_@@_bfliteral_bool
3147   {
3148     \@@_set_normal_Greek:nn {bfit}{#1}
3149     \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
3150   }
3151   {
3152     \bool_if:NF \g_@@_bfupGreek_bool
3153     {
3154       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3155       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3156     }
3157   }
3158 }
3159
3160 \@@_new_alphabet_config:nnn {bfit} {greek}
3161 {
3162   \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
3163   \bool_if:NTF \g_@@_bfliteral_bool
3164   {
3165     \@@_set_normal_greek:nn {bfit} {#1}
3166     \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
3167   }
3168   {
3169     \bool_if:NF \g_@@_bfupgreek_bool
3170     {
3171       \@@_set_normal_greek:nn {bfit,bfup} {#1}
3172       \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3173     }
3174   }
3175 }
3176
3177 \@@_new_alphabet_config:nnn {bfit} {misc}
3178 {
3179   \bool_if:NTF \g_@@_literal_Nabla_bool
3180   { \@@_set_normal_char:nnn {Nabla}{bfit}{#1} }
3181   {
3182     \bool_if:NF \g_@@_upNabla_bool
3183     { \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1} }
3184   }
3185   \bool_if:NTF \g_@@_literal_partial_bool
3186   { \@@_set_normal_char:nnn {partial}{bfit}{#1} }
3187   {
3188     \bool_if:NF \g_@@_uppartial_bool
3189     { \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1} }
3190   }

```

```

3191 \@@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it}{#1}
3192 \@@_set_mathalphabet_pos:nnnn {bfit} {Nabla} {up,it}{#1}
3193 \bool_if:NTF \g_@@_literal_partial_bool
3194 {
3195   \@@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
3196 }
3197 {
3198   \bool_if:NF \g_@@_uppartial_bool
3199   {
3200     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3201   }
3202 }
3203 \bool_if:NTF \g_@@_literal_Nabla_bool
3204 {
3205   \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {it}{#1}
3206 }
3207 {
3208   \bool_if:NF \g_@@_upNabla_bool
3209   {
3210     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3211   }
3212 }
3213 }

```

N.3.10 *Bold Upright: bfup*

```

3214 \@@_new_alphabet_config:nnn {bfup} {num}
3215 {
3216   \@@_set_mathalphabet_numbers:nnn {bf} {up}{#1}
3217   \@@_set_mathalphabet_numbers:nnn {bfup} {up}{#1}
3218 }
3219
3220 \@@_new_alphabet_config:nnn {bfup} {Latin}
3221 {
3222   \bool_if:NT \g_@@_bfupLatin_bool
3223   {
3224     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3225   }
3226   \@@_set_mathalphabet_Latin:nnn {bfup} {up,it}{#1}
3227   \bool_if:NTF \g_@@_bfliteral_bool
3228   {
3229     \@@_set_normal_Latin:nn {bfup} {#1}
3230     \@@_set_mathalphabet_Latin:nnn {bf} {up}{#1}
3231   }
3232   {
3233     \bool_if:NT \g_@@_bfupLatin_bool
3234     {
3235       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3236       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3237     }

```

```

3238     }
3239 }
3240
3241 \@@_new_alphabet_config:nnn {bfup} {latin}
3242 {
3243   \bool_if:NT \g_@@_bfuplatin_bool
3244   {
3245     \@@_set_normal_latin:nn {bfup,bfit} {#1}
3246   }
3247   \@@_set_mathalphabet_latin:nnn {bfup} {up,it}{#1}
3248   \bool_if:NTF \g_@@_bfliteral_bool
3249   {
3250     \@@_set_normal_latin:nn {bfup} {#1}
3251     \@@_set_mathalphabet_latin:nnn {bf} {up}{#1}
3252   }
3253   {
3254     \bool_if:NT \g_@@_bfuplatin_bool
3255     {
3256       \@@_set_normal_latin:nn {bfup,bfit} {#1}
3257       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3258     }
3259   }
3260 }
3261 \@@_new_alphabet_config:nnn {bfup} {Greek}
3262 {
3263   \@@_set_mathalphabet_Greek:nnn {bfup} {up,it}{#1}
3264   \bool_if:NTF \g_@@_bfliteral_bool
3265   {
3266     \@@_set_normal_Greek:nn {bfup}{#1}
3267     \@@_set_mathalphabet_Greek:nnn {bf} {up}{#1}
3268   }
3269   {
3270     \bool_if:NT \g_@@_bfupGreek_bool
3271     {
3272       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3273       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3274     }
3275   }
3276 }
3277
3278 \@@_new_alphabet_config:nnn {bfup} {greek}
3279 {
3280   \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
3281   \bool_if:NTF \g_@@_bfliteral_bool
3282   {
3283     \@@_set_normal_greek:nn {bfup} {#1}
3284     \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
3285   }
3286   {

```

