

Experimental Unicode mathematical typesetting: The unicode-math package

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

2014/07/30 v0.8

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 | 6 |
| 4.3 | Maths ‘versions’ | 7 |
| 4.4 | Legacy maths ‘alphabet’ commands | 7 |
| 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 | 11 |
| 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 (usage: `\symbfsf{g}` → **g**) to select specific ‘alphabets’ within the unicode maths font:

```
\symnormal \symliteral \symup \symbfup \symbfit \symsfup \symsfit  
\symbfsfup \symbfsfit \symbfsf \sybbb \sybbbfit \symscr \symbfscr  
\symcal \symbfcal \symfrak \symbffrak \symup \symsf \symbf \symtt  
\symit
```

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

```
\mathnormal \mathbb \mathbbfit \mathscr \mathbfscr \mathcal \mathbfcal  
\mathfrak \mathbffrak \mathbfup \mathbfit \mathsfup \mathsfit \mathbfsfup  
\mathbfsfit \mathbfsf
```

So what about `\mathup`, `\mathit`, `\mathbf`, `\mathsf`, and `\mathtt`? (N.B.: `\mathrm` is defined as a synonym for `\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., `\symbf` and `\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 `\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 `\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 `math-style` will not take effect until a new maths font is set up.

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

```
\unimathsetup{math-style=TeX}  
\setmathfont{Cambria Math}  
% OR  
\setmathfont{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., \mathscr{H} to \mathcal{H} and so on, all for Unicode glyphs within a single font.

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

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

4.1 Using multiple fonts

There will probably be few cases where a single Unicode maths font suffices (simply due to glyph coverage). The `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`, `bfcalf`,
`frak`, `bffrak`, `sfup`, `sfit`, `bfsup`, `bfsfit`, `bfsf`

Fonts can be selected for specified ranges only; use the following syntax:

- `[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).

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 fonts will possibly use entirely separate fonts.

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

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

4.3 Maths ‘versions’

L^AT_EX 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{<font name>}[version=<version name>,<font features>]
```

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`

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> |

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 `\symbf` 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 `\symbf` 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

\LaTeX 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\symbf
```

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

5 Maths input

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

5.1 Math ‘style’

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

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

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

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

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

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

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

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

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> | $(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$ | $(\alpha, \beta, \Gamma, \Xi)$ |

5.2 Bold style

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

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

In `unicode-math`, the `\mathbf{bf}` 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.

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

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

The `bold-style` options' effects are shown in brief in table 5.

5.3 Sans serif style

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

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

Therefore I reluctantly add the package options `[sans-style=upright]` and `[sans-style=italic]` to control the behaviour of `\mathsf`. The `upright` style sets up the command to use 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 `\mathsfit`, respectively. Please let me know if more granular control is necessary here.

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

5.3.1 What about bold sans serif?

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

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

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

5.4 All (the rest) of the mathematical 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 `\mathbfsf{...}` rather than `\mathbf{\mathsf{...}}`. 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 $\{0\text{--}z, \mathbb{A}\mathbb{Z}\}$, numerals $0\text{--}9$, summation symbol Σ , and four Greek letters only: $\{0, \pi, \Gamma, \Pi\}$.

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`.

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

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

There are also five Latin *italic* double-struck letters: $\mathbb{Ddei\j}$. 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 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={\mathcal,\mathbfcal},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: $\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{X}\mathcal{Y}\mathcal{Z}$

Table 7: The various forms of nabla.

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

5.5 Miscellanea

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 `\mathbf`; `\mathit` and `\mathup` can be used to force one way or the other.

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

5.5.2 Partial

The same applies to the symbols ∂ partial differential and ∂ math italic partial differential.

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

Use the `partial=upright` or `partial=italic` package options to specify which one you would like, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests and argues otherwise) `partial=italic`.¹ `partial=literal` is activated following `math-style=literal`.

See table 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.

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

Table 8: The various forms of the partial differential. Note that in the fonts used to display these glyphs, the first upright partial is incorrectly shown in an italic style.

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

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 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 back tick (`) or the Unicode reverse prime U+2035 ('). The command to access the back-prime is `\backprime`, and multiple backwards primes can accessed with `\backdprime`, `\backtrprime`, and `\backqprime`.

In all cases above, no error checking is performed if you attempt to access a multi-prime glyph in a font that doesn't contain one. For this reason, it may be

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

safer to write `x''''` instead of `x\qprime` in general.

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

5.5.5 Unicode subscripts and superscripts

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

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

5.5.6 Colon

The colon is one of the few confusing characters of Unicode maths. In $\mathrm{T}_{\mathrm{E}}\mathrm{X}$, `:` 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.

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 |

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

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

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

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

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

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

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

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

Finally, U+29F9 big reverse solidus is a ‘big operator’ (like Σ).

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

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

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

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

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`:

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

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

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

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

5.5.8 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 \XTeX and \LuaTeX did not support this distinction, however, and *all* accents there were growing automatically. (I.e., `\hat` and `\widehat` are equivalent.) As of \LuaTeX v0.65 and \XTeX 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.

$\frac{1}{4}$ $\frac{1}{2}$ $\frac{3}{4}$ $\frac{0}{3}$ $\frac{1}{7}$ $\frac{1}{9}$ $\frac{1}{10}$ $\frac{1}{3}$ $\frac{2}{3}$ $\frac{1}{5}$ $\frac{2}{5}$ $\frac{3}{5}$ $\frac{4}{5}$ $\frac{1}{6}$ $\frac{5}{6}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{5}{8}$ $\frac{7}{8}$

For example, instead of writing `\tfrac{12}{x}`, you may consider it more readable to have `\frac{12}{x}` 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.

| 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>\smbkcircle</code> | • | ◦ | <code>\smwhtcircle</code> | U+25E6 |
| U+2981 | <code>\mdsmbkcircle</code> | ● | ◦ | <code>\mdsmwhtcircle</code> | U+26AC |
| U+26AB | <code>\mdblkcircle</code> | ● | ◯ | <code>\mdwhtcircle</code> | U+26AA |
| U+25CF | <code>\mdlgbkcircle</code> | ● | ◯ | <code>\mdlgwhtcircle</code> | U+25CB |
| U+2B24 | <code>\lgblkcircle</code> | ● | ◯ | <code>\lgwhtcircle</code> | U+25EF |

Table 10: Filled and hollow Unicode circles.

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

Table 11: Different upwards pointing triangles.

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.

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

5.5.11 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.

These triangles all have different intended meanings. Note for backwards compatibility with T_EX, 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, `\mathbf { \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 $ABC-xyz, \mathfrak{ABC}-\mathcal{XYZ}$, etc.

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

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

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

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

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

`\SetSymbolFont{<name>}\{<maths version>\}\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 | 29 |
| E.1 Extras | 32 |
| E.2 Alphabet Unicode positions | 33 |
| E.3 Package options | 33 |
| E.4 Programmers' interface | 39 |
| F Bifurcation | 39 |
| F.1 Engine differences | 39 |
| F.2 Overcoming <code>\onlypreamble</code> | 39 |
| G Fundamentals | 40 |
| G.1 Setting math chars, math codes, etc. | 40 |
| G.2 <code>\setmathalphabet</code> | 43 |
| G.3 Hooks into <code>fontspec</code> | 44 |
| G.4 The main <code>\setmathfont</code> macro | 46 |
| G.5 (Big) operators | 54 |
| G.6 Radicals | 54 |
| G.7 Maths accents | 55 |
| G.8 Common interface for font parameters | 55 |
| H Font features | 60 |
| H.1 Math version | 60 |
| H.2 Script and scriptscript font options | 60 |
| H.3 Range processing | 60 |
| H.4 Resolving Greek symbol name control sequences | 64 |
| I Maths alphabets | 64 |
| I.1 Hooks into $\LaTeX 2_{\epsilon}$ | 65 |
| I.2 Setting styles | 65 |
| I.3 Defining the math style macros | 66 |
| I.4 Definition of alphabets and styles | 67 |
| I.5 Defining the math alphabets per style | 70 |
| I.6 Mapping 'naked' math characters | 73 |

| | | |
|----------|---|-----------|
| I.7 | Mapping chars inside a math style | 75 |
| J | A token list to contain the data of the math table | 78 |
| K | Definitions of the active math characters | 78 |
| L | Fall-back font | 80 |
| M | Epilogue | 80 |
| M.1 | Primes | 80 |
| M.2 | Unicode radicals | 87 |
| M.3 | Unicode sub- and super-scripts | 89 |
| M.4 | Synonyms and all the rest | 93 |
| N | Error messages | 96 |
| N.1 | Alphabet Unicode positions | 98 |
| N.2 | STIX fonts | 104 |
| N.3 | Alphabets | 107 |
| N.4 | Compatibility | 125 |

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

35 \ExplSyntaxOn

    For fontspec:

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

```

Conditionals

```

38 \bool_new:N \l_@@_ot_math_bool
39 \bool_new:N \l_@@_init_bool
40 \bool_new:N \l_@@_implicit_alph_bool
41 \bool_new:N \g_@@_mainfont_already_set_bool

```

For math-style:

```

42 \bool_new:N \g_@@_literal_bool
43 \bool_new:N \g_@@_upLatin_bool
44 \bool_new:N \g_@@_uplatin_bool
45 \bool_new:N \g_@@_upGreek_bool
46 \bool_new:N \g_@@_upgreek_bool

```

For bold-style:

```

47 \bool_new:N \g_@@_bfliteral_bool
48 \bool_new:N \g_@@_bfupLatin_bool
49 \bool_new:N \g_@@_bfuplatin_bool
50 \bool_new:N \g_@@_bfupGreek_bool
51 \bool_new:N \g_@@_bfupgreek_bool

```

For sans-style:

```

52 \bool_new:N \g_@@_upsans_bool
53 \bool_new:N \g_@@_sfliteral_bool

```

For assorted package options:

```

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

```

Variables

```

67 \int_new:N \g_@@_fam_int

```

For displaying in warning messages, etc.:

```

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

```

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

```

75 \tl_new:N \g_@@_operator_mathfont_tl

```

Variables:

```

76 \seq_new:N \l_@@_missing_alph_seq
77 \seq_new:N \l_@@_mathalph_seq
78 \seq_new:N \l_@@_char_range_seq
79 \seq_new:N \l_@@_mclass_range_seq
80 \seq_new:N \l_@@_cmd_range_seq

```

`\g_@@_mathclasses_seq` Every math class.

```

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

```

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

```

90 \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!

```

91 \seq_new:N \g_@@_named_ranges_seq
92 \seq_new:N \g_@@_mathstyles_seq

93 \muskip_new:N \g_@@_primekern_muskip
94 \muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }% arbitrary
95 \int_new:N \l_@@_primecount_int
96 \prop_new:N \g_@@_supers_prop
97 \prop_new:N \g_@@_subs_prop
98 \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!!

```

99 \prg_new_conditional:Nnn \@@_glyph_if_exist:n {p,TF,T,F}
100 {
101   \etex_iffontchar:D \l_@@_font #1 \scan_stop:
102   \prg_return_true:
103   \else:
104     \prg_return_false:
105   \fi:
106 }

```

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

```

\@@_set_mathcode:nnn 107 \cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4 {
\@@_set_mathchar:NNnn 108   \Umathcode \int_eval:n {#1} =
\@@_set_mathchar:cNnn 109   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
\@@_set_delcode:nnn 110 }
\@@_radical:nn 111 \cs_set:Npn \@@_set_mathcode:nnn #1#2#3 {
\@@_delimiter:Nnn 112   \Umathcode \int_eval:n {#1} =
\@@_accent:nnn 113   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
\@@_accent_keyword: 114 }
115 \cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4 {
116   \Umathchardef #1 =
117   \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
118 }
119 \cs_new:Nn \@@_set_delcode:nnn {
120   \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
121 }
122 \cs_new:Nn \@@_radical:nn {
123   \Uradical \csname sym#1\endcsname #2 \scan_stop:
124 }

```



```

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

```

\@@_char_gmake_mathactive:N

\@@_char_gmake_mathactive:n

```

132 \cs_new:Nn \@@_char_gmake_mathactive:N
133 {
134   \global\mathcode `#1 = "8000 \scan_stop:
135 }
136 \cs_new:Nn \@@_char_gmake_mathactive:n
137 {
138   \global\mathcode #1 = "8000 \scan_stop:
139 }

```

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.⁴

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

```

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

```

E.3 Package options

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

```

149 \DeclareDocumentCommand \unimathsetup {m}
150 { \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.

```

151 \cs_new:Nn \@@_keys_choices:nn
152 {
153   \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
154   \use:x

```

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

```

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

```

Compatibility

```

176 \@@_keys_choices:nn {mathup}
177 {
178   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
179   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
180 }
181 \@@_keys_choices:nn {mathrm}
182 {
183   {sym} { \bool_set_false:N \g_@@_mathrm_text_bool }
184   {text} { \bool_set_true:N \g_@@_mathrm_text_bool }
185 }
186 \@@_keys_choices:nn {mathit}
187 {
188   {sym} { \bool_set_false:N \g_@@_mathit_text_bool }
189   {text} { \bool_set_true:N \g_@@_mathit_text_bool }
190 }
191 \@@_keys_choices:nn {mathbf}
192 {
193   {sym} { \bool_set_false:N \g_@@_mathbf_text_bool }
194   {text} { \bool_set_true:N \g_@@_mathbf_text_bool }
195 }
196 \@@_keys_choices:nn {mathsf}
197 {
198   {sym} { \bool_set_false:N \g_@@_mathsf_text_bool }
199   {text} { \bool_set_true:N \g_@@_mathsf_text_bool }
200 }
201 \@@_keys_choices:nn {mathtt}

```

```

202 {
203   {sym} { \bool_set_false:N \g_@@_mathtt_text_bool }
204   {text} { \bool_set_true:N \g_@@_mathtt_text_bool }
205 }

```

math-style

```

206 \@@_keys_choices:nn {normal-style}
207 {
208   {ISO} {
209     \bool_set_false:N \g_@@_literal_bool
210     \bool_set_false:N \g_@@_upGreek_bool
211     \bool_set_false:N \g_@@_upgreek_bool
212     \bool_set_false:N \g_@@_upLatin_bool
213     \bool_set_false:N \g_@@_uplatin_bool
214   }
215   {TeX} {
216     \bool_set_false:N \g_@@_literal_bool
217     \bool_set_true:N \g_@@_upGreek_bool
218     \bool_set_false:N \g_@@_upgreek_bool
219     \bool_set_false:N \g_@@_upLatin_bool
220     \bool_set_false:N \g_@@_uplatin_bool
221   }
222   {french} {
223     \bool_set_false:N \g_@@_literal_bool
224     \bool_set_true:N \g_@@_upGreek_bool
225     \bool_set_true:N \g_@@_upgreek_bool
226     \bool_set_true:N \g_@@_upLatin_bool
227     \bool_set_false:N \g_@@_uplatin_bool
228   }
229   {upright} {
230     \bool_set_false:N \g_@@_literal_bool
231     \bool_set_true:N \g_@@_upGreek_bool
232     \bool_set_true:N \g_@@_upgreek_bool
233     \bool_set_true:N \g_@@_upLatin_bool
234     \bool_set_true:N \g_@@_uplatin_bool
235   }
236   {literal} {
237     \bool_set_true:N \g_@@_literal_bool
238   }
239 }
240 \@@_keys_choices:nn {math-style}
241 {
242   {ISO} {
243     \unimathsetup { nabla=upright, partial=italic,
244                   normal-style=ISO, bold-style=ISO, sans-style=italic }
245   }
246   {TeX} {
247     \unimathsetup { nabla=upright, partial=italic,

```

```

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

```

bold-style

```

263 \@@_keys_choices:nn {bold-style}
264 {
265     {ISO} {
266         \bool_set_false:N \g_@@_bfliteral_bool
267         \bool_set_false:N \g_@@_bfupGreek_bool
268         \bool_set_false:N \g_@@_bfupgreek_bool
269         \bool_set_false:N \g_@@_bfupLatin_bool
270         \bool_set_false:N \g_@@_bfuplatin_bool
271     }
272     {TeX} {
273         \bool_set_false:N \g_@@_bfliteral_bool
274         \bool_set_true:N \g_@@_bfupGreek_bool
275         \bool_set_false:N \g_@@_bfupgreek_bool
276         \bool_set_true:N \g_@@_bfupLatin_bool
277         \bool_set_true:N \g_@@_bfuplatin_bool
278     }
279     {upright} {
280         \bool_set_false:N \g_@@_bfliteral_bool
281         \bool_set_true:N \g_@@_bfupGreek_bool
282         \bool_set_true:N \g_@@_bfupgreek_bool
283         \bool_set_true:N \g_@@_bfupLatin_bool
284         \bool_set_true:N \g_@@_bfuplatin_bool
285     }
286     {literal} {
287         \bool_set_true:N \g_@@_bfliteral_bool
288     }
289 }

```

sans-style

```

290 \@@_keys_choices:nn {sans-style}
291 {

```

```

292 {italic} { \bool_set_false:N \g_@@_upsans_bool }
293 {upright} { \bool_set_true:N \g_@@_upsans_bool }
294 {literal} { \bool_set_true:N \g_@@_sfliteral_bool }
295 }

```

Nabla and partial

```

296 \@@_keys_choices:nn {nabla}
297 {
298   {upright} {
299     \bool_set_false:N \g_@@_literal_Nabla_bool
300     \bool_set_true:N \g_@@_upNabla_bool
301   }
302   {italic} {
303     \bool_set_false:N \g_@@_literal_Nabla_bool
304     \bool_set_false:N \g_@@_upNabla_bool
305   }
306   {literal} { \bool_set_true:N \g_@@_literal_Nabla_bool }
307 }
308 \@@_keys_choices:nn {partial}
309 {
310   {upright} {
311     \bool_set_false:N \g_@@_literal_partial_bool
312     \bool_set_true:N \g_@@_uppartial_bool
313   }
314   {italic} {
315     \bool_set_false:N \g_@@_literal_partial_bool
316     \bool_set_false:N \g_@@_uppartial_bool
317   }
318   {literal} { \bool_set_true:N \g_@@_literal_partial_bool }
319 }

```

Epsilon and phi shapes

```

320 \@@_keys_choices:nn {vargreek-shape}
321 {
322   {unicode} { \bool_set_false:N \g_@@_texgreek_bool }
323   {TeX} { \bool_set_true:N \g_@@_texgreek_bool }
324 }

```

Colon style

```

325 \@@_keys_choices:nn {colon}
326 {
327   {literal} { \bool_set_true:N \g_@@_literal_colon_bool }
328   {TeX} { \bool_set_false:N \g_@@_literal_colon_bool }
329 }

```

Slash delimiter style

```

330 \@@_keys_choices:nn {slash-delimiter}

```

```

331 {
332 {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
333 {frac} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
334 {div} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
335 }

```

Active fraction style

```

336 \@@_keys_choices:nn {active-frac}
337 {
338 {small}
339 {
340 \cs_if_exist:NTF \tfrac
341 { \bool_set_true:N \l_@@_smallfrac_bool }
342 {
343 \@@_warning:n {no-tfrac}
344 \bool_set_false:N \l_@@_smallfrac_bool
345 }
346 \use:c {@@_setup_active_frac:}
347 }
348
349 {normalsize}
350 {
351 \bool_set_false:N \l_@@_smallfrac_bool
352 \use:c {@@_setup_active_frac:}
353 }
354 }

```

Debug/tracing

```

355 \keys_define:nn {unicode-math}
356 {
357 warnings-off .code:n =
358 {
359 \clist_map_inline:nn {#1}
360 { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
361 }
362 }
363 \@@_keys_choices:nn {trace}
364 {
365 {on} {} % default
366 {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
367 {off} { \msg_redirect_module:nnn { unicode-math } { log } { none } }
368 }
369 \unimathsetup {math-style=TeX}
370 \unimathsetup {slash-delimiter=ascii}
371 \unimathsetup {trace=off}
372 \unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}
373 \cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
374 \ProcessKeysOptions {unicode-math}

```

E.4 Programmers' interface

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

```
375 \cs_new:Nn \unimath_get_mathstyle:
376 {
377   \tl_use:N \l_@@_mathstyle_tl
378 }
```

End of preamble code.

```
379 </preamble&!XE&!LU>
```

(Error messages and warning definitions go here from the `msg` chunk defined in section §N on page 96.)

F Bifurcation

And here the split begins. Most of the code is still shared, but code for Lua_T_EX uses the ‘LU’ flag and code for X_Y_T_EX uses ‘XE’.

