

Experimental Unicode mathematical typesetting: The unicode-math package

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

2017/07/30 v0.8e

Abstract

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

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

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

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

Part I

User documentation

Table of Contents

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

1 Introduction

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

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

2 Acknowledgements

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

3 Getting started

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

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

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

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

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

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

3.1 New commands

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

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

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

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

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

Additional similar commands can be defined using

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

3.2 Package options

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

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

Note, however, that some package options affect how maths is initialised and changing an option such as $\text{\textbackslash math-style}$ will not take effect until a new maths font is set up.

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

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

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

Table 1: Package options.

Option	Description	See...
<code>math-style</code>	Style of letters	section §5.1
<code>bold-style</code>	Style of bold letters	section §5.2
<code>sans-style</code>	Style of sans serif letters	section §5.3
<code>nabla</code>	Style of the nabla symbol	section §5.5.1
<code>partial</code>	Style of the partial symbol	section §5.5.2
<code>colon</code>	Behaviour of <code>\colon</code>	section §5.5.5
<code>slash-delimiter</code>	Glyph to use for ‘stretchy’ slash	section §5.5.6

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 every symbol and alphabetic variant. That means x to x , ξ to ξ , \leq to \leq , etc., 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:

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

where $\langle \text{unicode range} \rangle$ is a comma-separated list of Unicode slot numbers and ranges such as $\{ "27D0-"27EB, "27FF, "295B-"297F \}$. Note that \TeX 's syntax for accessing the slot number of a character, such as $\backslash +$, will also work here.

You may also use the macro for accessing the glyph, such as $\backslash \text{int}$, or whole collection of symbols with the same math type, such as $\backslash \text{mathopen}$, or complete math styles such as $\backslash \text{sybbb}$. (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 , bfcal ,
 frak , bffrak , sfup , sfit , bfsfup , bfsfit , bfsf

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

- $[\text{range}=\text{bb}]$ to use the font for 'bb' letters only.
- $[\text{range}=\text{bfsfit}/\{\text{greek}, \text{Greek}\}]$ for Greek lowercase and uppercase only (also with latin , Latin , num as possible options for Latin lower-/upper-case and numbers, resp.).
- $[\text{range}=\text{up} \rightarrow \text{sfup}]$ 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 $[\text{range}=\text{bfup}]$ and $[\text{range}=\text{bfit}]$ 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

$\backslash \text{setmathfont}\{\text{SomeFrakturFont}\}[\text{range}=\text{frak}]$

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 $[\text{range}=\text{frak} \rightarrow \text{up}]$, since the 'up' range corresponds to ASCII letters.

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

$\backslash \text{setmathfont}\{\text{MinionMath-Regular.otf}\}$
 $\backslash \text{setmathfont}\{\text{MinionMath-Bold.otf}\}[\text{range}=\{\text{bfup} \rightarrow \text{up}, \text{bfit} \rightarrow \text{it}\}]$

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

```

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

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

```

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

4.2 Script and scriptscript fonts/features

Cambria Math uses OpenType font features to activate smaller optical sizes for scriptsize and scriptscriptsize symbols (the B and C , respectively, in A_{BC}). Other typefaces (such as Minion Math) may use entirely separate font files.

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

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

4.3 Maths ‘versions’

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 `\setmathfontface` 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\sfddefault{...}`. With `unicode-math` loaded, after writing `\setsansfont{Helvetica}`, `\mathsf` will now be set in Helvetica.

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

4.4.2 Replacing ‘text math’ fonts by symbols

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

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

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>

4.4.3 Operator font

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

```
\setoperatorfont\mathit
```

Or, to select a unicode-math range:

```
\setoperatorfont\symscr
```

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

5 Maths input

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

5.1 Math ‘style’

Classically, T_EX uses italic lowercase Greek letters and *upright* uppercase Greek letters for variables in mathematics. This is contrary to the iso standards of using italic forms for both upper- and lowercase. Furthermore, in various historical contexts, often associated with French typesetting, it was common to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek, but italic lowercase latin. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

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

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

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

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

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

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

‘Literal’ interface Some may not like this convention of normalising their input. For them, an upright `x` is an upright ‘ x ’ and that’s that. (This will be the case when obtaining source text from copy/pasting PDF or Microsoft Word documents, for example.) For these users, the `literal` option to `math-style` will effect this behaviour. The `\symliteral{<syms>}` command can also be used, regardless of package setting, to force the style to match the literal input characters. This is a ‘mirror’ to `\symnormal{<syms>}` (also alias `\mathnormal`) which ‘resets’ the character mapping in its argument to that originally set up through package options.

5.2 Bold style

Similar as in the previous section, ISO standards differ somewhat to \TeX 's conventions (and classical typesetting) for ‘boldness’ in mathematics. In the past, it has been customary to use bold *upright* letters to denote things like vectors and matrices. For example, $\mathbf{M} = (M_x, M_y, M_z)$. Presumably, this was due to the relatively scarcity of bold italic fonts in the pre-digital typesetting era. It has been suggested by some that *italic* bold symbols should be used nowadays instead, but this practise is certainly not widespread.

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

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

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

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

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

5.3 Sans serif style

Unicode contains upright and italic, medium and bold mathematical style characters. These may be explicitly selected with the `\mathsfup`, `\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 con-

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>\mathbfi</code>	•	•	•
Sans serif	Upright	Normal	<code>\mathsfup</code>	•		•
	Italic	Normal	<code>\mathsf</code>	•		•
	Upright	Bold	<code>\mathbfsfup</code>	•	•	•
	Italic	Bold	<code>\mathbfsfi</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>	•		

ceivable). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

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

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

5.4 All (the rest) of the mathematical styles

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

The math font switching commands do not nest; therefore if you want sans serif bold, you must write `\ymbfsf{...}` rather than `\ymbf{\symsf{...}}`. 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 $\{\mathbb{a}-\mathbb{z}, \mathbb{A}-\mathbb{Z}\}$, numerals $\mathbb{0}-\mathbb{9}$, summation symbol $\mathbb{\Sigma}$, and four Greek letters only: $\{\Upsilon, \Pi, \Gamma, \mathbb{I}\}$.

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

There are also five Latin *italic* double-struck letters: $\mathbb{D}, \mathbb{d}, \mathbb{E}, \mathbb{I}, \mathbb{J}$. These can be accessed (if not with their literal characters or control sequences) with the `\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={cal,bfcal},StylisticSet=1]
```

An example is shown below.

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

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

5.5 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). $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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 `\sympbf`; `\symit` and `\symup` can be used to force one way or the other.

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

5.5.2 Partial

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

Table 7: The various forms of nabla.

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

Table 8: The partial differential.

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

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

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

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

5.5.3 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 backwards prime is `\backprime`, and multiple backwards primes can be accessed with `\backdprime`, `\backtrprime`, and `\backqprime`.

²A good argument would revolve around some international standards body recommending upright over italic. I just don't have the time right now to look it up.

A 0 1 2 3 4 5 6 7 8 9 + - = () i n n h j r w y Z

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

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

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

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

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

5.5.4 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.

5.5.5 Colon

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

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

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

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

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

Table 9: Slashes and backslashes.

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

5.5.6 Slashes and backslashes

There are several slash-like symbols defined in Unicode. The complete list is shown in table 9.

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

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

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

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

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

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

MathML uses U+2216 set minus like this: $A \setminus B$.³ The \LaTeX command name `\smallsetminus` is used for backwards compatibility.

Presumably, U+29F5 reverse solidus operator is intended to be used in a similar way, but it could also (perhaps?) be used to represent ‘inverse division’: $\pi \approx 7 \setminus 22$.⁴ The \LaTeX name for this character is `\setminus`.

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

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

⁴This is valid syntax in the Octave and Matlab programming languages, in which it means matrix inverse pre-multiplication. I.e., $A \setminus B \equiv A^{-1}B$.

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

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

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

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

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

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

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

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

5.5.7 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 \XETEX 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 \XETEX v0.9998, these wide/non-wide commands will again behave in their expected manner.

5.5.8 Pre-drawn fraction characters

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

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

For example, instead of writing `\tfrac{12}{x}`, you may consider it more readable to have `'\frac{1}{2}x'` in the source instead.

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>\smbllkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbllkcircle</code>	●	◦	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdbllkcircle</code>	●	◯	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlgbllkcircle</code>	●	◯	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	◯	<code>\lgwhtcircle</code>	U+25EF

Table 10: Filled and hollow Unicode circles.

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

5.5.9 Circles

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

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

5.5.10 Triangles

While there aren't as many different sizes of triangle as there are circle, there's some important distinctions to make between a few similar characters. 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.

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

Table 11: Different upwards pointing triangles.

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, `\sympf { \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: $\alpha, \leq, \rightarrow$

`\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 \TeX font dimensions

Text fonts		Maths font, $\backslash\text{fam2}$		Maths font, $\backslash\text{fam3}$	
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 (\XTeX 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 \XTeX math font dimensions

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

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

\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	SPACEAFTERSCRIP	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	38
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	77
K	Definitions of the active math characters	78
L	Fall-back font	78
M	Epilogue	79
M.1	Primes	79
M.2	Unicode radicals	85
M.3	Unicode sub- and super-scripts	87
M.4	Synonyms and all the rest	92
N	Error messages	95
N.1	Alphabet Unicode positions	97
N.2	STIX fonts	102
N.3	Alphabets	106
N.4	Compatibility	124

The prefix for unicode-math is um:

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

E Header code

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

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

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

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

```

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

```

Packages

```

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

35 \ExplSyntaxOn

    Variants needed from expl3:

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

    For fontspec:

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

```

Conditionals

```

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

```

For math-style:

```

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

```

For bold-style:

```

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

```

For sans-style:

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

For assorted package options:

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

Variables

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

For displaying in warning messages, etc.:

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

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

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

Variables:

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

\g_@@_mathclasses_seq Every math class.

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

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

```
89 \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 `\symsfit` and `\symsfit`). ‘Mathstyles’ are a superset of named ranges and also include commands such as `\symsf` and `\symsf`.

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

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

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

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

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

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

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

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

```
97 \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!!

```
98 \prg_new_conditional:Nnn \@@_glyph_if_exist:n {p,TF,T,F}
```

```
99 {
```

```
100 \etex_iffontchar:D \l_@@_font #1 \scan_stop:
```

```
101 \prg_return_true:
```

```
102 \else:
```

```
103 \prg_return_false:
```

```
104 \fi:
```

```
105 }
```

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

`\@@_set_mathcode:nnn`

```
106 \cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4 {
```

`\@@_set_mathchar:NNnn`

```
107 \Umathcode \int_eval:n {#1} =
```

`\@@_set_mathchar:cNnn`

```
108 \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
```

`\@@_set_delcode:nnn`

```
109 }
```

`\@@_radical:nn`

```
110 \cs_set:Npn \@@_set_mathcode:nnn #1#2#3 {
```

`\@@_delimiter:Nnn`

```
111 \Umathcode \int_eval:n {#1} =
```

`\@@_accent:nnn`

```
112 \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
```

```
113 }
```

`\@@_accent_keyword:`

```
114 \cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4 {
```

```
115 \Umathchardef #1 =
```

```
116 \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
```

```
117 }
```

```
118 \cs_new:Nn \@@_set_delcode:nnn {
```

```
119 \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
```

```
120 }
```



```

121 \cs_new:Nn \@@_radical:nn {
122   \Uradical \csname sym#1\endcsname #2 \scan_stop:
123 }
124 \cs_new:Nn \@@_delimiter:Nnn {
125   \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
126 }
127 \cs_new:Nn \@@_accent:nnn {
128   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
129 }
130 \cs_generate_variant:Nn \@@_set_mathchar:NNnn {c}

```

\@@_char_gmake_mathactive:N

\@@_char_gmake_mathactive:n

```

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

```

E.2 Alphabet Unicode positions

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

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

```

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

```

E.3 Package options

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

```

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

```

150 \cs_new:Nn \@@_keys_choices:nn

```

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

```

151 {
152   \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
153   \use:x
154   {
155     \exp_not:N \keys_define:nn {unicode-math}
156     {
157       #1 .choice: ,
158       \@@_tl_map_dbl:nN {#2} \@@_keys_choices_fn:nn
159     }
160   }
161 }
162 \cs_new:Nn \@@_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
163
164 \cs_new:Nn \@@_tl_map_dbl:nN
165 {
166   \__@@_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
167 }
168 \cs_new:Nn \__@@_tl_map_dbl:Nnn
169 {
170   \quark_if_recursion_tail_stop:n {#2}
171   \quark_if_recursion_tail_stop:n {#3}
172   #1 {#2} {#3}
173   \__@@_tl_map_dbl:Nnn #1
174 }

```

Compatibility

```

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

```

```

198 {text} { \bool_set_true:N \g_@@_mathsf_text_bool }
199 }
200 \@@_keys_choices:nn {mathtt}
201 {
202 {sym} { \bool_set_false:N \g_@@_mathtt_text_bool }
203 {text} { \bool_set_true:N \g_@@_mathtt_text_bool }
204 }

```

math-style

```

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

```

```

244     }
245     {TeX} {
246         \unimathsetup { nabra=upright, partial=italic,
247             normal-style=TeX, bold-style=TeX, sans-style=upright }
248     }
249     {french} {
250         \unimathsetup { nabra=upright, partial=upright,
251             normal-style=french, bold-style=upright, sans-style=upright }
252     }
253     {upright} {
254         \unimathsetup { nabra=upright, partial=upright,
255             normal-style=upright, bold-style=upright, sans-style=upright }
256     }
257     {literal} {
258         \unimathsetup { colon=literal, nabra=literal, partial=literal,
259             normal-style=literal, bold-style=literal, sans-style=literal }
260     }
261 }

```

bold-style

```

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

```

sans-style

```
289 \@@_keys_choices:nn {sans-style}
290 {
291   {italic} { \bool_set_false:N \g_@@_upsans_bool }
292   {upright} { \bool_set_true:N \g_@@_upsans_bool }
293   {literal} { \bool_set_true:N \g_@@_sfliteral_bool }
294 }
```

Nabla and partial

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

Colon style

```
319 \@@_keys_choices:nn {colon}
320 {
321   {literal} { \bool_set_true:N \g_@@_literal_colon_bool }
322   {TeX} { \bool_set_false:N \g_@@_literal_colon_bool }
323 }
```

Slash delimiter style

```
324 \@@_keys_choices:nn {slash-delimiter}
325 {
326   {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
327   {frac} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
328   {div} { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
329 }
```

Active fraction style

```
330 \@@_keys_choices:nn {active-frac}
331 {
332   {small}
333   {
334     \cs_if_exist:NTF \tfrac
335     { \bool_set_true:N \l_@@_smallfrac_bool }
336     {
337       \@@_warning:n {no-tfrac}
338       \bool_set_false:N \l_@@_smallfrac_bool
339     }
340     \use:c {@@_setup_active_frac:}
341   }
342
343   {normalsize}
344   {
345     \bool_set_false:N \l_@@_smallfrac_bool
346     \use:c {@@_setup_active_frac:}
347   }
348 }
```

Debug/tracing

```
349 \keys_define:nn {unicode-math}
350 {
351   warnings-off .code:n =
352   {
353     \clist_map_inline:nn {#1}
354     { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
355   }
356 }
357 \@@_keys_choices:nn {trace}
358 {
359   {on}    {} % default
360   {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
361   {off}   { \msg_redirect_module:nnn { unicode-math } { log } { none } }
362 }
363 \unimathsetup {math-style=TeX}
364 \unimathsetup {slash-delimiter=ascii}
365 \unimathsetup {trace=off}
366 \unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}
367 \cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
368 \ProcessKeysOptions {unicode-math}
```

E.4 Programmers' interface

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

```
369 \cs_new:Nn \unimath_get_mathstyle:
```

```

370 {
371   \tl_use:N \l_@@_mathstyle_tl
372 }

```

End of preamble code.

```

373 </preamble&!XE&!LU>

```

(Error messages and warning definitions go here from the msg chunk defined in section §N on page 95.)