```

3287 \bool_if:NT \g_@@_bfupgreek_bool
3288 {
3289 \@@_set_normal_greek:nn {bfup,bfit} {#1}
3290 \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3291 }
3292 }
3293 }
3294
3295 \@@_new_alphabet_config:nnn {bfup} {misc}
3296 {
3297 \bool_if:NTF \g_@@_literal_Nabla_bool
3298 {
3299 \@@_set_normal_char:nnn {Nabla}{bfup}{#1}
3300 }
3301 {
3302 \bool_if:NT \g_@@_upNabla_bool
3303 {
3304 \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1}
3305 }
3306 }
3307 \bool_if:NTF \g_@@_literal_partial_bool
3308 {
3309 \@@_set_normal_char:nnn {partial}{bfup}{#1}
3310 }
3311 {
3312 \bool_if:NT \g_@@_uppartial_bool
3313 {
3314 \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1}
3315 }
3316 }
3317 \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it}{#1}
3318 \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla} {up,it}{#1}
3319 \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up}{#1}
3320 \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up}{#1}
3321 \@@_set_mathalphabet_pos:nnnn {bf} {digamma} {up}{#1}
3322 \@@_set_mathalphabet_pos:nnnn {bf} {Digamma} {up}{#1}
3323 \bool_if:NTF \g_@@_literal_partial_bool
3324 {
3325 \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up}{#1}
3326 }
3327 {
3328 \bool_if:NT \g_@@_uppartial_bool
3329 {
3330 \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3331 }
3332 }
3333 \bool_if:NTF \g_@@_literal_Nabla_bool
3334 {
3335 \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}

```

```

3336 }
3337 {
3338   \bool_if:NT \g_@@_upNabla_bool
3339   {
3340     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3341   }
3342 }
3343 }

```

N.3.11 *Bold fractur or fraktur or blackletter: bffrak*

```

3344 \@@_new_alphabet_config:nnn {bffrak} {Latin}
3345 {
3346   \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
3347 }
3348
3349 \@@_new_alphabet_config:nnn {bffrak} {latin}
3350 {
3351   \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
3352 }

```

N.3.12 *Bold script or calligraphic: bfscr*

```

3353 \@@_new_alphabet_config:nnn {bfscr} {Latin}
3354 {
3355   \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
3356 }
3357 \@@_new_alphabet_config:nnn {bfscr} {latin}
3358 {
3359   \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
3360 }
3361 \@@_new_alphabet_config:nnn {bfcal} {Latin}
3362 {
3363   \@@_set_mathalphabet_Latin:nnn {bfcal} {up,it}{#1}
3364 }

```

N.3.13 *Bold upright sans serif: bfsfup*

```

3365 \@@_new_alphabet_config:nnn {bfsfup} {num}
3366 {
3367   \@@_set_mathalphabet_numbers:nnn {bfsf} {up}{#1}
3368   \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
3369 }
3370 \@@_new_alphabet_config:nnn {bfsfup} {Latin}
3371 {
3372   \bool_if:NTF \g_@@_sfliteral_bool
3373   {
3374     \@@_set_normal_Latin:nn {bfsfup} {#1}
3375     \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
3376   }
3377   {
3378     \bool_if:NT \g_@@_upsans_bool

```