```
380 <*package&(XE|LU)>
381 \ExplSyntaxOn
```

F.1 Engine differences

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

```
382 <*LU>
383 \RequirePackage{luaotfload} [2014/05/18]
384 \RequirePackage{lualatex-math}[2011/08/07]
385 </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.

```
386 \tl_map_inline:nn
387 {
388   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
389   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
390   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
391   \version@list\version@elt\alpha@list\alpha@elt
392   \restore@mathversion\init@restore@version\dorestore@version\process@table
393   \new@mathversion\DeclareSymbolFont\group@list\group@elt
394   \new@symbolfont\SetSymbolFont\SetSymbolFont@get@cdp
395   \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
396   \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
```

```

397 \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
398 \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
399 \set@mathdelimiter\DeclareMathRadical\mathchar@type
400 \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
401 }
402 {
403 \tl_remove_once:Nn \@preamblecmds {\do#1}
404 }

```

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.

```

405 \cs_set:Nn \@@_set_mathsymbol:nNNn
406 {
407 \int_compare:nT { \char_value_catcode:n {#4} == 11 }
408 { \char_set_catcode_other:n {#4} }
409
410 \tl_case:Nn #3
411 {
412 \mathord { \@@_set_mathcode:nnn {#4} {#3} {#1} }
413 \mathalpha { \@@_set_mathcode:nnn {#4} {#3} {#1} }
414 \mathbin { \@@_set_mathcode:nnn {#4} {#3} {#1} }
415 \mathrel { \@@_set_mathcode:nnn {#4} {#3} {#1} }
416 \mathpunct { \@@_set_mathcode:nnn {#4} {#3} {#1} }
417 \mathop { \@@_set_big_operator:nnn {#1} {#2} {#4} }
418 \mathopen { \@@_set_math_open:nnn {#1} {#2} {#4} }
419 \mathclose { \@@_set_math_close:nnn {#1} {#2} {#4} }
420 \mathfence { \@@_set_math_fence:nnnn {#1} {#2} {#3} {#4} }
421 \mathaccent
422 { \@@_set_math_accent:Nnnn #2 {fixed} {#1} {#4} }
423 \mathbotaccent
424 { \@@_set_math_accent:Nnnn #2 {bottom~ fixed} {#1} {#4} }
425 \mathaccentwide
426 { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
427 \mathbotaccentwide
428 { \@@_set_math_accent:Nnnn #2 {bottom} {#1} {#4} }
429 \mathover
430 { \@@_set_math_overunder:Nnnn #2 {} {#1} {#4} }

```



```

431 \mathunder
432 { \@@_set_math_overunder:Nnn #2 {bottom} {#1} {#4} }
433 }
434 }

435 \edef\mathfence{\string\mathfence}
436 \edef\mathover{\string\mathover}
437 \edef\mathunder{\string\mathunder}
438 \edef\mathbotaccent{\string\mathbotaccent}
439 \edef\mathaccentwide{\string\mathaccentwide}
440 \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 \sum_{op} . (This follows the convention of $\text{\LaTeX}/\text{amsmath}$.)
- Define \sum_{sym} as \sum_{op} , followed by \nolimits if necessary.

Whether the \nolimits suffix is inserted is controlled by the token list \l1_no_limits_tl , which contains a list of such characters. This list is checked dynamically to allow it to be updated mid-document.

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

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

```

441 \cs_new:Nn \@@_set_big_operator:nnn
442 {
443   \group_begin:
444     \char_set_catcode_active:n {#3}
445     \@@_char_gmake_mathactive:n {#3}
446     \@@_active_char_set:wc #3 \q_nil { \cs_to_str:N #2 _sym }
447   \group_end:
448
449   \@@_set_mathchar:cNnn { \cs_to_str:N #2 op } \mathop {#1} {#3}
450
451   \cs_gset:cpx { \cs_to_str:N #2 _sym }
452   {
453     \exp_not:c { \cs_to_str:N #2 op }
454     \exp_not:n { \tl_if_in:NnT \l1\_no\_limits\_tl {#2} \nolimits }
455   }
456 }

```

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

```

457 \cs_new:Nn \@@_set_math_open:nnn
458 {
459   \tl_if_in:NnTF \l_@@_radicals_tl {#2}
460   {
461     \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
462     { \@@_radical:nn {#1} {#3} }
463     \tl_set:cn {l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
464   }
465   {
466     \@@_set_delcode:nnn {#1} {#3} {#3}
467     \@@_set_mathcode:nnn {#3} \mathopen {#1}
468     \cs_gset_protected_nopar:Npx #2
469     { \@@_delimiter:Nnn \mathopen {#1} {#3} }
470   }
471 }
  
```

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

```

472 \cs_new:Nn \@@_set_math_close:nnn
473 {
474   \@@_set_delcode:nnn {#1} {#3} {#3}
475   \@@_set_mathcode:nnn {#3} \mathclose {#1}
476   \cs_gset_protected_nopar:Npx #2
477   { \@@_delimiter:Nnn \mathclose {#1} {#3} }
478 }
  
```

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

```

479 \cs_new:Nn \@@_set_math_fence:nnnn
480 {
481   \@@_set_mathcode:nnn {#4} {#3} {#1}
482   \@@_set_delcode:nnn {#1} {#4} {#4}
483   \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
484   { \@@_delimiter:Nnn \mathopen {#1} {#4} }
485   \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
486   { \@@_delimiter:Nnn \mathclose {#1} {#4} }
487 }
  
```

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

```

488 \cs_new:Nn \@@_set_math_accent:Nnnn
489 {
490   \cs_gset_protected_nopar:Npx #1
491   { \@@_accent:nnn {#2} {#3} {#4} }
492 }

```

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

```

493 \cs_new:Nn \@@_set_math_overunder:Nnnn
494 {
495   \cs_gset_protected_nopar:Npx #1 ##1
496   {
497     \mathop
498     { \@@_accent:nnn {#2} {#3} {#4} {##1} }
499     \limits
500   }
501 }

```

G.2 `\setmathalphabet`

`\setmathalphabet`

```

502 \keys_define:nn {@@_mathface}
503 {
504   version .code:n =
505   { \tl_set:Nn \l_@@_mversion_tl {#1} }
506 }
507
508 \DeclareDocumentCommand \setmathfontface { m O{} m O{} }
509 {
510   \tl_clear:N \l_@@_mversion_tl
511
512   \keys_set_known:nnN {@@_mathface} {#2,#4} \l_@@_keyval_clist
513   \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
514   { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}
515
516   \tl_if_empty:NT \l_@@_mversion_tl
517   {
518     \tl_set:Nn \l_@@_mversion_tl {normal}
519     \DeclareMathAlphabet #1 {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\mdde-
520     fault} {\updefault}
521   }
522   \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\md-
523   default} {\updefault}
524
525   % integrate with fontspec's \setmathrm etc:
526   \tl_case:Nn #1
527   {

```

```

526 \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
527 \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
528 \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
529 }
530 }
531
532 \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?

```

533 \DeclareDocumentCommand \setoperatorfont {m}
534 { \tl_set:Nn \g_@@_operator_mathfont_tl {#1} }
535 \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

```

536 \cs_generate_variant:Nn \tl_if_eq:nnT {o}
537 \cs_set:Nn \__fontspec_setmainfont:nn
538 {
539   \fontspec_set_family:Nnn \rmdefault {#1}{#2}
540   \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault}
541   {
542     (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
543     (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
544     \SetMathAlphabet\mathrm{normal}\g__fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
545     \SetMathAlphabet\mathit{normal}\g__fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
546     \SetMathAlphabet\mathbf{normal}\g__fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
547   }
548   \normalfont
549   \ignorespaces
550 }
551
552 \cs_set:Nn \__fontspec_setsansfont:nn
553 {
554   \fontspec_set_family:Nnn \sfdefault {#1}{#2}
555   \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault}
556   {
557     (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
558     (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}

```

```

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

```

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.

```

579 \cs_set:Nn \__fontspec_setmathrm:nn
580 {
581   (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
582   (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
583   \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
584   \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
585   \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
586 }
587 \cs_set:Nn \__fontspec_setboldmathrm:nn
588 {
589   (XE) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {#1} {#2}
590   (LU) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {Renderer=Basic,#1} {#2}
591   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\updefault
592   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
593   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\itdefault
594 }
595 \cs_set:Nn \__fontspec_setmathsf:nn
596 {
597   (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
598   (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
599   \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
600   \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
601 }
602 \cs_set:Nn \__fontspec_setmathtt:nn
603 {

```

```

604 (XE) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
605 (LU) \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
606 \SetMathAlphabet\mathtt{normal}\g__fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
607 \SetMathAlphabet\mathtt{bold} \g__fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
608 }

```

G.4 The main `\setmathfont` macro

Using a range including large character sets such as `\mathrel`, `\mathalpha`, `ifnextchar`.*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

```

609 \DeclareDocumentCommand \setmathfont { O{ } m O{ } }
610 {
611   \tl_set:Nn \l_@@_fontname_tl {#2}
612   \@@_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.

```

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

```

Parse options and tell people what's going on:

```

615 \keys_set_known:nnN {unicode-math} {#1,#3} \l_@@_unknown_keys_clist
616 \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.

```

617
618 \csname TIC\endcsname
619 \@@_fontspec_select_font:
620 \csname TOC\endcsname
621 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
622 {
623   \@@_declare_math_sizes:
624   \@@_fontspec_select_font:
625 }

```

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

```

626 \cs_if_exist:cF { sym \@@_symfont_tl }
627 {
628   \DeclareSymbolFont{\@@_symfont_tl}
629     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
630 }
631 \SetSymbolFont{\@@_symfont_tl}{\l_@@_mversion_tl}
632   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}

```

Set the bold math version.

```
633 \tl_set:Nn \l_@@_tmpa_tl {normal}
634 \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
635 {
636   \SetSymbolFont{\@@_symfont_tl}{bold}
637   {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
638 }
```

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:

```
639 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
640 {
641   \bool_set_true:N \g_@@_mainfont_already_set_bool
642   \@@_setup_legacy_fam_two:
643   \@@_setup_legacy_fam_three:
644 }
```

And now we input every single maths char.

```
645 \csname TIC\endcsname
646 \@@_input_math_symbol_table:
647 \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.)

```
648 \@@_remap_symbols:
649 \@@_setup_mathactives:
650 \@@_setup_delcodes:
651 \csname TIC\endcsname
652 \@@_setup_alphabets:
653 \csname TOC\endcsname
654 \@@_setup_negations:
```

Prevent spaces, and that's it:

```
655 \ignorespaces
656 }
```

Backward compatibility alias.

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

`\@@_init:`

```
658 \cs_new:Nn \@@_init:
659 {
```

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

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

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

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

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

```
662 \bool_set_true:N \l_@@_init_bool
663 \seq_clear:N \l_@@_char_range_seq
664 \clist_clear:N \l_@@_char_nrange_clist
665 \seq_clear:N \l_@@_mathalph_seq
666 \seq_clear:N \l_@@_missing_alph_seq
```

- By default use the ‘normal’ math version.

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

- Other range initialisations.

```
668 \tl_set:Nn \l_@@_symfont_tl {operators}
669 \cs_set_eq:NN \_@@_sym:nnn \l_@@_process_symbol_noparse:nnn
670 \cs_set_eq:NN \l_@@_set_mathalphabet_char:nnn \l_@@_mathmap_noparse:nnn
671 \cs_set_eq:NN \l_@@_remap_symbol:nnn \l_@@_remap_symbol_noparse:nnn
672 \cs_set_eq:NN \l_@@_maybe_init_alphabet:n \l_@@_init_alphabet:n
673 \cs_set_eq:NN \l_@@_map_char_single:nn \l_@@_map_char_noparse:nn
674 \cs_set_eq:NN \l_@@_assign_delcode:nn \l_@@_assign_delcode_noparse:nn
675 \cs_set_eq:NN \l_@@_make_mathactive:nNN \l_@@_make_mathactive_noparse:nNN
```

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

```
676 \tl_set:Nn \l_@@_script_features_tl {Style=MathScript}
677 \tl_set:Nn \l_@@_sscript_features_tl {Style=MathScriptScript}
678 \tl_set_eq:NN \l_@@_script_font_tl \l_@@_fontname_tl
679 \tl_set_eq:NN \l_@@_sscript_font_tl \l_@@_fontname_tl
```

```
680 }
```

\l_@@_declare_math_sizes: Set the math sizes according to the recommended font parameters:

```
681 \cs_new:Nn \l_@@_declare_math_sizes:
682 {
683   \dim_compare:nF { \fontdimen 10 \l_@@_font == 0pt }
684   {
685     \DeclareMathSizes { \f@size } { \f@size }
686       { \l_@@_fontdimen_to_scale:nn {10} { \l_@@_font } }
687       { \l_@@_fontdimen_to_scale:nn {11} { \l_@@_font } }
688   }
689 }
```


\@@_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 imperceptable amount.

```

690 \cs_new:Nn \@@_setup_legacy_fam_two:
691 {
692   \fontspec_set_family:Nxn \l_@@_family_tl
693   {
694     \l_@@_font_keyval_tl,
695     Scale=1.00001,
696     FontAdjustment =
697     {
698       \fontdimen8\font= \@@_get_fontparam:nn {43} {FractionNumeratorDis-
        playStyleShiftUp}\relax
699       \fontdimen9\font= \@@_get_fontparam:nn {42} {FractionNumerator-
        ShiftUp}\relax
700       \fontdimen10\font=\@@_get_fontparam:nn {32} {StackTopShiftUp}\relax
701       \fontdimen11\font=\@@_get_fontparam:nn {45} {FractionDenominatorDis-
        playStyleShiftDown}\relax
702       \fontdimen12\font=\@@_get_fontparam:nn {44} {FractionDenominatorShift-
        Down}\relax
703       \fontdimen13\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
704       \fontdimen14\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
705       \fontdimen15\font=\@@_get_fontparam:nn {22} {SuperscriptShif-
        tUpCramped}\relax
706       \fontdimen16\font=\@@_get_fontparam:nn {18} {SubscriptShiftDown}\relax
707       \fontdimen17\font=\@@_get_fontparam:nn {18} {SubscriptShiftDownWith-
        Superscript}\relax
708       \fontdimen18\font=\@@_get_fontparam:nn {24} {SuperscriptBaselineDrop-
        Max}\relax
709       \fontdimen19\font=\@@_get_fontparam:nn {20} {SubscriptBaselineDrop-
        Min}\relax
710       \fontdimen20\font=\opt\relax % delim1 = FractionDelimiterDisplaySize
711       \fontdimen21\font=\opt\relax % delim2 = FractionDelimiterSize
712       \fontdimen22\font=\@@_get_fontparam:nn {15} {AxisHeight}\relax
713     }
714   } {\l_@@_fontname_tl}
715   \SetSymbolFont{symbols}{\l_@@_mversion_tl}
716   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
717
718   \tl_set:Nn \l_@@_tmpa_tl {normal}
719   \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
720   {
721     \SetSymbolFont{symbols}{bold}
722     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
723   }
724 }

```

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

```

725 \cs_new:Nn \@@_setup_legacy_fam_three:
726 {

```

```

727 \fontspec_set_family:Nxn \l_@@_family_tl
728 {
729   \l_@@_font_keyval_tl,
730   Scale=0.99999,
731   FontAdjustment={
732     \fontdimen8\font= \@@_get_fontparam:nn {48} {FractionRuleThick-
ness}\relax
733     \fontdimen9\font= \@@_get_fontparam:nn {28} {UpperLimitGapMin}\relax
734     \fontdimen10\font=\@@_get_fontparam:nn {30} {LowerLimitGapMin}\relax
735     \fontdimen11\font=\@@_get_fontparam:nn {29} {UpperLimitBaselineR-
iseMin}\relax
736     \fontdimen12\font=\@@_get_fontparam:nn {31} {LowerLimitBaselineDrop-
Min}\relax
737     \fontdimen13\font=0pt\relax
738   }
739 } {\l_@@_fontname_tl}
740 \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
741 {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
742
743 \tl_set:Nn \l_@@_tmpa_tl {normal}
744 \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
745 {
746   \SetSymbolFont{largesymbols}{bold}
747   {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
748 }
749 }

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

```

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

```

753 \cs_new:Nn \@@_fontspec_select_font:
754 {
755   \tl_set:Nx \l_@@_font_keyval_tl {
756 (LU)     Renderer = Basic,
757     BoldItalicFont = {}, ItalicFont = {},
758     Script = Math,
759     SizeFeatures =
760     {
761       {
762         Size = \tf@size-
763       } ,
764       {
765         Size = \sf@size-\tf@size ,
766         Font = \l_@@_script_font_tl ,
767         \l_@@_script_features_tl
768       } ,
769       {
770         Size = -\sf@size ,

```

```

771     Font = \l_@@_sscript_font_tl ,
772     \l_@@_sscript_features_tl
773   }
774   } ,
775   \l_@@_unknown_keys_clist
776 }
777 \fontspec_set_fontface:NNxn \l_@@_font \l_@@_family_tl
778   {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

```

Check whether we're using a real maths font:

```

779 \group_begin:
780   \fontfamily{\l_@@_family_tl}\selectfont
781   \fontspec_if_script:nF {math} {\bool_gset_false:N \l_@@_ot_math_bool}
782 \group_end:
783 }

```

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.

```

784 \cs_set:Nn \@@_process_symbol_noparse:nnn
785 {
786   \@@_set_mathsymbol:nNNn {\@@_symfont_tl} #2 #3 {#1}
787 }
788 \cs_set:Nn \@@_process_symbol_parse:nnn
789 {
790   \@@_if_char_spec:nNNT {#1} {#2} {#3}
791   {
792     \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
793   }
794 }

```

`\@@_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
795 \cs_new:Npn \@@_remap_symbols:
796 {
797   \@@_remap_symbol:nnn{\-}{\mathbin}{"02212}% hyphen to minus
798   \@@_remap_symbol:nnn{\*}{\mathbin}{"02217}% text asterisk to "centred as-
       terisk"
799   \bool_if:NF \g_@@_literal_colon_bool
800   {
801     \@@_remap_symbol:nnn{\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
802   }
803 }

```

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

```

804 \cs_new:Nn \@@_remap_symbol_parse:nnn
805 {

```

```

806 \@@_if_char_spec:nNTT {#3} {\@nil} {#2}
807 { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
808 }
809 \cs_new:Nn \@@_remap_symbol_noparse:nnn
810 {
811   \clist_map_inline:nn {#1}
812   { \@@_set_mathcode:nnnn {##1} {#2} {\@@_symfont_tl} {#3} }
813 }

```

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:`

```

814 \cs_new:Npn \@@_setup_mathactives:
815 {
816   \@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
817   \@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
818   \@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
819   \@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar \mathord
820   \@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
821   \@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
822   \@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
823   \@@_make_mathactive:nNN {'\'} \mathstraightquote \mathord
824   \@@_make_mathactive:nNN {'\'} \mathbacktick \mathord
825 }

```

`\@@_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!

```

826 \cs_new:Nn \@@_make_mathactive_parse:nNN
827 {
828   \@@_if_char_spec:nNTT {#1} #2 #3
829   { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
830 }
831 \cs_new:Nn \@@_make_mathactive_noparse:nNN
832 {
833   \@@_set_mathchar:NNnn #2 #3 {\@@_symfont_tl} {#1}
834   \@@_char_gmake_mathactive:n {#1}
835 }

```

G.4.3 Delimiter codes

`\@@_assign_delcode:nn`

```

836 \cs_new:Nn \@@_assign_delcode_noparse:nn
837 {
838   \@@_set_delcode:nnn \@@_symfont_tl {#1} {#2}
839 }
840 \cs_new:Nn \@@_assign_delcode_parse:nn

```

```

841 {
842   \@@_if_char_spec:nNNT {#2} {\@nil} {\@nil}
843   {
844     \@@_assign_delcode_noparse:nn {#1} {#2}
845   }
846 }

```

\@@_assign_delcode:n Shorthand.

```

847 \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.

```

848 \cs_new:Npn \@@_setup_delcodes:
849 {
850   % ensure \left. and \right. work:
851   \@@_set_delcode:nnn \@@_symfont_tl {\'.} {\c_zero}
852   % this is forcefully done to fix a bug -- indicates a larger problem!
853
854   \@@_assign_delcode:nn {\'/} {\g_@@_slash_delimiter_usv}
855   \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
856   \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
857   \@@_assign_delcode:n {"005C} % backslash
858   \@@_assign_delcode:nn {\'\<} {"27E8} % angle brackets with ascii notation
859   \@@_assign_delcode:nn {\'\>} {"27E9} % angle brackets with ascii notation
860   \@@_assign_delcode:n {"2191} % up arrow
861   \@@_assign_delcode:n {"2193} % down arrow
862   \@@_assign_delcode:n {"2195} % updown arrow
863   \@@_assign_delcode:n {"219F} % up arrow twohead
864   \@@_assign_delcode:n {"21A1} % down arrow twohead
865   \@@_assign_delcode:n {"21A5} % up arrow from bar
866   \@@_assign_delcode:n {"21A7} % down arrow from bar
867   \@@_assign_delcode:n {"21A8} % updown arrow from bar
868   \@@_assign_delcode:n {"21BE} % up harpoon right
869   \@@_assign_delcode:n {"21BF} % up harpoon left
870   \@@_assign_delcode:n {"21C2} % down harpoon right
871   \@@_assign_delcode:n {"21C3} % down harpoon left
872   \@@_assign_delcode:n {"21C5} % arrows up down
873   \@@_assign_delcode:n {"21F5} % arrows down up
874   \@@_assign_delcode:n {"21C8} % arrows up up
875   \@@_assign_delcode:n {"21CA} % arrows down down
876   \@@_assign_delcode:n {"21D1} % double up arrow
877   \@@_assign_delcode:n {"21D3} % double down arrow
878   \@@_assign_delcode:n {"21D5} % double updown arrow
879   \@@_assign_delcode:n {"21DE} % up arrow double stroke
880   \@@_assign_delcode:n {"21DF} % down arrow double stroke
881   \@@_assign_delcode:n {"21E1} % up arrow dashed
882   \@@_assign_delcode:n {"21E3} % down arrow dashed

```

```

883 \@@_assign_delcode:n {"21E7} % up white arrow
884 \@@_assign_delcode:n {"21E9} % down white arrow
885 \@@_assign_delcode:n {"21EA} % up white arrow from bar
886 \@@_assign_delcode:n {"21F3} % updown white arrow
887 }

```

G.5 (Big) operators

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

However, the limits aren't set automatically; that is, we want to define, a la Plain \TeX `ifnextchar.etcetc.`, `\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 `\@@_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 \iiint , but that might be a matter of preference.

```

888 \tl_new:N \l_@@_nolimits_tl
889 \tl_set:Nn \l_@@_nolimits_tl
890 {
891   \int\iint\iiint\iiiint\oint\oiint\oiint
892   \intclockwise\varointclockwise\ointctrackwise\sumint
893   \intbar\intBar\find\cirfnint\awint\rrppolint
894   \scpolint\ncpolint\pointint\sqint\intlarhk\intx
895   \intcap\intcup\upint\lowint
896 }

```

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

```

897 \DeclareDocumentCommand \addnolimits {m}
898 {
899   \tl_put_right:Nn \l_@@_nolimits_tl {#1}
900 }

```

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

```

901 \DeclareDocumentCommand \removenolimits {m}
902 {
903   \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
904 }

```

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 `\cuberoor` 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.

```
905 \tl_new:N \l_@@_radicals_tl
906 \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 LuaTeX has `\crampeddisplaystyle` etc., but they are loaded as `\luatexcrampeddisplaystyle` etc. by the `luatextra` package. X_YTeX, 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 *⟨command⟩* as a new cramped style switch. For LuaTeX, simply rename the corresponding primitive. For X_YTeX, define *⟨command⟩* as a new quark.