F Bifurcation

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

```

374 <*package&(XE|LU)>
375 \ExplSyntaxOn

```

F.1 Engine differences

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

```

376 <*LU>
377 \RequirePackage{luaotfload} [2014/05/18]
378 \RequirePackage{lualatex-math}[2011/08/07]
379 </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.

```

380 \tl_map_inline:nn
381 {
382   \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
383   \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
384   \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
385   \version@list\version@elt\alpha@list\alpha@elt
386   \restore@mathversion\init@restore@version\dorestore@version\process@table
387   \new@mathversion\DeclareSymbolFont\group@list\group@elt
388   \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
389   \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
390   \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
391   \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
392   \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
393   \set@mathdelimiter\DeclareMathRadical\mathchar@type
394   \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@

```

```

395 }
396 {
397   \tl_remove_once:Nn \@preamblecmds {\do#1}
398 }

```

G Fundamentals

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

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

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

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

```

399 \cs_set:Nn \@@_set_mathsymbol:nNn
400 {
401   \bool_lazy_and:nnT
402   {
403     \int_compare_p:nNn {#4} > {127}
404   }
405   {
406     \int_compare_p:nNn { \char_value_catcode:n {#4} } = {11}
407   }
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:Nnnn #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 (\sum). In order for literal Unicode characters to be used in the source and still have the correct limits behaviour, big operators are made math-active. This involves three steps:

- The active math char is defined to expand to the macro \sum_{sym} . (Later, the control sequence \sum will be assigned the math char.)
- Declare the plain old \mathchardef for the control sequence \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 $\l_1_{@@_nolimits_t1}$, 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:

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

```

441 \cs_new:Nn \@@_set_big_operator:nnn
442 {
443   \@@_char_gmake_mathactive:n {#3}
444   \cs_set_protected_nopar:Npx \@@_tmpa: { \exp_not:c { \cs_to_str:N #2 _sym } }
445   \char_gset_active_eq:nN {#3} \@@_tmpa:
446
447   \@@_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}
448
449   \cs_gset:cpx { \cs_to_str:N #2 _sym }
450   {
451     \exp_not:c { \cs_to_str:N #2 op }
452   }

```

```

452     \exp_not:n { \tl_if_in:NnT \l_@@_nolimits_tl {#2} \nolimits }
453   }
454 }

\@@_set_math_open:nnn #1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot
455 \cs_new:Nn \@@_set_math_open:nnn
456 {
457   \tl_if_in:NnTF \l_@@_radicals_tl {#2}
458   {
459     \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
460     { \@@_radical:nn {#1} {#3} }
461     \tl_set:cn {l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
462   }
463   {
464     \@@_set_delcode:nnn {#1} {#3} {#3}
465     \@@_set_mathcode:nnn {#3} \mathopen {#1}
466     \cs_gset_protected_nopar:Npx #2
467     { \@@_delimiter:Nnn \mathopen {#1} {#3} }
468   }
469 }

\@@_set_math_close:nnn #1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot
470 \cs_new:Nn \@@_set_math_close:nnn
471 {
472   \@@_set_delcode:nnn {#1} {#3} {#3}
473   \@@_set_mathcode:nnn {#3} \mathclose {#1}
474   \cs_gset_protected_nopar:Npx #2
475   { \@@_delimiter:Nnn \mathclose {#1} {#3} }
476 }

\@@_set_math_fence:nnnn #1 : Symbol font name
#2 : Macro to assign
#3 : Type, e.g., \mathalpha
#4 : Glyph slot
477 \cs_new:Nn \@@_set_math_fence:nnnn
478 {
479   \@@_set_mathcode:nnn {#4} {#3} {#1}
480   \@@_set_delcode:nnn {#1} {#4} {#4}
481   \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
482   { \@@_delimiter:Nnn \mathopen {#1} {#4} }
483   \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
484   { \@@_delimiter:Nnn \mathclose {#1} {#4} }
485 }

```

```

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

486 \cs_new:Nn \@@_set_math_accent:Nnnn
487 {
488   \cs_gset_protected_nopar:Npx #1
489   { \@@_accent:nnn {#2} {#3} {#4} }
490 }

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

491 \cs_new:Nn \@@_set_math_overunder:Nnnn
492 {
493   \cs_gset_protected_nopar:Npx #1 ##1
494   {
495     \mathop
496     { \@@_accent:nnn {#2} {#3} {#4} {##1} }
497     \limits
498   }
499 }

```

G.2 \setmathalphabet

\setmathalphabet

```

500 \keys_define:nn {@@_mathface}
501 {
502   version .code:n =
503   { \tl_set:Nn \l_@@_mversion_tl {#1} }
504 }
505
506 \DeclareDocumentCommand \setmathfontface { m O{} m O{} }
507 {
508   \tl_clear:N \l_@@_mversion_tl
509
510   \keys_set_known:nnN {@@_mathface} {#2,#4} \l_@@_keyval_clist
511   \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
512   { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}
513
514   \tl_if_empty:NT \l_@@_mversion_tl
515   {
516     \tl_set:Nn \l_@@_mversion_tl {normal}
517     \DeclareMathAlphabet #1 {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\mddefault} {\updefault}
518   }
519   \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\mddefault} {\V
520

```

```

521 % integrate with fontspec's \setmathrm etc:
522 \tl_case:Nn #1
523 {
524   \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
525   \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
526   \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
527 }
528 }
529
530 \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?

```

531 \DeclareDocumentCommand \setoperatorfont {m}
532 { \tl_set:Nn \g_@@_operator_mathfont_tl {#1} }
533 \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

```

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

```

```

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

```

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.

```

577 \cs_set:Nn \__fontspec_setmathrm:nn
578 {
579 (XE) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
580 (LU) \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
581 \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
582 \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
583 \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
584 }
585 \cs_set:Nn \__fontspec_setboldmathrm:nn
586 {
587 (XE) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {#1} {#2}
588 (LU) \fontspec_set_family:Nnn \g__fontspec_bfmathrm_tl {Renderer=Basic,#1} {#2}
589 \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\updefault
590 \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
591 \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\itdefault
592 }
593 \cs_set:Nn \__fontspec_setmathsf:nn
594 {
595 (XE) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
596 (LU) \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
597 \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
598 \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault

```

```

599 }
600 \cs_set:Nn \__fontspec_setmathhtt:nn
601 {
602 (XE) \fontspec_set_family:Nnn \g__fontspec_mathhtt_tl {#1} {#2}
603 (LU) \fontspec_set_family:Nnn \g__fontspec_mathhtt_tl {Renderer=Basic,#1} {#2}
604 \SetMathAlphabet\mathhtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathhtt_tl\mddefault\updefault
605 \SetMathAlphabet\mathhtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathhtt_tl\bfdefault\updefault
606 }

```

G.4 The main `\setmathfont` macro

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

`\setmathfont` [#1]: font features (first optional argument retained for backwards compatibility)
 #2 : font name
 [#3]: font features

```

607 \DeclareDocumentCommand \setmathfont { O{} m O{} }
608 {
609 \tl_set:Nn \l_@@_fontname_tl {#2}
610 \@@_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.

```

611 \cs_if_exist:cf { S@ \f@size } { \calculate@math@sizes }
612 \csname S@\f@size\endcsname

```

Parse options and tell people what's going on:

```

613 \keys_set_known:nnN {unicode-math} {#1,#3} \l_@@_unknown_keys_clist
614 \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.

```

615
616 (debug) \csname TIC\endcsname
617 \@@_fontspec_select_font:
618 (debug) \csname TOC\endcsname
619 \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
620 {
621 \@@_declare_math_sizes:
622 \@@_fontspec_select_font:
623 }

```

Now define `\@@_symfont_tl` as the L^AT_EX math font to access everything:

```

624 \cs_if_exist:cf { sym \@@_symfont_tl }
625 {
626 \DeclareSymbolFont{\@@_symfont_tl}

```

```

627         {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
628     }
629     \SetSymbolFont{\@@_symfont_tl}{\l_@@_mversion_tl}
630     {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}

```

Set the bold math version.

```

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

```

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:

```

637     \bool_if:nT { \l_@@_ot_math_bool && !\g_@@_mainfont_already_set_bool }
638     {
639         \bool_set_true:N \g_@@_mainfont_already_set_bool
640         \@@_setup_legacy_fam_two:
641         \@@_setup_legacy_fam_three:
642     }

```

And now we input every single maths char.

```

643 <debug> \csname TIC\endcsname
644 \@@_input_math_symbol_table:
645 <debug> \csname TOC\endcsname

```

Finally,

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

```

646 \@@_remap_symbols:
647 \@@_setup_mathactives:
648 \@@_setup_delcodes:
649 <debug> \csname TIC\endcsname
650 \@@_setup_alphabets:
651 <debug> \csname TOC\endcsname
652 \@@_setup_negations:

```

Prevent spaces, and that's it:

```

653 \ignorespaces
654 }

```

Backward compatibility alias.

```

655 \cs_set_eq:NN \resetmathfont \setmathfont

```

`\@@_init:`

```
656 \cs_new:Nn \@@_init:
657 {
```

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

```
658     \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.

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

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

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

- By default use the ‘normal’ math version.

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

- Other range initialisations.

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

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

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

```
678 }
```

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

```
679 \cs_new:Nn \@@_declare_math_sizes:
680 {
```



```

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

```

\@@_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.

```

688 \cs_new:Nn \@@_setup_legacy_fam_two:
689 {
690   \fontspec_set_family:Nxn \l_@@_family_tl
691   {
692     \l_@@_font_keyval_tl,
693     Scale=1.00001,
694     FontAdjustment =
695     {
696       \fontdimen8\font= \@@_get_fontparam:nn {43} {FractionNumeratorDisplayStyleShiftUp}\relax
697       \fontdimen9\font= \@@_get_fontparam:nn {42} {FractionNumeratorShiftUp}\relax
698       \fontdimen10\font=\@@_get_fontparam:nn {32} {StackTopShiftUp}\relax
699       \fontdimen11\font=\@@_get_fontparam:nn {45} {FractionDenominatorDisplayStyleShiftDown}\relax
700       \fontdimen12\font=\@@_get_fontparam:nn {44} {FractionDenominatorShiftDown}\relax
701       \fontdimen13\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
702       \fontdimen14\font=\@@_get_fontparam:nn {21} {SuperscriptShiftUp}\relax
703       \fontdimen15\font=\@@_get_fontparam:nn {22} {SuperscriptShiftUpCramped}\relax
704       \fontdimen16\font=\@@_get_fontparam:nn {18} {SubscriptShiftDown}\relax
705       \fontdimen17\font=\@@_get_fontparam:nn {18} {SubscriptShiftDownWithSuperscript}\relax
706       \fontdimen18\font=\@@_get_fontparam:nn {24} {SuperscriptBaselineDropMax}\relax
707       \fontdimen19\font=\@@_get_fontparam:nn {20} {SubscriptBaselineDropMin}\relax
708       \fontdimen20\font=0pt\relax % delim1 = FractionDelimiterDisplayStyle
709       \fontdimen21\font=0pt\relax % delim2 = FractionDelimiterSize
710       \fontdimen22\font=\@@_get_fontparam:nn {15} {AxisHeight}\relax
711     }
712   } {\l_@@_fontname_tl}
713   \SetSymbolFont{symbols}{\l_@@_mversion_tl}
714   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
715
716   \tl_set:Nn \l_@@_tmpa_tl {normal}
717   \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
718   {
719     \SetSymbolFont{symbols}{bold}
720     {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
721   }
722 }

```

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

```

723 \cs_new:Nn \@@_setup_legacy_fam_three:
724 {

```

```

725 \fontspec_set_family:Nxn \l_@@_family_tl
726 {
727   \l_@@_font_keyval_tl,
728   Scale=0.99999,
729   FontAdjustment={
730     \fontdimen8\font= \@@_get_fontparam:nn {48} {FractionRuleThickness}\relax
731     \fontdimen9\font= \@@_get_fontparam:nn {28} {UpperLimitGapMin}\relax
732     \fontdimen10\font=\@@_get_fontparam:nn {30} {LowerLimitGapMin}\relax
733     \fontdimen11\font=\@@_get_fontparam:nn {29} {UpperLimitBaselineRiseMin}\relax
734     \fontdimen12\font=\@@_get_fontparam:nn {31} {LowerLimitBaselineDropMin}\relax
735     \fontdimen13\font=0pt\relax
736   }
737 } {\l_@@_fontname_tl}
738 \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
739   {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
740
741 \tl_set:Nn \l_@@_tmpa_tl {normal}
742 \tl_if_eq:NNT \l_@@_mversion_tl \l_@@_tmpa_tl
743 {
744   \SetSymbolFont{largesymbols}{bold}
745   {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
746 }
747 }

```

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

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

```

751 \cs_new:Nn \@@_fontspec_select_font:
752 {
753   \tl_set:Nx \l_@@_font_keyval_tl {
754     (LU) Renderer = Basic,
755     BoldItalicFont = {}, ItalicFont = {},
756     Script = Math,
757     SizeFeatures =
758     {
759       {
760         Size = \tf@size-
761       } ,
762       {
763         Size = \sf@size-\tf@size ,
764         Font = \l_@@_script_font_tl ,
765         \l_@@_script_features_tl
766       } ,
767       {
768         Size = -\sf@size ,
769         Font = \l_@@_sscript_font_tl ,
770         \l_@@_sscript_features_tl
771       }

```

```

772     } ,
773     \l_@@_unknown_keys_clist
774 }
775 \fontspec_set_fontface:NNxn \l_@@_font \l_@@_family_tl
776 {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

```

Check whether we're using a real maths font:

```

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

```

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.

```

782 \cs_set:Nn \@@_process_symbol_noparse:nnn
783 {
784   \@@_set_mathsymbol:nNNn {\@@_symfont_tl} #2 #3 {#1}
785 }
786 \cs_set:Nn \@@_process_symbol_parse:nnn
787 {
788   \@@_if_char_spec:nNNT {#1} {#2} {#3}
789   {
790     \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
791   }
792 }

```

`\@@_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`

```

793 \cs_new:Npn \@@_remap_symbols:
794 {
795   \@@_remap_symbol:nnn{'-}{\mathbin}{"02212}% hyphen to minus
796   \@@_remap_symbol:nnn{'*}{\mathbin}{"02217}% text asterisk to "centred asterisk"
797   \bool_if:NF \g_@@_literal_colon_bool
798   {
799     \@@_remap_symbol:nnn{'\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
800   }
801 }

```

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

```

802 \cs_new:Nn \@@_remap_symbol_parse:nnn
803 {
804   \@@_if_char_spec:nNNT {#3} {\@nil} {#2}
805   { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
806 }
807 \cs_new:Nn \@@_remap_symbol_noparse:nnn

```

```

808 {
809   \clist_map_inline:nn {#1}
810   { \@@_set_mathcode:nnnn {##1} {#2} {\@@_symfont_tl} {#3} }
811 }

```

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:`

```

812 \cs_new:Npn \@@_setup_mathactives:
813 {
814   \@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
815   \@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
816   \@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
817   \@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar \mathord
818   \@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
819   \@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
820   \@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
821   \@@_make_mathactive:nNN {'\'} \mathstraightquote \mathord
822   \@@_make_mathactive:nNN {'\'} \mathbacktick \mathord
823 }

```

`\@@_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!