```

3379     {
3380         \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
3381         \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
3382     }
3383 }
3384 \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
3385 }
3386
3387 \@@_new_alphabet_config:nnn {bfsfup} {latin}
3388 {
3389     \bool_if:NTF \g_@@_sfliteral_bool
3390     {
3391         \@@_set_normal_latin:nn {bfsfup} {#1}
3392         \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
3393     }
3394     {
3395         \bool_if:NT \g_@@_upsans_bool
3396         {
3397             \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
3398             \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
3399         }
3400     }
3401     \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
3402 }
3403
3404 \@@_new_alphabet_config:nnn {bfsfup} {Greek}
3405 {
3406     \bool_if:NTF \g_@@_sfliteral_bool
3407     {
3408         \@@_set_normal_Greek:nn {bfsfup}{#1}
3409         \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
3410     }
3411     {
3412         \bool_if:NT \g_@@_upsans_bool
3413         {
3414             \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
3415             \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3416         }
3417     }
3418     \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
3419 }
3420
3421 \@@_new_alphabet_config:nnn {bfsfup} {greek}
3422 {
3423     \bool_if:NTF \g_@@_sfliteral_bool
3424     {
3425         \@@_set_normal_greek:nn {bfsfup} {#1}
3426         \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
3427     }

```

```

3428 {
3429   \bool_if:NT \g_@@_upsans_bool
3430   {
3431     \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3432     \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3433   }
3434 }
3435 \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
3436 }
3437 \@@_new_alphabet_config:nnn {bfsfup} {misc}
3438 {
3439   \bool_if:NTF \g_@@_literal_Nabla_bool
3440   {
3441     \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
3442   }
3443   {
3444     \bool_if:NT \g_@@_upNabla_bool
3445     {
3446       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3447     }
3448   }
3449   \bool_if:NTF \g_@@_literal_partial_bool
3450   {
3451     \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
3452   }
3453   {
3454     \bool_if:NT \g_@@_uppartial_bool
3455     {
3456       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3457     }
3458   }
3459   \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
3460   \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla} {up,it}{#1}
3461   \bool_if:NTF \g_@@_literal_partial_bool
3462   {
3463     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
3464   }
3465   {
3466     \bool_if:NT \g_@@_uppartial_bool
3467     {
3468       \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3469     }
3470   }
3471   \bool_if:NTF \g_@@_literal_Nabla_bool
3472   {
3473     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up}{#1}
3474   }
3475   {
3476     \bool_if:NT \g_@@_upNabla_bool

```

```

3477 {
3478   \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3479 }
3480 }
3481 }

```

N.3.14 *Bold italic sans serif: bfsfit*

```

3482 \@@_new_alphabet_config:nnn {bfsfit} {Latin}
3483 {
3484   \bool_if:NTF \g_@@_sfliteral_bool
3485   {
3486     \@@_set_normal_Latin:nn {bfsfit} {#1}
3487     \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}
3488   }
3489   {
3490     \bool_if:NF \g_@@_upsans_bool
3491     {
3492       \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
3493       \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
3494     }
3495   }
3496   \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}
3497 }
3498
3499 \@@_new_alphabet_config:nnn {bfsfit} {latin}
3500 {
3501   \bool_if:NTF \g_@@_sfliteral_bool
3502   {
3503     \@@_set_normal_latin:nn {bfsfit} {#1}
3504     \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}
3505   }
3506   {
3507     \bool_if:NF \g_@@_upsans_bool
3508     {
3509       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
3510       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
3511     }
3512   }
3513   \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}
3514 }
3515
3516 \@@_new_alphabet_config:nnn {bfsfit} {Greek}
3517 {
3518   \bool_if:NTF \g_@@_sfliteral_bool
3519   {
3520     \@@_set_normal_Greek:nn {bfsfit}{#1}
3521     \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}
3522   }
3523   {

```