```
907 \cs_new_protected_nopar:Nn \@@_new_cramped_style:N
908 (XE) { \quark_new:N #1 }
909 (LU) { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
```

```

\crampeddisplaystyle The cramped style commands.
\crampedtextstyle    910 \@@_new_cramped_style:N \crampeddisplaystyle
\crampedscriptstyle  911 \@@_new_cramped_style:N \crampedtextstyle
\crampedscriptscriptstyle 912 \@@_new_cramped_style:N \crampedscriptstyle
                        913 \@@_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 $\Xint\TeX$, 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  $\Xint\TeX$  math font dimension. <style token> must be one of the
style switches (\displaystyle, \crampeddisplaystyle, ...). The other parameters
are integer constants referring to font dimension numbers. The macro expands to
a dimension which contains the appropriate font dimension.

```

```

914 (*XE)
915 \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
916   \fontdimen
917   \cs_if_eq:NNTF #1 \displaystyle {
918     #2 \textfont
919   } {
920     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
921       #3 \textfont
922     } {
923       \cs_if_eq:NNTF #1 \textstyle {
924         #4 \textfont
925       } {
926         \cs_if_eq:NNTF #1 \crampedtextstyle {
927           #5 \textfont
928         } {
929           \cs_if_eq:NNTF #1 \scriptstyle {
930             #4 \scriptfont
931           } {
932             \cs_if_eq:NNTF #1 \crampedscriptstyle {
933               #5 \scriptfont
934             } {
935               \cs_if_eq:NNTF #1 \scriptscriptstyle {
936                 #4 \scriptscriptfont
937               } {
Should we check here if the style is invalid?
938                 #5 \scriptscriptfont
939               }

```



```

940         }
941     }
942 }
943 }
944 }
945 }

```

Which family to use?

```

946     \c_two
947 }
948 </XE>

```

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

`\@@_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 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.

```

949 \cs_new_protected_nopar:Nn \@@_font_param:nnnnn
950 < *XE>
951 {
952   \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
953     { #2 } { #3 } { #4 } { #5 }
954 }
955 < /XE>
956 < *LU>
957 {
958   \tl_set:Nn \l_@@_tmpa_tl { #1 }
959   \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
960   \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
961     { luatexUmath \l_@@_tmpa_tl }
962 }
963 < /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 $\langle name \rangle$. 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.

```

964 \cs_new_protected_nopar:Nn \@@_font_param:nnn

```

```

965 {
966   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
967 }

```

\@@_font_param:nn #1 : name

#2 : font dimension

This macro defines getter and setter functions for the font parameter $\langle name \rangle$. 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.

```

968 \cs_new_protected_nopar:Nn \@@_font_param:nn
969 {
970   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
971 }

```

\@@_font_param:n #1 : name

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

```

972 \cs_new_protected_nopar:Nn \@@_font_param:n
973 (XE) { }
974 (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

```

975 (*XE)
976 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNnnnn
977 {
978   \cs_new_nopar:Npn #1 ##1
979   {
980     \@@_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
981   }
982   \cs_new_protected_nopar:Npn #2 ##1 ##2
983   {
984     #1 ##1 \dim_eval:n { ##2 }
985   }
986 }
987 \cs_generate_variant:Nn \@@_font_param_aux:NNnnnn { cc }
988 (/XE)
989 (*LU)
990 \cs_new_protected_nopar:Nn \@@_font_param_aux:NNN
991 {
992   \cs_new_nopar:Npn #1 ##1
993   {
994     #3 ##1
995   }
996   \cs_new_protected_nopar:Npn #2 ##1 ##2
997   {
998     #3 ##1 \dim_eval:n { ##2 }

```

```

999     }
1000 }
1001 \cs_generate_variant:Nn \@@_font_param_aux:NNN { ccc }
1002 </LU>

```

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

```

1003 \@@_font_param:nn { axis } { 15 }
1004 \@@_font_param:nn { operator_size } { 13 }
1005 \@@_font_param:n { fraction_del_size }
1006 \@@_font_param:nnn { fraction_denom_down } { 45 } { 44 }
1007 \@@_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
1008 \@@_font_param:nnn { fraction_num_up } { 43 } { 42 }
1009 \@@_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
1010 \@@_font_param:nn { fraction_rule } { 48 }
1011 \@@_font_param:nn { limit_above_bgap } { 29 }
1012 \@@_font_param:n { limit_above_kern }
1013 \@@_font_param:nn { limit_above_vgap } { 28 }
1014 \@@_font_param:nn { limit_below_bgap } { 31 }
1015 \@@_font_param:n { limit_below_kern }
1016 \@@_font_param:nn { limit_below_vgap } { 30 }
1017 \@@_font_param:nn { over_delimiter_vgap } { 41 }
1018 \@@_font_param:nn { over_delimiter_bgap } { 38 }
1019 \@@_font_param:nn { under_delimiter_vgap } { 40 }
1020 \@@_font_param:nn { under_delimiter_bgap } { 39 }
1021 \@@_font_param:nn { overbar_kern } { 55 }
1022 \@@_font_param:nn { overbar_rule } { 54 }
1023 \@@_font_param:nn { overbar_vgap } { 53 }
1024 \@@_font_param:n { quad }
1025 \@@_font_param:nn { radical_kern } { 62 }
1026 \@@_font_param:nn { radical_rule } { 61 }
1027 \@@_font_param:nnn { radical_vgap } { 60 } { 59 }
1028 \@@_font_param:nn { radical_degree_before } { 63 }
1029 \@@_font_param:nn { radical_degree_after } { 64 }
1030 \@@_font_param:nn { radical_degree_raise } { 65 }
1031 \@@_font_param:nn { space_after_script } { 27 }
1032 \@@_font_param:nnn { stack_denom_down } { 35 } { 34 }
1033 \@@_font_param:nnn { stack_num_up } { 33 } { 32 }
1034 \@@_font_param:nnn { stack_vgap } { 37 } { 36 }
1035 \@@_font_param:nn { sub_shift_down } { 18 }
1036 \@@_font_param:nn { sub_shift_drop } { 20 }
1037 \@@_font_param:n { subsup_shift_down }
1038 \@@_font_param:nn { sub_top_max } { 19 }
1039 \@@_font_param:nn { subsup_vgap } { 25 }
1040 \@@_font_param:nn { sup_bottom_min } { 23 }
1041 \@@_font_param:nn { sup_shift_drop } { 24 }
1042 \@@_font_param:nnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
1043 \@@_font_param:nn { supsub_bottom_max } { 26 }
1044 \@@_font_param:nn { underbar_kern } { 58 }
1045 \@@_font_param:nn { underbar_rule } { 57 }

```

```

1046 \@@_font_param:nn { underbar_vgap } { 56 }
1047 \@@_font_param:n { connector_overlap_min }

```

H Font features

H.1 Math version

```

1048 \keys_define:nn {unicode-math}
1049 {
1050   version .code:n =
1051   {
1052     \tl_set:Nn \l_@@_mversion_tl {#1}
1053     \DeclareMathVersion {\l_@@_mversion_tl}
1054   }
1055 }

```

H.2 Script and scriptscript font options

```

1056 \keys_define:nn {unicode-math}
1057 {
1058   script-features .tl_set:N = \l_@@_script_features_tl ,
1059   sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
1060   script-font .tl_set:N = \l_@@_script_font_tl ,
1061   sscript-font .tl_set:N = \l_@@_sscript_font_tl ,
1062 }

```

H.3 Range processing

```

1063 \keys_define:nn {unicode-math}
1064 {
1065   range .code:n =
1066   {
1067     \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

```

1068   \int_incr:N \g_@@_fam_int
1069   \tl_set:Nx \@@_symfont_tl {@@_fam\int_use:N\g_@@_fam_int}
1070   \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
1071   \cs_set_eq:NN \@@_set_mathalphabet_char:Nnn \@@_mathmap_parse:Nnn
1072   \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
1073   \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
1074   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
1075   \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
1076   \cs_set_eq:NN \@@_make_mathactive:nnn \@@_make_mathactive_parse:nnn

```

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

```

1077   \seq_clear:N \l_@@_char_range_seq
1078   \seq_clear:N \l_@@_mclass_range_seq
1079   \seq_clear:N \l_@@_cmd_range_seq

```

```

1080 \seq_clear:N \l_@@_mathalph_seq
1081
1082 \clist_map_inline:nn {#1}
1083 {
1084   \@@_if_mathalph_decl:nTF {##1}
1085   {
1086     \seq_put_right:Nx \l_@@_mathalph_seq
1087     {
1088       { \exp_not:V \l_@@_tmpa_tl }
1089       { \exp_not:V \l_@@_tmpb_tl }
1090       { \exp_not:V \l_@@_tmpc_tl }
1091     }
1092   }
1093 {

```

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.

```

1094   \seq_if_in:NnTF \g_@@_mathclasses_seq {##1}
1095   { \seq_put_right:Nn \l_@@_mclass_range_seq {##1} }
1096   {
1097     \bool_if:nTF { \tl_if_single_p:n {##1} && \token_if_cs_p:N ##1 }
1098     { \seq_put_right:Nn \l_@@_cmd_range_seq {##1} }
1099     { \seq_put_right:Nn \l_@@_char_range_seq {##1} }
1100   }
1101 }
1102 }
1103 }
1104 }

```

\@@_if_mathalph_decl:nTF Possible forms of input:

```

\mathscr
\mathscr->\mathup
\mathscr/{Latin}
\mathscr/{Latin}->\mathup

```

Outputs:

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

tmpb: alphabets (*e.g.*, Latin)

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

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

```

1105 \prg_new_conditional:Nnn \@@_if_mathalph_decl:n {TF}
1106 {
1107   \tl_set:Nn \l_@@_tmpa_tl {#1}
1108   \tl_clear:N \l_@@_tmpb_tl
1109   \tl_clear:N \l_@@_tmpc_tl
1110
1111   \tl_if_in:NnT \l_@@_tmpa_tl {->}
1112   { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }

```

```

1113
1114 \tl_if_in:NnT \l_@@_tmpa_tl {/}
1115 { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }
1116
1117 \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
1118 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
1119 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
1120 \tl_trim_spaces:N \l_@@_tmpa_tl
1121
1122 \tl_if_empty:NT \l_@@_tmpc_tl
1123 { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }
1124
1125 \seq_if_in:NVTF \g_@@_named_ranges_seq \l_@@_tmpa_tl
1126 { \prg_return_true: } { \prg_return_false: }
1127 }
1128 \cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
1129 {
1130   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1131   \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
1132 }
1133 \cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
1134 {
1135   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1136   \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
1137 }

```

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

`\@@_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.

```

1138 \cs_new:Nn \@@_if_char_spec:nnNT
1139 {
1140   % math class:
1141   \seq_if_in:NnT \l_@@_mclass_range_seq {#3}
1142   { \use_none_delimit_by_q_nil:w }
1143
1144   % command name:
1145   \seq_if_in:NnT \l_@@_cmd_range_seq {#2}
1146   { \use_none_delimit_by_q_nil:w }
1147
1148   % character slot:
1149   \seq_map_inline:Nn \l_@@_char_range_seq
1150   {
1151     \@@_int_if_slot_in_range:nnT {#1} {##1}
1152     { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
1153   }
1154
1155   % the following expands to nil if no match was found:
1156   \use_none:nnn
1157   \q_nil
1158   \use:n
1159   {
1160     \clist_put_right:Nx \l_@@_char_nrange_clist { \int_eval:n {#1} }
1161     #4
1162   }
1163 }

```

\@@_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

```

```

1164 \cs_new:Nn \@@_int_if_slot_in_range:nnT
1165 { \@@_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3} }
1166 \cs_set:Npn \@@_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
1167 {
1168   \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
1169   {
1170     \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
1171     {
1172       \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
1173       {
1174         \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
1175       } } }

```

1176 }

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.

```

1177 \AtBeginDocument{\@@_resolve_greek:}
1178 \cs_new:Npn \@@_resolve_greek:
1179 {
1180   \clist_map_inline:nn
1181   {
1182     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1183     alpha,beta,gamma,delta,          zeta,eta,theta,iota,kappa,lambda,
1184     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1185     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,  chi,psi,omega,
1186     varTheta,
1187     varsigma,vartheta,varkappa,varrho,varpi
1188   }
1189   {
1190     \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
1191     \tl_set:cx {up ##1} { \exp_not:N \symup \exp_not:c { ##1 } }
1192     \tl_set:cx {it ##1} { \exp_not:N \symit \exp_not:c { ##1 } }
1193   }
1194   \tl_set:Nn \epsilon
1195   { \bool_if:NTF \g_@@_texgreek_bool \mitvarepsilon \mitepsilon }
1196   \tl_set:Nn \phi
1197   { \bool_if:NTF \g_@@_texgreek_bool \mitvarphi \mitphi }
1198   \tl_set:Nn \varepsilon
1199   { \bool_if:NTF \g_@@_texgreek_bool \mitepsilon \mitvarepsilon }
1200   \tl_set:Nn \varphi
1201   { \bool_if:NTF \g_@@_texgreek_bool \mitphi \mitvarphi }
1202 }
```

I Maths alphabets

Defining commands like `\mathrm` is not as simple with Unicode fonts. In traditional T_EX 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 $\LaTeX 2_{\epsilon}$

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 `\DeclareSymbolFontAlphabet`.) 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.

```
1203 \cs_set:Npn \use@mathgroup #1 #2
1204 {
1205   \mode_if_math:T % <- not sure if this is really necessary since we’ve just checked for mmode and raised
      ror if not!
1206   {
1207     \math@bgroup
1208     \cs_if_eq:cNF {M@f@encoding} #1 {#1}
1209     \@_switchto_literal:
1210     \mathgroup #2 \relax
1211     \math@egroup
1212   }
1213 }
```

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.

1.3 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.

```

1214 \cs_new:Nn \@@_prepare_mathstyle:n
1215 {
1216   \seq_put_right:Nn \g_@@_mathstyles_seq {#1}
1217   \@@_init_alphabet:n {#1}
1218   \cs_set:cpn {_@@_sym_#1_aux:n}
1219   { \use:c {@@_switchto_#1:} \math@egroup }
1220   \cs_set_protected:cpx {sym#1}
1221   {
1222     \exp_not:n
1223     {
1224       \math@bgroup
1225       \mode_if_math:F
1226       {
1227         \egroup\expandafter
1228         \non@alpherr\expandafter{\csname sym#1\endcsname\space}
1229       }
1230       \tl_set:Nn \l_@@_mathstyle_tl {#1}
1231     }
1232     \exp_not:c {_@@_sym_#1_aux:n}
1233   }
1234 }
```

`\@@_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.

```

1235 \cs_set:Nn \@@_init_alphabet:n
1236 {
1237   \@@_log:nx {alph-initialise} {#1}
1238   \cs_set_eq:cN {\@@_switchto_#1:} \prg_do_nothing:
1239 }

```

1.4 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).

```

1240 \cs_new:Nn \@@_new_named_range:n
1241 {
1242   \prop_new:c {g_@@_named_range_#1_prop}
1243 }
1244 \clist_set:Nn \g_@@_named_ranges_clist
1245 {
1246   up, it, tt, bfup, bfit, bb , bbit, scr, bfscr, cal, bfcalf,
1247   frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
1248 }
1249 \clist_map_inline:Nn \g_@@_named_ranges_clist
1250 { \@@_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.

```

1251 \clist_new:N \g_@@_alphabets_seq
1252 \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.

```

1253 \cs_new:Nn \@@_new_alphabet_config:nnn
1254 {
1255   \prop_if_exist:cF {g_@@_named_range_#1_prop}
1256   { \@@_warning:nnn {no-named-range} {#1} {#2} }
1257
1258   \prop_gput:cnn {g_@@_named_range_#1_prop} { alpha_t1 }
1259   {
1260     \prop_item:cn {g_@@_named_range_#1_prop} { alpha_t1 }
1261     {#2}
1262   }
1263   % Q: do I need to bother removing duplicates?
1264
1265   \cs_new:cn { @@_config_#1_#2:n } {#3}
1266 }

```

```

1267 \cs_new:Nn \@@_alphabet_config:nnn { \use:c {@@_config_#1_#2:n} {#3} }
1268 \prg_new_conditional:Nnn \@@_if_alphabet_exists:nn {T,TF}
1269 {
1270   \cs_if_exist:cTF {@@_config_#1_#2:n}
1271   \prg_return_true: \prg_return_false:
1272 }

```

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.

```

1273 \cs_new:Nn \@@_default_mathalph:nnn
1274 {
1275   \seq_put_right:Nx \g_@@_named_ranges_seq { \tl_to_str:n {#1} }
1276   \seq_put_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
1277 }
1278 \@@_default_mathalph:nnn {up} {latin, Latin, greek, Greek, num, misc} {up}
1279 \@@_default_mathalph:nnn {it} {latin, Latin, greek, Greek, misc} {it}
1280 \@@_default_mathalph:nnn {bb} {latin, Latin, num, misc} {bb}
1281 \@@_default_mathalph:nnn {bbit} {misc} {bbit}
1282 \@@_default_mathalph:nnn {scr} {latin, Latin} {scr}
1283 \@@_default_mathalph:nnn {cal} {Latin} {scr}
1284 \@@_default_mathalph:nnn {bfcal} {Latin} {bfscr}
1285 \@@_default_mathalph:nnn {frak} {latin, Latin} {frak}
1286 \@@_default_mathalph:nnn {tt} {latin, Latin, num} {tt}
1287 \@@_default_mathalph:nnn {sfup} {latin, Latin, num} {sfup}
1288 \@@_default_mathalph:nnn {sfit} {latin, Latin} {sfit}
1289 \@@_default_mathalph:nnn {bfup} {latin, Latin, greek, Greek, num, misc} {bfup}
1290 \@@_default_mathalph:nnn {bfit} {latin, Latin, greek, Greek, misc} {bfit}
1291 \@@_default_mathalph:nnn {bfscr} {latin, Latin} {bfscr}
1292 \@@_default_mathalph:nnn {bffrak} {latin, Latin} {bffrak}
1293 \@@_default_mathalph:nnn {bfsfup} {latin, Latin, greek, Greek, num, misc} {bfsfup}
1294 \@@_default_mathalph:nnn {bfsfit} {latin, Latin, greek, Greek, misc} {bfsfit}

```

1.4.1 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!

```

1295 \clist_map_inline:nn
1296 {
1297   up, it, bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf,
1298   tt, bb, bbit, scr, bfscr, cal, bfcal, frak, bffrak,
1299   normal, literal, sf, bf,
1300 }
1301 { \@@_prepare_mathstyle:n {#1} }

```

1.4.2 New names for legacy textmath alphabet selection

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

```

1302 \clist_map_inline:nn
1303 { rm, it, bf, sf, tt }
1304 { \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.

```

1305 \clist_map_inline:nn
1306 {
1307   normal, bb , bbit, scr, bfscr, cal, bfcac, frak, bffrak, tt,
1308   bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
1309 }
1310 {
1311   \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
1312 }

```

I.4.4 New commands for ambiguous alphabets

```

1313 \AtBeginDocument{
1314 \clist_map_inline:nn
1315 { rm, it, bf, sf, tt }
1316 {
1317   \cs_set_protected:cpx { math #1 }
1318   {
1319     \exp_not:n { \bool_if:NTF } \exp_not:c { g_@@_ math #1 _text_bool }
1320     { \exp_not:c { mathtext #1 } }
1321     { \exp_not:c { sym #1 } }
1322   }
1323 }}

```

Alias \mathrm as legacy name for \mathup

```

1324 \cs_set_protected:Npn \mathup { \mathrm }
1325 \cs_set_protected:Npn \symrm { \symup }

```

I.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`

```
1326 \cs_set:Npn \operator@font
1327 {
1328   \@_switchto_literal:
1329   \@fontswitch {} { \g_@_operator_mathfont_tl }
1330 }
```

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.

```
1331 \cs_new:Npn \l_@@_setup_alphabets:
```

```
1332 {
```

If `range=` has been used to configure styles, those choices will be in `\l_@@_mathalph_seq`.

If not, set up the styles implicitly:

```
1333 \seq_if_empty:NTF \l_@@_mathalph_seq
```

```
1334 {
```

```
1335 \l_@@_log:n {setup-implicit}
```

```
1336 \seq_set_eq:NN \l_@@_mathalph_seq \g_@@_default_mathalph_seq
```

```
1337 \bool_set_true:N \l_@@_implicit_alph_bool
```

```
1338 \l_@@_maybe_init_alphabet:n {sf}
```

```
1339 \l_@@_maybe_init_alphabet:n {bf}
```

```
1340 \l_@@_maybe_init_alphabet:n {bfsf}
```

```
1341 }
```

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

```
1342 {
```

```
1343 \l_@@_log:n {setup-explicit}
```

```
1344 \bool_set_false:N \l_@@_implicit_alph_bool
```

```
1345 \cs_set_eq:NN \l_@@_set_mathalphabet_char:nnn \l_@@_mathmap_noparse:nnn
```

```
1346 \cs_set_eq:NN \l_@@_map_char_single:nn \l_@@_map_char_noparse:nn
```

```
1347 }
```

Now perform the mapping:

```
1348 \seq_map_inline:Nn \l_@@_mathalph_seq
```

```
1349 {
```

```
1350 \tl_set:Nn \l_@@_style_tl { \use_i:nnn ##1 }
```

```
1351 \clist_set:Nn \l_@@_alphabet_clist { \use_ii:nnn ##1 }
```

```
1352 \tl_set:Nn \l_@@_remap_style_tl { \use_iii:nnn ##1 }
```

```
1353
```

```
1354 \clist_if_empty:NT \l_@@_alphabet_clist
```

```
1355 {
```

```
1356 \cs_set_eq:NN \l_@@_maybe_init_alphabet:n \l_@@_init_alphabet:n
```

```
1357 \clist_set:Nn \l_@@_alphabet_clist { latin, Latin, greek, Greek, num, misc }
```

```
1358 }
```

```
1359
```

```
1360 \l_@@_setup_math_alphabet:NNN
```

```
1361 \l_@@_style_tl \l_@@_alphabet_clist \l_@@_remap_style_tl
```

```
1362 }
```

```
1363 \seq_if_empty:NF \l_@@_missing_alph_seq { \l_@@_log:n { missing-alphabets } }
```

```
1364 }
```

`\l_@@_setup_math_alphabet:NNN` #1 : Math font style (e.g., `\bb`)

#2 : Math alphabets, comma separated of {latin, Latin, greek, Greek, num}

#3 : Name of the output math style (usually same as the name of the style `bb`)

```
1365 \cs_new:Npn \l_@@_setup_math_alphabet:NNN
```

```
1366 {
```

```
1367 \tl_set_eq:NN \l_@@_style_tl #1% actually a no-op but makes the code clearer!
```

First check that at least one of the alphabets for the font shape is defined (this process is fast) ...

```

1368 \clist_map_inline:Nn #2
1369 {
1370   \tl_set:Nn \l_@@_alphabet_tl {##1}
1371   \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
1372   {
1373     \str_if_eq_x:nnTF {\l_@@_alphabet_tl} {misc}
1374     {
1375       \@@_maybe_init_alphabet:n \l_@@_style_tl
1376       \clist_map_break:
1377     }
1378     {
1379       \@@_glyph_if_exist:nT { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
1380       {
1381         \@@_maybe_init_alphabet:n \l_@@_style_tl
1382         \clist_map_break:
1383       }
1384     }
1385   }
1386   { \msg_warning:nx {unicode-math} {no-alphabet} { \l_@@_style_tl / \l_@@_alphabet_tl } }
1387 }