```

824 \cs_new:Nn \@@_make_mathactive_parse:nNN
825 {
826   \@@_if_char_spec:nNNT {#1} #2 #3
827   { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
828 }
829 \cs_new:Nn \@@_make_mathactive_noparse:nNN
830 {
831   \@@_set_mathchar:NNnn #2 #3 {\@@_symfont_tl} {#1}
832   \@@_char_gmake_mathactive:n {#1}
833 }

```

G.4.3 Delimiter codes

`\@@_assign_delcode:nn`

```

834 \cs_new:Nn \@@_assign_delcode_noparse:nn
835 {
836   \@@_set_delcode:nnn \@@_symfont_tl {#1} {#2}
837 }
838 \cs_new:Nn \@@_assign_delcode_parse:nn
839 {
840   \@@_if_char_spec:nNNT {#2} {\@nil} {\@nil}
841   {
842     \@@_assign_delcode_noparse:nn {#1} {#2}

```

```

843 }
844 }

```

`\@@_assign_delcode:n` Shorthand.

```

845 \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.

```

846 \cs_new:Npn \@@_setup_delcodes:
847 {
848 % ensure \left. and \right. work:
849 \@@_set_delcode:nnn \@@_symfont_tl {\'.} {\c_zero}
850 % this is forcefully done to fix a bug -- indicates a larger problem!
851
852 \@@_assign_delcode:nn {\'/} {\g_@@_slash_delimiter_usv}
853 \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
854 \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
855 \@@_assign_delcode:n {"005C} % backslash
856 \@@_assign_delcode:nn {\'<} {"27E8} % angle brackets with ascii notation
857 \@@_assign_delcode:nn {\'>} {"27E9} % angle brackets with ascii notation
858 \@@_assign_delcode:n {"2191} % up arrow
859 \@@_assign_delcode:n {"2193} % down arrow
860 \@@_assign_delcode:n {"2195} % updown arrow
861 \@@_assign_delcode:n {"219F} % up arrow twohead
862 \@@_assign_delcode:n {"21A1} % down arrow twohead
863 \@@_assign_delcode:n {"21A5} % up arrow from bar
864 \@@_assign_delcode:n {"21A7} % down arrow from bar
865 \@@_assign_delcode:n {"21A8} % updown arrow from bar
866 \@@_assign_delcode:n {"21BE} % up harpoon right
867 \@@_assign_delcode:n {"21BF} % up harpoon left
868 \@@_assign_delcode:n {"21C2} % down harpoon right
869 \@@_assign_delcode:n {"21C3} % down harpoon left
870 \@@_assign_delcode:n {"21C5} % arrows up down
871 \@@_assign_delcode:n {"21F5} % arrows down up
872 \@@_assign_delcode:n {"21C8} % arrows up up
873 \@@_assign_delcode:n {"21CA} % arrows down down
874 \@@_assign_delcode:n {"21D1} % double up arrow
875 \@@_assign_delcode:n {"21D3} % double down arrow
876 \@@_assign_delcode:n {"21D5} % double updown arrow
877 \@@_assign_delcode:n {"21DE} % up arrow double stroke
878 \@@_assign_delcode:n {"21DF} % down arrow double stroke
879 \@@_assign_delcode:n {"21E1} % up arrow dashed
880 \@@_assign_delcode:n {"21E3} % down arrow dashed
881 \@@_assign_delcode:n {"21E7} % up white arrow
882 \@@_assign_delcode:n {"21E9} % down white arrow
883 \@@_assign_delcode:n {"21EA} % up white arrow from bar
884 \@@_assign_delcode:n {"21F3} % updown white arrow

```

885 }

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 *etc.*, `\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.

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

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

```
895 \DeclareDocumentCommand \addnolimits {m}
896 {
897   \tl_put_right:Nn \l_@@_nolimits_tl {#1}
898 }
```

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

```
899 \DeclareDocumentCommand \removenolimits {m}
900 {
901   \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
902 }
```

G.6 Radicals

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

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

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

```
903 \tl_new:N \l_@@_radicals_tl
```

```
904 \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 `\fontdimens`; 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 if it is not already defined. For X_YTeX, define *⟨command⟩* as a new quark.

```
905 \cs_new_protected_nopar:Nn \@@_new_cramped_style:N
```

```
906 <XE> { \quark_new:N #1 }
```

```
907 <LU> {
```

```

908 <LU> \cs_if_exist:NF #1
909 <LU> { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
910 <LU> }

```

`\crampeddisplaystyle` The cramped style commands.

```

\crampedtextstyle 911 \@@_new_cramped_style:N \crampeddisplaystyle
\crampedscriptstyle 912 \@@_new_cramped_style:N \crampedtextstyle
\crampedscriptscriptstyle 913 \@@_new_cramped_style:N \crampedscriptstyle
914 \@@_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 X \LaTeX , 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 X \LaTeX 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.

```

915 <*XE>
916 \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
917   \fontdimen
918   \cs_if_eq:NNTF #1 \displaystyle {
919     #2 \textfont
920   } {
921     \cs_if_eq:NNTF #1 \crampeddisplaystyle {
922       #3 \textfont
923     } {
924       \cs_if_eq:NNTF #1 \textstyle {
925         #4 \textfont
926       } {
927         \cs_if_eq:NNTF #1 \crampedtextstyle {
928           #5 \textfont
929         } {
930           \cs_if_eq:NNTF #1 \scriptstyle {
931             #4 \scriptfont
932           } {
933             \cs_if_eq:NNTF #1 \crampedscriptstyle {
934               #5 \scriptfont
935             } {
936               \cs_if_eq:NNTF #1 \scriptscriptstyle {
937                 #4 \scriptscriptfont
938               } {

```


Should we check here if the style is invalid?

```

939             #5 \scriptscriptfont
940         }
941     }
942 }
943 }
944 }
945 }
946 }
```

Which family to use?

```

947     \c_two
948 }
949 </XE>
```

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

`\@@_font_param:nnnnn` #1 : name
 #2 : font dimension for non-cramped display style
 #3 : font dimension for cramped display style
 #4 : font dimension for non-cramped non-display styles
 #5 : font dimension for cramped non-display styles
 This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The Lua_T_EX font parameter name is produced by removing all underscores and prefixing the result with `Umath`. The X_Y_T_EX font dimension numbers must be integer constants.

```

950 \cs_new_protected_nopar:Nn \@@_font_param:nnnnn
951 <*XE>
952 {
953   \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
954     { #2 } { #3 } { #4 } { #5 }
955 }
956 </XE>
957 <*LU>
958 {
959   \tl_set:Nn \l_@@_tmpa_tl { #1 }
960   \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
961   \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
962     { Umath \l_@@_tmpa_tl }
963 }
964 </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 Lua_T_EX font parameter name is produced by removing all underscores and pre-

fixing the result with `Umath`. The $\text{X}\text{\TeX}$ font dimension numbers must be integer constants.

```

965 \cs_new_protected_nopar:Nn \@@_font_param:nnn
966 {
967   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
968 }

```

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

#2 : font dimension

This macro defines getter and setter functions for the font parameter $\langle name \rangle$. The $\text{Lua}\text{\TeX}$ font parameter name is produced by removing all underscores and prefixing the result with `Umath`. The $\text{X}\text{\TeX}$ font dimension number must be an integer constant.

```

969 \cs_new_protected_nopar:Nn \@@_font_param:nn
970 {
971   \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
972 }

```

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

This macro defines getter and setter functions for the font parameter $\langle name \rangle$, which is considered unavailable in $\text{X}\text{\TeX}$. The $\text{Lua}\text{\TeX}$ font parameter name is produced by removing all underscores and prefixing the result with `Umath`.

```

973 \cs_new_protected_nopar:Nn \@@_font_param:n
974 (XE) { }
975 (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`

```

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

```

```

997 \cs_new_protected_nopar:Npn #2 ##1 ##2
998 {
999     #3 ##1 \dim_eval:n { ##2 }
1000 }
1001 }
1002 \cs_generate_variant:Nn \@@_font_param_aux:NNN { ccc }
1003 </LU>

```

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

```

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

```

```

1044 \@@_font_param:nn { supsub_bottom_max } { 26 }
1045 \@@_font_param:nn { underbar_kern } { 58 }
1046 \@@_font_param:nn { underbar_rule } { 57 }
1047 \@@_font_param:nn { underbar_vgap } { 56 }
1048 \@@_font_param:n { connector_overlap_min }

```

H Font features

H.1 Math version

```

1049 \keys_define:nn {unicode-math}
1050 {
1051   version .code:n =
1052   {
1053     \tl_set:Nn \l_@@_mversion_tl {#1}
1054     \DeclareMathVersion {\l_@@_mversion_tl}
1055   }
1056 }

```

H.2 Script and scriptscript font options

```

1057 \keys_define:nn {unicode-math}
1058 {
1059   script-features .tl_set:N = \l_@@_script_features_tl ,
1060   sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
1061   script-font .tl_set:N = \l_@@_script_font_tl ,
1062   sscript-font .tl_set:N = \l_@@_sscript_font_tl ,
1063 }

```

H.3 Range processing

```

1064 \keys_define:nn {unicode-math}
1065 {
1066   range .code:n =
1067   {
1068     \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

```

1069   \int_incr:N \g_@@_fam_int
1070   \tl_set:Nx \@@_symfont_tl {@@_fam\int_use:N\g_@@_fam_int}
1071   \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
1072   \cs_set_eq:NN \@@_set_mathalphabet_char:Nnn \@@_mathmap_parse:Nnn
1073   \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
1074   \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
1075   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
1076   \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
1077   \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_parse:nNN

```

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

```

1078 \seq_clear:N \l_@@_char_range_seq
1079 \seq_clear:N \l_@@_mclass_range_seq
1080 \seq_clear:N \l_@@_cmd_range_seq
1081 \seq_clear:N \l_@@_mathalph_seq
1082
1083 \clist_map_inline:nn {#1}
1084 {
1085   \@@_if_mathalph_decl:nTF {##1}
1086   {
1087     \seq_put_right:Nx \l_@@_mathalph_seq
1088     {
1089       { \exp_not:V \l_@@_tmpa_tl }
1090       { \exp_not:V \l_@@_tmpb_tl }
1091       { \exp_not:V \l_@@_tmpc_tl }
1092     }
1093   }
1094   {

```

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.

```

1095     \seq_if_in:NnTF \g_@@_mathclasses_seq {##1}
1096     { \seq_put_right:Nn \l_@@_mclass_range_seq {##1} }
1097     {
1098       \bool_lazy_and:nnTF { \tl_if_single_p:n {##1} } { \token_if_cs_p:N ##1 }
1099       { \seq_put_right:Nn \l_@@_cmd_range_seq {##1} }
1100       { \seq_put_right:Nn \l_@@_char_range_seq {##1} }
1101     }
1102   }
1103 }
1104 }
1105 }

```

\@@_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.

```

1106 \prg_new_conditional:Nnn \@@_if_mathalph_decl:n {TF}
1107 {
1108   \tl_set:Nn \l_@@_tmpa_tl {#1}
1109   \tl_clear:N \l_@@_tmpb_tl
1110   \tl_clear:N \l_@@_tmpc_tl

```

```

1111
1112 \tl_if_in:NnT \l_@@_tmpa_tl {->}
1113 { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }
1114
1115 \tl_if_in:NnT \l_@@_tmpa_tl {/}
1116 { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }
1117
1118 \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
1119 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
1120 \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
1121 \tl_trim_spaces:N \l_@@_tmpa_tl
1122
1123 \tl_if_empty:NT \l_@@_tmpc_tl
1124 { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }
1125
1126 \seq_if_in:NVTF \g_@@_named_ranges_seq \l_@@_tmpa_tl
1127 { \prg_return_true: } { \prg_return_false: }
1128 }
1129 \cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
1130 {
1131   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1132   \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
1133 }
1134 \cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
1135 {
1136   \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
1137   \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
1138 }

```

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.

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

```

1139 \cs_new:Nn \@@_if_char_spec:nNNT
1140 {
1141   % math class:
1142   \seq_if_in:NnT \l_@@_mclass_range_seq {#3}

```

Table 13: Ranges accepted by `\@@_if_char_spec:nNT`.

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

```

1143 { \use_none_delimit_by_q_nil:w }
1144
1145 % command name:
1146 \seq_if_in:NnT \l_@@_cmd_range_seq {#2}
1147 { \use_none_delimit_by_q_nil:w }
1148
1149 % character slot:
1150 \seq_map_inline:Nn \l_@@_char_range_seq
1151 {
1152   \@@_int_if_slot_in_range:nnT {#1} {##1}
1153   { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
1154 }
1155
1156 % the following expands to nil if no match was found:
1157 \use_none:nnn
1158 \q_nil
1159 \use:n
1160 {
1161   \clist_put_right:Nx \l_@@_char_nrange_clist { \int_eval:n {#1} }
1162   #4
1163 }
1164 }
```

`\@@_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
```

```

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

```

1174     {
1175     \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
1176     } } }
1177 }

```

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.

```

1178 \AtBeginDocument{\@@_resolve_greek:}
1179 \cs_new:Npn \@@_resolve_greek:
1180 {
1181   \clist_map_inline:nn
1182   {
1183     Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
1184     alpha,beta,gamma,delta,epsilon,zeta,eta,theta,iota,kappa,lambda,
1185     Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
1186     mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,phi,chi,psi,omega,
1187     varTheta,varsigma,vartheta,varkappa,varrho,varpi,varepsilon,varphi
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 }

```

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 $\mathcal{L}\mathcal{T}_{\mathcal{E}}\mathcal{X}2_{\epsilon}$

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.

```

1195 \cs_set:Npn \use@mathgroup #1 #2
1196 {
1197   \mode_if_math:T % <- not sure if this is really necessary since we’ve just checked for mmode and
1198   {
1199     \math@bgroup
1200     \cs_if_eq:cNF {M@f@encoding} #1 {#1}
1201     \@_switchto_literal:
1202     \mathgroup #2 \relax
1203     \math@egroup
1204   }
1205 }

```

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.

```

1206 \cs_new:Nn \@@_prepare_mathstyle:n
1207 {
1208   \seq_put_right:Nn \g_@@_mathstyles_seq {#1}
1209   \@@_init_alphabet:n {#1}
1210   \cs_set:cpn {_@@_sym_#1_aux:n}
1211   { \use:c {@@_switchto_#1:} \math@egroup }
1212   \cs_set_protected:cpx {sym#1}
1213   {
1214     \exp_not:n
1215     {
1216       \math@bgroup
1217       \mode_if_math:F
1218       {
1219         \egroup\expandafter
1220         \non@alpherr\expandafter{\csname sym#1\endcsname\space}
1221       }
1222       \tl_set:Nn \l_@@_mathstyle_tl {#1}
1223     }
1224     \exp_not:c {_@@_sym_#1_aux:n}
1225   }
1226 }
```

`\@@_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.

```

1227 \cs_set:Nn \@@_init_alphabet:n
1228 {
1229   \@@_log:nx {alph-initialise} {#1}
1230   \cs_set_eq:cN {@@_switchto_#1:} \prg_do_nothing:
```

```
1231 }
```

I.4 Definition of alphabets and styles

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

```
1232 \cs_new:Nn \@@_new_named_range:n
1233 {
1234   \prop_new:c {g_@@_named_range_#1_prop}
1235 }
1236 \clist_set:Nn \g_@@_named_ranges_clist
1237 {
1238   up, it, tt, bfup, bfit, bb , bbit, scr, bfscr, cal, bfcalf,
1239   frak, bffrak, sfup, sfit, bfsfup, bfsfit, bfsf
1240 }
1241 \clist_map_inline:Nn \g_@@_named_ranges_clist
1242 { \@@_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.

```
1243 \clist_new:N \g_@@_alphabets_seq
1244 \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.