```

3524 \bool_if:NF \g_@@_upsans_bool
3525 {
3526   \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
3527   \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3528 }
3529 }
3530 \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
3531 }
3532
3533 \@@_new_alphabet_config:nnn {bfsfit} {greek}
3534 {
3535   \bool_if:NTF \g_@@_sfliteral_bool
3536   {
3537     \@@_set_normal_greek:nn {bfsfit} {#1}
3538     \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
3539   }
3540   {
3541     \bool_if:NF \g_@@_upsans_bool
3542     {
3543       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3544       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3545     }
3546   }
3547   \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
3548 }
3549
3550 \@@_new_alphabet_config:nnn {bfsfit} {misc}
3551 {
3552   \bool_if:NTF \g_@@_literal_Nabla_bool
3553   {
3554     \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
3555   }
3556   {
3557     \bool_if:NF \g_@@_upNabla_bool
3558     {
3559       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3560     }
3561   }
3562   \bool_if:NTF \g_@@_literal_partial_bool
3563   {
3564     \@@_set_normal_char:nnn {partial}{bfsfit}{#1}
3565   }
3566   {
3567     \bool_if:NF \g_@@_uppartial_bool
3568     {
3569       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3570     }
3571   }
3572   \@@_set_mathalphabet_pos:nnnn {bfsfit} {partial} {up,it}{#1}

```

```

3573 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3574 \bool_if:NTF \g_@@_literal_partial_bool
3575 {
3576   \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {it}{#1}
3577 }
3578 {
3579   \bool_if:NF \g_@@_uppartial_bool
3580   {
3581     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3582   }
3583 }
3584 \bool_if:NTF \g_@@_literal_Nabla_bool
3585 {
3586   \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {it}{#1}
3587 }
3588 {
3589   \bool_if:NF \g_@@_upNabla_bool
3590   {
3591     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3592   }
3593 }
3594 }
3595 </alphabets>

```

N.4 Compatibility

```

3596 <*compat>

```

`\@@_check_and_fix:NNnnnn` #1 : command
 #2 : factory command
 #3 : parameter text
 #4 : expected replacement text
 #5 : new replacement text for Lua \TeX
 #6 : new replacement text for X \TeX

Tries to patch $\langle command \rangle$. If $\langle command \rangle$ is undefined, do nothing. Otherwise it must be a macro with the given $\langle parameter text \rangle$ and $\langle expected replacement text \rangle$, created by the given $\langle factory command \rangle$ or equivalent. In this case it will be overwritten using the $\langle parameter text \rangle$ and the $\langle new replacement text for Lua\TeX \rangle$ or the $\langle new replacement text for X\TeX \rangle$, depending on the engine. Otherwise issue a warning and don't overwrite.

```

3597 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnnn
3598 {
3599   \cs_if_exist:NT #1
3600   {
3601     \token_if_macro:NTF #1
3602     {
3603       \group_begin:
3604       #2 \@@_tmpa:w #3 { #4 }
3605       \cs_if_eq:NNTF #1 \@@_tmpa:w

```

```

3606 {
3607   \msg_info:nxx { unicode-math } { patch-macro }
3608   { \token_to_str:N #1 }
3609   \group_end:
3610   #2 #1 #3
3611   (XE) { #6 }
3612   (LU) { #5 }
3613 }
3614 {
3615   \msg_warning:nxxxx { unicode-math } { wrong-meaning }
3616   { \token_to_str:N #1 } { \token_to_meaning:N #1 }
3617   { \token_to_meaning:N \@@_tmpa:w }
3618   \group_end:
3619 }
3620 }
3621 {
3622   \msg_warning:nxx { unicode-math } { macro-expected }
3623   { \token_to_str:N #1 }
3624 }
3625 }
3626 }

```

`\@@_check_and_fix:NNnnn` #1 : command
 #2 : factory command
 #3 : parameter text
 #4 : expected replacement text
 #5 : new replacement text

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

```

3627 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
3628 {
3629   \@@_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
3630 }

```

`\@@_check_and_fix luatex:NNnnn` #1 : command
`\@@_check_and_fix luatex:cNnnn` #2 : factory command
 #3 : parameter text
 #4 : expected replacement text
 #5 : new replacement text

Tries to patch $\langle command \rangle$. If Xe_{La}TeX is the current engine or $\langle command \rangle$ is undefined, do nothing. Otherwise it must be a macro with the given $\langle parameter text \rangle$ and $\langle expected replacement text \rangle$, created by the given $\langle factory command \rangle$ or equivalent. In this case it will be overwritten using the $\langle parameter text \rangle$ and the $\langle new replacement text \rangle$. Otherwise issue a warning and don't overwrite.