```

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

```

1388 \csname TIC\endcsname
1389 \clist_map_inline:Nn #2
1390 {
1391   \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
1392   \cs_if_exist:cT {@@_config_ \l_@@_style_tl _ \l_@@_alphabet_tl :n}
1393   {
1394     \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}
1395     {
1396       \@@_log:nx {setup-alph} {math \l_@@_style_tl~(\l_@@_alphabet_tl)}
1397       \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {#3}
1398     }
1399     {
1400       \@@_glyph_if_exist:nTF { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
1401       {
1402         \@@_log:nx {setup-alph} {math \l_@@_style_tl~(\l_@@_alphabet_tl)}
1403         \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {#3}
1404       }
1405       {
1406         \bool_if:NTF \l_@@_implicit_alph_bool
1407         {
1408           \seq_put_right:Nx \l_@@_missing_alph_seq
1409           {
1410             \@backslashchar sym \l_@@_style_tl \space
1411             (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
1412           }
1413         }
1414       }

```



```

1415         \l_@_style_tl} {\l_@_alphabet_tl} {up}
1416     }
1417 }
1418 }
1419 }
1420 }
1421 \csname TOC\endcsname
1422 }

```

I.6 Mapping ‘naked’ math characters

Before we show the definitions of the alphabet mappings using the functions `\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

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

```

\l_@_map_char_noparse:nn
\l_@_map_char_parse:nn
1423 \cs_new:Nn \l_@_map_char_noparse:nn
1424 { \l_@_set_mathcode:nnnn {#1}{\mathalpha}{\l_@_symfont_tl}{#2} }
1425 \cs_new:Nn \l_@_map_char_parse:nn
1426 {
1427   \l_@_if_char_spec:nNNT {#1} {\@nil} {\mathalpha}
1428   { \l_@_map_char_noparse:nn {#1}{#2} }
1429 }

```

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

```

1430 \cs_new:Nn \l_@_map_char_single:nnn
1431 {
1432   \l_@_map_char_single:nn { \l_@_to_usv:nn {#1}{#3} }
1433   { \l_@_to_usv:nn {#2}{#3} }
1434 }

```

`\l_@_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:

```

1435 \cs_set:Nn \l_@_map_chars_range:nnn
1436 {
1437   \int_step_inline:nnnn {0}{1}{#1-1}
1438   { \l_@_map_char_single:nn {#2+##1}{#3+##1} }
1439 }

```

And the wrapper with names:

```

1440 \cs_new:Nn \@@_map_chars_range:nnnn
1441 {
1442   \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2}{#4} }
1443   { \@@_to_usv:nn {#3}{#4} }
1444 }

```

I.6.2 Functions for ‘normal’ alphabet symbols

\@@_set_normal_char:nnn

```

1445 \cs_set:Nn \@@_set_normal_char:nnn
1446 {
1447   \@@_usv_if_exist:nnT {#3} {#1}
1448   {
1449     \clist_map_inline:nn {#2}
1450     {
1451       \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
1452       \@@_map_char_single:nnn {##1} {#3} {#1}
1453     }
1454   }
1455 }

1456 \cs_new:Nn \@@_set_normal_Latin:nn
1457 {
1458   \clist_map_inline:nn {#1}
1459   {
1460     \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
1461     \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}
1462   }
1463 }

1464 \cs_new:Nn \@@_set_normal_latin:nn
1465 {
1466   \clist_map_inline:nn {#1}
1467   {
1468     \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
1469     \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
1470   }
1471 }

1472 \cs_new:Nn \@@_set_normal_greek:nn
1473 {
1474   \clist_map_inline:nn {#1}
1475   {
1476     \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
1477     \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
1478     \@@_map_char_single:nnn {##1} {#2} {varepsilon}
1479     \@@_map_char_single:nnn {##1} {#2} {vartheta}
1480     \@@_map_char_single:nnn {##1} {#2} {varkappa}
1481     \@@_map_char_single:nnn {##1} {#2} {varphi}
1482     \@@_map_char_single:nnn {##1} {#2} {varrho}

```

```

1483 \@@_map_char_single:nnn {##1} {#2} {\varpi}
1484 \@@_set_mathalphabet_pos:nnnn {normal} {\varepsilon} {##1} {#2}
1485 \@@_set_mathalphabet_pos:nnnn {normal} {\vartheta} {##1} {#2}
1486 \@@_set_mathalphabet_pos:nnnn {normal} {\varkappa} {##1} {#2}
1487 \@@_set_mathalphabet_pos:nnnn {normal} {\varphi} {##1} {#2}
1488 \@@_set_mathalphabet_pos:nnnn {normal} {\varrho} {##1} {#2}
1489 \@@_set_mathalphabet_pos:nnnn {normal} {\varpi} {##1} {#2}
1490 }
1491 }

1492 \cs_new:Nn \@@_set_normal_Greek:nn
1493 {
1494 \clist_map_inline:nn {#1}
1495 {
1496 \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
1497 \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
1498 \@@_map_char_single:nnn {##1} {#2} {\varTheta}
1499 \@@_set_mathalphabet_pos:nnnn {normal} {\varTheta} {##1} {#2}
1500 }
1501 }

1502 \cs_new:Nn \@@_set_normal_numbers:nn
1503 {
1504 \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
1505 \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
1506 }

```

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.

```

1507 \cs_new:Nn \@@_mathmap_noparse:nnn
1508 {
1509 \clist_map_inline:nn {#2}
1510 {
1511 \tl_put_right:cx {\@@_switchto_#1:}
1512 {
1513 \@@_set_mathcode:nnnn {##1} {\mathalpha} {\@@_symfont_t1} {#3}
1514 }
1515 }
1516 }

```

`\@@_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.

```

1517 \cs_new:Nn \@@_mathmap_parse:nnn
1518 {
1519   \clist_if_in:NnT \l_@@_char_nrange_clist {#3}
1520   {
1521     \@@_mathmap_noparse:nnn {#1}{#2}{#3}
1522   }
1523 }
```

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

```

1524 \cs_new:Nn \@@_set_mathalphabet_char:nnnn
1525 {
1526   \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
1527   { \@@_to_usv:nn {#3} {#4} }
1528 }
```

`\@@_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:

```

1529 \cs_new:Nn \@@_set_mathalph_range:nnnn
1530 {
1531   \int_step_inline:nnnn {0} {1} {#1-1}
1532   { \@@_set_mathalphabet_char:nnn {#2} { ##1 + #3 } { ##1 + #4 } }
1533 }
```

Then the wrapper version that uses names:

```

1534 \cs_new:Nn \@@_set_mathalph_range:nnnnn
1535 {
1536   \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
1537   { \@@_to_usv:nn {#4} {#5} }
1538 }
```

1.7.2 Individual mapping functions for different alphabets

```

1539 \cs_new:Nn \@@_set_mathalphabet_pos:nnnn
1540 {
```

```

1541 \@@_usv_if_exist:nnT {#4} {#2}
1542 {
1543   \clist_map_inline:nn {#3}
1544     { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }
1545   }
1546 }

1547 \cs_new:Nn \@@_set_mathalphabet_numbers:nnn
1548 {
1549   \clist_map_inline:nn {#2}
1550     { \@@_set_mathalph_range:nnnn {10} {#1} {##1} {#3} {num} }
1551   }

1552 \cs_new:Nn \@@_set_mathalphabet_Latin:nnn
1553 {
1554   \clist_map_inline:nn {#2}
1555     { \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {Latin} }
1556   }

1557 \cs_new:Nn \@@_set_mathalphabet_latin:nnn
1558 {
1559   \clist_map_inline:nn {#2}
1560     {
1561       \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {latin}
1562       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {h}
1563     }
1564   }

1565 \cs_new:Nn \@@_set_mathalphabet_Greek:nnn
1566 {
1567   \clist_map_inline:nn {#2}
1568     {
1569       \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {Greek}
1570       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varTheta}
1571     }
1572   }

1573 \cs_new:Nn \@@_set_mathalphabet_greek:nnn
1574 {
1575   \clist_map_inline:nn {#2}
1576     {
1577       \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {greek}
1578       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varepsilon}
1579       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {vartheta}
1580       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varkappa}
1581       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varphi}
1582       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varrho}
1583       \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varpi}
1584     }
1585   }

```

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.

```

1586 \cs_new:Npn \@@_symbol_setup:
1587 {
1588   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
1589   {
1590     \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
1591   }
1592 }

1593 \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.

```

1594 \cs_new:Nn \@@_input_math_symbol_table: {\g_@@_mathtable_tl}

```

K Definitions of the active math characters

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

`\@@_cs_set_eq_active_char:Nw` We need to do some trickery to transform the `_@@_sym:nnn` argument "ABCDEF into the \XeTeX '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.

```

1595 \group_begin:
1596   \char_set_catcode_other:N \^
1597   \cs_gset:Npn \@@_cs_set_eq_active_char:Nw #1 = "#2 \q_nil
1598   {
1599     \tex_lowercase:D
1600     {
1601       \tl_rescan:nn
1602       {
1603         \ExplSyntaxOn
1604         \char_set_catcode_other:N \{
1605         \char_set_catcode_other:N \}
1606         \char_set_catcode_other:N \&
1607         \char_set_catcode_other:N \%
1608         \char_set_catcode_other:N \$
1609       }

```

```

1610     {
1611       \cs_gset_eq:NN #1 ^^^^#2
1612     }
1613   }
1614 }

```

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

```

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

```

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`.

```

1624 \AtBeginDocument{\@@_define_math_chars:}
1625 \cs_new:Nn \@@_define_math_chars:
1626 {
1627   \group_begin:
1628   \char_set_catcode_math_superscript:N \^
1629   \cs_set:Npn \_@@_sym:nnn ##1##2##3
1630   {
1631     \tl_if_in:nnT
1632     { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
1633     {##3}
1634     {
1635       \@@_cs_set_eq_active_char:Nw ##2 = ##1 \q_nil \ignorespaces
1636     }
1637   }
1638   \char_set_catcode_other:N \#
1639   \@@_input_math_symbol_table:
1640   \group_end:
1641 }

```

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

```

1642 \group_begin:
1643 \lcode'\*='\
1644 \char_set_catcode_escape:N \|
1645 \char_set_catcode_other:N \
1646 |lowercase
1647 {
1648 |AtBeginDocument

```

```

1649     {
1650         |let|backslash=*
1651     }
1652 }
1653 |group_end:

```

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.

```

1654 \AtBeginDocument { \@@_load_lm_if_necessary: }
1655 \cs_new:Nn \@@_load_lm_if_necessary:
1656 {
1657     \cs_if_exist:NF \l_@@_fontname_tl
1658     {
1659         % TODO: update this when lmmath-bold.otf is released
1660         \setmathfont{latinmodern-math.otf}[BoldFont={latinmodern-math.otf}]
1661         \bool_set_false:N \g_@@_mainfont_already_set_bool
1662     }
1663 }

```

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:

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

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

In regular \LaTeX , primes can be entered with the straight quote character `'`, and multiple straight quotes chain together to produce multiple primes. Better

results can be achieved in unicode-math by chaining multiple single primes into a pre-drawn multi-prime glyph; consider x''' vs. x''' .

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

Our algorithm is:

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

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

```

1664 \cs_new:Nn \@@_arg_i_before_egroup:n {#1\egroup}
1665 \cs_new:Nn \@@_superscript:n
1666 {
1667   ^\bgroup #1
1668   \peek_meaning_remove:NTF ^ \@@_arg_i_before_egroup:n \egroup
1669 }

1670 \cs_new:Nn \@@_nprimes:Nn
1671 {
1672   \@@_superscript:n
1673   {
1674     #1
1675     \prg_replicate:nn {#2-1} { \mskip \g_@@_primekern_muskip #1 }
1676   }
1677 }

1678
1679 \cs_new:Nn \@@_nprimes_select:nn
1680 {
1681   \int_case:nnF {#2}
1682   {
1683     {1} { \@@_superscript:n {#1} }
1684     {2} {
1685       \@@_glyph_if_exist:nTF {"2033}
1686       { \@@_superscript:n {\@@_prime_double_mchar} }
1687       { \@@_nprimes:Nn #1 {#2} }
1688     }
1689     {3} {
1690       \@@_glyph_if_exist:nTF {"2034}
1691       { \@@_superscript:n {\@@_prime_triple_mchar} }

```

```

1692     { \@@_nprimes:Nn #1 {#2} }
1693   }
1694   {4} {
1695     \@@_glyph_if_exist:nTF {"2057}
1696     { \@@_superscript:n {\@@_prime_quad_mchar} }
1697     { \@@_nprimes:Nn #1 {#2} }
1698   }
1699 }
1700 {
1701   \@@_nprimes:Nn #1 {#2}
1702 }
1703 }
1704 \cs_new:Nn \@@_nbackprimes_select:nn
1705 {
1706   \int_case:nnF {#2}
1707   {
1708     {1} { \@@_superscript:n {#1} }
1709     {2} {
1710       \@@_glyph_if_exist:nTF {"2036}
1711       { \@@_superscript:n {\@@_backprime_double_mchar} }
1712       { \@@_nprimes:Nn #1 {#2} }
1713     }
1714     {3} {
1715       \@@_glyph_if_exist:nTF {"2037}
1716       { \@@_superscript:n {\@@_backprime_triple_mchar} }
1717       { \@@_nprimes:Nn #1 {#2} }
1718     }
1719   }
1720   {
1721     \@@_nprimes:Nn #1 {#2}
1722   }
1723 }

```

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

```

1724 \cs_new:Npn \@@_scan_prime:
1725 {
1726   \cs_set_eq:NN \@@_superscript:n \use:n
1727   \int_zero:N \l_@@_primecount_int
1728   \@@_scanprime_collect:N \@@_prime_single_mchar
1729 }
1730 \cs_new:Npn \@@_scan_dprime:
1731 {
1732   \cs_set_eq:NN \@@_superscript:n \use:n
1733   \int_set:Nn \l_@@_primecount_int {1}
1734   \@@_scanprime_collect:N \@@_prime_single_mchar
1735 }
1736 \cs_new:Npn \@@_scan_trprime:
1737 {
1738   \cs_set_eq:NN \@@_superscript:n \use:n
1739   \int_set:Nn \l_@@_primecount_int {2}

```

```

1740 \@@_scanprime_collect:N \@@_prime_single_mchar
1741 }
1742 \cs_new:Npn \@@_scan_qprime:
1743 {
1744   \cs_set_eq:NN \@@_superscript:n \use:n
1745   \int_set:Nn \l_@@_primecount_int {3}
1746   \@@_scanprime_collect:N \@@_prime_single_mchar
1747 }
1748 \cs_new:Npn \@@_scan_sup_prime:
1749 {
1750   \int_zero:N \l_@@_primecount_int
1751   \@@_scanprime_collect:N \@@_prime_single_mchar
1752 }
1753 \cs_new:Npn \@@_scan_sup_dprime:
1754 {
1755   \int_set:Nn \l_@@_primecount_int {1}
1756   \@@_scanprime_collect:N \@@_prime_single_mchar
1757 }
1758 \cs_new:Npn \@@_scan_sup_trprime:
1759 {
1760   \int_set:Nn \l_@@_primecount_int {2}
1761   \@@_scanprime_collect:N \@@_prime_single_mchar
1762 }
1763 \cs_new:Npn \@@_scan_sup_qprime:
1764 {
1765   \int_set:Nn \l_@@_primecount_int {3}
1766   \@@_scanprime_collect:N \@@_prime_single_mchar
1767 }
1768 \cs_new:Nn \@@_scanprime_collect:N
1769 {
1770   \int_incr:N \l_@@_primecount_int
1771   \peek_meaning_remove:NTF '
1772   { \@@_scanprime_collect:N #1 }
1773   {
1774     \peek_meaning_remove:NTF \@@_scan_prime:
1775     { \@@_scanprime_collect:N #1 }
1776     {
1777       \peek_meaning_remove:NTF ^^^^2032
1778       { \@@_scanprime_collect:N #1 }
1779       {
1780         \peek_meaning_remove:NTF \@@_scan_dprime:
1781         {
1782           \int_incr:N \l_@@_primecount_int
1783           \@@_scanprime_collect:N #1
1784         }
1785         {
1786           \peek_meaning_remove:NTF ^^^^2033
1787           {
1788             \int_incr:N \l_@@_primecount_int

```

```

1789         \@@_scanprime_collect:N #1
1790     }
1791     {
1792         \peek_meaning_remove:NTF \@@_scan_trprime:
1793         {
1794             \int_add:Nn \l_@@_primecount_int {2}
1795             \@@_scanprime_collect:N #1
1796         }
1797         {
1798             \peek_meaning_remove:NTF ^^^^2034
1799             {
1800                 \int_add:Nn \l_@@_primecount_int {2}
1801                 \@@_scanprime_collect:N #1
1802             }
1803             {
1804                 \peek_meaning_remove:NTF \@@_scan_qprime:
1805                 {
1806                     \int_add:Nn \l_@@_primecount_int {3}
1807                     \@@_scanprime_collect:N #1
1808                 }
1809                 {
1810                     \peek_meaning_remove:NTF ^^^^2057
1811                     {
1812                         \int_add:Nn \l_@@_primecount_int {3}
1813                         \@@_scanprime_collect:N #1
1814                     }
1815                     {
1816                         \@@_nprimes_select:nn {#1} {\l_@@_primecount_int}
1817                     }
1818                 }
1819             }
1820         }
1821     }
1822 }
1823 }
1824 }
1825 }
1826 }
1827 \cs_new:Npn \@@_scan_backprime:
1828 {
1829     \cs_set_eq:NN \@@_superscript:n \use:n
1830     \int_zero:N \l_@@_primecount_int
1831     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1832 }
1833 \cs_new:Npn \@@_scan_backdprime:
1834 {
1835     \cs_set_eq:NN \@@_superscript:n \use:n
1836     \int_set:Nn \l_@@_primecount_int {1}
1837     \@@_scanbackprime_collect:N \@@_backprime_single_mchar

```

```

1838 }
1839 \cs_new:Npn \@@_scan_backtrprime:
1840 {
1841   \cs_set_eq:NN \@@_superscript:n \use:n
1842   \int_set:Nn \l_@@_primecount_int {2}
1843   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1844 }
1845 \cs_new:Npn \@@_scan_sup_backprime:
1846 {
1847   \int_zero:N \l_@@_primecount_int
1848   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1849 }
1850 \cs_new:Npn \@@_scan_sup_backdprime:
1851 {
1852   \int_set:Nn \l_@@_primecount_int {1}
1853   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1854 }
1855 \cs_new:Npn \@@_scan_sup_backtrprime:
1856 {
1857   \int_set:Nn \l_@@_primecount_int {2}
1858   \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1859 }
1860 \cs_new:Nn \@@_scanbackprime_collect:N
1861 {
1862   \int_incr:N \l_@@_primecount_int
1863   \peek_meaning_remove:NTF `
1864   {
1865     \@@_scanbackprime_collect:N #1
1866   }
1867   {
1868     \peek_meaning_remove:NTF \@@_scan_backprime:
1869     {
1870       \@@_scanbackprime_collect:N #1
1871     }
1872     {
1873       \peek_meaning_remove:NTF ^^^^2035
1874       {
1875         \@@_scanbackprime_collect:N #1
1876       }
1877       {
1878         \peek_meaning_remove:NTF \@@_scan_backdprime:
1879         {
1880           \int_incr:N \l_@@_primecount_int
1881           \@@_scanbackprime_collect:N #1
1882         }
1883         {
1884           \peek_meaning_remove:NTF ^^^^2036
1885           {
1886             \int_incr:N \l_@@_primecount_int

```

```

1887         \@@_scanbackprime_collect:N #1
1888     }
1889     {
1890         \peek_meaning_remove:NTF \@@_scan_backtrprime:
1891         {
1892             \int_add:Nn \l_@@_primecount_int {2}
1893             \@@_scanbackprime_collect:N #1
1894         }
1895         {
1896             \peek_meaning_remove:NTF ^^^^2037
1897             {
1898                 \int_add:Nn \l_@@_primecount_int {2}
1899                 \@@_scanbackprime_collect:N #1
1900             }
1901             {
1902                 \@@_nbackprimes_select:nn {#1} {\l_@@_primecount_int}
1903             }
1904         }
1905     }
1906 }
1907 }
1908 }
1909 }
1910 }

1911 \AtBeginDocument{\@@_define_prime_commands: \@@_define_prime_chars:}
1912 \cs_new:Nn \@@_define_prime_commands:
1913 {
1914     \cs_set_eq:NN \prime \@@_prime_single_mchar
1915     \cs_set_eq:NN \dprime \@@_prime_double_mchar
1916     \cs_set_eq:NN \trprime \@@_prime_triple_mchar
1917     \cs_set_eq:NN \qprime \@@_prime_quad_mchar
1918     \cs_set_eq:NN \backprime \@@_backprime_single_mchar
1919     \cs_set_eq:NN \backdprime \@@_backprime_double_mchar
1920     \cs_set_eq:NN \backtrprime \@@_backprime_triple_mchar
1921 }
1922 \group_begin:
1923     \char_set_catcode_active:N \'
1924     \char_set_catcode_active:N `
1925     \char_set_catcode_active:n {"2032}
1926     \char_set_catcode_active:n {"2033}
1927     \char_set_catcode_active:n {"2034}
1928     \char_set_catcode_active:n {"2057}
1929     \char_set_catcode_active:n {"2035}
1930     \char_set_catcode_active:n {"2036}
1931     \char_set_catcode_active:n {"2037}
1932     \cs_gset:Nn \@@_define_prime_chars:
1933     {
1934         \cs_set_eq:NN ' \@@_scan_sup_prime:
1935         \cs_set_eq:NN ^^^^2032 \@@_scan_sup_prime:

```

```

1936 \cs_set_eq:NN ^^^^2033 \@@_scan_sup_dprime:
1937 \cs_set_eq:NN ^^^^2034 \@@_scan_sup_trprime:
1938 \cs_set_eq:NN ^^^^2057 \@@_scan_sup_qprime:
1939 \cs_set_eq:NN ` \@@_scan_sup_backprime:
1940 \cs_set_eq:NN ^^^^2035 \@@_scan_sup_backprime:
1941 \cs_set_eq:NN ^^^^2036 \@@_scan_sup_backdprime:
1942 \cs_set_eq:NN ^^^^2037 \@@_scan_sup_backtrprime:
1943 }
1944 \group_end:

```

M.2 Unicode radicals

```

1945 \AtBeginDocument{\@@_redefine_radical:}
1946 \cs_new:Nn \@@_redefine_radical:
1947 (*XE)
1948 {
1949 \ifpackageloaded { amsmath } { }
1950 {

```

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

#2 : Leading superscript for the sqrt sign

A re-implementation of L^AT_EX's hard-coded n-root sign using the appropriate \fontdimens.