```
1245 \cs_new:Nn \@@_new_alphabet_config:nnn
1246 {
1247   \prop_if_exist:cF {g_@@_named_range_#1_prop}
1248   { \@@_warning:nnn {no-named-range} {#1} {#2} }
1249
1250   \prop_gput:cnn {g_@@_named_range_#1_prop} { alpha_tl }
1251   {
1252     \prop_item:cn {g_@@_named_range_#1_prop} { alpha_tl }
1253     {#2}
1254   }
1255   % Q: do I need to bother removing duplicates?
1256
1257   \cs_new:cn { @@_config_#1_#2:n } {#3}
1258 }
1259 \cs_new:Nn \@@_alphabet_config:nnn
1260 {
1261   \use:c {@@_config_#1_#2:n} {#3}
1262 }
1263 \prg_new_conditional:Nnn \@@_if_alphabet_exists:nn {T,TF}
1264 {
1265   \cs_if_exist:cTF {@@_config_#1_#2:n}
```

```

1266 \prg_return_true: \prg_return_false:
1267 }

```

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.

```

1268 \cs_new:Nn \@@_default_mathalph:nnn
1269 {
1270   \seq_put_right:Nx \g_@@_named_ranges_seq { \tl_to_str:n {#1} }
1271   \seq_put_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
1272   \prop_gput:cnn { g_@@_named_range_#1_prop } { default-alpha } {#2}
1273 }
1274 \@@_default_mathalph:nnn {up} { } {latin, Latin, greek, Greek, num, misc} {up} { }
1275 \@@_default_mathalph:nnn {it} { } {latin, Latin, greek, Greek, misc} {it} { }
1276 \@@_default_mathalph:nnn {bb} { } {latin, Latin, num, misc} {bb} { }
1277 \@@_default_mathalph:nnn {bbit} { } {misc} {bbit} { }
1278 \@@_default_mathalph:nnn {scr} { } {latin, Latin} {scr} { }
1279 \@@_default_mathalph:nnn {cal} { } {Latin} {scr} { }
1280 \@@_default_mathalph:nnn {bfcal} { } {Latin} {bfscr} { }
1281 \@@_default_mathalph:nnn {frak} { } {latin, Latin} {frak} { }
1282 \@@_default_mathalph:nnn {tt} { } {latin, Latin, num} {tt} { }
1283 \@@_default_mathalph:nnn {sfup} { } {latin, Latin, num} {sfup} { }
1284 \@@_default_mathalph:nnn {sfit} { } {latin, Latin} {sfit} { }
1285 \@@_default_mathalph:nnn {bfup} { } {latin, Latin, greek, Greek, num, misc} {bfup} { }
1286 \@@_default_mathalph:nnn {bfit} { } {latin, Latin, greek, Greek, misc} {bfit} { }
1287 \@@_default_mathalph:nnn {bfscr} { } {latin, Latin} {bfscr} { }
1288 \@@_default_mathalph:nnn {bffrak} { } {latin, Latin} {bffrak} { }
1289 \@@_default_mathalph:nnn {bfsfup} { } {latin, Latin, greek, Greek, num, misc} {bfsfup} { }
1290 \@@_default_mathalph:nnn {bfsfit} { } {latin, Latin, greek, Greek, misc} {bfsfit} { }

```

I.4.1 Define symbol style commands

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

```

1291 \clist_map_inline:nn
1292 {
1293   up, it, bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf,
1294   tt, bb, bbit, scr, bfscr, cal, bfcal, frak, bffrak,
1295   normal, literal, sf, bf,
1296 }
1297 { \@@_prepare_mathstyle:n {#1} }

```

I.4.2 New names for legacy textmath alphabet selection

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

```

1298 \clist_map_inline:nn
1299 { rm, it, bf, sf, tt }
1300 { \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.

```

1301 \clist_map_inline:nn
1302 {
1303   normal, bb , bbit, scr, bfscr, cal, bfcalf, frak, bffrak, tt,
1304   bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
1305 }
1306 {
1307   \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
1308 }
```

I.4.4 New commands for ambiguous alphabets

```

1309 \AtBeginDocument{
1310 \clist_map_inline:nn
1311 { rm, it, bf, sf, tt }
1312 {
1313   \cs_set_protected:cpx { math #1 }
1314   {
1315     \exp_not:n { \bool_if:NTF } \exp_not:c { g_@_ math #1 _text_bool }
1316     { \exp_not:c { mathtext #1 } }
1317     { \exp_not:c { sym #1 } }
1318   }
1319 }}
```

Alias \mathrm as legacy name for \mathup

```

1320 \cs_set_protected:Npn \mathup { \mathrm }
1321 \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
1322 \cs_set:Npn \operator@font
1323 {
1324   \@@_switchto_literal:
1325   \@fontswitch {} { \g_@@_operator_mathfont_tl }
1326 }

```

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.

```

1327 \cs_new:Npn \@@_setup_alphabets:
1328 {

```

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

```

1329 \seq_if_empty:NTF \l_@@_mathalph_seq
1330 {
1331   \@@_log:n {setup-implicit}
1332   \seq_set_eq:NN \l_@@_mathalph_seq \g_@@_default_mathalph_seq
1333   \bool_set_true:N \l_@@_implicit_alph_bool
1334   \@@_maybe_init_alphabet:n {sf}
1335   \@@_maybe_init_alphabet:n {bf}
1336   \@@_maybe_init_alphabet:n {bfsf}
1337 }

```

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

```

1338 {
1339   \@@_log:n {setup-explicit}
1340   \bool_set_false:N \l_@@_implicit_alph_bool
1341   \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
1342   \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
1343 }
1344
1345 % Now perform the mapping:
1346 \seq_map_inline:Nn \l_@@_mathalph_seq
1347 {
1348   \tl_set:No \l_@@_style_tl { \use_i:nnn ##1 }
1349   \clist_set:No \l_@@_alphabet_clist { \use_ii:nnn ##1 }
1350   \tl_set:No \l_@@_remap_style_tl { \use_iii:nnn ##1 }
1351
1352   % If no set of alphabets is defined:
1353   \clist_if_empty:NT \l_@@_alphabet_clist
1354   {
1355     \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
1356     \prop_get:cnN { g_@@_named_range_ \l_@@_style_tl _prop }
1357     { default-alpha } \l_@@_alphabet_clist
1358   }
1359
1360   \@@_setup_math_alphabet:
1361 }
1362 \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
1363 }

```

\@@_setup_math_alphabet:

```

1364 \cs_new:Nn \@@_setup_math_alphabet:
1365 {

```

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

```

1366 \clist_map_inline:Nn \l_@@_alphabet_clist
1367 {
1368   \tl_set:Nn \l_@@_alphabet_tl {##1}
1369   \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
1370   {
1371     \str_if_eq:x:nnTF {\l_@@_alphabet_tl} {misc}

```

```

1372 {
1373   \@@_maybe_init_alphabet:n \l_@@_style_tl
1374   \clist_map_break:
1375 }
1376 {
1377   \@@_glyph_if_exist:nT { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
1378   {
1379     \@@_maybe_init_alphabet:n \l_@@_style_tl
1380     \clist_map_break:
1381   }
1382 }
1383 }
1384 { \msg_warning:nnx {unicode-math} {no-alphabet} { \l_@@_style_tl / \l_@@_alphabet_tl } }
1385 }

```

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

```

1386 (debug) \csname TIC\endcsname
1387 \clist_map_inline:Nn \l_@@_alphabet_clist
1388 {
1389   \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
1390   \cs_if_exist:cT {@@_config_ \l_@@_style_tl _ \l_@@_alphabet_tl :n}
1391   {
1392     \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}
1393     {
1394       \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1395       \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
1396     }
1397     {
1398       \@@_glyph_if_exist:nTF { \@@_to_usv:nn {\l_@@_remap_style_tl} {\l_@@_alphabet_tl} }
1399       {
1400         \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
1401         \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
1402       }
1403       {
1404         \bool_if:NTF \l_@@_implicit_alph_bool
1405         {
1406           \seq_put_right:Nx \l_@@_missing_alph_seq
1407           {
1408             \@backslashchar sym \l_@@_style_tl \space
1409             (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
1410           }
1411         }
1412         {
1413           \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {up}
1414         }
1415       }
1416     }
1417   }
1418 }

```



```

1419 <debug> \csname TOC\endcsname
1420 }

```

I.6 Mapping ‘naked’ math characters

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

I.6.1 Functions

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

```

\@@_map_char_noparse:nn
\@@_map_char_parse:nn
1421 \cs_new:Nn \@@_map_char_noparse:nn
1422 { \@@_set_mathcode:nnnn {#1}{\mathalpha}{\@@_symfont_tl}{#2} }
1423 \cs_new:Nn \@@_map_char_parse:nn
1424 {
1425   \@@_if_char_spec:nNNT {#1} {\@nil} {\mathalpha}
1426   { \@@_map_char_noparse:nn {#1}{#2} }
1427 }

```

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

```

1428 \cs_new:Nn \@@_map_char_single:nnn
1429 {
1430   \@@_map_char_single:nn { \@@_to_usv:nn {#1}{#3} }
1431   { \@@_to_usv:nn {#2}{#3} }
1432 }

```

`\@@_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:

```

1433 \cs_set:Nn \@@_map_chars_range:nnn
1434 {
1435   \int_step_inline:nnnn {0}{1}{#1-1}
1436   { \@@_map_char_single:nn {#2+##1}{#3+##1} }
1437 }

```

And the wrapper with names:

```

1438 \cs_new:Nn \@@_map_chars_range:nnnn
1439 {
1440   \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2}{#4} }

```

```

1441 { \@@_to_usv:nn {#3}{#4} }
1442 }

```

I.6.2 Functions for ‘normal’ alphabet symbols

\@@_set_normal_char:nnn

```

1443 \cs_set:Nn \@@_set_normal_char:nnn
1444 {
1445   \@@_usv_if_exist:nnT {#3} {#1}
1446   {
1447     \clist_map_inline:nn {#2}
1448     {
1449       \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
1450       \@@_map_char_single:nnn {##1} {#3} {#1}
1451     }
1452   }
1453 }

1454 \cs_new:Nn \@@_set_normal_Latin:nn
1455 {
1456   \clist_map_inline:nn {#1}
1457   {
1458     \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
1459     \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}
1460   }
1461 }

1462 \cs_new:Nn \@@_set_normal_latin:nn
1463 {
1464   \clist_map_inline:nn {#1}
1465   {
1466     \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
1467     \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
1468   }
1469 }

1470 \cs_new:Nn \@@_set_normal_greek:nn
1471 {
1472   \clist_map_inline:nn {#1}
1473   {
1474     \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
1475     \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
1476     \@@_map_char_single:nnn {##1} {#2} {epsilon}
1477     \@@_map_char_single:nnn {##1} {#2} {vartheta}
1478     \@@_map_char_single:nnn {##1} {#2} {varkappa}
1479     \@@_map_char_single:nnn {##1} {#2} {phi}
1480     \@@_map_char_single:nnn {##1} {#2} {varrho}
1481     \@@_map_char_single:nnn {##1} {#2} {varpi}
1482     \@@_set_mathalphabet_pos:nnnn {normal} {epsilon} {##1} {#2}
1483     \@@_set_mathalphabet_pos:nnnn {normal} {vartheta} {##1} {#2}
1484     \@@_set_mathalphabet_pos:nnnn {normal} {varkappa} {##1} {#2}

```

```

1485 \@@_set_mathalphabet_pos:nnnn {normal} {phi} {##1} {#2}
1486 \@@_set_mathalphabet_pos:nnnn {normal} {varrho} {##1} {#2}
1487 \@@_set_mathalphabet_pos:nnnn {normal} {varpi} {##1} {#2}
1488 }
1489 }

1490 \cs_new:Nn \@@_set_normal_Greek:nn
1491 {
1492   \clist_map_inline:nn {#1}
1493   {
1494     \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
1495     \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
1496     \@@_map_char_single:nnn {##1} {#2} {varTheta}
1497     \@@_set_mathalphabet_pos:nnnn {normal} {varTheta} {##1} {#2}
1498   }
1499 }

1500 \cs_new:Nn \@@_set_normal_numbers:nn
1501 {
1502   \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
1503   \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
1504 }

```

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.

```

1505 \cs_new:Nn \@@_mathmap_noparse:nnn
1506 {
1507   \clist_map_inline:nn {#2}
1508   {
1509     \tl_put_right:cx {@@_switchto_#1:}
1510     {
1511       \@@_set_mathcode:nnnn {##1} {\mathalpha} {\@@_symfont_tl} {#3}
1512     }
1513   }
1514 }

```

`\@@_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:nnn` declarations to the maths alphabet definition.

```
1515 \cs_new:Nn \@@_mathmap_parse:nnn
1516 {
1517   \clist_if_in:NnT \l_@@_char_nrange_clist {#3}
1518   {
1519     \@@_mathmap_noparse:nnn {#1}{#2}{#3}
1520   }
1521 }
```

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

```
1522 \cs_new:Nn \@@_set_mathalphabet_char:nnnn
1523 {
1524   \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
1525                                     { \@@_to_usv:nn {#3} {#4} }
1526 }
```

`\@@_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:

```
1527 \cs_new:Nn \@@_set_mathalph_range:nnnn
1528 {
1529   \int_step_inline:nnnn {0} {1} {#1-1}
1530   { \@@_set_mathalphabet_char:nnn {#2} { ##1 + #3 } { ##1 + #4 } }
1531 }
```

Then the wrapper version that uses names:

```
1532 \cs_new:Nn \@@_set_mathalph_range:nnnnn
1533 {
1534   \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
1535                                     { \@@_to_usv:nn {#4} {#5} }
1536 }
```

1.7.2 Individual mapping functions for different alphabets

```
1537 \cs_new:Nn \@@_set_mathalphabet_pos:nnnn
1538 {
1539   \@@_usv_if_exist:nnT {#4} {#2}
1540   {
1541     \clist_map_inline:nn {#3}
1542     { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }
```

```

1543     }
1544 }
1545 \cs_new:Nn \@@_set_mathalphabet_numbers:nnn
1546 {
1547   \clist_map_inline:nn {#2}
1548   { \@@_set_mathalph_range:nnnn {10} {#1} {##1} {#3} {num} }
1549 }
1550 \cs_new:Nn \@@_set_mathalphabet_Latin:nnn
1551 {
1552   \clist_map_inline:nn {#2}
1553   { \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {Latin} }
1554 }
1555 \cs_new:Nn \@@_set_mathalphabet_latin:nnn
1556 {
1557   \clist_map_inline:nn {#2}
1558   {
1559     \@@_set_mathalph_range:nnnn {26} {#1} {##1} {#3} {latin}
1560     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {h}
1561   }
1562 }
1563 \cs_new:Nn \@@_set_mathalphabet_Greek:nnn
1564 {
1565   \clist_map_inline:nn {#2}
1566   {
1567     \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {Greek}
1568     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varTheta}
1569   }
1570 }
1571 \cs_new:Nn \@@_set_mathalphabet_greek:nnn
1572 {
1573   \clist_map_inline:nn {#2}
1574   {
1575     \@@_set_mathalph_range:nnnn {25} {#1} {##1} {#3} {greek}
1576     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {epsilon}
1577     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {vartheta}
1578     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varkappa}
1579     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {phi}
1580     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varrho}
1581     \@@_set_mathalphabet_char:nnnn {#1} {##1} {#3} {varpi}
1582   }
1583 }