```

3631 \cs_new_protected_nopar:Nn \@@_check_and_fix luatex:NNnnn

```

```

3632 {
3633 (LU) \@@_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
3634 }
3635 \cs_generate_variant:Nn \@@_check_and_fix_luatex:NNnnn { c }

```

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

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

```

3636 \AtEndOfPackageFile * {url}
3637 {
3638 \tl_put_left:Nn \Url@FormatString { \@@_switchto_up: }
3639 \tl_put_right:Nn \Url@Specials
3640 {
3641 \do\{\mathchar`\}
3642 \do\'\mathchar`\'
3643 \do\$\mathchar`\$
3644 \do\&\mathchar`\&
3645 }
3646 }

```

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:

```

3647 \AtEndOfPackageFile * {amsmath}
3648 {
3649 (*XE)
3650 \tl_remove_once:Nn \@begindocumenthook
3651 {
3652 \mathchardef\std@minus\mathcode`\-\relax
3653 \mathchardef\std@equal\mathcode`\=\relax
3654 }
3655 \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
3656 \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
3657 (/XE)
3658 \cs_set:Npn \@cdots {\mathinner{\cdots}}
3659 \cs_set_eq:NN \dotso@ \cdots

```

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

```

3660 (*XE)
3661 \def \resetMathstrut@
3662 {%
3663 \setbox\z@\hbox{${}$}
3664 \ht\Mathstrutbox@ \ht\z@ \dp\Mathstrutbox@ \dp\z@
3665 }

```

The subarray environment uses inappropriate font dimensions.

```

3666 \@@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
3667 {
3668   \vcenter
3669   \bgroup
3670   \Let@
3671   \restore@math@cr
3672   \default@tag
3673   \baselineskip \fontdimen 10~ \scriptfont \tw@
3674   \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
3675   \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
3676   \lineskiplimit \lineskip
3677   \ialign
3678   \bgroup
3679   \ifx c #1 \hfil \fi
3680   $ \m@th \scriptstyle ## $
3681   \hfil
3682   \crr
3683 }
3684 {
3685   \vcenter
3686   \c_group_begin_token
3687   \Let@
3688   \restore@math@cr
3689   \default@tag
3690   \skip_set:Nn \baselineskip
3691   {

```

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

```

3692 \@@_stack_num_up:N \scriptstyle
3693 + \@@_stack_denom_down:N \scriptstyle
3694 }

```

Here we use the minimum stack gap.

```

3695 \lineskip \@@_stack_vgap:N \scriptstyle
3696 \lineskiplimit \lineskip
3697 \ialign
3698 \c_group_begin_token
3699 \token_if_eq_meaning:NNT c #1 { \hfil }
3700 \c_math_toggle_token
3701 \m@th
3702 \scriptstyle
3703 \c_parameter_token \c_parameter_token
3704 \c_math_toggle_token
3705 \hfil
3706 \crr
3707 }
3708 </XE>

```

The roots need a complete rework.


```

3709 \@@_check_and_fix luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
3710 {
3711   \setbox \rootbox \hbox
3712   {
3713     $ \math \scriptscriptstyle { #1 } $
3714   }
3715   \mathchoice
3716     { \r@@@t \displaystyle { #2 } }
3717     { \r@@@t \textstyle { #2 } }~
3718     { \r@@@t \scriptstyle { #2 } }
3719     { \r@@@t \scriptscriptstyle { #2 } }
3720   \egroup
3721 }
3722 {
3723   \bool_if:nTF
3724   {
3725     \int_compare_p:nNn { \uproot@ } = { \c_zero }
3726     && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
3727   }
3728   {
3729     \Uroot \l_@@_radical_sqrt_tl { #1 } { #2 }
3730   }
3731   {
3732     \hbox_set:Nn \rootbox
3733     {
3734       \c_math_toggle_token
3735       \math
3736       \scriptscriptstyle { #1 }
3737       \c_math_toggle_token
3738     }
3739     \mathchoice
3740       { \r@@@t \displaystyle { #2 } }
3741       { \r@@@t \textstyle { #2 } }
3742       { \r@@@t \scriptstyle { #2 } }
3743       { \r@@@t \scriptscriptstyle { #2 } }
3744   }
3745   \c_group_end_token
3746 }
3747 \@@_check_and_fix:NNnnnn \r@@@t \cs_set_nopar:Npn { #1 #2 }
3748 {
3749   \setboxz@h { $ \math #1 \sqrt{sign { #2 } } $ }
3750   \dimen@ \ht\z@
3751   \advance \dimen@ -\dp\z@
3752   \setbox\@ne \hbox { $ \math #1 \mskip \uproot@ mu $ }
3753   \advance \dimen@ by 1.667 \wd\@ne
3754   \mkern -\leftroot@ mu
3755   \mkern 5mu
3756   \raise .6\dimen@ \copy\rootbox
3757   \mkern -10mu

```