```

1951 \cs_set_nopar:Npn \r@@@t ##1 ##2
1952 {
1953 \hbox_set:Nn \l_tmpa_box
1954 {
1955 \c_math_toggle_token
1956 \m@th
1957 ##1
1958 \sqrtsign { ##2 }
1959 \c_math_toggle_token
1960 }
1961 \@@_mathstyle_scale:Nnn ##1 { \kern }
1962 { \fontdimen 63 \l_@@_font }
1963 \box_move_up:nn
1964 {
1965 (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
1966 * \number \fontdimen 65 \l_@@_font / 100
1967 }
1968 { \box_use:N \rootbox }
1969 \@@_mathstyle_scale:Nnn ##1 { \kern }
1970 { \fontdimen 64 \l_@@_font }
1971 \box_use_clear:N \l_tmpa_box
1972 }
1973 }
1974 }
1975 (/XE)
1976 (*LU)

```

```

1977 {
1978 \ifpackageloaded { amsmath } { }
1979 {

```

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

```

1980 \cs_set:Npn \root ##1 \of ##2
1981 {
1982 \luatexUroot \l_@@_radical_sqrt_tl { ##1 } { ##2 }
1983 }
1984 }
1985 }
1986 </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.

```

1987 \cs_new:Nn \@@_fontdimen_to_percent:nn
1988 {
1989 \fp_eval:n { \dim_to_decimal:n { \fontdimen #1 #2 } * 65536 / 100 }
1990 }
1991 \cs_new:Nn \@@_fontdimen_to_scale:nn
1992 {
1993 \fp_eval:n { \@@_fontdimen_to_percent:nn {#1} {#2} * \f@size } pt
1994 }

```

`@@_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.

```

1995 \cs_new:Nn \@@_mathstyle_scale:Nnn
1996 {
1997 \ifx#1\scriptstyle
1998 #2 \@@_fontdimen_to_percent:nn {10} \l_@@_font #3
1999 \else
2000 \ifx#1\scriptscriptstyle
2001 #2 \@@_fontdimen_to_percent:nn {11} \l_@@_font #3
2002 \else
2003 #2 #3
2004 \fi
2005 \fi
2006 }

```


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?

2007 \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.

```
2008 \cs_new:Nn \@@_setup_active_superscript:nn
2009 {
2010   \prop_gput:Nn \g_@@_supers_prop   {\meaning #1} {#2}
2011   \char_set_catcode_active:N #1
2012   \@@_char_gmake_mathactive:N #1
2013   \scantokens
2014   {
2015     \cs_gset:Npn #1
2016     {
2017       \tl_set:Nn \l_@@_ss_chain_tl {#2}
2018       \cs_set_eq:NN \@@_sub_or_super:n \sp
2019       \tl_set:Nn \l_@@_tmpa_tl {supers}
2020       \@@_scan_sscript:
2021     }
2022   }
2023 }
```

Bam:

```
2024 \@@_setup_active_superscript:nn {^^^2070} {0}
2025 \@@_setup_active_superscript:nn {^^^00b9} {1}
2026 \@@_setup_active_superscript:nn {^^^00b2} {2}
2027 \@@_setup_active_superscript:nn {^^^00b3} {3}
2028 \@@_setup_active_superscript:nn {^^^2074} {4}
2029 \@@_setup_active_superscript:nn {^^^2075} {5}
2030 \@@_setup_active_superscript:nn {^^^2076} {6}
2031 \@@_setup_active_superscript:nn {^^^2077} {7}
2032 \@@_setup_active_superscript:nn {^^^2078} {8}
2033 \@@_setup_active_superscript:nn {^^^2079} {9}
2034 \@@_setup_active_superscript:nn {^^^207a} {+}
2035 \@@_setup_active_superscript:nn {^^^207b} {-}
```

```

2036 \@@_setup_active_superscript:nn {^^^^207c} {=}
2037 \@@_setup_active_superscript:nn {^^^^207d} {(}
2038 \@@_setup_active_superscript:nn {^^^^207e} {)}
2039 \@@_setup_active_superscript:nn {^^^^2071} {i}
2040 \@@_setup_active_superscript:nn {^^^^207f} {n}
2041 \@@_setup_active_superscript:nn {^^^^02b0} {h}
2042 \@@_setup_active_superscript:nn {^^^^02b2} {j}
2043 \@@_setup_active_superscript:nn {^^^^02b3} {r}
2044 \@@_setup_active_superscript:nn {^^^^02b7} {w}
2045 \@@_setup_active_superscript:nn {^^^^02b8} {y}

```

Subscripts Ditto above.

```

2046 \cs_new:Nn \@@_setup_active_subscript:nn
2047 {
2048   \prop_gput:Non \g_@@_subs_prop   {\meaning #1} {#2}
2049   \char_set_catcode_active:N #1
2050   \@@_char_gmake_mathactive:N #1
2051   \scantokens
2052   {
2053     \cs_gset:Npn #1
2054     {
2055       \tl_set:Nn \l_@@_ss_chain_tl {#2}
2056       \cs_set_eq:NN \@@_sub_or_super:n \sb
2057       \tl_set:Nn \l_@@_tmpa_tl {subs}
2058       \@@_scan_sscript:
2059     }
2060   }
2061 }

```

A few more subscripts than superscripts:

```

2062 \@@_setup_active_subscript:nn {^^^^2080} {0}
2063 \@@_setup_active_subscript:nn {^^^^2081} {1}
2064 \@@_setup_active_subscript:nn {^^^^2082} {2}
2065 \@@_setup_active_subscript:nn {^^^^2083} {3}
2066 \@@_setup_active_subscript:nn {^^^^2084} {4}
2067 \@@_setup_active_subscript:nn {^^^^2085} {5}
2068 \@@_setup_active_subscript:nn {^^^^2086} {6}
2069 \@@_setup_active_subscript:nn {^^^^2087} {7}
2070 \@@_setup_active_subscript:nn {^^^^2088} {8}
2071 \@@_setup_active_subscript:nn {^^^^2089} {9}
2072 \@@_setup_active_subscript:nn {^^^^208a} {+}
2073 \@@_setup_active_subscript:nn {^^^^208b} {-}
2074 \@@_setup_active_subscript:nn {^^^^208c} {=}
2075 \@@_setup_active_subscript:nn {^^^^208d} {(}
2076 \@@_setup_active_subscript:nn {^^^^208e} {)}
2077 \@@_setup_active_subscript:nn {^^^^2090} {a}
2078 \@@_setup_active_subscript:nn {^^^^2091} {e}
2079 \@@_setup_active_subscript:nn {^^^^1d62} {i}
2080 \@@_setup_active_subscript:nn {^^^^2092} {o}

```

```

2081 \@@_setup_active_subscript:nn {^^^1d63} {r}
2082 \@@_setup_active_subscript:nn {^^^1d64} {u}
2083 \@@_setup_active_subscript:nn {^^^1d65} {v}
2084 \@@_setup_active_subscript:nn {^^^2093} {x}
2085 \@@_setup_active_subscript:nn {^^^1d66} {\beta}
2086 \@@_setup_active_subscript:nn {^^^1d67} {\gamma}
2087 \@@_setup_active_subscript:nn {^^^1d68} {\rho}
2088 \@@_setup_active_subscript:nn {^^^1d69} {\phi}
2089 \@@_setup_active_subscript:nn {^^^1d6a} {\chi}
2090 \group_end:

```

The scanning command, evident in its purpose:

```

2091 \cs_new:Npn \@@_scan_sscript:
2092 {
2093   \@@_scan_sscript:TF
2094   {
2095     \@@_scan_sscript:
2096   }
2097   {
2098     \@@_sub_or_super:n {\l_@@_ss_chain_tl}
2099   }
2100 }

```

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.

```

2101 \cs_new:Npn \@@_scan_sscript:TF #1#2
2102 {
2103   \tl_set:Nx \__peek_true_aux:w { \exp_not:n{ #1 } }
2104   \tl_set_eq:NN \__peek_true:w \__peek_true_remove:w
2105   \tl_set:Nx \__peek_false:w { \exp_not:n { \group_align_safe_end: #2 } }
2106   \group_align_safe_begin:
2107   \peek_after:Nw \@@_peek_execute_branches_ss:
2108 }

```

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

```

2109 \cs_new:Npn \@@_peek_execute_branches_ss:
2110 {
2111   \bool_if:nTF
2112   {
2113     \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
2114     \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
2115     \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
2116   }
2117   { \__peek_false:w }
2118   { \@@_peek_execute_branches_ss_aux: }
2119 }

```

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.)

```

2120 \cs_new:Npn \@@_peek_execute_branches_ss_aux:
2121 {
2122   \prop_if_in:coTF
2123     {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token}
2124   {
2125     \prop_get:coN
2126       {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token} \l_@@_tmpb_tl
2127     \tl_put_right:NV \l_@@_ss_chain_tl \l_@@_tmpb_tl
2128     \__peek_true:w
2129   }
2130   { \__peek_false:w }
2131 }

```

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.

```

2132 \cs_new:Npn \@@_define_active_frac:Nw #1 #2/#3
2133 {
2134   \char_set_catcode_active:N #1
2135   \@@_char_gmake_mathactive:N #1
2136   \tl_rescan:nn
2137   {
2138     \catcode`\_ =11\relax
2139     \catcode`\: =11\relax
2140   }
2141   {
2142     \cs_gset:Npx #1
2143     {
2144       \bool_if:NTF \l_@@_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2145       {#2} {#3}
2146     }
2147   }
2148 }

```

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

```

2149 \cs_new:Npn \@@_setup_active_frac:
2150 {
2151   \group_begin:
2152   \@@_define_active_frac:Nw ^^^^2189 0/3
2153   \@@_define_active_frac:Nw ^^^^2152 1/{10}
2154   \@@_define_active_frac:Nw ^^^^2151 1/9
2155   \@@_define_active_frac:Nw ^^^^215b 1/8

```

```

2156 \@@_define_active_frac:Nw ^^^^2150 1/7
2157 \@@_define_active_frac:Nw ^^^^2159 1/6
2158 \@@_define_active_frac:Nw ^^^^2155 1/5
2159 \@@_define_active_frac:Nw ^^^^00bc 1/4
2160 \@@_define_active_frac:Nw ^^^^2153 1/3
2161 \@@_define_active_frac:Nw ^^^^215c 3/8
2162 \@@_define_active_frac:Nw ^^^^2156 2/5
2163 \@@_define_active_frac:Nw ^^^^00bd 1/2
2164 \@@_define_active_frac:Nw ^^^^2157 3/5
2165 \@@_define_active_frac:Nw ^^^^215d 5/8
2166 \@@_define_active_frac:Nw ^^^^2154 2/3
2167 \@@_define_active_frac:Nw ^^^^00be 3/4
2168 \@@_define_active_frac:Nw ^^^^2158 4/5
2169 \@@_define_active_frac:Nw ^^^^215a 5/6
2170 \@@_define_active_frac:Nw ^^^^215e 7/8
2171 \group_end:
2172 }
2173 \@@_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.

```

2174 \def\to{\rightarrow}
2175 \def\le{\leq}
2176 \def\ge{\geq}
2177 \def\neq{\neq}
2178 \def\triangle{\mathord{\bigtriangleup}}
2179 \def\bigcirc{\mdlgwhtcircle}
2180 \def\circ{\vysmwhtcircle}
2181 \def\bullet{\smbllkcircle}
2182 \def\mathyen{\yen}
2183 \def\mathsterling{\sterling}
2184 \def\diamond{\smwhtdiamond}
2185 \def\emptyset{\varnothing}
2186 \def\hbar{\hslash}
2187 \def\land{\wedge}
2188 \def\lor{\vee}
2189 \def\owns{\ni}
2190 \def\gets{\leftarrow}
2191 \def\mathring{\ocirc}
2192 \def\lnot{\neg}
2193 \def\longdivision{\longdivisionsign}

```

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

```

2194 \def\backepsilon{\upbackepsilon}
2195 \def\eth{\matheth}

```

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

```

2196 \def\dbkarow {\dbkarrow}
2197 \def\drbkarow{\drbkarrow}
2198 \def\hksearrow{\hksearrow}
2199 \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:

```

2200 \def\smallint{\mathop{\textstyle\int}\limits}

```

`\underbar`

```

2201 \cs_set_eq:NN \latexe_underbar:n \underbar
2202 \renewcommand\underbar
2203 {
2204   \mode_if_math:TF \mathunderbar \latexe_underbar:n
2205 }

```

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

```

2206 \ifpackageloaded{amsmath}
2207 {
2208   % define their own colon, perhaps I should just steal it. (It does look much bet-
2209   % ter.)
2210 }
2211 {
2212   \cs_set_protected:Npn \colon
2213   {
2214     \bool_if:NTF \g_@@_literal_colon_bool {::} { \mathpunct{:} }
2215   }
2216 }

```

`\mathrm`

```

2216 \let\mathfence\mathord

```

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

```

\Digamma 2217 \def\digamma{\updigamma}
2218 \def\Digamma{\upDigamma}

```

Symbols

```

2219 \cs_set:Npn \l {\Vert}
\mathinner items:
2220 \cs_set:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2221 \cs_set:Npn \cdots {\mathinner{\unicodedots}}
2222 \cs_set_eq:NN \@@_text_slash: \slash
2223 \cs_set_protected:Npn \slash
2224 {
2225   \mode_if_math:TF {\mathslash} {\@@_text_slash:}
2226 }

```

`\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_E_X nor X_Y_T_E_X 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 Gregorio⁵.

```

2227 \cs_new:Npn \@@_newnot:N #1
2228 {
2229   \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
2230   \exp_args:Nx \tl_if_empty:nF { \tl_tail:V \l_not_token_name_tl }
2231   {
2232     \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }
2233   }
2234   \cs_if_exist:cTF { n \l_not_token_name_tl }
2235   {
2236     \use:c { n \l_not_token_name_tl }
2237   }
2238   {
2239     \cs_if_exist:cTF { not \l_not_token_name_tl }
2240     {
2241       \use:c { not \l_not_token_name_tl }
2242     }
2243     {
2244       \@@_oldnot: #1
2245     }
2246   }
2247 }

2248 \cs_set_eq:NN \@@_oldnot: \not
2249 \AtBeginDocument{\cs_set_eq:NN \not \@@_newnot:N}

2250 \cs_new_protected_nopar:Nn \@@_setup_negations:
2251 {
2252   \cs_gset:cpn { not= } { \neq }
2253   \cs_gset:cpn { not< } { \nless }
2254   \cs_gset:cpn { not> } { \ngtr }
2255   \cs_gset:Npn \ngets { \leftarrow }
2256   \cs_gset:Npn \nsimeq { \simeq }
2257   \cs_gset:Npn \nequal { \ne }
2258   \cs_gset:Npn \nle { \nleq }
2259   \cs_gset:Npn \nge { \ngeq }
2260   \cs_gset:Npn \ngreater { \ngtr }
2261   \cs_gset:Npn \nforksnot { \forks }
2262 }

```

⁵<http://tex.stackexchange.com/a/47260/729>

2263 </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.

2264 <*msg>

Wrapper functions:

```
2265 \cs_new:Npn \@@_error:n { \msg_error:nn {unicode-math} }
2266 \cs_new:Npn \@@_warning:n { \msg_warning:nn {unicode-math} }
2267 \cs_new:Npn \@@_warning:nnn { \msg_warning:nnxx {unicode-math} }
2268 \cs_new:Npn \@@_log:n { \msg_log:nn {unicode-math} }
2269 \cs_new:Npn \@@_log:nx { \msg_log:nnx {unicode-math} }

2270 \msg_new:nnn {unicode-math} {no-tfrac}
2271 {
2272   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\!
2273   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
2274 }
2275 \msg_new:nnn {unicode-math} {default-math-font}
2276 {
2277   Defining~ the~ default~ maths~ font~ as~ '\l_@@_fontname_tl'.
2278 }
2279 \msg_new:nnn {unicode-math} {setup-implicit}
2280 {
2281   Setup~ alphabets:~ implicit~ mode.
2282 }
2283 \msg_new:nnn {unicode-math} {setup-explicit}
2284 {
2285   Setup~ alphabets:~ explicit~ mode.
2286 }
2287 \msg_new:nnn {unicode-math} {alph-initialise}
2288 {
2289   Initialising~ \@backslashchar math#1.
2290 }
2291 \msg_new:nnn {unicode-math} {setup-alph}
2292 {
2293   Setup~ alphabet:~ #1.
2294 }
2295 \msg_new:nnn {unicode-math} {no-alphabet}
2296 {
2297   I~ am~ trying~ to~ set~ up~ alphabet~"#1"~ but~ there~ are~ no~ configura-
2298   tion~ settings~ for~ it.
2299   See~ file~ "unicode-math-alphabets.dtx".
2300 }
2301 \msg_new:nnn { unicode-math } { no-named-range }
2302 {
```



```

2302 I'm trying to define new alphabet "#2" in range "#1", but range "#1" hasn't been de-
2303 fined yet.
2304 }
2305 \msg_new:nnn { unicode-math } { missing-alphabets }
2306 {
2307   Missing~math~alphabets~in~font~ "\fontname\l_@@_font" \ \ \
2308   \seq_map_function:NN \l_@@_missing_alph_seq \@@_print_indent:n
2309 }
2310 \cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \ \ }
2311 \msg_new:nnn { unicode-math } { macro-expected }
2312 {
2313   I've expected that #1 is a macro, but it isn't.
2314 }
2315 \msg_new:nnn { unicode-math } { wrong-meaning }
2316 {
2317   I've expected #1 to have the meaning #3, but it has the mean-
2318   ing #2.
2319 }
2320 \msg_new:nnn { unicode-math } { patch-macro }
2321 {
2322   I'm going to patch macro #1.
2323 }
2324 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
2325   Using~ \token_to_str:N \overbracket~ and~
2326   \token_to_str:N \underbracket~ from~
2327   'mathtools'~ package.\ \
2328   \ \
2329   Use~ \token_to_str:N \Uoverbracket~ and~
2330   \token_to_str:N \Uunderbracket~ for~
2331   original~ 'unicode-math'~ definition.
2332 }
2333 \msg_new:nnn { unicode-math } { mathtools-colon } {
2334   I'm going to overwrite the following commands from
2335   the 'mathtools' package: \ \ \
2336   \ \ \ \ \token_to_str:N \dblcolon,~
2337   \token_to_str:N \coloneqq,~
2338   \token_to_str:N \Coloneqq,~
2339   \token_to_str:N \eqqcolon. \ \ \
2340   Note that since I won't overwrite the other colon-like
2341   commands, using them will lead to inconsistencies.
2342 }
2343 \msg_new:nnn { unicode-math } { colonequals } {
2344   I'm going to overwrite the following commands from
2345   the 'colonequals' package: \ \ \
2346   \ \ \ \ \token_to_str:N \ratio,~
2347   \token_to_str:N \coloncolon,~
2348   \token_to_str:N \minuscolon, \ \
2349   \ \ \ \ \token_to_str:N \colonequals,~
2350   \token_to_str:N \equalscolon,~

```

```

2349         \token_to_str:N \coloncolonequals. \\\
2350 Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
2351 commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
2352 Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl
2353 or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
2354 any~ effect~ on~ the~ re-defined~ commands.
2355 }
2356 </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.⁶