```

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.

```

1584 \cs_new:Npn \@@_symbol_setup:
1585 {
1586   \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
1587   {
1588     \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
1589   }
1590 }

1591 \tl_set_from_file:x:Nnn \g_@@_mathtable_tl {\@@_symbol_setup:} {unicode-math-table.tex}

```

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

```

1592 \cs_new:Nn \@@_input_math_symbol_table: {\g_@@_mathtable_tl}

```

K Definitions of the active math characters

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`.

```

1593 \AtBeginDocument{\@@_define_math_chars:}
1594 \cs_new:Nn \@@_define_math_chars:
1595 {
1596   \group_begin:
1597   \cs_set:Npn \_@@_sym:nnn ##1##2##3
1598   {
1599     \tl_if_in:nnT
1600     { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
1601     {##3}
1602     {
1603       \exp_last_unbraced:NNx \cs_gset_eq:NN ##2 { \Ucharcat ##1 ~ 12 ~ }
1604     }
1605   }
1606   \@@_input_math_symbol_table:
1607   \group_end:
1608 }

```

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.

```

1609 \AtBeginDocument { \@@_load_lm_if_necessary: }
1610 \cs_new:Nn \@@_load_lm_if_necessary:
1611 {
1612   \cs_if_exist:NF \l_@@_fontname_tl

```

```

1613 {
1614   % TODO: update this when lmmath-bold.otf is released
1615   \setmathfont{latinmodern-math.otf}[BoldFont={latinmodern-math.otf}]
1616   \bool_set_false:N \g_@@_mainfont_already_set_bool
1617 }
1618 }

```

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.

```

1619 \cs_new:Nn \@@_arg_i_before_egroup:n {#1\egroup}
1620 \cs_new:Nn \@@_superscript:n
1621 {
1622   ^\bgroup #1
1623   \peek_meaning_remove:NTF ^ \@@_arg_i_before_egroup:n \egroup
1624 }

1625 \cs_new:Nn \@@_nprimes:Nn
1626 {
1627   \@@_superscript:n
1628   {
1629     #1
1630     \prg_replicate:nn {#2-1} { \mskip \g_@@_primekern_muskip #1 }
1631   }
1632 }

1633
1634 \cs_new:Nn \@@_nprimes_select:nn
1635 {
1636   \int_case:nnF {#2}
1637   {
1638     {1} { \@@_superscript:n {#1} }
1639     {2} {
1640       \@@_glyph_if_exist:nTF {"2033}
1641       { \@@_superscript:n {\@@_prime_double_mchar} }
1642       { \@@_nprimes:Nn #1 {#2} }
1643     }
1644     {3} {
1645       \@@_glyph_if_exist:nTF {"2034}
1646       { \@@_superscript:n {\@@_prime_triple_mchar} }
1647       { \@@_nprimes:Nn #1 {#2} }
1648     }
1649     {4} {
1650       \@@_glyph_if_exist:nTF {"2057}
1651       { \@@_superscript:n {\@@_prime_quad_mchar} }
1652       { \@@_nprimes:Nn #1 {#2} }
1653     }
1654   }
1655   {
1656     \@@_nprimes:Nn #1 {#2}
1657   }
1658 }

1659 \cs_new:Nn \@@_nbackprimes_select:nn
1660 {
1661   \int_case:nnF {#2}

```



```

1662 {
1663   {1} { \@@_superscript:n {#1} }
1664   {2} {
1665     \@@_glyph_if_exist:nTF {"2036}
1666     { \@@_superscript:n {\@@_backprime_double_mchar} }
1667     { \@@_nprimes:Nn #1 {#2} }
1668   }
1669   {3} {
1670     \@@_glyph_if_exist:nTF {"2037}
1671     { \@@_superscript:n {\@@_backprime_triple_mchar} }
1672     { \@@_nprimes:Nn #1 {#2} }
1673   }
1674 }
1675 {
1676   \@@_nprimes:Nn #1 {#2}
1677 }
1678 }

```

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

```

1679 \cs_new:Npn \@@_scan_prime:
1680 {
1681   \cs_set_eq:NN \@@_superscript:n \use:n
1682   \int_zero:N \l_@@_primecount_int
1683   \@@_scanprime_collect:N \@@_prime_single_mchar
1684 }
1685 \cs_new:Npn \@@_scan_dprime:
1686 {
1687   \cs_set_eq:NN \@@_superscript:n \use:n
1688   \int_set:Nn \l_@@_primecount_int {1}
1689   \@@_scanprime_collect:N \@@_prime_single_mchar
1690 }
1691 \cs_new:Npn \@@_scan_trprime:
1692 {
1693   \cs_set_eq:NN \@@_superscript:n \use:n
1694   \int_set:Nn \l_@@_primecount_int {2}
1695   \@@_scanprime_collect:N \@@_prime_single_mchar
1696 }
1697 \cs_new:Npn \@@_scan_qprime:
1698 {
1699   \cs_set_eq:NN \@@_superscript:n \use:n
1700   \int_set:Nn \l_@@_primecount_int {3}
1701   \@@_scanprime_collect:N \@@_prime_single_mchar
1702 }
1703 \cs_new:Npn \@@_scan_sup_prime:
1704 {
1705   \int_zero:N \l_@@_primecount_int
1706   \@@_scanprime_collect:N \@@_prime_single_mchar
1707 }
1708 \cs_new:Npn \@@_scan_sup_dprime:
1709 {

```

```

1710 \int_set:Nn \l_@@_primecount_int {1}
1711 \@@_scanprime_collect:N \@@_prime_single_mchar
1712 }
1713 \cs_new:Npn \@@_scan_sup_trprime:
1714 {
1715   \int_set:Nn \l_@@_primecount_int {2}
1716   \@@_scanprime_collect:N \@@_prime_single_mchar
1717 }
1718 \cs_new:Npn \@@_scan_sup_qprime:
1719 {
1720   \int_set:Nn \l_@@_primecount_int {3}
1721   \@@_scanprime_collect:N \@@_prime_single_mchar
1722 }
1723 \cs_new:Nn \@@_scanprime_collect:N
1724 {
1725   \int_incr:N \l_@@_primecount_int
1726   \peek_meaning_remove:NTF '
1727   { \@@_scanprime_collect:N #1 }
1728   {
1729     \peek_meaning_remove:NTF \@@_scan_prime:
1730     { \@@_scanprime_collect:N #1 }
1731     {
1732       \peek_meaning_remove:NTF ^^^^2032
1733       { \@@_scanprime_collect:N #1 }
1734       {
1735         \peek_meaning_remove:NTF \@@_scan_dprime:
1736         {
1737           \int_incr:N \l_@@_primecount_int
1738           \@@_scanprime_collect:N #1
1739         }
1740         {
1741           \peek_meaning_remove:NTF ^^^^2033
1742           {
1743             \int_incr:N \l_@@_primecount_int
1744             \@@_scanprime_collect:N #1
1745           }
1746           {
1747             \peek_meaning_remove:NTF \@@_scan_trprime:
1748             {
1749               \int_add:Nn \l_@@_primecount_int {2}
1750               \@@_scanprime_collect:N #1
1751             }
1752             {
1753               \peek_meaning_remove:NTF ^^^^2034
1754               {
1755                 \int_add:Nn \l_@@_primecount_int {2}
1756                 \@@_scanprime_collect:N #1
1757               }
1758               {

```

```

1759         \peek_meaning_remove:NTF \@@_scan_qprime:
1760         {
1761             \int_add:Nn \l_@@_primecount_int {3}
1762             \@@_scanprime_collect:N #1
1763         }
1764         {
1765             \peek_meaning_remove:NTF ^^^^2057
1766             {
1767                 \int_add:Nn \l_@@_primecount_int {3}
1768                 \@@_scanprime_collect:N #1
1769             }
1770             {
1771                 \@@_nprimes_select:nn {#1} {\l_@@_primecount_int}
1772             }
1773         }
1774     }
1775 }
1776 }
1777 }
1778 }
1779 }
1780 }
1781 }
1782 \cs_new:Npn \@@_scan_backprime:
1783 {
1784     \cs_set_eq:NN \@@_superscript:n \use:n
1785     \int_zero:N \l_@@_primecount_int
1786     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1787 }
1788 \cs_new:Npn \@@_scan_backdprime:
1789 {
1790     \cs_set_eq:NN \@@_superscript:n \use:n
1791     \int_set:Nn \l_@@_primecount_int {1}
1792     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1793 }
1794 \cs_new:Npn \@@_scan_backtrprime:
1795 {
1796     \cs_set_eq:NN \@@_superscript:n \use:n
1797     \int_set:Nn \l_@@_primecount_int {2}
1798     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1799 }
1800 \cs_new:Npn \@@_scan_sup_backprime:
1801 {
1802     \int_zero:N \l_@@_primecount_int
1803     \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1804 }
1805 \cs_new:Npn \@@_scan_sup_backdprime:
1806 {
1807     \int_set:Nn \l_@@_primecount_int {1}

```

```

1808 \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1809 }
1810 \cs_new:Npn \@@_scan_sup_backtrprime:
1811 {
1812 \int_set:Nn \l_@@_primecount_int {2}
1813 \@@_scanbackprime_collect:N \@@_backprime_single_mchar
1814 }
1815 \cs_new:Nn \@@_scanbackprime_collect:N
1816 {
1817 \int_incr:N \l_@@_primecount_int
1818 \peek_meaning_remove:NTF `
1819 {
1820 \@@_scanbackprime_collect:N #1
1821 }
1822 {
1823 \peek_meaning_remove:NTF \@@_scan_backprime:
1824 {
1825 \@@_scanbackprime_collect:N #1
1826 }
1827 {
1828 \peek_meaning_remove:NTF ^^^^2035
1829 {
1830 \@@_scanbackprime_collect:N #1
1831 }
1832 {
1833 \peek_meaning_remove:NTF \@@_scan_backdprime:
1834 {
1835 \int_incr:N \l_@@_primecount_int
1836 \@@_scanbackprime_collect:N #1
1837 }
1838 {
1839 \peek_meaning_remove:NTF ^^^^2036
1840 {
1841 \int_incr:N \l_@@_primecount_int
1842 \@@_scanbackprime_collect:N #1
1843 }
1844 {
1845 \peek_meaning_remove:NTF \@@_scan_backtrprime:
1846 {
1847 \int_add:Nn \l_@@_primecount_int {2}
1848 \@@_scanbackprime_collect:N #1
1849 }
1850 {
1851 \peek_meaning_remove:NTF ^^^^2037
1852 {
1853 \int_add:Nn \l_@@_primecount_int {2}
1854 \@@_scanbackprime_collect:N #1
1855 }
1856 {

```

```

1857         \@@_nbackprimes_select:nn {#1} {\l_@@_primecount_int}
1858     }
1859 }
1860 }
1861 }
1862 }
1863 }
1864 }
1865 }

1866 \AtBeginDocument{\@@_define_prime_commands: \@@_define_prime_chars:}
1867 \cs_new:Nn \@@_define_prime_commands:
1868 {
1869     \cs_set_eq:NN \prime \@@_prime_single_mchar
1870     \cs_set_eq:NN \dprime \@@_prime_double_mchar
1871     \cs_set_eq:NN \trprime \@@_prime_triple_mchar
1872     \cs_set_eq:NN \qprime \@@_prime_quad_mchar
1873     \cs_set_eq:NN \backprime \@@_backprime_single_mchar
1874     \cs_set_eq:NN \backdprime \@@_backprime_double_mchar
1875     \cs_set_eq:NN \backtrprime \@@_backprime_triple_mchar
1876 }
1877 \group_begin:
1878     \char_set_catcode_active:N \'
1879     \char_set_catcode_active:N `
1880     \char_set_catcode_active:n {"2032}
1881     \char_set_catcode_active:n {"2033}
1882     \char_set_catcode_active:n {"2034}
1883     \char_set_catcode_active:n {"2057}
1884     \char_set_catcode_active:n {"2035}
1885     \char_set_catcode_active:n {"2036}
1886     \char_set_catcode_active:n {"2037}
1887     \cs_gset:Nn \@@_define_prime_chars:
1888     {
1889         \cs_set_eq:NN ' \@@_scan_sup_prime:
1890         \cs_set_eq:NN ^^^^2032 \@@_scan_sup_prime:
1891         \cs_set_eq:NN ^^^^2033 \@@_scan_sup_dprime:
1892         \cs_set_eq:NN ^^^^2034 \@@_scan_sup_trprime:
1893         \cs_set_eq:NN ^^^^2057 \@@_scan_sup_qprime:
1894         \cs_set_eq:NN ` \@@_scan_sup_backprime:
1895         \cs_set_eq:NN ^^^^2035 \@@_scan_sup_backprime:
1896         \cs_set_eq:NN ^^^^2036 \@@_scan_sup_backdprime:
1897         \cs_set_eq:NN ^^^^2037 \@@_scan_sup_backtrprime:
1898     }
1899 \group_end:

```

M.2 Unicode radicals

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

```
1900 (*LU)
```

```

1901 \cs_if_exist:NF \Uroot
1902 { \cs_new_eq:NN \Uroot \luatexUroot }
1903 </LU>

1904 \AtBeginDocument{\@@_redefine_radical:}
1905 \cs_new:Nn \@@_redefine_radical:
1906 (*XE)
1907 {
1908   \ifpackageloaded { amsmath } { }
1909   {

```

`\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`.

```

1910   \cs_set_nopar:Npn \r@@@t ##1 ##2
1911   {
1912     \hbox_set:Nn \l_tmpa_box
1913     {
1914       \c_math_toggle_token
1915       \m@th
1916       ##1
1917       \sqrtsign { ##2 }
1918       \c_math_toggle_token
1919     }
1920     \@@_mathstyle_scale:Nnn ##1 { \kern }
1921     { \fontdimen 63 \l_@@_font }
1922     \box_move_up:nn
1923     {
1924       (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
1925       * \number \fontdimen 65 \l_@@_font / 100
1926     }
1927     { \box_use:N \rootbox }
1928     \@@_mathstyle_scale:Nnn ##1 { \kern }
1929     { \fontdimen 64 \l_@@_font }
1930     \box_use_clear:N \l_tmpa_box
1931   }

1932 }
1933 }
1934 </XE>
1935 (*LU)
1936 {
1937   \ifpackageloaded { amsmath } { }
1938   {

```

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

```

1939   \cs_set:Npn \root ##1 \of ##2
1940   {
1941     \Uroot \l_@@_radical_sqrt_tl { ##1 } { ##2 }

```

```

1942     }
1943   }
1944 }
1945 </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.

```

1946 \cs_new:Nn \@@_fontdimen_to_percent:nn
1947 {
1948   \fp_eval:n { \dim_to_decimal:n { \fontdimen #1 #2 } * 65536 / 100 }
1949 }
1950 \cs_new:Nn \@@_fontdimen_to_scale:nn
1951 {
1952   \fp_eval:n {\@@_fontdimen_to_percent:nn {#1} {#2} * \f@size } pt
1953 }

```

`\@@_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.

```

1954 \cs_new:Nn \@@_mathstyle_scale:Nnn
1955 {
1956   \ifx#1\scriptstyle
1957     #2 \@@_fontdimen_to_percent:nn {10} \l_@@_font #3
1958   \else
1959     \ifx#1\scriptscriptstyle
1960       #2 \@@_fontdimen_to_percent:nn {11} \l_@@_font #3
1961     \else
1962       #2 #3
1963     \fi
1964   \fi
1965 }

```

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 \TeX 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?

1966 \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.