```

3758 \mkern \leftroot@ mu
3759 \boxz@
3760 }
3761 {
3762 \hbox_set:Nn \l_tmpa_box
3763 {
3764 \c_math_toggle_token
3765 \m@th
3766 #1
3767 \mskip \uproot@ mu
3768 \c_math_toggle_token
3769 }
3770 \Uroot \l_@@_radical_sqrt_tl
3771 {
3772 \box_move_up:nn { \box_wd:N \l_tmpa_box }
3773 {
3774 \hbox:n
3775 {
3776 \c_math_toggle_token
3777 \m@th
3778 \mkern -\leftroot@ mu
3779 \box_use:N \rootbox
3780 \mkern \leftroot@ mu
3781 \c_math_toggle_token
3782 }
3783 }
3784 }
3785 { #2 }
3786 }
3787 {
3788 \hbox_set:Nn \l_tmpa_box
3789 {
3790 \c_math_toggle_token
3791 \m@th
3792 #1
3793 \sqrtsign { #2 }
3794 \c_math_toggle_token
3795 }
3796 \hbox_set:Nn \l_tmpb_box
3797 {
3798 \c_math_toggle_token
3799 \m@th
3800 #1
3801 \mskip \uproot@ mu
3802 \c_math_toggle_token
3803 }
3804 \mkern -\leftroot@ mu
3805 \@@_mathstyle_scale:Nnn #1 { \kern }
3806 {

```

```

3807     \fontdimen 63 \l_@@_font
3808   }
3809   \box_move_up:nn
3810   {
3811     \box_wd:N \l_tmpb_box
3812     + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
3813     * \number \fontdimen 65 \l_@@_font / 100
3814   }
3815   {
3816     \box_use:N \rootbox
3817   }
3818   \@@_mathstyle_scale:Nnn #1 { \kern }
3819   {
3820     \fontdimen 64 \l_@@_font
3821   }
3822   \mkern \leftroot@ mu
3823   \box_use_clear:N \l_tmpa_box
3824 }
3825 }

```

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

```

3826 (*XE)
3827 \AtEndOfPackageFile * {amsopn}
3828 {
3829   \cs_set:Npn \newmcodes@
3830   {
3831     \mathcode'\ '39\scan_stop:
3832     \mathcode'\ *42\scan_stop:
3833     \mathcode'\ ."613A\scan_stop:
3834     %% \ifnum\mathcode'\ -=45 \else
3835     %%   \mathchardef\std@minus\mathcode'\ -\relax
3836     %% \fi
3837     \mathcode'\ -45\scan_stop:
3838     \mathcode'\ /47\scan_stop:
3839     \mathcode'\ ":"603A\scan_stop:
3840   }
3841 }
3842 (/XE)

```

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

```

3843 \AtEndOfPackageFile * { mathtools }
3844 {
3845 (*XE)
3846   \newfam \g_@@_empty_fam
3847   \@@_check_and_fix:NNnnn
3848     \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }

```