```

2357 <*usv>

```

Alphabets

```

2358 \usv_set:nnn {normal} {num} {48}
2359 \usv_set:nnn {normal} {Latin}{1D434}
2360 \usv_set:nnn {normal} {latin}{1D44E}
2361 \usv_set:nnn {normal} {Greek}{1D6E2}
2362 \usv_set:nnn {normal} {greek}{1D6FC}
2363 \usv_set:nnn {normal}{varTheta} {1D6F3}
2364 \usv_set:nnn {normal}{varepsilon}{1D716}
2365 \usv_set:nnn {normal}{vartheta} {1D717}
2366 \usv_set:nnn {normal}{varkappa} {1D718}
2367 \usv_set:nnn {normal}{varphi} {1D719}
2368 \usv_set:nnn {normal}{varrho} {1D71A}
2369 \usv_set:nnn {normal}{varpi} {1D71B}
2370 \usv_set:nnn {normal} {Nabla}{1D6FB}
2371 \usv_set:nnn {normal} {partial}{1D715}
2372
2373 \usv_set:nnn {up} {num} {48}
2374 \usv_set:nnn {up} {Latin}{65}
2375 \usv_set:nnn {up} {latin}{97}
2376 \usv_set:nnn {up} {Greek}{391}
2377 \usv_set:nnn {up} {greek}{3B1}
2378 \usv_set:nnn {it} {Latin}{1D434}
2379 \usv_set:nnn {it} {latin}{1D44E}
2380 \usv_set:nnn {it} {Greek}{1D6E2}
2381 \usv_set:nnn {it} {greek}{1D6FC}
2382 \usv_set:nnn {bb} {num} {1D7D8}
2383 \usv_set:nnn {bb} {Latin}{1D538}
2384 \usv_set:nnn {bb} {latin}{1D552}
2385 \usv_set:nnn {scr} {Latin}{1D49C}
2386 \usv_set:nnn {cal} {Latin}{1D49C}
2387 \usv_set:nnn {scr} {latin}{1D4B6}

```

⁶'u.s.v.' stands for 'Unicode scalar value'.

2388 \usv_set:nnn {frak}{Latin}{ "1D504}
 2389 \usv_set:nnn {frak}{latin}{ "1D51E}
 2390 \usv_set:nnn {sf} {num} { "1D7E2}
 2391 \usv_set:nnn {sfup}{num} { "1D7E2}
 2392 \usv_set:nnn {sfit}{num} { "1D7E2}
 2393 \usv_set:nnn {sfup}{Latin}{ "1D5A0}
 2394 \usv_set:nnn {sf} {Latin}{ "1D5A0}
 2395 \usv_set:nnn {sfup}{latin}{ "1D5BA}
 2396 \usv_set:nnn {sf} {latin}{ "1D5BA}
 2397 \usv_set:nnn {sfit}{Latin}{ "1D608}
 2398 \usv_set:nnn {sfit}{latin}{ "1D622}
 2399 \usv_set:nnn {tt} {num} { "1D7F6}
 2400 \usv_set:nnn {tt} {Latin}{ "1D670}
 2401 \usv_set:nnn {tt} {latin}{ "1D68A}

Bold:

2402 \usv_set:nnn {bf} {num} { "1D7CE}
 2403 \usv_set:nnn {bfup} {num} { "1D7CE}
 2404 \usv_set:nnn {bfit} {num} { "1D7CE}
 2405 \usv_set:nnn {bfup} {Latin}{ "1D400}
 2406 \usv_set:nnn {bfup} {latin}{ "1D41A}
 2407 \usv_set:nnn {bfup} {Greek}{ "1D6A8}
 2408 \usv_set:nnn {bfup} {greek}{ "1D6C2}
 2409 \usv_set:nnn {bfit} {Latin}{ "1D468}
 2410 \usv_set:nnn {bfit} {latin}{ "1D482}
 2411 \usv_set:nnn {bfit} {Greek}{ "1D71C}
 2412 \usv_set:nnn {bfit} {greek}{ "1D736}
 2413 \usv_set:nnn {bffrak}{Latin}{ "1D56C}
 2414 \usv_set:nnn {bffrak}{latin}{ "1D586}
 2415 \usv_set:nnn {bfscr} {Latin}{ "1D4D0}
 2416 \usv_set:nnn {bfcal} {Latin}{ "1D4D0}
 2417 \usv_set:nnn {bfscr} {latin}{ "1D4EA}
 2418 \usv_set:nnn {bfsf} {num} { "1D7EC}
 2419 \usv_set:nnn {bfsfup}{num} { "1D7EC}
 2420 \usv_set:nnn {bfsfit}{num} { "1D7EC}
 2421 \usv_set:nnn {bfsfup}{Latin}{ "1D5D4}
 2422 \usv_set:nnn {bfsfup}{latin}{ "1D5EE}
 2423 \usv_set:nnn {bfsfup}{Greek}{ "1D756}
 2424 \usv_set:nnn {bfsfup}{greek}{ "1D770}
 2425 \usv_set:nnn {bfsfit}{Latin}{ "1D63C}
 2426 \usv_set:nnn {bfsfit}{latin}{ "1D656}
 2427 \usv_set:nnn {bfsfit}{Greek}{ "1D790}
 2428 \usv_set:nnn {bfsfit}{greek}{ "1D7AA}

2429 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_@@_upLatin_bool \g_@@_bfsfup_Latin_usv \g_@@_bfsfit_Lat
 2430 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_@@_uplatin_bool \g_@@_bfsfup_latin_usv \g_@@_bfsfit_lat
 2431 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_@@_upGreek_bool \g_@@_bfsfup_Greek_usv \g_@@_bfsfit_Gre
 2432 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_@@_upgreek_bool \g_@@_bfsfup_greek_usv \g_@@_bfsfit_gre
 2433 \usv_set:nnn {bf} {Latin}{ \bool_if:NTF \g_@@_bfupLatin_bool \g_@@_bfup_Latin_usv \g_@@_bfit_Latin
 2434 \usv_set:nnn {bf} {latin}{ \bool_if:NTF \g_@@_bfuplatin_bool \g_@@_bfup_latin_usv \g_@@_bfit_latin

2435 \usv_set:nnn {bf} {Greek}{ \bool_if:NTF \g_@@_bfupGreek_bool \g_@@_bfup_Greek_usv \g_@@_bfit_Greek_

2436 \usv_set:nnn {bf} {greek}{ \bool_if:NTF \g_@@_bfupgreek_bool \g_@@_bfup_greek_usv \g_@@_bfit_greek_

Greek variants:

2437 \usv_set:nnn {up}{varTheta} {"3F4}

2438 \usv_set:nnn {up}{Digamma} {"3DC}

2439 \usv_set:nnn {up}{varepsilon}{ "3F5}

2440 \usv_set:nnn {up}{vartheta} {"3D1}

2441 \usv_set:nnn {up}{varkappa} {"3F0}

2442 \usv_set:nnn {up}{varphi} {"3D5}

2443 \usv_set:nnn {up}{varrho} {"3F1}

2444 \usv_set:nnn {up}{varpi} {"3D6}

2445 \usv_set:nnn {up}{digamma} {"3DD}

Bold:

2446 \usv_set:nnn {bfup}{varTheta} {"1D6B9}

2447 \usv_set:nnn {bfup}{Digamma} {"1D7CA}

2448 \usv_set:nnn {bfup}{varepsilon}{ "1D6DC}

2449 \usv_set:nnn {bfup}{vartheta} {"1D6DD}

2450 \usv_set:nnn {bfup}{varkappa} {"1D6DE}

2451 \usv_set:nnn {bfup}{varphi} {"1D6DF}

2452 \usv_set:nnn {bfup}{varrho} {"1D6E0}

2453 \usv_set:nnn {bfup}{varpi} {"1D6E1}

2454 \usv_set:nnn {bfup}{digamma} {"1D7CB}

Italic Greek variants:

2455 \usv_set:nnn {it}{varTheta} {"1D6F3}

2456 \usv_set:nnn {it}{varepsilon}{ "1D716}

2457 \usv_set:nnn {it}{vartheta} {"1D717}

2458 \usv_set:nnn {it}{varkappa} {"1D718}

2459 \usv_set:nnn {it}{varphi} {"1D719}

2460 \usv_set:nnn {it}{varrho} {"1D71A}

2461 \usv_set:nnn {it}{varpi} {"1D71B}

Bold italic:

2462 \usv_set:nnn {bfit}{varTheta} {"1D72D}

2463 \usv_set:nnn {bfit}{varepsilon}{ "1D750}

2464 \usv_set:nnn {bfit}{vartheta} {"1D751}

2465 \usv_set:nnn {bfit}{varkappa} {"1D752}

2466 \usv_set:nnn {bfit}{varphi} {"1D753}

2467 \usv_set:nnn {bfit}{varrho} {"1D754}

2468 \usv_set:nnn {bfit}{varpi} {"1D755}

Bold sans:

2469 \usv_set:nnn {bfsfup}{varTheta} {"1D767}

2470 \usv_set:nnn {bfsfup}{varepsilon}{ "1D78A}

2471 \usv_set:nnn {bfsfup}{vartheta} {"1D78B}

2472 \usv_set:nnn {bfsfup}{varkappa} {"1D78C}

2473 \usv_set:nnn {bfsfup}{varphi} {"1D78D}

2474 \usv_set:nnn {bfsfup}{varrho} {"1D78E}

2475 \usv_set:nnn {bfsfup}{varpi} {"1D78F}

Bold sans italic:

```
2476 \usv_set:nnn {bfsfit}{varTheta} {"1D7A1}
2477 \usv_set:nnn {bfsfit}{varepsilon} {"1D7C4}
2478 \usv_set:nnn {bfsfit}{vartheta} {"1D7C5}
2479 \usv_set:nnn {bfsfit}{varkappa} {"1D7C6}
2480 \usv_set:nnn {bfsfit}{varphi} {"1D7C7}
2481 \usv_set:nnn {bfsfit}{varrho} {"1D7C8}
2482 \usv_set:nnn {bfsfit}{varpi} {"1D7C9}
```

Nabla:

```
2483 \usv_set:nnn {up} {Nabla} {"02207}
2484 \usv_set:nnn {it} {Nabla} {"1D6FB}
2485 \usv_set:nnn {bfup} {Nabla} {"1D6C1}
2486 \usv_set:nnn {bfit} {Nabla} {"1D735}
2487 \usv_set:nnn {bfsfup}{Nabla} {"1D76F}
2488 \usv_set:nnn {bfsfit}{Nabla} {"1D7A9}
```

Partial:

```
2489 \usv_set:nnn {up} {partial} {"02202}
2490 \usv_set:nnn {it} {partial} {"1D715}
2491 \usv_set:nnn {bfup} {partial} {"1D6DB}
2492 \usv_set:nnn {bfit} {partial} {"1D74F}
2493 \usv_set:nnn {bfsfup}{partial} {"1D789}
2494 \usv_set:nnn {bfsfit}{partial} {"1D7C3}
```

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

```
2495 \usv_set:nnn {up}{B}{'\B}
2496 \usv_set:nnn {up}{C}{'\C}
2497 \usv_set:nnn {up}{D}{'\D}
2498 \usv_set:nnn {up}{E}{'\E}
2499 \usv_set:nnn {up}{F}{'\F}
2500 \usv_set:nnn {up}{H}{'\H}
2501 \usv_set:nnn {up}{I}{'\I}
2502 \usv_set:nnn {up}{L}{'\L}
2503 \usv_set:nnn {up}{M}{'\M}
2504 \usv_set:nnn {up}{N}{'\N}
2505 \usv_set:nnn {up}{P}{'\P}
2506 \usv_set:nnn {up}{Q}{'\Q}
2507 \usv_set:nnn {up}{R}{'\R}
2508 \usv_set:nnn {up}{Z}{'\Z}

2509 \usv_set:nnn {it}{B} {"1D435}
2510 \usv_set:nnn {it}{C} {"1D436}
2511 \usv_set:nnn {it}{D} {"1D437}
2512 \usv_set:nnn {it}{E} {"1D438}
2513 \usv_set:nnn {it}{F} {"1D439}
2514 \usv_set:nnn {it}{H} {"1D43B}
2515 \usv_set:nnn {it}{I} {"1D43C}
2516 \usv_set:nnn {it}{L} {"1D43F}
```

2517 \usv_set:nnn {it}{M}{1D440}
 2518 \usv_set:nnn {it}{N}{1D441}
 2519 \usv_set:nnn {it}{P}{1D443}
 2520 \usv_set:nnn {it}{Q}{1D444}
 2521 \usv_set:nnn {it}{R}{1D445}
 2522 \usv_set:nnn {it}{Z}{1D44D}

 2523 \usv_set:nnn {up}{d}{\d}
 2524 \usv_set:nnn {up}{e}{\e}
 2525 \usv_set:nnn {up}{g}{\g}
 2526 \usv_set:nnn {up}{h}{\h}
 2527 \usv_set:nnn {up}{i}{\i}
 2528 \usv_set:nnn {up}{j}{\j}
 2529 \usv_set:nnn {up}{o}{\o}

 2530 \usv_set:nnn {it}{d}{1D451}
 2531 \usv_set:nnn {it}{e}{1D452}
 2532 \usv_set:nnn {it}{g}{1D454}
 2533 \usv_set:nnn {it}{h}{0210E}
 2534 \usv_set:nnn {it}{i}{1D456}
 2535 \usv_set:nnn {it}{j}{1D457}
 2536 \usv_set:nnn {it}{o}{1D45C}

Latin ‘h’:

2537 \usv_set:nnn {bb} {h}{1D559}
 2538 \usv_set:nnn {tt} {h}{1D691}
 2539 \usv_set:nnn {scr} {h}{1D4BD}
 2540 \usv_set:nnn {frak} {h}{1D525}
 2541 \usv_set:nnn {bfup} {h}{1D421}
 2542 \usv_set:nnn {bfit} {h}{1D489}
 2543 \usv_set:nnn {sfup} {h}{1D5C1}
 2544 \usv_set:nnn {sfit} {h}{1D629}
 2545 \usv_set:nnn {bffrak}{h}{1D58D}
 2546 \usv_set:nnn {bfscr} {h}{1D4F1}
 2547 \usv_set:nnn {bfsfup}{h}{1D5F5}
 2548 \usv_set:nnn {bfsfit}{h}{1D65D}

Dotless ‘i’ and ‘j’:

2549 \usv_set:nnn {up}{dotlessi}{00131}
 2550 \usv_set:nnn {up}{dotlessj}{00237}
 2551 \usv_set:nnn {it}{dotlessi}{1D6A4}
 2552 \usv_set:nnn {it}{dotlessj}{1D6A5}

Blackboard:

2553 \usv_set:nnn {bb}{C}{2102}
 2554 \usv_set:nnn {bb}{H}{210D}
 2555 \usv_set:nnn {bb}{N}{2115}
 2556 \usv_set:nnn {bb}{P}{2119}
 2557 \usv_set:nnn {bb}{Q}{211A}
 2558 \usv_set:nnn {bb}{R}{211D}
 2559 \usv_set:nnn {bb}{Z}{2124}
 2560 \usv_set:nnn {up}{Pi} {003A0}

2561 \usv_set:nnn {up}{pi} {"003C0}
 2562 \usv_set:nnn {up}{Gamma} {"00393}
 2563 \usv_set:nnn {up}{gamma} {"003B3}
 2564 \usv_set:nnn {up}{summation}{"02211}
 2565 \usv_set:nnn {it}{Pi} {"1D6F1}
 2566 \usv_set:nnn {it}{pi} {"1D70B}
 2567 \usv_set:nnn {it}{Gamma} {"1D6E4}
 2568 \usv_set:nnn {it}{gamma} {"1D6FE}
 2569 \usv_set:nnn {bb}{Pi} {"0213F}
 2570 \usv_set:nnn {bb}{pi} {"0213C}
 2571 \usv_set:nnn {bb}{Gamma} {"0213E}
 2572 \usv_set:nnn {bb}{gamma} {"0213D}
 2573 \usv_set:nnn {bb}{summation}{"02140}

Italic blackboard:

2574 \usv_set:nnn {bbit}{D}{"2145}
 2575 \usv_set:nnn {bbit}{d}{"2146}
 2576 \usv_set:nnn {bbit}{e}{"2147}
 2577 \usv_set:nnn {bbit}{i}{"2148}
 2578 \usv_set:nnn {bbit}{j}{"2149}

Script exceptions:

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

 2590 \usv_set:nnn {cal}{B}{"212C}
 2591 \usv_set:nnn {cal}{E}{"2130}
 2592 \usv_set:nnn {cal}{F}{"2131}
 2593 \usv_set:nnn {cal}{H}{"210B}
 2594 \usv_set:nnn {cal}{I}{"2110}
 2595 \usv_set:nnn {cal}{L}{"2112}
 2596 \usv_set:nnn {cal}{M}{"2133}
 2597 \usv_set:nnn {cal}{R}{"211B}

Fraktur exceptions:

2598 \usv_set:nnn {frak}{C}{"212D}
 2599 \usv_set:nnn {frak}{H}{"210C}
 2600 \usv_set:nnn {frak}{I}{"2111}
 2601 \usv_set:nnn {frak}{R}{"211C}
 2602 \usv_set:nnn {frak}{Z}{"2128}

 2603 < *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.

2604 *(*stix)*

Upright

```
2605 \usv_set:nnn {stixsfup}{partial}{ "E17C}
2606 \usv_set:nnn {stixsfup}{Greek}{ "E17D}
2607 \usv_set:nnn {stixsfup}{greek}{ "E196}
2608 \usv_set:nnn {stixsfup}{varTheta}{ "E18E}
2609 \usv_set:nnn {stixsfup}{varepsilon}{ "E1AF}
2610 \usv_set:nnn {stixsfup}{vartheta}{ "E1B0}
2611 \usv_set:nnn {stixsfup}{varkappa}{0000} % ???
2612 \usv_set:nnn {stixsfup}{varphi}{ "E1B1}
2613 \usv_set:nnn {stixsfup}{varrho}{ "E1B2}
2614 \usv_set:nnn {stixsfup}{varpi}{ "E1B3}
2615 \usv_set:nnn {stixupslash}{Greek}{ "E2FC}
```

Italic

```
2616 \usv_set:nnn {stixbbit}{A}{ "E154}
2617 \usv_set:nnn {stixbbit}{B}{ "E155}
2618 \usv_set:nnn {stixbbit}{E}{ "E156}
2619 \usv_set:nnn {stixbbit}{F}{ "E157}
2620 \usv_set:nnn {stixbbit}{G}{ "E158}
2621 \usv_set:nnn {stixbbit}{I}{ "E159}
2622 \usv_set:nnn {stixbbit}{J}{ "E15A}
2623 \usv_set:nnn {stixbbit}{K}{ "E15B}
2624 \usv_set:nnn {stixbbit}{L}{ "E15C}
2625 \usv_set:nnn {stixbbit}{M}{ "E15D}
2626 \usv_set:nnn {stixbbit}{O}{ "E15E}
2627 \usv_set:nnn {stixbbit}{S}{ "E15F}
2628 \usv_set:nnn {stixbbit}{T}{ "E160}
2629 \usv_set:nnn {stixbbit}{U}{ "E161}
2630 \usv_set:nnn {stixbbit}{V}{ "E162}
2631 \usv_set:nnn {stixbbit}{W}{ "E163}
2632 \usv_set:nnn {stixbbit}{X}{ "E164}
2633 \usv_set:nnn {stixbbit}{Y}{ "E165}

2634 \usv_set:nnn {stixbbit}{a}{ "E166}
2635 \usv_set:nnn {stixbbit}{b}{ "E167}
2636 \usv_set:nnn {stixbbit}{c}{ "E168}
2637 \usv_set:nnn {stixbbit}{f}{ "E169}
2638 \usv_set:nnn {stixbbit}{g}{ "E16A}
2639 \usv_set:nnn {stixbbit}{h}{ "E16B}
2640 \usv_set:nnn {stixbbit}{k}{ "E16C}
2641 \usv_set:nnn {stixbbit}{l}{ "E16D}
```


2642 \usv_set:nnn {stixbbit}{m}{ "E16E}
 2643 \usv_set:nnn {stixbbit}{n}{ "E16F}
 2644 \usv_set:nnn {stixbbit}{o}{ "E170}
 2645 \usv_set:nnn {stixbbit}{p}{ "E171}
 2646 \usv_set:nnn {stixbbit}{q}{ "E172}
 2647 \usv_set:nnn {stixbbit}{r}{ "E173}
 2648 \usv_set:nnn {stixbbit}{s}{ "E174}
 2649 \usv_set:nnn {stixbbit}{t}{ "E175}
 2650 \usv_set:nnn {stixbbit}{u}{ "E176}
 2651 \usv_set:nnn {stixbbit}{v}{ "E177}
 2652 \usv_set:nnn {stixbbit}{w}{ "E178}
 2653 \usv_set:nnn {stixbbit}{x}{ "E179}
 2654 \usv_set:nnn {stixbbit}{y}{ "E17A}
 2655 \usv_set:nnn {stixbbit}{z}{ "E17B}

 2656 \usv_set:nnn {stixsfit}{Numerals}{ "E1B4}
 2657 \usv_set:nnn {stixsfit}{partial}{ "E1BE}
 2658 \usv_set:nnn {stixsfit}{Greek}{ "E1BF}
 2659 \usv_set:nnn {stixsfit}{greek}{ "E1D8}
 2660 \usv_set:nnn {stixsfit}{varTheta}{ "E1D0}
 2661 \usv_set:nnn {stixsfit}{varepsilon}{ "E1F1}
 2662 \usv_set:nnn {stixsfit}{vartheta}{ "E1F2}
 2663 \usv_set:nnn {stixsfit}{varkappa}{0000} % ???
 2664 \usv_set:nnn {stixsfit}{varphi}{ "E1F3}
 2665 \usv_set:nnn {stixsfit}{varrho}{ "E1F4}
 2666 \usv_set:nnn {stixsfit}{varpi}{ "E1F5}

 2667 \usv_set:nnn {stixcal}{Latin}{ "E22D}
 2668 \usv_set:nnn {stixcal}{num}{ "E262}
 2669 \usv_set:nnn {scr}{num}{48}
 2670 \usv_set:nnn {it}{num}{48}

 2671 \usv_set:nnn {stixsfitslash}{Latin}{ "E294}
 2672 \usv_set:nnn {stixsfitslash}{latin}{ "E2C8}
 2673 \usv_set:nnn {stixsfitslash}{greek}{ "E32C}
 2674 \usv_set:nnn {stixsfitslash}{varepsilon}{ "E37A}
 2675 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35E}
 2676 \usv_set:nnn {stixsfitslash}{varkappa}{ "E374}
 2677 \usv_set:nnn {stixsfitslash}{varphi}{ "E360}
 2678 \usv_set:nnn {stixsfitslash}{varrho}{ "E376}
 2679 \usv_set:nnn {stixsfitslash}{varpi}{ "E362}
 2680 \usv_set:nnn {stixsfitslash}{digamma}{ "E36A}

Bold

2681 \usv_set:nnn {stixbfupslash}{Greek}{ "E2FD}
 2682 \usv_set:nnn {stixbfupslash}{Digamma}{ "E369}

 2683 \usv_set:nnn {stixbfbb}{A}{ "E38A}
 2684 \usv_set:nnn {stixbfbb}{B}{ "E38B}
 2685 \usv_set:nnn {stixbfbb}{E}{ "E38D}
 2686 \usv_set:nnn {stixbfbb}{F}{ "E38E}

2687 \usv_set:nnn {stixbfbb}{G}{ "E38F}
 2688 \usv_set:nnn {stixbfbb}{I}{ "E390}
 2689 \usv_set:nnn {stixbfbb}{J}{ "E391}
 2690 \usv_set:nnn {stixbfbb}{K}{ "E392}
 2691 \usv_set:nnn {stixbfbb}{L}{ "E393}
 2692 \usv_set:nnn {stixbfbb}{M}{ "E394}
 2693 \usv_set:nnn {stixbfbb}{O}{ "E395}
 2694 \usv_set:nnn {stixbfbb}{S}{ "E396}
 2695 \usv_set:nnn {stixbfbb}{T}{ "E397}
 2696 \usv_set:nnn {stixbfbb}{U}{ "E398}
 2697 \usv_set:nnn {stixbfbb}{V}{ "E399}
 2698 \usv_set:nnn {stixbfbb}{W}{ "E39A}
 2699 \usv_set:nnn {stixbfbb}{X}{ "E39B}
 2700 \usv_set:nnn {stixbfbb}{Y}{ "E39C}

 2701 \usv_set:nnn {stixbfbb}{a}{ "E39D}
 2702 \usv_set:nnn {stixbfbb}{b}{ "E39E}
 2703 \usv_set:nnn {stixbfbb}{c}{ "E39F}
 2704 \usv_set:nnn {stixbfbb}{f}{ "E3A2}
 2705 \usv_set:nnn {stixbfbb}{g}{ "E3A3}
 2706 \usv_set:nnn {stixbfbb}{h}{ "E3A4}
 2707 \usv_set:nnn {stixbfbb}{k}{ "E3A7}
 2708 \usv_set:nnn {stixbfbb}{l}{ "E3A8}
 2709 \usv_set:nnn {stixbfbb}{m}{ "E3A9}
 2710 \usv_set:nnn {stixbfbb}{n}{ "E3AA}
 2711 \usv_set:nnn {stixbfbb}{o}{ "E3AB}
 2712 \usv_set:nnn {stixbfbb}{p}{ "E3AC}
 2713 \usv_set:nnn {stixbfbb}{q}{ "E3AD}
 2714 \usv_set:nnn {stixbfbb}{r}{ "E3AE}
 2715 \usv_set:nnn {stixbfbb}{s}{ "E3AF}
 2716 \usv_set:nnn {stixbfbb}{t}{ "E3B0}
 2717 \usv_set:nnn {stixbfbb}{u}{ "E3B1}
 2718 \usv_set:nnn {stixbfbb}{v}{ "E3B2}
 2719 \usv_set:nnn {stixbfbb}{w}{ "E3B3}
 2720 \usv_set:nnn {stixbfbb}{x}{ "E3B4}
 2721 \usv_set:nnn {stixbfbb}{y}{ "E3B5}
 2722 \usv_set:nnn {stixbfbb}{z}{ "E3B6}

 2723 \usv_set:nnn {stixbfsfup}{Numerals}{ "E3B7}

Bold Italic

2724 \usv_set:nnn {stixbfsfit}{Numerals}{ "E1F6}

 2725 \usv_set:nnn {stixbfbbbit}{A}{ "E200}
 2726 \usv_set:nnn {stixbfbbbit}{B}{ "E201}
 2727 \usv_set:nnn {stixbfbbbit}{E}{ "E203}
 2728 \usv_set:nnn {stixbfbbbit}{F}{ "E204}
 2729 \usv_set:nnn {stixbfbbbit}{G}{ "E205}
 2730 \usv_set:nnn {stixbfbbbit}{I}{ "E206}
 2731 \usv_set:nnn {stixbfbbbit}{J}{ "E207}
 2732 \usv_set:nnn {stixbfbbbit}{K}{ "E208}