```

1967 \cs_new:Nn \@@_setup_active_superscript:nn
1968 {
1969   \prop_gput:Nn \g_@@_supers_prop   {\meaning #1} {#2}
1970   \char_set_catcode_active:N #1
1971   \@@_char_gmake_mathactive:N #1
1972   \scantokens
1973   {
1974     \cs_gset:Npn #1
1975     {
1976       \tl_set:Nn \l_@@_ss_chain_tl {#2}
1977       \cs_set_eq:NN \@@_sub_or_super:n \sp
1978       \tl_set:Nn \l_@@_tmpa_tl {supers}
1979       \@@_scan_sscript:
1980     }
1981   }
1982 }
```

Bam:

```

1983 \@@_setup_active_superscript:nn {^^^2070} {0}
1984 \@@_setup_active_superscript:nn {^^^00b9} {1}
1985 \@@_setup_active_superscript:nn {^^^00b2} {2}
1986 \@@_setup_active_superscript:nn {^^^00b3} {3}
1987 \@@_setup_active_superscript:nn {^^^2074} {4}
1988 \@@_setup_active_superscript:nn {^^^2075} {5}
1989 \@@_setup_active_superscript:nn {^^^2076} {6}
1990 \@@_setup_active_superscript:nn {^^^2077} {7}
1991 \@@_setup_active_superscript:nn {^^^2078} {8}
1992 \@@_setup_active_superscript:nn {^^^2079} {9}
1993 \@@_setup_active_superscript:nn {^^^207a} {+}
1994 \@@_setup_active_superscript:nn {^^^207b} {-}
1995 \@@_setup_active_superscript:nn {^^^207c} {=}
1996 \@@_setup_active_superscript:nn {^^^207d} {(}
1997 \@@_setup_active_superscript:nn {^^^207e} {)}
1998 \@@_setup_active_superscript:nn {^^^2071} {i}
1999 \@@_setup_active_superscript:nn {^^^207f} {n}
2000 \@@_setup_active_superscript:nn {^^^02b0} {h}
2001 \@@_setup_active_superscript:nn {^^^02b2} {j}
2002 \@@_setup_active_superscript:nn {^^^02b3} {r}
```



```

2003 \@@_setup_active_superscript:nn {^^^^02b7} {w}
2004 \@@_setup_active_superscript:nn {^^^^02b8} {y}

```

Subscripts Ditto above.

```

2005 \cs_new:Nn \@@_setup_active_subscript:nn
2006 {
2007   \prop_gput:Non \g_@@_subs_prop {\meaning #1} {#2}
2008   \char_set_catcode_active:N #1
2009   \@@_char_gmake_mathactive:N #1
2010   \scantokens
2011   {
2012     \cs_gset:Npn #1
2013     {
2014       \tl_set:Nn \l_@@_ss_chain_tl {#2}
2015       \cs_set_eq:NN \@@_sub_or_super:n \sb
2016       \tl_set:Nn \l_@@_tmpa_tl {subs}
2017       \@@_scan_sscript:
2018     }
2019   }
2020 }

```

A few more subscripts than superscripts:

```

2021 \@@_setup_active_subscript:nn {^^^^2080} {0}
2022 \@@_setup_active_subscript:nn {^^^^2081} {1}
2023 \@@_setup_active_subscript:nn {^^^^2082} {2}
2024 \@@_setup_active_subscript:nn {^^^^2083} {3}
2025 \@@_setup_active_subscript:nn {^^^^2084} {4}
2026 \@@_setup_active_subscript:nn {^^^^2085} {5}
2027 \@@_setup_active_subscript:nn {^^^^2086} {6}
2028 \@@_setup_active_subscript:nn {^^^^2087} {7}
2029 \@@_setup_active_subscript:nn {^^^^2088} {8}
2030 \@@_setup_active_subscript:nn {^^^^2089} {9}
2031 \@@_setup_active_subscript:nn {^^^^208a} {+}
2032 \@@_setup_active_subscript:nn {^^^^208b} {-}
2033 \@@_setup_active_subscript:nn {^^^^208c} {=}
2034 \@@_setup_active_subscript:nn {^^^^208d} {(}
2035 \@@_setup_active_subscript:nn {^^^^208e} {)}
2036 \@@_setup_active_subscript:nn {^^^^2090} {a}
2037 \@@_setup_active_subscript:nn {^^^^2091} {e}
2038 \@@_setup_active_subscript:nn {^^^^2095} {h}
2039 \@@_setup_active_subscript:nn {^^^^1d62} {i}
2040 \@@_setup_active_subscript:nn {^^^^2c7c} {j}
2041 \@@_setup_active_subscript:nn {^^^^2096} {k}
2042 \@@_setup_active_subscript:nn {^^^^2097} {l}
2043 \@@_setup_active_subscript:nn {^^^^2098} {m}
2044 \@@_setup_active_subscript:nn {^^^^2099} {n}
2045 \@@_setup_active_subscript:nn {^^^^2092} {o}
2046 \@@_setup_active_subscript:nn {^^^^209a} {p}
2047 \@@_setup_active_subscript:nn {^^^^1d63} {r}

```

```

2048 \@@_setup_active_subscript:nn {^^^209b} {s}
2049 \@@_setup_active_subscript:nn {^^^209c} {t}
2050 \@@_setup_active_subscript:nn {^^^1d64} {u}
2051 \@@_setup_active_subscript:nn {^^^1d65} {v}
2052 \@@_setup_active_subscript:nn {^^^2093} {x}
2053 \@@_setup_active_subscript:nn {^^^1d66} {\beta}
2054 \@@_setup_active_subscript:nn {^^^1d67} {\gamma}
2055 \@@_setup_active_subscript:nn {^^^1d68} {\rho}
2056 \@@_setup_active_subscript:nn {^^^1d69} {\phi}
2057 \@@_setup_active_subscript:nn {^^^1d6a} {\chi}
2058 \group_end:

```

The scanning command, evident in its purpose:

```

2059 \cs_new:Npn \@@_scan_sscript:
2060 {
2061   \@@_scan_sscript:TF
2062   {
2063     \@@_scan_sscript:
2064   }
2065   {
2066     \@@_sub_or_super:n {\l_@@_ss_chain_tl}
2067   }
2068 }

```

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.

```

2069 \cs_new:Npn \@@_scan_sscript:TF #1#2
2070 {
2071   \tl_set:Nx \__peek_true_aux:w { \exp_not:n{ #1 } }
2072   \tl_set_eq:NN \__peek_true:w \__peek_true_remove:w
2073   \tl_set:Nx \__peek_false:w { \exp_not:n { \group_align_safe_end: #2 } }
2074   \group_align_safe_begin:
2075   \peek_after:Nw \@@_peek_execute_branches_ss:
2076 }

```

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

```

2077 \cs_new:Npn \@@_peek_execute_branches_ss:
2078 {
2079   \bool_lazy_any:nTF
2080   {
2081     { \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token }
2082     { \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token }
2083     { \token_if_eq_meaning_p:NN \l_peek_token \c_space_token }
2084   }
2085   { \__peek_false:w }
2086   { \@@_peek_execute_branches_ss_aux: }
2087 }

```

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.)

```

2088 \cs_new:Npn \@@_peek_execute_branches_ss_aux:
2089 {
2090   \prop_if_in:coTF
2091     {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token}
2092     {
2093       \prop_get:coN
2094         {g_@@_l_@@_tmpa_tl _prop} {\meaning\l_peek_token} \l_@@_tmpb_tl
2095       \tl_put_right:NV \l_@@_ss_chain_tl \l_@@_tmpb_tl
2096       \__peek_true:w
2097     }
2098     { \__peek_false:w }
2099 }

```

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.

```

2100 \cs_new:Npn \@@_define_active_frac:Nw #1 #2/#3
2101 {
2102   \char_set_catcode_active:N #1
2103   \@@_char_gmake_mathactive:N #1
2104   \tl_rescan:nn
2105     {
2106       \catcode`\_ =11\relax
2107       \catcode`\: =11\relax
2108     }
2109     {
2110       \cs_gset:Npx #1
2111       {
2112         \bool_if:NTF \l_@@_smallfrac_bool {\exp_not:N\tfrac} {\exp_not:N\frac}
2113         {#2} {#3}
2114       }
2115     }
2116 }

```

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

```

2117 \cs_new:Npn \@@_setup_active_frac:
2118 {
2119   \group_begin:
2120   \@@_define_active_frac:Nw ^^^^2189 0/3

```

```

2121 \@@_define_active_frac:Nw ^^^^2152 1/{10}
2122 \@@_define_active_frac:Nw ^^^^2151 1/9
2123 \@@_define_active_frac:Nw ^^^^215b 1/8
2124 \@@_define_active_frac:Nw ^^^^2150 1/7
2125 \@@_define_active_frac:Nw ^^^^2159 1/6
2126 \@@_define_active_frac:Nw ^^^^2155 1/5
2127 \@@_define_active_frac:Nw ^^^^00bc 1/4
2128 \@@_define_active_frac:Nw ^^^^2153 1/3
2129 \@@_define_active_frac:Nw ^^^^215c 3/8
2130 \@@_define_active_frac:Nw ^^^^2156 2/5
2131 \@@_define_active_frac:Nw ^^^^00bd 1/2
2132 \@@_define_active_frac:Nw ^^^^2157 3/5
2133 \@@_define_active_frac:Nw ^^^^215d 5/8
2134 \@@_define_active_frac:Nw ^^^^2154 2/3
2135 \@@_define_active_frac:Nw ^^^^00be 3/4
2136 \@@_define_active_frac:Nw ^^^^2158 4/5
2137 \@@_define_active_frac:Nw ^^^^215a 5/6
2138 \@@_define_active_frac:Nw ^^^^215e 7/8
2139 \group_end:
2140 }
2141 \@@_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.

```

2142 \protected\def\to{\rightarrow}
2143 \protected\def\le{\leq}
2144 \protected\def\ge{\geq}
2145 \protected\def\neq{\neq}
2146 \protected\def\triangle{\mathord{\bigtriangleup}}
2147 \protected\def\bigcirc{\mdlgwhtcircle}
2148 \protected\def\circ{\vysmwhtcircle}
2149 \protected\def\bullet{\smbllkcircle}
2150 \protected\def\mathyen{\yen}
2151 \protected\def\mathsterling{\sterling}
2152 \protected\def\diamond{\smwhtdiamond}
2153 \protected\def\emptyset{\varnothing}
2154 \protected\def\hbar{\hslash}
2155 \protected\def\land{\wedge}
2156 \protected\def\lor{\vee}
2157 \protected\def\owns{\ni}
2158 \protected\def\gets{\leftarrow}
2159 \protected\def\mathring{\ocirc}
2160 \protected\def\not{\neg}
2161 \protected\def\longdivision{\longdivisionsign}

```

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

```

2162 \protected\def\backepsilon{\upbackepsilon}
2163 \protected\def\eth{\matheth}

```

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

```

2164 \protected\def\dbkarow {\dbkarrow}
2165 \protected\def\drbkarow{\drbkarrow}
2166 \protected\def\hksearrow{\hksearrow}
2167 \protected\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:

```

2168 \protected\def\smallint{\mathop{\textstyle\int}\limits}

```

`\underbar`

```

2169 \cs_set_eq:NN \latexe_underbar:n \underbar
2170 \renewcommand\underbar
2171 {
2172   \mode_if_math:TF \mathunderbar \latexe_underbar:n
2173 }

```

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

```

2174 \ifpackageloaded{amsmath}
2175 {
2176   % define their own colon, perhaps I should just steal it. (It does look much better.)
2177 }
2178 {
2179   \cs_set_protected:Npn \colon
2180   {
2181     \bool_if:NTF \g_@@_literal_colon_bool {:} { \mathpunct{:} }
2182   }
2183 }

```

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

```

\Digamma 2184 \protected\def\digamma{\updigamma}
2185 \protected\def\Digamma{\upDigamma}

```

Symbols

```

2186 \cs_set_protected:Npn \l {\Vert}

```

`\mathinner` items:

```

2187 \cs_set_protected:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
2188 \cs_set_protected:Npn \cdots {\mathinner{\unicodecdots}}

2189 \cs_set_eq:NN \@@_text_slash: \slash
2190 \cs_set_protected:Npn \slash
2191 {
2192   \mode_if_math:TF {\mathslash} {\@@_text_slash:}
2193 }

```

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

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

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

```

2194 \cs_new:Npn \@@_newnot:N #1
2195 {
2196   \tl_set:Nx \l_not_token_name_tl { \token_to_str:N #1 }
2197   \exp_args:Nx \tl_if_empty:nF { \tl_tail:V \l_not_token_name_tl }
2198   {
2199     \tl_set:Nx \l_not_token_name_tl { \tl_tail:V \l_not_token_name_tl }
2200   }
2201   \cs_if_exist:cTF { n \l_not_token_name_tl }
2202   {
2203     \use:c { n \l_not_token_name_tl }
2204   }
2205   {
2206     \cs_if_exist:cTF { not \l_not_token_name_tl }
2207     {
2208       \use:c { not \l_not_token_name_tl }
2209     }
2210     {
2211       \@@_oldnot: #1
2212     }
2213   }
2214 }
2215 \cs_set_eq:NN \@@_oldnot: \not
2216 \AtBeginDocument{\cs_set_eq:NN \not \@@_newnot:N}
2217 \cs_new_protected_nopar:Nn \@@_setup_negations:
2218 {
2219   \cs_gset:cpn { not= } { \neq }
2220   \cs_gset:cpn { not< } { \nless }
2221   \cs_gset:cpn { not> } { \ngtr }
2222   \cs_gset:Npn \ngets { \leftarrow }
2223   \cs_gset:Npn \nsimeq { \simeq }
2224   \cs_gset:Npn \nequal { \ne }
2225   \cs_gset:Npn \nle { \nleq }
2226   \cs_gset:Npn \nge { \ngeq }
2227   \cs_gset:Npn \ngreater { \ngtr }
2228   \cs_gset:Npn \nforksnot { \forks }
2229 }

```

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

2230 $\langle /package \& (X E | L U) \rangle$

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.

2231 $\langle *msg \rangle$

Wrapper functions:

```
2232 \cs_new:Npn \@@_error:n { \msg_error:nn {unicode-math} }
2233 \cs_new:Npn \@@_warning:n { \msg_warning:nn {unicode-math} }
2234 \cs_new:Npn \@@_warning:nnn { \msg_warning:nnxx {unicode-math} }
2235 \cs_new:Npn \@@_log:n { \msg_log:nn {unicode-math} }
2236 \cs_new:Npn \@@_log:nx { \msg_log:nnx {unicode-math} }

2237 \msg_new:nnn {unicode-math} {no-tfrac}
2238 {
2239   Small~ fraction~ command~ \protect\tfrac~ not~ defined.\.
2240   Load~ amsmath~ or~ define~ it~ manually~ before~ loading~ unicode-math.
2241 }
2242 \msg_new:nnn {unicode-math} {default-math-font}
2243 {
2244   Defining~ the~ default~ maths~ font~ as~ '\l_@@_fontname_tl'.
2245 }
2246 \msg_new:nnn {unicode-math} {setup-implicit}
2247 {
2248   Setup~ alphabets:~ implicit~ mode.
2249 }
2250 \msg_new:nnn {unicode-math} {setup-explicit}
2251 {
2252   Setup~ alphabets:~ explicit~ mode.
2253 }
2254 \msg_new:nnn {unicode-math} {alph-initialise}
2255 {
2256   Initialising~ \@backslashchar math#1.
2257 }
2258 \msg_new:nnn {unicode-math} {setup-alph}
2259 {
2260   Setup~ alphabet:~ #1.
2261 }
2262 \msg_new:nnn {unicode-math} {no-alphabet}
2263 {
2264   I~ am~ trying~ to~ set~ up~ alphabet~"#1"~ but~ there~ are~ no~ configuration~ settings~ for~ it.
2265   (See~ source~ file~ "unicode-math-alphabets.dtx"~ to~ debug.)
2266 }
2267 \msg_new:nnn { unicode-math } { no-named-range }
2268 {
2269   I~ am~ trying~ to~ define~ new~ alphabet~ "#2"~ in~ range~ "#1",~ but~ range~ "#1"~ hasn't~ been~
2270 }
```