```

3849 {
3850   \sbox \z@
3851   {
3852     $
3853     \math
3854     #1
3855     \nulldelimiterspace = \z@
3856     \radical \z@ { #2 }
3857     $
3858   }
3859   \ifx #1 \displaystyle
3860     \dimen@ = \fontdimen 8 \textfont 3
3861     \advance \dimen@ .25 \fontdimen 5 \textfont 2
3862   \else
3863     \dimen@ = 1.25 \fontdimen 8
3864     \ifx #1 \textstyle
3865       \textfont
3866     \else
3867       \ifx #1 \scriptstyle
3868         \scriptfont
3869       \else
3870         \scriptscriptfont
3871       \fi
3872     \fi
3873     3
3874     \fi
3875     \advance \dimen@ -\ht\z@
3876     \ht\z@ = -\dimen@
3877     \box\z@
3878   }

```

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

```

3879 {
3880   \hbox_set:Nn \l_tmpa_box
3881   {
3882     \color@setgroup
3883     \c_math_toggle_token
3884     \math
3885     #1
3886     \dim_zero:N \nulldelimiterspace
3887     \XeTeXradical \g_@@_empty_fam \c_zero { #2 }
3888     \c_math_toggle_token
3889     \color@endgroup
3890   }
3891   \box_set_ht:Nn \l_tmpa_box
3892   {
3893     \box_ht:N \l_tmpa_box

```

Here we use the radical vertical gap.

```

3894         - \@@_radical_vgap:N #1
3895     }
3896     \box_use_clear:N \l_tmpa_box
3897 }
3898 (/XE)

```

`\overbracket` and `\underbracket` take optional arguments and are defined in terms of rules, so we keep them, and rename ours to `\Uoverbracket` and `\Underbracket`.

```

3899 \AtEndOfPackageFile * { mathtools }
3900 {
3901     \cs_set_eq:NN \MToverbracket \overbracket
3902     \cs_set_eq:NN \MTunderbracket \underbracket
3903
3904     \AtBeginDocument
3905     {
3906         \msg_warning:nn { unicode-math } { mathtools-overbracket }
3907
3908     \def\downbracketfill#1#2
3909     {%

```

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

```

3910         \edef\l_MT_bracketheight_fdim{.27ex}%
3911         \downbracketend{#1}{#2}
3912         \leaders \vrule \@height #1 \@depth \z@ \hfill
3913         \downbracketend{#1}{#2}%
3914     }
3915 \def\upbracketfill#1#2
3916 {%
3917         \edef\l_MT_bracketheight_fdim{.27ex}%
3918         \upbracketend{#1}{#2}
3919         \leaders \vrule \@height \z@ \@depth #1 \hfill
3920         \upbracketend{#1}{#2}%
3921     }
3922 \let\Uoverbracket =\overbracket
3923 \let\Underbracket=\underbracket
3924     \let\overbracket =\MToverbracket
3925     \let\underbracket =\MTunderbracket
3926 }
3927 }

```

`\dblcolon`, `\coloneqq`, `\Coloneqq`, and `\eqqcolon` are defined by mathtools as combinations of colons and other characters, but with meanings incompatible to unicode-math. Thus we issue a warning. Because mathtools uses `\providecommand` `\AtBeginDocument`, we can just define the offending commands here.

```

3928 \msg_warning:nn { unicode-math } { mathtools-colon }

```

```

3929 \NewDocumentCommand \dblcolon { } { \Colon }
3930 \NewDocumentCommand \coloneqq { } { \coloneq }
3931 \NewDocumentCommand \Coloneqq { } { \Coloneq }
3932 \NewDocumentCommand \eqqcolon { } { \eqcolon }
3933 }

```

colonequals

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

```

\coloncolon 3934 \AtEndOfPackageFile * { colonequals }
\minuscolon 3935 {
\colonequals 3936 \msg_warning:nn { unicode-math } { colonequals }
\equalscolon 3937 \RenewDocumentCommand \ratio { } { \mathratio }
\coloncolonequals 3938 \RenewDocumentCommand \coloncolon { } { \Colon }
3939 \RenewDocumentCommand \minuscolon { } { \dashcolon }
3940 \RenewDocumentCommand \colonequals { } { \coloneq }
3941 \RenewDocumentCommand \equalscolon { } { \eqcolon }
3942 \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
3943 }
3944 </compat>

```