```

2733 \usv_set:nnn {stixbfbbbit}{L}{ "E209}
2734 \usv_set:nnn {stixbfbbbit}{M}{ "E20A}
2735 \usv_set:nnn {stixbfbbbit}{O}{ "E20B}
2736 \usv_set:nnn {stixbfbbbit}{S}{ "E20C}
2737 \usv_set:nnn {stixbfbbbit}{T}{ "E20D}
2738 \usv_set:nnn {stixbfbbbit}{U}{ "E20E}
2739 \usv_set:nnn {stixbfbbbit}{V}{ "E20F}
2740 \usv_set:nnn {stixbfbbbit}{W}{ "E210}
2741 \usv_set:nnn {stixbfbbbit}{X}{ "E211}
2742 \usv_set:nnn {stixbfbbbit}{Y}{ "E212}

2743 \usv_set:nnn {stixbfbbbit}{a}{ "E213}
2744 \usv_set:nnn {stixbfbbbit}{b}{ "E214}
2745 \usv_set:nnn {stixbfbbbit}{c}{ "E215}
2746 \usv_set:nnn {stixbfbbbit}{e}{ "E217}
2747 \usv_set:nnn {stixbfbbbit}{f}{ "E218}
2748 \usv_set:nnn {stixbfbbbit}{g}{ "E219}
2749 \usv_set:nnn {stixbfbbbit}{h}{ "E21A}
2750 \usv_set:nnn {stixbfbbbit}{k}{ "E21D}
2751 \usv_set:nnn {stixbfbbbit}{l}{ "E21E}
2752 \usv_set:nnn {stixbfbbbit}{m}{ "E21F}
2753 \usv_set:nnn {stixbfbbbit}{n}{ "E220}
2754 \usv_set:nnn {stixbfbbbit}{o}{ "E221}
2755 \usv_set:nnn {stixbfbbbit}{p}{ "E222}
2756 \usv_set:nnn {stixbfbbbit}{q}{ "E223}
2757 \usv_set:nnn {stixbfbbbit}{r}{ "E224}
2758 \usv_set:nnn {stixbfbbbit}{s}{ "E225}
2759 \usv_set:nnn {stixbfbbbit}{t}{ "E226}
2760 \usv_set:nnn {stixbfbbbit}{u}{ "E227}
2761 \usv_set:nnn {stixbfbbbit}{v}{ "E228}
2762 \usv_set:nnn {stixbfbbbit}{w}{ "E229}
2763 \usv_set:nnn {stixbfbbbit}{x}{ "E22A}
2764 \usv_set:nnn {stixbfbbbit}{y}{ "E22B}
2765 \usv_set:nnn {stixbfbbbit}{z}{ "E22C}

2766 \usv_set:nnn {stixbfcal}{Latin}{ "E247}

2767 \usv_set:nnn {stixbfitslash}{Latin}{ "E295}
2768 \usv_set:nnn {stixbfitslash}{latin}{ "E2C9}
2769 \usv_set:nnn {stixbfitslash}{greek}{ "E32D}
2770 \usv_set:nnn {stixsfitslash}{varepsilon}{ "E37B}
2771 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35F}
2772 \usv_set:nnn {stixsfitslash}{varkappa}{ "E375}
2773 \usv_set:nnn {stixsfitslash}{varphi}{ "E361}
2774 \usv_set:nnn {stixsfitslash}{varrho}{ "E377}
2775 \usv_set:nnn {stixsfitslash}{varpi}{ "E363}
2776 \usv_set:nnn {stixsfitslash}{digamma}{ "E36B}

2777 (/stix)

```

N.3 Alphabets

2778 *(*alphabets)*

N.3.1 Upright: up

```
2779 \@@_new_alphabet_config:nnn {up} {num}
2780 {
2781   \@@_set_normal_numbers:nn {up} {#1}
2782   \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
2783 }
2784
2785 \@@_new_alphabet_config:nnn {up} {Latin}
2786 {
2787   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
2788   {
2789     \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2790   }
2791   \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
2792   \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
2793   \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
2794 }
2795
2796 \@@_new_alphabet_config:nnn {up} {latin}
2797 {
2798   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
2799   {
2800     \bool_if:NT \g_@@_uplatin_bool
2801     {
2802       \@@_set_normal_latin:nn {up,it} {#1}
2803       \@@_set_normal_char:nnn {h} {up,it} {#1}
2804       \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
2805       \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
2806     }
2807   }
2808   \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
2809   \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
2810   \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
2811 }
2812
2813 \@@_new_alphabet_config:nnn {up} {Greek}
2814 {
2815   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
2816   {
2817     \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2818   }
2819   \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
2820   \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
2821   \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
2822 }
2823
2824 \@@_new_alphabet_config:nnn {up} {greek}
```

```

2825 {
2826 \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
2827 {
2828 \bool_if:NT \g_@@_upgreek_bool
2829 {
2830 \@@_set_normal_greek:nn {up,it} {#1}
2831 }
2832 }
2833 \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
2834 \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
2835 \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
2836 }
2837
2838 \@@_new_alphabet_config:nnn {up} {misc}
2839 {
2840 \bool_if:NTF \g_@@_literal_Nabla_bool
2841 {
2842 \@@_set_normal_char:nnn {Nabla}{up}{up}
2843 }
2844 {
2845 \bool_if:NT \g_@@_upNabla_bool
2846 {
2847 \@@_set_normal_char:nnn {Nabla}{up,it}{up}
2848 }
2849 }
2850 \bool_if:NTF \g_@@_literal_partial_bool
2851 {
2852 \@@_set_normal_char:nnn {partial}{up}{up}
2853 }
2854 {
2855 \bool_if:NT \g_@@_uppartial_bool
2856 {
2857 \@@_set_normal_char:nnn {partial}{up,it}{up}
2858 }
2859 }
2860 \@@_set_mathalphabet_pos:nnnn {up} {partial} {up,it} {#1}
2861 \@@_set_mathalphabet_pos:nnnn {up} {Nabla} {up,it} {#1}
2862 \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
2863 \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
2864 }

```

N.3.2 *Italic: it*

```

2865 \@@_new_alphabet_config:nnn {it} {Latin}
2866 {
2867 \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
2868 {
2869 \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2870 }
2871 \@@_set_mathalphabet_Latin:nnn {it}{up,it}{#1}

```

```

2872 }
2873
2874 \@@_new_alphabet_config:nnn {it} {latin}
2875 {
2876   \bool_if:NTF \g_@@_literal_bool
2877   {
2878     \@@_set_normal_latin:nn {it} {#1}
2879     \@@_set_normal_char:nnn {h}{it}{#1}
2880   }
2881   {
2882     \bool_if:NF \g_@@_uplatin_bool
2883     {
2884       \@@_set_normal_latin:nn {up,it} {#1}
2885       \@@_set_normal_char:nnn {h}{up,it}{#1}
2886       \@@_set_normal_char:nnn {dotlessi}{up,it}{#1}
2887       \@@_set_normal_char:nnn {dotlessj}{up,it}{#1}
2888     }
2889   }
2890   \@@_set_mathalphabet_latin:nnn {it} {up,it} {#1}
2891   \@@_set_mathalphabet_pos:nnnn {it} {dotlessi} {up,it} {#1}
2892   \@@_set_mathalphabet_pos:nnnn {it} {dotlessj} {up,it} {#1}
2893 }
2894
2895 \@@_new_alphabet_config:nnn {it} {Greek}
2896 {
2897   \bool_if:NTF \g_@@_literal_bool
2898   {
2899     \@@_set_normal_Greek:nn {it}{#1}
2900   }
2901   {
2902     \bool_if:NF \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2903   }
2904   \@@_set_mathalphabet_Greek:nnn {it} {up,it}{#1}
2905 }
2906
2907 \@@_new_alphabet_config:nnn {it} {greek}
2908 {
2909   \bool_if:NTF \g_@@_literal_bool
2910   {
2911     \@@_set_normal_greek:nn {it} {#1}
2912   }
2913   {
2914     \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
2915   }
2916   \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
2917 }
2918
2919 \@@_new_alphabet_config:nnn {it} {misc}
2920 {

```

```

2921 \bool_if:NTF \g_@@_literal_Nabla_bool
2922 {
2923   \@@_set_normal_char:nnn {Nabla}{it}{it}
2924 }
2925 {
2926   \bool_if:NF \g_@@_upNabla_bool
2927   {
2928     \@@_set_normal_char:nnn {Nabla}{up,it}{it}
2929   }
2930 }
2931 \bool_if:NTF \g_@@_literal_partial_bool
2932 {
2933   \@@_set_normal_char:nnn {partial}{it}{it}
2934 }
2935 {
2936   \bool_if:NF \g_@@_uppartial_bool
2937   {
2938     \@@_set_normal_char:nnn {partial}{up,it}{it}
2939   }
2940 }
2941 \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
2942 \@@_set_mathalphabet_pos:nnnn {it} {Nabla} {up,it}{#1}
2943 }

```

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

```

2944 \@@_new_alphabet_config:nnn {bb} {latin}
2945 {
2946   \@@_set_mathalphabet_latin:nnn {bb} {up,it}{#1}
2947 }
2948
2949 \@@_new_alphabet_config:nnn {bb} {Latin}
2950 {
2951   \@@_set_mathalphabet_Latin:nnn {bb} {up,it}{#1}
2952   \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
2953   \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
2954   \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
2955   \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
2956   \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
2957   \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
2958   \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
2959 }
2960
2961 \@@_new_alphabet_config:nnn {bb} {num}
2962 {
2963   \@@_set_mathalphabet_numbers:nnn {bb} {up}{#1}
2964 }
2965
2966 \@@_new_alphabet_config:nnn {bb} {misc}
2967 {

```

```

2968 \@@_set_mathalphabet_pos:nnnn {bb}      {Pi} {up,it} {#1}
2969 \@@_set_mathalphabet_pos:nnnn {bb}      {pi} {up,it} {#1}
2970 \@@_set_mathalphabet_pos:nnnn {bb}      {Gamma} {up,it} {#1}
2971 \@@_set_mathalphabet_pos:nnnn {bb}      {gamma} {up,it} {#1}
2972 \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up} {#1}
2973 }
2974
2975 \@@_new_alphabet_config:nnn {bbit} {misc}
2976 {
2977 \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
2978 \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
2979 \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
2980 \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
2981 \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
2982 }

```

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

```

2983 \@@_new_alphabet_config:nnn {scr} {Latin}
2984 {
2985 \@@_set_mathalphabet_Latin:nnn {scr} {up,it}{#1}
2986 \@@_set_mathalphabet_pos:nnnn {scr} {B}{up,it}{#1}
2987 \@@_set_mathalphabet_pos:nnnn {scr} {E}{up,it}{#1}
2988 \@@_set_mathalphabet_pos:nnnn {scr} {F}{up,it}{#1}
2989 \@@_set_mathalphabet_pos:nnnn {scr} {H}{up,it}{#1}
2990 \@@_set_mathalphabet_pos:nnnn {scr} {I}{up,it}{#1}
2991 \@@_set_mathalphabet_pos:nnnn {scr} {L}{up,it}{#1}
2992 \@@_set_mathalphabet_pos:nnnn {scr} {M}{up,it}{#1}
2993 \@@_set_mathalphabet_pos:nnnn {scr} {R}{up,it}{#1}
2994 }
2995
2996 \@@_new_alphabet_config:nnn {scr} {latin}
2997 {
2998 \@@_set_mathalphabet_latin:nnn {scr} {up,it}{#1}
2999 \@@_set_mathalphabet_pos:nnnn {scr} {e}{up,it}{#1}
3000 \@@_set_mathalphabet_pos:nnnn {scr} {g}{up,it}{#1}
3001 \@@_set_mathalphabet_pos:nnnn {scr} {o}{up,it}{#1}
3002 }

```

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

```

3003 \@@_new_alphabet_config:nnn {cal} {Latin}
3004 {
3005 \@@_set_mathalphabet_Latin:nnn {cal} {up,it}{#1}
3006 \@@_set_mathalphabet_pos:nnnn {cal} {B}{up,it}{#1}
3007 \@@_set_mathalphabet_pos:nnnn {cal} {E}{up,it}{#1}
3008 \@@_set_mathalphabet_pos:nnnn {cal} {F}{up,it}{#1}
3009 \@@_set_mathalphabet_pos:nnnn {cal} {H}{up,it}{#1}
3010 \@@_set_mathalphabet_pos:nnnn {cal} {I}{up,it}{#1}
3011 \@@_set_mathalphabet_pos:nnnn {cal} {L}{up,it}{#1}
3012 \@@_set_mathalphabet_pos:nnnn {cal} {M}{up,it}{#1}

```



```

3013 \@@_set_mathalphabet_pos:nnnn {cal} {R}{up,it}{#1}
3014 }

```

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

```

3015 \@@_new_alphabet_config:nnn {frak} {Latin}
3016 {
3017 \@@_set_mathalphabet_Latin:nnn {frak} {up,it}{#1}
3018 \@@_set_mathalphabet_pos:nnnn {frak} {C}{up,it}{#1}
3019 \@@_set_mathalphabet_pos:nnnn {frak} {H}{up,it}{#1}
3020 \@@_set_mathalphabet_pos:nnnn {frak} {I}{up,it}{#1}
3021 \@@_set_mathalphabet_pos:nnnn {frak} {R}{up,it}{#1}
3022 \@@_set_mathalphabet_pos:nnnn {frak} {Z}{up,it}{#1}
3023 }
3024 \@@_new_alphabet_config:nnn {frak} {latin}
3025 {
3026 \@@_set_mathalphabet_latin:nnn {frak} {up,it}{#1}
3027 }

```

N.3.6 *Sans serif upright: sfup*

```

3028 \@@_new_alphabet_config:nnn {sfup} {num}
3029 {
3030 \@@_set_mathalphabet_numbers:nnn {sf} {up}{#1}
3031 \@@_set_mathalphabet_numbers:nnn {sfup} {up}{#1}
3032 }
3033 \@@_new_alphabet_config:nnn {sfup} {Latin}
3034 {
3035 \bool_if:NTF \g_@@_sfliteral_bool
3036 {
3037 \@@_set_normal_Latin:nn {sfup} {#1}
3038 \@@_set_mathalphabet_Latin:nnn {sf} {up}{#1}
3039 }
3040 {
3041 \bool_if:NT \g_@@_upsans_bool
3042 {
3043 \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3044 \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3045 }
3046 }
3047 \@@_set_mathalphabet_Latin:nnn {sfup} {up,it}{#1}
3048 }
3049 \@@_new_alphabet_config:nnn {sfup} {latin}
3050 {
3051 \bool_if:NTF \g_@@_sfliteral_bool
3052 {
3053 \@@_set_normal_latin:nn {sfup} {#1}
3054 \@@_set_mathalphabet_latin:nnn {sf} {up}{#1}
3055 }
3056 {
3057 \bool_if:NT \g_@@_upsans_bool

```

```

3058     {
3059       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3060       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
3061     }
3062   }
3063   \@@_set_mathalphabet_latin:nnn {sfup} {up,it}{#1}
3064 }

```

N.3.7 *Sans serif italic: sfit*

```

3065 \@@_new_alphabet_config:nnn {sfit} {Latin}
3066 {
3067   \bool_if:NTF \g_@@_sfliteral_bool
3068   {
3069     \@@_set_normal_Latin:nn {sfit} {#1}
3070     \@@_set_mathalphabet_Latin:nnn {sf} {it}{#1}
3071   }
3072   {
3073     \bool_if:NF \g_@@_upsans_bool
3074     {
3075       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3076       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3077     }
3078   }
3079   \@@_set_mathalphabet_Latin:nnn {sfit} {up,it}{#1}
3080 }
3081 \@@_new_alphabet_config:nnn {sfit} {latin}
3082 {
3083   \bool_if:NTF \g_@@_sfliteral_bool
3084   {
3085     \@@_set_normal_latin:nn {sfit} {#1}
3086     \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
3087   }
3088   {
3089     \bool_if:NF \g_@@_upsans_bool
3090     {
3091       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3092       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
3093     }
3094   }
3095   \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
3096 }

```

N.3.8 *Typewriter or monospaced: tt*

```

3097 \@@_new_alphabet_config:nnn {tt} {num}
3098 {
3099   \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
3100 }
3101 \@@_new_alphabet_config:nnn {tt} {Latin}
3102 {

```

```

3103 \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
3104 }
3105 \@@_new_alphabet_config:nnn {tt} {latin}
3106 {
3107   \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
3108 }

```

N.3.9 *Bold Italic: bfit*

```

3109 \@@_new_alphabet_config:nnn {bfit} {Latin}
3110 {
3111   \bool_if:NF \g_@@_bfupLatin_bool
3112   {
3113     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3114   }
3115   \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
3116   \bool_if:NTF \g_@@_bfliteral_bool
3117   {
3118     \@@_set_normal_Latin:nn {bfit} {#1}
3119     \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
3120   }
3121   {
3122     \bool_if:NF \g_@@_bfupLatin_bool
3123     {
3124       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3125       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3126     }
3127   }
3128 }
3129
3130 \@@_new_alphabet_config:nnn {bfit} {latin}
3131 {
3132   \bool_if:NF \g_@@_bfuplatin_bool
3133   {
3134     \@@_set_normal_latin:nn {bfup,bfit} {#1}
3135   }
3136   \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
3137   \bool_if:NTF \g_@@_bfliteral_bool
3138   {
3139     \@@_set_normal_latin:nn {bfit} {#1}
3140     \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
3141   }
3142   {
3143     \bool_if:NF \g_@@_bfuplatin_bool
3144     {
3145       \@@_set_normal_latin:nn {bfup,bfit} {#1}
3146       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3147     }
3148   }
3149 }

```

```

3150
3151 \@@_new_alphabet_config:nnn {bfit} {Greek}
3152 {
3153   \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
3154   \bool_if:NTF \g_@@_bfliteral_bool
3155   {
3156     \@@_set_normal_Greek:nn {bfit}{#1}
3157     \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
3158   }
3159   {
3160     \bool_if:NF \g_@@_bfupGreek_bool
3161     {
3162       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3163       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3164     }
3165   }
3166 }
3167
3168 \@@_new_alphabet_config:nnn {bfit} {greek}
3169 {
3170   \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
3171   \bool_if:NTF \g_@@_bfliteral_bool
3172   {
3173     \@@_set_normal_greek:nn {bfit} {#1}
3174     \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
3175   }
3176   {
3177     \bool_if:NF \g_@@_bfupgreek_bool
3178     {
3179       \@@_set_normal_greek:nn {bfit,bfup} {#1}
3180       \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3181     }
3182   }
3183 }
3184
3185 \@@_new_alphabet_config:nnn {bfit} {misc}
3186 {
3187   \bool_if:NTF \g_@@_literal_Nabla_bool
3188   { \@@_set_normal_char:nnn {Nabla}{bfit}{#1} }
3189   {
3190     \bool_if:NF \g_@@_upNabla_bool
3191     { \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1} }
3192   }
3193   \bool_if:NTF \g_@@_literal_partial_bool
3194   { \@@_set_normal_char:nnn {partial}{bfit}{#1} }
3195   {
3196     \bool_if:NF \g_@@_uppartial_bool
3197     { \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1} }
3198   }

```

```

3199 \@@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it}{#1}
3200 \@@_set_mathalphabet_pos:nnnn {bfit} {Nabla} {up,it}{#1}
3201 \bool_if:NTF \g_@@_literal_partial_bool
3202 {
3203   \@@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
3204 }
3205 {
3206   \bool_if:NF \g_@@_uppartial_bool
3207   {
3208     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3209   }
3210 }
3211 \bool_if:NTF \g_@@_literal_Nabla_bool
3212 {
3213   \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {it}{#1}
3214 }
3215 {
3216   \bool_if:NF \g_@@_upNabla_bool
3217   {
3218     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3219   }
3220 }
3221 }

```

N.3.10 *Bold Upright: bfup*

```

3222 \@@_new_alphabet_config:nnn {bfup} {num}
3223 {
3224   \@@_set_mathalphabet_numbers:nnn {bf} {up}{#1}
3225   \@@_set_mathalphabet_numbers:nnn {bfup} {up}{#1}
3226 }
3227
3228 \@@_new_alphabet_config:nnn {bfup} {Latin}
3229 {
3230   \bool_if:NT \g_@@_bfupLatin_bool
3231   {
3232     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3233   }
3234   \@@_set_mathalphabet_Latin:nnn {bfup} {up,it}{#1}
3235   \bool_if:NTF \g_@@_bfliteral_bool
3236   {
3237     \@@_set_normal_Latin:nn {bfup} {#1}
3238     \@@_set_mathalphabet_Latin:nnn {bf} {up}{#1}
3239   }
3240   {
3241     \bool_if:NT \g_@@_bfupLatin_bool
3242     {
3243       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3244       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3245     }

```