```

2271 \msg_new:nnn { unicode-math } { missing-alphabets }
2272 {
2273   Missing~math~alphabets~in~font~ "\fontname\l_@@_font" \ \ \
2274   \seq_map_function:NN \l_@@_missing_alph_seq \@@_print_indent:n
2275 }
2276 \cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \ \ }
2277 \msg_new:nnn { unicode-math } { macro-expected }
2278 {
2279   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
2280 }
2281 \msg_new:nnn { unicode-math } { wrong-meaning }
2282 {
2283   I've~ expected~ #1~ to~ have~ the~ meaning~ #3,~ but~ it~ has~ the~ meaning~ #2.
2284 }
2285 \msg_new:nnn { unicode-math } { patch-macro }
2286 {
2287   I'm~ going~ to~ patch~ macro~ #1.
2288 }
2289 \msg_new:nnn { unicode-math } { mathtools-overbracket } {
2290   Using~ \token_to_str:N \overbracket~ and~
2291   \token_to_str:N \underbracket~ from~
2292   'mathtools'~ package.\ \
2293   \ \
2294   Use~ \token_to_str:N \Uoverbracket~ and~
2295   \token_to_str:N \Uunderbracket~ for~
2296   original~ 'unicode-math'~ definition.
2297 }
2298 \msg_new:nnn { unicode-math } { mathtools-colon } {
2299   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
2300   the~ 'mathtools'~ package: \ \ \
2301   \ \ \ \ \token_to_str:N \dblcolon,~
2302   \token_to_str:N \coloneqq,~
2303   \token_to_str:N \Coloneqq,~
2304   \token_to_str:N \eqqcolon. \ \ \
2305   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
2306   commands,~ using~ them~ will~ lead~ to~ inconsistencies.
2307 }
2308 \msg_new:nnn { unicode-math } { colonequals } {
2309   I'm~ going~ to~ overwrite~ the~ following~ commands~ from~
2310   the~ 'colonequals'~ package: \ \ \
2311   \ \ \ \ \token_to_str:N \ratio,~
2312   \token_to_str:N \coloncolon,~
2313   \token_to_str:N \minuscolon, \ \
2314   \ \ \ \ \token_to_str:N \colonequals,~
2315   \token_to_str:N \equalscolon,~
2316   \token_to_str:N \coloncolonequals. \ \ \
2317   Note~ that~ since~ I~ won't~ overwrite~ the~ other~ colon-like~
2318   commands,~ using~ them~ will~ lead~ to~ inconsistencies.~
2319   Furthermore,~ changing~ \token_to_str:N \colonsep \c_space_tl

```



```

2320 or~ \token_to_str:N \doublecolonsep \c_space_tl won't~ have~
2321 any~ effect~ on~ the~ re-defined~ commands.
2322 }
2323 </msg>

```

N.1 Alphabet Unicode positions

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.⁷

```

2324 <*usv>

```

Alphabets

```

2325 \usv_set:nnn {normal} {num} {48}
2326 \usv_set:nnn {normal} {Latin}{1D434}
2327 \usv_set:nnn {normal} {latin}{1D44E}
2328 \usv_set:nnn {normal} {Greek}{1D6E2}
2329 \usv_set:nnn {normal} {greek}{1D6FC}
2330 \usv_set:nnn {normal}{varTheta} {1D6F3}
2331 \usv_set:nnn {normal}{epsilon}{1D716}
2332 \usv_set:nnn {normal}{vartheta} {1D717}
2333 \usv_set:nnn {normal}{varkappa} {1D718}
2334 \usv_set:nnn {normal}{phi} {1D719}
2335 \usv_set:nnn {normal}{varrho} {1D71A}
2336 \usv_set:nnn {normal}{varpi} {1D71B}
2337 \usv_set:nnn {normal} {Nabla}{1D6FB}
2338 \usv_set:nnn {normal} {partial}{1D715}
2339
2340 \usv_set:nnn {up} {num} {48}
2341 \usv_set:nnn {up} {Latin}{65}
2342 \usv_set:nnn {up} {latin}{97}
2343 \usv_set:nnn {up} {Greek}{391}
2344 \usv_set:nnn {up} {greek}{3B1}
2345 \usv_set:nnn {it} {Latin}{1D434}
2346 \usv_set:nnn {it} {latin}{1D44E}
2347 \usv_set:nnn {it} {Greek}{1D6E2}
2348 \usv_set:nnn {it} {greek}{1D6FC}
2349 \usv_set:nnn {bb} {num} {1D7D8}
2350 \usv_set:nnn {bb} {Latin}{1D538}
2351 \usv_set:nnn {bb} {latin}{1D552}
2352 \usv_set:nnn {scr} {Latin}{1D49C}
2353 \usv_set:nnn {cal} {Latin}{1D49C}
2354 \usv_set:nnn {scr} {latin}{1D4B6}
2355 \usv_set:nnn {frak}{Latin}{1D504}
2356 \usv_set:nnn {frak}{latin}{1D51E}
2357 \usv_set:nnn {sf} {num} {1D7E2}
2358 \usv_set:nnn {sfup}{num} {1D7E2}

```

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

2359 \usv_set:nnn {sfit}{num} {"1D7E2}
 2360 \usv_set:nnn {sfup}{Latin}{"1D5A0}
 2361 \usv_set:nnn {sf} {Latin}{"1D5A0}
 2362 \usv_set:nnn {sfup}{latin}{"1D5BA}
 2363 \usv_set:nnn {sf} {latin}{"1D5BA}
 2364 \usv_set:nnn {sfit}{Latin}{"1D608}
 2365 \usv_set:nnn {sfit}{latin}{"1D622}
 2366 \usv_set:nnn {tt} {num} {"1D7F6}
 2367 \usv_set:nnn {tt} {Latin}{"1D670}
 2368 \usv_set:nnn {tt} {latin}{"1D68A}

Bold:

2369 \usv_set:nnn {bf} {num} {"1D7CE}
 2370 \usv_set:nnn {bfup} {num} {"1D7CE}
 2371 \usv_set:nnn {bfit} {num} {"1D7CE}
 2372 \usv_set:nnn {bfup} {Latin}{"1D400}
 2373 \usv_set:nnn {bfup} {latin}{"1D41A}
 2374 \usv_set:nnn {bfup} {Greek}{"1D6A8}
 2375 \usv_set:nnn {bfup} {greek}{"1D6C2}
 2376 \usv_set:nnn {bfit} {Latin}{"1D468}
 2377 \usv_set:nnn {bfit} {latin}{"1D482}
 2378 \usv_set:nnn {bfit} {Greek}{"1D71C}
 2379 \usv_set:nnn {bfit} {greek}{"1D736}
 2380 \usv_set:nnn {bffrak}{Latin}{"1D56C}
 2381 \usv_set:nnn {bffrak}{latin}{"1D586}
 2382 \usv_set:nnn {bfscr} {Latin}{"1D4D0}
 2383 \usv_set:nnn {bfcal} {Latin}{"1D4D0}
 2384 \usv_set:nnn {bfscr} {latin}{"1D4EA}
 2385 \usv_set:nnn {bfsf} {num} {"1D7EC}
 2386 \usv_set:nnn {bfsfup}{num} {"1D7EC}
 2387 \usv_set:nnn {bfsfit}{num} {"1D7EC}
 2388 \usv_set:nnn {bfsfup}{Latin}{"1D5D4}
 2389 \usv_set:nnn {bfsfup}{latin}{"1D5EE}
 2390 \usv_set:nnn {bfsfup}{Greek}{"1D756}
 2391 \usv_set:nnn {bfsfup}{greek}{"1D770}
 2392 \usv_set:nnn {bfsfit}{Latin}{"1D63C}
 2393 \usv_set:nnn {bfsfit}{latin}{"1D656}
 2394 \usv_set:nnn {bfsfit}{Greek}{"1D790}
 2395 \usv_set:nnn {bfsfit}{greek}{"1D7AA}

2396 \usv_set:nnn {bfsf}{Latin}{ \bool_if:NTF \g_@@_upLatin_bool \g_@@_bfsfup_Latin_usv \g_@@_bfsfit_L
 2397 \usv_set:nnn {bfsf}{latin}{ \bool_if:NTF \g_@@_uplatin_bool \g_@@_bfsfup_latin_usv \g_@@_bfsfit_l
 2398 \usv_set:nnn {bfsf}{Greek}{ \bool_if:NTF \g_@@_upGreek_bool \g_@@_bfsfup_Greek_usv \g_@@_bfsfit_G
 2399 \usv_set:nnn {bfsf}{greek}{ \bool_if:NTF \g_@@_upgreek_bool \g_@@_bfsfup_greek_usv \g_@@_bfsfit_g
 2400 \usv_set:nnn {bf} {Latin}{ \bool_if:NTF \g_@@_bfupLatin_bool \g_@@_bfup_Latin_usv \g_@@_bfit_Lat
 2401 \usv_set:nnn {bf} {latin}{ \bool_if:NTF \g_@@_bfuplatin_bool \g_@@_bfup_latin_usv \g_@@_bfit_lat
 2402 \usv_set:nnn {bf} {Greek}{ \bool_if:NTF \g_@@_bfupGreek_bool \g_@@_bfup_Greek_usv \g_@@_bfit_Gre
 2403 \usv_set:nnn {bf} {greek}{ \bool_if:NTF \g_@@_bfupgreek_bool \g_@@_bfup_greek_usv \g_@@_bfit_gre

Greek variants:

2404 \usv_set:nnn {up}{varTheta} {"3F4}

2405 \usv_set:nnn {up}{Digamma} {"3DC}
 2406 \usv_set:nnn {up}{epsilon}{"3F5}
 2407 \usv_set:nnn {up}{vartheta} {"3D1}
 2408 \usv_set:nnn {up}{varkappa} {"3F0}
 2409 \usv_set:nnn {up}{phi} {"3D5}
 2410 \usv_set:nnn {up}{varrho} {"3F1}
 2411 \usv_set:nnn {up}{varpi} {"3D6}
 2412 \usv_set:nnn {up}{digamma} {"3DD}

Bold:

2413 \usv_set:nnn {bfup}{varTheta} {"1D6B9}
 2414 \usv_set:nnn {bfup}{Digamma} {"1D7CA}
 2415 \usv_set:nnn {bfup}{epsilon}{"1D6DC}
 2416 \usv_set:nnn {bfup}{vartheta} {"1D6DD}
 2417 \usv_set:nnn {bfup}{varkappa} {"1D6DE}
 2418 \usv_set:nnn {bfup}{phi} {"1D6DF}
 2419 \usv_set:nnn {bfup}{varrho} {"1D6E0}
 2420 \usv_set:nnn {bfup}{varpi} {"1D6E1}
 2421 \usv_set:nnn {bfup}{digamma} {"1D7CB}

Italic Greek variants:

2422 \usv_set:nnn {it}{varTheta} {"1D6F3}
 2423 \usv_set:nnn {it}{epsilon}{"1D716}
 2424 \usv_set:nnn {it}{vartheta} {"1D717}
 2425 \usv_set:nnn {it}{varkappa} {"1D718}
 2426 \usv_set:nnn {it}{phi} {"1D719}
 2427 \usv_set:nnn {it}{varrho} {"1D71A}
 2428 \usv_set:nnn {it}{varpi} {"1D71B}

Bold italic:

2429 \usv_set:nnn {bfit}{varTheta} {"1D72D}
 2430 \usv_set:nnn {bfit}{epsilon}{"1D750}
 2431 \usv_set:nnn {bfit}{vartheta} {"1D751}
 2432 \usv_set:nnn {bfit}{varkappa} {"1D752}
 2433 \usv_set:nnn {bfit}{phi} {"1D753}
 2434 \usv_set:nnn {bfit}{varrho} {"1D754}
 2435 \usv_set:nnn {bfit}{varpi} {"1D755}

Bold sans:

2436 \usv_set:nnn {bfsfup}{varTheta} {"1D767}
 2437 \usv_set:nnn {bfsfup}{epsilon}{"1D78A}
 2438 \usv_set:nnn {bfsfup}{vartheta} {"1D78B}
 2439 \usv_set:nnn {bfsfup}{varkappa} {"1D78C}
 2440 \usv_set:nnn {bfsfup}{phi} {"1D78D}
 2441 \usv_set:nnn {bfsfup}{varrho} {"1D78E}
 2442 \usv_set:nnn {bfsfup}{varpi} {"1D78F}

Bold sans italic:

2443 \usv_set:nnn {bfsfit}{varTheta} {"1D7A1}
 2444 \usv_set:nnn {bfsfit}{epsilon}{"1D7C4}
 2445 \usv_set:nnn {bfsfit}{vartheta} {"1D7C5}

```

2446 \usv_set:nnn {bfsfit}{varkappa} {"1D7C6}
2447 \usv_set:nnn {bfsfit}{phi} {"1D7C7}
2448 \usv_set:nnn {bfsfit}{varrho} {"1D7C8}
2449 \usv_set:nnn {bfsfit}{varpi} {"1D7C9}

```

Nabla:

```

2450 \usv_set:nnn {up} {Nabla}{"02207}
2451 \usv_set:nnn {it} {Nabla}{"1D6FB}
2452 \usv_set:nnn {bfup} {Nabla}{"1D6C1}
2453 \usv_set:nnn {bfit} {Nabla}{"1D735}
2454 \usv_set:nnn {bfsfup}{Nabla}{"1D76F}
2455 \usv_set:nnn {bfsfit}{Nabla}{"1D7A9}

```

Partial:

```

2456 \usv_set:nnn {up} {partial}{"02202}
2457 \usv_set:nnn {it} {partial}{"1D715}
2458 \usv_set:nnn {bfup} {partial}{"1D6DB}
2459 \usv_set:nnn {bfit} {partial}{"1D74F}
2460 \usv_set:nnn {bfsfup}{partial}{"1D789}
2461 \usv_set:nnn {bfsfit}{partial}{"1D7C3}

```

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

```

2462 \usv_set:nnn {up}{B}{`\B}
2463 \usv_set:nnn {up}{C}{`\C}
2464 \usv_set:nnn {up}{D}{`\D}
2465 \usv_set:nnn {up}{E}{`\E}
2466 \usv_set:nnn {up}{F}{`\F}
2467 \usv_set:nnn {up}{H}{`\H}
2468 \usv_set:nnn {up}{I}{`\I}
2469 \usv_set:nnn {up}{L}{`\L}
2470 \usv_set:nnn {up}{M}{`\M}
2471 \usv_set:nnn {up}{N}{`\N}
2472 \usv_set:nnn {up}{P}{`\P}
2473 \usv_set:nnn {up}{Q}{`\Q}
2474 \usv_set:nnn {up}{R}{`\R}
2475 \usv_set:nnn {up}{Z}{`\Z}