```

3246     }
3247 }
3248
3249 \@@_new_alphabet_config:nnn {bfup} {latin}
3250 {
3251   \bool_if:NT \g_@@_bfuplatin_bool
3252   {
3253     \@@_set_normal_latin:nn {bfup,bfit} {#1}
3254   }
3255   \@@_set_mathalphabet_latin:nnn {bfup} {up,it}{#1}
3256   \bool_if:NTF \g_@@_bfliteral_bool
3257   {
3258     \@@_set_normal_latin:nn {bfup} {#1}
3259     \@@_set_mathalphabet_latin:nnn {bf} {up}{#1}
3260   }
3261   {
3262     \bool_if:NT \g_@@_bfuplatin_bool
3263     {
3264       \@@_set_normal_latin:nn {bfup,bfit} {#1}
3265       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3266     }
3267   }
3268 }
3269 \@@_new_alphabet_config:nnn {bfup} {Greek}
3270 {
3271   \@@_set_mathalphabet_Greek:nnn {bfup} {up,it}{#1}
3272   \bool_if:NTF \g_@@_bfliteral_bool
3273   {
3274     \@@_set_normal_Greek:nn {bfup}{#1}
3275     \@@_set_mathalphabet_Greek:nnn {bf} {up}{#1}
3276   }
3277   {
3278     \bool_if:NT \g_@@_bfupGreek_bool
3279     {
3280       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3281       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3282     }
3283   }
3284 }
3285
3286 \@@_new_alphabet_config:nnn {bfup} {greek}
3287 {
3288   \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
3289   \bool_if:NTF \g_@@_bfliteral_bool
3290   {
3291     \@@_set_normal_greek:nn {bfup} {#1}
3292     \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
3293   }
3294   {

```

```

3295 \bool_if:NT \g_@@_bfupgreek_bool
3296 {
3297   \@@_set_normal_greek:nn {bfup,bfit} {#1}
3298   \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3299 }
3300 }
3301 }
3302
3303 \@@_new_alphabet_config:nnn {bfup} {misc}
3304 {
3305   \bool_if:NTF \g_@@_literal_Nabla_bool
3306   {
3307     \@@_set_normal_char:nnn {Nabla}{bfup}{#1}
3308   }
3309   {
3310     \bool_if:NT \g_@@_upNabla_bool
3311     {
3312       \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1}
3313     }
3314   }
3315   \bool_if:NTF \g_@@_literal_partial_bool
3316   {
3317     \@@_set_normal_char:nnn {partial}{bfup}{#1}
3318   }
3319   {
3320     \bool_if:NT \g_@@_uppartial_bool
3321     {
3322       \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1}
3323     }
3324   }
3325   \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it}{#1}
3326   \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla} {up,it}{#1}
3327   \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up}{#1}
3328   \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up}{#1}
3329   \@@_set_mathalphabet_pos:nnnn {bf} {digamma} {up}{#1}
3330   \@@_set_mathalphabet_pos:nnnn {bf} {Digamma} {up}{#1}
3331   \bool_if:NTF \g_@@_literal_partial_bool
3332   {
3333     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up}{#1}
3334   }
3335   {
3336     \bool_if:NT \g_@@_uppartial_bool
3337     {
3338       \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3339     }
3340   }
3341   \bool_if:NTF \g_@@_literal_Nabla_bool
3342   {
3343     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}

```

```

3344 }
3345 {
3346   \bool_if:NT \g_@@_upNabla_bool
3347   {
3348     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3349   }
3350 }
3351 }

```

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

```

3352 \@@_new_alphabet_config:nnn {bffrak} {Latin}
3353 {
3354   \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
3355 }
3356
3357 \@@_new_alphabet_config:nnn {bffrak} {latin}
3358 {
3359   \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
3360 }

```

N.3.12 *Bold script or calligraphic: bfscr*

```

3361 \@@_new_alphabet_config:nnn {bfscr} {Latin}
3362 {
3363   \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
3364 }
3365 \@@_new_alphabet_config:nnn {bfscr} {latin}
3366 {
3367   \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
3368 }
3369 \@@_new_alphabet_config:nnn {bfcal} {Latin}
3370 {
3371   \@@_set_mathalphabet_Latin:nnn {bfcal} {up,it}{#1}
3372 }

```

N.3.13 *Bold upright sans serif: bfsfup*

```

3373 \@@_new_alphabet_config:nnn {bfsfup} {num}
3374 {
3375   \@@_set_mathalphabet_numbers:nnn {bfsf} {up}{#1}
3376   \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
3377 }
3378 \@@_new_alphabet_config:nnn {bfsfup} {Latin}
3379 {
3380   \bool_if:NTF \g_@@_sfliteral_bool
3381   {
3382     \@@_set_normal_Latin:nn {bfsfup} {#1}
3383     \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
3384   }
3385   {
3386     \bool_if:NT \g_@@_upsans_bool

```



```

3387     {
3388         \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
3389         \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
3390     }
3391 }
3392 \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
3393 }
3394
3395 \@@_new_alphabet_config:nnn {bfsfup} {latin}
3396 {
3397     \bool_if:NTF \g_@@_sfliteral_bool
3398     {
3399         \@@_set_normal_latin:nn {bfsfup} {#1}
3400         \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
3401     }
3402     {
3403         \bool_if:NT \g_@@_upsans_bool
3404         {
3405             \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
3406             \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
3407         }
3408     }
3409     \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
3410 }
3411
3412 \@@_new_alphabet_config:nnn {bfsfup} {Greek}
3413 {
3414     \bool_if:NTF \g_@@_sfliteral_bool
3415     {
3416         \@@_set_normal_Greek:nn {bfsfup}{#1}
3417         \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
3418     }
3419     {
3420         \bool_if:NT \g_@@_upsans_bool
3421         {
3422             \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
3423             \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3424         }
3425     }
3426     \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
3427 }
3428
3429 \@@_new_alphabet_config:nnn {bfsfup} {greek}
3430 {
3431     \bool_if:NTF \g_@@_sfliteral_bool
3432     {
3433         \@@_set_normal_greek:nn {bfsfup} {#1}
3434         \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
3435     }

```

```

3436 {
3437   \bool_if:NT \g_@@_upsans_bool
3438   {
3439     \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3440     \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3441   }
3442 }
3443 \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
3444 }
3445 \@@_new_alphabet_config:nnn {bfsfup} {misc}
3446 {
3447   \bool_if:NTF \g_@@_literal_Nabla_bool
3448   {
3449     \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
3450   }
3451   {
3452     \bool_if:NT \g_@@_upNabla_bool
3453     {
3454       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3455     }
3456   }
3457   \bool_if:NTF \g_@@_literal_partial_bool
3458   {
3459     \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
3460   }
3461   {
3462     \bool_if:NT \g_@@_uppartial_bool
3463     {
3464       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3465     }
3466   }
3467   \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
3468   \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla} {up,it}{#1}
3469   \bool_if:NTF \g_@@_literal_partial_bool
3470   {
3471     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
3472   }
3473   {
3474     \bool_if:NT \g_@@_uppartial_bool
3475     {
3476       \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3477     }
3478   }
3479   \bool_if:NTF \g_@@_literal_Nabla_bool
3480   {
3481     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up}{#1}
3482   }
3483   {
3484     \bool_if:NT \g_@@_upNabla_bool

```

```

3485     {
3486       \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}   {up,it}{#1}
3487     }
3488   }
3489 }

```

N.3.14 *Bold italic sans serif: bfsfit*

```

3490 \@@_new_alphabet_config:nnn {bfsfit} {Latin}
3491 {
3492   \bool_if:NTF \g_@@_sfliteral_bool
3493   {
3494     \@@_set_normal_Latin:nn {bfsfit} {#1}
3495     \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}
3496   }
3497   {
3498     \bool_if:NF \g_@@_upsans_bool
3499     {
3500       \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
3501       \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
3502     }
3503   }
3504   \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}
3505 }
3506
3507 \@@_new_alphabet_config:nnn {bfsfit} {latin}
3508 {
3509   \bool_if:NTF \g_@@_sfliteral_bool
3510   {
3511     \@@_set_normal_latin:nn {bfsfit} {#1}
3512     \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}
3513   }
3514   {
3515     \bool_if:NF \g_@@_upsans_bool
3516     {
3517       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
3518       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
3519     }
3520   }
3521   \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}
3522 }
3523
3524 \@@_new_alphabet_config:nnn {bfsfit} {Greek}
3525 {
3526   \bool_if:NTF \g_@@_sfliteral_bool
3527   {
3528     \@@_set_normal_Greek:nn {bfsfit}{#1}
3529     \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}
3530   }
3531   {

```

```

3532 \bool_if:NF \g_@@_upsans_bool
3533 {
3534   \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
3535   \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3536 }
3537 }
3538 \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
3539 }
3540
3541 \@@_new_alphabet_config:nnn {bfsfit} {greek}
3542 {
3543   \bool_if:NTF \g_@@_sfliteral_bool
3544   {
3545     \@@_set_normal_greek:nn {bfsfit} {#1}
3546     \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
3547   }
3548   {
3549     \bool_if:NF \g_@@_upsans_bool
3550     {
3551       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3552       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3553     }
3554   }
3555   \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
3556 }
3557
3558 \@@_new_alphabet_config:nnn {bfsfit} {misc}
3559 {
3560   \bool_if:NTF \g_@@_literal_Nabla_bool
3561   {
3562     \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
3563   }
3564   {
3565     \bool_if:NF \g_@@_upNabla_bool
3566     {
3567       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3568     }
3569   }
3570   \bool_if:NTF \g_@@_literal_partial_bool
3571   {
3572     \@@_set_normal_char:nnn {partial}{bfsfit}{#1}
3573   }
3574   {
3575     \bool_if:NF \g_@@_uppartial_bool
3576     {
3577       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3578     }
3579   }
3580   \@@_set_mathalphabet_pos:nnnn {bfsfit} {partial} {up,it}{#1}

```

```

3581 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3582 \bool_if:NTF \g_@@_literal_partial_bool
3583 {
3584   \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {it}{#1}
3585 }
3586 {
3587   \bool_if:NF \g_@@_uppartial_bool
3588   {
3589     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3590   }
3591 }
3592 \bool_if:NTF \g_@@_literal_Nabla_bool
3593 {
3594   \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {it}{#1}
3595 }
3596 {
3597   \bool_if:NF \g_@@_upNabla_bool
3598   {
3599     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3600   }
3601 }
3602 }
3603 </alphabets>

```

N.4 Compatibility

3604 <*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.

```

3605 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnnn
3606 {
3607   \cs_if_exist:NT #1
3608   {
3609     \token_if_macro:NTF #1
3610     {
3611       \group_begin:
3612       #2 \@@_tmpa:w #3 { #4 }
3613       \cs_if_eq:NNTF #1 \@@_tmpa:w

```

```

3614 {
3615   \msg_info:nxx { unicode-math } { patch-macro }
3616   { \token_to_str:N #1 }
3617   \group_end:
3618   #2 #1 #3
3619   (XE) { #6 }
3620   (LU) { #5 }
3621 }
3622 {
3623   \msg_warning:nxxxx { unicode-math } { wrong-meaning }
3624   { \token_to_str:N #1 } { \token_to_meaning:N #1 }
3625   { \token_to_meaning:N \@@_tmpa:w }
3626   \group_end:
3627 }
3628 }
3629 {
3630   \msg_warning:nxx { unicode-math } { macro-expected }
3631   { \token_to_str:N #1 }
3632 }
3633 }
3634 }

```

`\@@_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.

```

3635 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
3636 {
3637   \@@_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
3638 }

```

`\@@_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.

```

3639 \cs_new_protected_nopar:Nn \@@_check_and_fix luatex:NNnnn

```

```

3640 {
3641 (LU) \@@_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
3642 }
3643 \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.

```

3644 \AtEndOfPackageFile * {url}
3645 {
3646 \tl_put_left:Nn \Url@FormatString { \@@_switchto_up: }
3647 \tl_put_right:Nn \Url$Specials
3648 {
3649 \do\{\mathchar`\}
3650 \do\'\mathchar`\}
3651 \do\$\mathchar`\}
3652 \do\&\mathchar`\&
3653 }
3654 }

```

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:

```

3655 \AtEndOfPackageFile * {amsmath}
3656 {
3657 (*XE)
3658 \tl_remove_once:Nn \@begindocumenthook
3659 {
3660 \mathchardef\std@minus\mathcode`\-\relax
3661 \mathchardef\std@equal\mathcode`\=\relax
3662 }
3663 \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
3664 \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
3665 (/XE)
3666 \cs_set:Npn \@cdots {\mathinner{\cdots}}
3667 \cs_set_eq:NN \dotso@ \cdots

```

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

```

3668 (*XE)
3669 \def \resetMathstrut@
3670 {%
3671 \setbox\z@\hbox{$(\$)%}
3672 \ht\Mathstrutbox@ \ht\z@ \dp\Mathstrutbox@ \dp\z@
3673 }

```

The subarray environment uses inappropriate font dimensions.

```

3674 \@@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
3675 {
3676   \vcenter
3677   \bgroup
3678   \Let@
3679   \restore@math@cr
3680   \default@tag
3681   \baselineskip \fontdimen 10~ \scriptfont \tw@
3682   \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
3683   \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
3684   \lineskiplimit \lineskip
3685   \ialign
3686   \bgroup
3687   \ifx c #1 \hfil \fi
3688   $ \m@th \scriptstyle ## $
3689   \hfil
3690   \crr
3691 }
3692 {
3693   \vcenter
3694   \c_group_begin_token
3695   \Let@
3696   \restore@math@cr
3697   \default@tag
3698   \skip_set:Nn \baselineskip
3699   {

```

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

```

3700 \@@_stack_num_up:N \scriptstyle
3701 + \@@_stack_denom_down:N \scriptstyle
3702 }

```

Here we use the minimum stack gap.

```

3703 \lineskip \@@_stack_vgap:N \scriptstyle
3704 \lineskiplimit \lineskip
3705 \ialign
3706 \c_group_begin_token
3707 \token_if_eq_meaning:NNT c #1 { \hfil }
3708 \c_math_toggle_token
3709 \m@th
3710 \scriptstyle
3711 \c_parameter_token \c_parameter_token
3712 \c_math_toggle_token
3713 \hfil
3714 \crr
3715 }
3716 </XE>

```

The roots need a complete rework.


```

3717 \@@_check_and_fix luatex:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
3718 {
3719   \setbox \rootbox \hbox
3720   {
3721     $ \math \scriptscriptstyle { #1 } $
3722   }
3723   \mathchoice
3724     { \r@@@t \displaystyle { #2 } }
3725     { \r@@@t \textstyle { #2 } }~
3726     { \r@@@t \scriptstyle { #2 } }
3727     { \r@@@t \scriptscriptstyle { #2 } }
3728   \egroup
3729 }
3730 {
3731   \bool_if:nTF
3732   {
3733     \int_compare_p:nNn { \uproot@ } = { \c_zero }
3734     && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
3735   }
3736   {
3737     \luatexUroot \l_@@_radical_sqrt_tl { #1 } { #2 }
3738   }
3739   {
3740     \hbox_set:Nn \rootbox
3741     {
3742       \c_math_toggle_token
3743       \math
3744       \scriptscriptstyle { #1 }
3745       \c_math_toggle_token
3746     }
3747     \mathchoice
3748       { \r@@@t \displaystyle { #2 } }
3749       { \r@@@t \textstyle { #2 } }
3750       { \r@@@t \scriptstyle { #2 } }
3751       { \r@@@t \scriptscriptstyle { #2 } }
3752   }
3753   \c_group_end_token
3754 }
3755 \@@_check_and_fix:NNnnnn \r@@@t \cs_set_nopar:Npn { #1 #2 }
3756 {
3757   \setboxz@h { $ \math #1 \sqrt{sign { #2 } } $ }
3758   \dimen@ \ht\z@
3759   \advance \dimen@ -\dp\z@
3760   \setbox\@ne \hbox { $ \math #1 \mskip \uproot@ mu $ }
3761   \advance \dimen@ by 1.667 \wd\@ne
3762   \mkern -\leftroot@ mu
3763   \mkern 5mu
3764   \raise .6\dimen@ \copy\rootbox
3765   \mkern -10mu

```

```

3766 \mkern \leftroot@ mu
3767 \boxz@
3768 }
3769 {
3770 \hbox_set:Nn \l_tmpa_box
3771 {
3772 \c_math_toggle_token
3773 \m@th
3774 #1
3775 \mskip \uproot@ mu
3776 \c_math_toggle_token
3777 }
3778 \luatexUroot \l_@@_radical_sqrt_tl
3779 {
3780 \box_move_up:nn { \box_wd:N \l_tmpa_box }
3781 {
3782 \hbox:n
3783 {
3784 \c_math_toggle_token
3785 \m@th
3786 \mkern -\leftroot@ mu
3787 \box_use:N \rootbox
3788 \mkern \leftroot@ mu
3789 \c_math_toggle_token
3790 }
3791 }
3792 }
3793 { #2 }
3794 }
3795 {
3796 \hbox_set:Nn \l_tmpa_box
3797 {
3798 \c_math_toggle_token
3799 \m@th
3800 #1
3801 \sqrtsign { #2 }
3802 \c_math_toggle_token
3803 }
3804 \hbox_set:Nn \l_tmpb_box
3805 {
3806 \c_math_toggle_token
3807 \m@th
3808 #1
3809 \mskip \uproot@ mu
3810 \c_math_toggle_token
3811 }
3812 \mkern -\leftroot@ mu
3813 \@@_mathstyle_scale:Nnn #1 { \kern }
3814 {

```

```

3815     \fontdimen 63 \l_@@_font
3816   }
3817   \box_move_up:nn
3818   {
3819     \box_wd:N \l_tmpb_box
3820     + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
3821     * \number \fontdimen 65 \l_@@_font / 100
3822   }
3823   {
3824     \box_use:N \rootbox
3825   }
3826   \@@_mathstyle_scale:Nnn #1 { \kern }
3827   {
3828     \fontdimen 64 \l_@@_font
3829   }
3830   \mkern \leftroot@ mu
3831   \box_use_clear:N \l_tmpa_box
3832 }
3833 }

```

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:

```

3834 (*XE)
3835 \AtEndOfPackageFile * {amsopn}
3836 {
3837   \cs_set:Npn \newmcodes@
3838   {
3839     \mathcode'\ '39\scan_stop:
3840     \mathcode'\ '*42\scan_stop:
3841     \mathcode'\ "."613A\scan_stop:
3842     %% \ifnum\mathcode'\-=45 \else
3843     %%   \mathchardef\std@minus\mathcode'\-\relax
3844     %% \fi
3845     \mathcode'\-45\scan_stop:
3846     \mathcode'\ /47\scan_stop:
3847     \mathcode'\ ":"603A\scan_stop:
3848   }
3849 }
3850 (/XE)

```

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

```

3851 \AtEndOfPackageFile * { mathtools }
3852 {
3853 (*XE)
3854   \newfam \g_@@_empty_fam
3855   \@@_check_and_fix:NNnnn
3856     \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }

```

```

3857 {
3858   \sbox \z@
3859   {
3860     $
3861     \m@th
3862     #1
3863     \nulldelimiterspace = \z@
3864     \radical \z@ { #2 }
3865     $
3866   }
3867   \ifx #1 \displaystyle
3868     \dimen@ = \fontdimen 8 \textfont 3
3869     \advance \dimen@ .25 \fontdimen 5 \textfont 2
3870   \else
3871     \dimen@ = 1.25 \fontdimen 8
3872     \ifx #1 \textstyle
3873       \textfont
3874     \else
3875       \ifx #1 \scriptstyle
3876         \scriptfont
3877       \else
3878         \scriptscriptfont
3879       \fi
3880     \fi
3881     3
3882   \fi
3883   \advance \dimen@ -\ht\z@
3884   \ht\z@ = -\dimen@
3885   \box\z@
3886 }

```

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.

```

3887 {
3888   \hbox_set:Nn \l_tmpa_box
3889   {
3890     \color@setgroup
3891     \c_math_toggle_token
3892     \m@th
3893     #1
3894     \dim_zero:N \nulldelimiterspace
3895     \XeTeXradical \g_@@_empty_fam \c_zero { #2 }
3896     \c_math_toggle_token
3897     \color@endgroup
3898   }
3899   \box_set_ht:Nn \l_tmpa_box
3900   {
3901     \box_ht:N \l_tmpa_box

```

Here we use the radical vertical gap.

```

3902         - \@@_radical_vgap:N #1
3903     }
3904     \box_use_clear:N \l_tmpa_box
3905 }
3906 (/XE)

```

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

```

3907 \AtEndOfPackageFile * { mathtools }
3908 {
3909     \cs_set_eq:NN \MToverbracket \overbracket
3910     \cs_set_eq:NN \MTunderbracket \underbracket
3911
3912     \AtBeginDocument
3913     {
3914         \msg_warning:nn { unicode-math } { mathtools-overbracket }
3915
3916     \def\downbracketfill#1#2
3917     {%

```

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.

```

3918         \edef\l_MT_bracketheight_fdim{.27ex}%
3919         \downbracketend{#1}{#2}
3920         \leaders \vrule \@height #1 \@depth \z@ \hfill
3921         \downbracketend{#1}{#2}%
3922     }
3923 \def\upbracketfill#1#2
3924 {%
3925         \edef\l_MT_bracketheight_fdim{.27ex}%
3926         \upbracketend{#1}{#2}
3927         \leaders \vrule \@height \z@ \@depth #1 \hfill
3928         \upbracketend{#1}{#2}%
3929     }
3930 \let\Uoverbracket =\overbracket
3931 \let\Underbracket=\underbracket
3932     \let\overbracket =\MToverbracket
3933     \let\underbracket =\MTunderbracket
3934 }
3935 }

```

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

```

3936 \msg_warning:nn { unicode-math } { mathtools-colon }

```

```

3937 \NewDocumentCommand \dblcolon { } { \Colon }
3938 \NewDocumentCommand \coloneqq { } { \coloneq }
3939 \NewDocumentCommand \Coloneqq { } { \Coloneq }
3940 \NewDocumentCommand \eqqcolon { } { \eqcolon }
3941 }

```

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 3942 \AtEndOfPackageFile * { colonequals }
\minuscolon 3943 {
\colonequals 3944 \msg_warning:nn { unicode-math } { colonequals }
\equalscolon 3945 \RenewDocumentCommand \ratio { } { \mathratio }
\coloncolonequals 3946 \RenewDocumentCommand \coloncolon { } { \Colon }
3947 \RenewDocumentCommand \minuscolon { } { \dashcolon }
3948 \RenewDocumentCommand \colonequals { } { \coloneq }
3949 \RenewDocumentCommand \equalscolon { } { \eqcolon }
3950 \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
3951 }
3952 </compat>

```