2476 \usv_set:nnn {it}{B}{"1D435}
2477 \usv_set:nnn {it}{C}{"1D436}
2478 \usv_set:nnn {it}{D}{"1D437}
2479 \usv_set:nnn {it}{E}{"1D438}
2480 \usv_set:nnn {it}{F}{"1D439}
2481 \usv_set:nnn {it}{H}{"1D43B}
2482 \usv_set:nnn {it}{I}{"1D43C}
2483 \usv_set:nnn {it}{L}{"1D43F}
2484 \usv_set:nnn {it}{M}{"1D440}
2485 \usv_set:nnn {it}{N}{"1D441}
2486 \usv_set:nnn {it}{P}{"1D443}
2487 \usv_set:nnn {it}{Q}{"1D444}

```

2488 \usv_set:nnn {it}{R}{\string"1D445}
 2489 \usv_set:nnn {it}{Z}{\string"1D44D}
 2490 \usv_set:nnn {up}{d}{\string"1D44E}
 2491 \usv_set:nnn {up}{e}{\string"1D44F}
 2492 \usv_set:nnn {up}{g}{\string"1D450}
 2493 \usv_set:nnn {up}{h}{\string"1D451}
 2494 \usv_set:nnn {up}{i}{\string"1D452}
 2495 \usv_set:nnn {up}{j}{\string"1D453}
 2496 \usv_set:nnn {up}{o}{\string"1D454}
 2497 \usv_set:nnn {it}{d}{\string"1D451}
 2498 \usv_set:nnn {it}{e}{\string"1D452}
 2499 \usv_set:nnn {it}{g}{\string"1D454}
 2500 \usv_set:nnn {it}{h}{\string"0210E}
 2501 \usv_set:nnn {it}{i}{\string"1D456}
 2502 \usv_set:nnn {it}{j}{\string"1D457}
 2503 \usv_set:nnn {it}{o}{\string"1D45C}

Latin ‘h’:

2504 \usv_set:nnn {bb} {h}{\string"1D559}
 2505 \usv_set:nnn {tt} {h}{\string"1D691}
 2506 \usv_set:nnn {scr} {h}{\string"1D4BD}
 2507 \usv_set:nnn {frak} {h}{\string"1D525}
 2508 \usv_set:nnn {bfup} {h}{\string"1D421}
 2509 \usv_set:nnn {bfit} {h}{\string"1D489}
 2510 \usv_set:nnn {sfup} {h}{\string"1D5C1}
 2511 \usv_set:nnn {sfrit} {h}{\string"1D629}
 2512 \usv_set:nnn {bffrak}{h}{\string"1D58D}
 2513 \usv_set:nnn {bfscr} {h}{\string"1D4F1}
 2514 \usv_set:nnn {bfsfup}{h}{\string"1D5F5}
 2515 \usv_set:nnn {bfsfit}{h}{\string"1D65D}

Dotless ‘i’ and ‘j’:

2516 \usv_set:nnn {up}{dotlessi}{\string"00131}
 2517 \usv_set:nnn {up}{dotlessj}{\string"00237}
 2518 \usv_set:nnn {it}{dotlessi}{\string"1D6A4}
 2519 \usv_set:nnn {it}{dotlessj}{\string"1D6A5}

Blackboard:

2520 \usv_set:nnn {bb}{C}{\string"2102}
 2521 \usv_set:nnn {bb}{H}{\string"210D}
 2522 \usv_set:nnn {bb}{N}{\string"2115}
 2523 \usv_set:nnn {bb}{P}{\string"2119}
 2524 \usv_set:nnn {bb}{Q}{\string"211A}
 2525 \usv_set:nnn {bb}{R}{\string"211D}
 2526 \usv_set:nnn {bb}{Z}{\string"2124}
 2527 \usv_set:nnn {up}{Pi} {\string"003A0}
 2528 \usv_set:nnn {up}{pi} {\string"003C0}
 2529 \usv_set:nnn {up}{Gamma} {\string"00393}
 2530 \usv_set:nnn {up}{gamma} {\string"003B3}
 2531 \usv_set:nnn {up}{summation}{\string"02211}

```

2532 \usv_set:nnn {it}{Pi}      {"1D6F1}
2533 \usv_set:nnn {it}{pi}      {"1D70B}
2534 \usv_set:nnn {it}{Gamma}   {"1D6E4}
2535 \usv_set:nnn {it}{gamma}    {"1D6FE}
2536 \usv_set:nnn {bb}{Pi}      {"0213F}
2537 \usv_set:nnn {bb}{pi}      {"0213C}
2538 \usv_set:nnn {bb}{Gamma}   {"0213E}
2539 \usv_set:nnn {bb}{gamma}    {"0213D}
2540 \usv_set:nnn {bb}{summation}{"02140}

```

Italic blackboard:

```

2541 \usv_set:nnn {bbit}{D}{"2145}
2542 \usv_set:nnn {bbit}{d}{"2146}
2543 \usv_set:nnn {bbit}{e}{"2147}
2544 \usv_set:nnn {bbit}{i}{"2148}
2545 \usv_set:nnn {bbit}{j}{"2149}

```

Script exceptions:

```

2546 \usv_set:nnn {scr}{B}{"212C}
2547 \usv_set:nnn {scr}{E}{"2130}
2548 \usv_set:nnn {scr}{F}{"2131}
2549 \usv_set:nnn {scr}{H}{"210B}
2550 \usv_set:nnn {scr}{I}{"2110}
2551 \usv_set:nnn {scr}{L}{"2112}
2552 \usv_set:nnn {scr}{M}{"2133}
2553 \usv_set:nnn {scr}{R}{"211B}
2554 \usv_set:nnn {scr}{e}{"212F}
2555 \usv_set:nnn {scr}{g}{"210A}
2556 \usv_set:nnn {scr}{o}{"2134}

2557 \usv_set:nnn {cal}{B}{"212C}
2558 \usv_set:nnn {cal}{E}{"2130}
2559 \usv_set:nnn {cal}{F}{"2131}
2560 \usv_set:nnn {cal}{H}{"210B}
2561 \usv_set:nnn {cal}{I}{"2110}
2562 \usv_set:nnn {cal}{L}{"2112}
2563 \usv_set:nnn {cal}{M}{"2133}
2564 \usv_set:nnn {cal}{R}{"211B}

```

Fraktur exceptions:

```

2565 \usv_set:nnn {frak}{C}{"212D}
2566 \usv_set:nnn {frak}{H}{"210C}
2567 \usv_set:nnn {frak}{I}{"2111}
2568 \usv_set:nnn {frak}{R}{"211C}
2569 \usv_set:nnn {frak}{Z}{"2128}

```

```

2570 (*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.

2571 *(***stix**)*

Upright

```
2572 \usv_set:nnn {stixsfup}{partial}{ "E17C}
2573 \usv_set:nnn {stixsfup}{Greek}{ "E17D}
2574 \usv_set:nnn {stixsfup}{greek}{ "E196}
2575 \usv_set:nnn {stixsfup}{varTheta}{ "E18E}
2576 \usv_set:nnn {stixsfup}{epsilon}{ "E1AF}
2577 \usv_set:nnn {stixsfup}{vartheta}{ "E1B0}
2578 \usv_set:nnn {stixsfup}{varkappa}{0000} % ???
2579 \usv_set:nnn {stixsfup}{phi}{ "E1B1}
2580 \usv_set:nnn {stixsfup}{varrho}{ "E1B2}
2581 \usv_set:nnn {stixsfup}{varpi}{ "E1B3}
2582 \usv_set:nnn {stixupslash}{Greek}{ "E2FC}
```

Italic

```
2583 \usv_set:nnn {stixbbit}{A}{ "E154}
2584 \usv_set:nnn {stixbbit}{B}{ "E155}
2585 \usv_set:nnn {stixbbit}{E}{ "E156}
2586 \usv_set:nnn {stixbbit}{F}{ "E157}
2587 \usv_set:nnn {stixbbit}{G}{ "E158}
2588 \usv_set:nnn {stixbbit}{I}{ "E159}
2589 \usv_set:nnn {stixbbit}{J}{ "E15A}
2590 \usv_set:nnn {stixbbit}{K}{ "E15B}
2591 \usv_set:nnn {stixbbit}{L}{ "E15C}
2592 \usv_set:nnn {stixbbit}{M}{ "E15D}
2593 \usv_set:nnn {stixbbit}{O}{ "E15E}
2594 \usv_set:nnn {stixbbit}{S}{ "E15F}
2595 \usv_set:nnn {stixbbit}{T}{ "E160}
2596 \usv_set:nnn {stixbbit}{U}{ "E161}
2597 \usv_set:nnn {stixbbit}{V}{ "E162}
2598 \usv_set:nnn {stixbbit}{W}{ "E163}
2599 \usv_set:nnn {stixbbit}{X}{ "E164}
2600 \usv_set:nnn {stixbbit}{Y}{ "E165}

2601 \usv_set:nnn {stixbbit}{a}{ "E166}
2602 \usv_set:nnn {stixbbit}{b}{ "E167}
2603 \usv_set:nnn {stixbbit}{c}{ "E168}
2604 \usv_set:nnn {stixbbit}{f}{ "E169}
2605 \usv_set:nnn {stixbbit}{g}{ "E16A}
2606 \usv_set:nnn {stixbbit}{h}{ "E16B}
2607 \usv_set:nnn {stixbbit}{k}{ "E16C}
2608 \usv_set:nnn {stixbbit}{l}{ "E16D}
2609 \usv_set:nnn {stixbbit}{m}{ "E16E}
2610 \usv_set:nnn {stixbbit}{n}{ "E16F}
2611 \usv_set:nnn {stixbbit}{o}{ "E170}
```

2612 \usv_set:nnn {stixbbit}{p}{ "E171}
 2613 \usv_set:nnn {stixbbit}{q}{ "E172}
 2614 \usv_set:nnn {stixbbit}{r}{ "E173}
 2615 \usv_set:nnn {stixbbit}{s}{ "E174}
 2616 \usv_set:nnn {stixbbit}{t}{ "E175}
 2617 \usv_set:nnn {stixbbit}{u}{ "E176}
 2618 \usv_set:nnn {stixbbit}{v}{ "E177}
 2619 \usv_set:nnn {stixbbit}{w}{ "E178}
 2620 \usv_set:nnn {stixbbit}{x}{ "E179}
 2621 \usv_set:nnn {stixbbit}{y}{ "E17A}
 2622 \usv_set:nnn {stixbbit}{z}{ "E17B}

 2623 \usv_set:nnn {stixsfit}{Numerals}{ "E1B4}
 2624 \usv_set:nnn {stixsfit}{partial}{ "E1BE}
 2625 \usv_set:nnn {stixsfit}{Greek}{ "E1BF}
 2626 \usv_set:nnn {stixsfit}{greek}{ "E1D8}
 2627 \usv_set:nnn {stixsfit}{varTheta}{ "E1D0}
 2628 \usv_set:nnn {stixsfit}{epsilon}{ "E1F1}
 2629 \usv_set:nnn {stixsfit}{vartheta}{ "E1F2}
 2630 \usv_set:nnn {stixsfit}{varkappa}{0000} % ???
 2631 \usv_set:nnn {stixsfit}{phi}{ "E1F3}
 2632 \usv_set:nnn {stixsfit}{varrho}{ "E1F4}
 2633 \usv_set:nnn {stixsfit}{varpi}{ "E1F5}

 2634 \usv_set:nnn {stixcal}{Latin}{ "E22D}
 2635 \usv_set:nnn {stixcal}{num}{ "E262}
 2636 \usv_set:nnn {scr}{num}{48}
 2637 \usv_set:nnn {it}{num}{48}

 2638 \usv_set:nnn {stixsfitslash}{Latin}{ "E294}
 2639 \usv_set:nnn {stixsfitslash}{latin}{ "E2C8}
 2640 \usv_set:nnn {stixsfitslash}{greek}{ "E32C}
 2641 \usv_set:nnn {stixsfitslash}{epsilon}{ "E37A}
 2642 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35E}
 2643 \usv_set:nnn {stixsfitslash}{varkappa}{ "E374}
 2644 \usv_set:nnn {stixsfitslash}{phi}{ "E360}
 2645 \usv_set:nnn {stixsfitslash}{varrho}{ "E376}
 2646 \usv_set:nnn {stixsfitslash}{varpi}{ "E362}
 2647 \usv_set:nnn {stixsfitslash}{digamma}{ "E36A}

Bold

2648 \usv_set:nnn {stixbfupslash}{Greek}{ "E2FD}
 2649 \usv_set:nnn {stixbfupslash}{Digamma}{ "E369}

 2650 \usv_set:nnn {stixbfbb}{A}{ "E38A}
 2651 \usv_set:nnn {stixbfbb}{B}{ "E38B}
 2652 \usv_set:nnn {stixbfbb}{E}{ "E38D}
 2653 \usv_set:nnn {stixbfbb}{F}{ "E38E}
 2654 \usv_set:nnn {stixbfbb}{G}{ "E38F}
 2655 \usv_set:nnn {stixbfbb}{I}{ "E390}
 2656 \usv_set:nnn {stixbfbb}{J}{ "E391}

2657 \usv_set:nnn {stixbfbb}{K}{ "E392}
 2658 \usv_set:nnn {stixbfbb}{L}{ "E393}
 2659 \usv_set:nnn {stixbfbb}{M}{ "E394}
 2660 \usv_set:nnn {stixbfbb}{O}{ "E395}
 2661 \usv_set:nnn {stixbfbb}{S}{ "E396}
 2662 \usv_set:nnn {stixbfbb}{T}{ "E397}
 2663 \usv_set:nnn {stixbfbb}{U}{ "E398}
 2664 \usv_set:nnn {stixbfbb}{V}{ "E399}
 2665 \usv_set:nnn {stixbfbb}{W}{ "E39A}
 2666 \usv_set:nnn {stixbfbb}{X}{ "E39B}
 2667 \usv_set:nnn {stixbfbb}{Y}{ "E39C}

 2668 \usv_set:nnn {stixbfbb}{a}{ "E39D}
 2669 \usv_set:nnn {stixbfbb}{b}{ "E39E}
 2670 \usv_set:nnn {stixbfbb}{c}{ "E39F}
 2671 \usv_set:nnn {stixbfbb}{f}{ "E3A2}
 2672 \usv_set:nnn {stixbfbb}{g}{ "E3A3}
 2673 \usv_set:nnn {stixbfbb}{h}{ "E3A4}
 2674 \usv_set:nnn {stixbfbb}{k}{ "E3A7}
 2675 \usv_set:nnn {stixbfbb}{l}{ "E3A8}
 2676 \usv_set:nnn {stixbfbb}{m}{ "E3A9}
 2677 \usv_set:nnn {stixbfbb}{n}{ "E3AA}
 2678 \usv_set:nnn {stixbfbb}{o}{ "E3AB}
 2679 \usv_set:nnn {stixbfbb}{p}{ "E3AC}
 2680 \usv_set:nnn {stixbfbb}{q}{ "E3AD}
 2681 \usv_set:nnn {stixbfbb}{r}{ "E3AE}
 2682 \usv_set:nnn {stixbfbb}{s}{ "E3AF}
 2683 \usv_set:nnn {stixbfbb}{t}{ "E3B0}
 2684 \usv_set:nnn {stixbfbb}{u}{ "E3B1}
 2685 \usv_set:nnn {stixbfbb}{v}{ "E3B2}
 2686 \usv_set:nnn {stixbfbb}{w}{ "E3B3}
 2687 \usv_set:nnn {stixbfbb}{x}{ "E3B4}
 2688 \usv_set:nnn {stixbfbb}{y}{ "E3B5}
 2689 \usv_set:nnn {stixbfbb}{z}{ "E3B6}

 2690 \usv_set:nnn {stixbfsfup}{Numerals}{ "E3B7}

Bold Italic

2691 \usv_set:nnn {stixbfsfit}{Numerals}{ "E1F6}

 2692 \usv_set:nnn {stixbfbbbit}{A}{ "E200}
 2693 \usv_set:nnn {stixbfbbbit}{B}{ "E201}
 2694 \usv_set:nnn {stixbfbbbit}{E}{ "E203}
 2695 \usv_set:nnn {stixbfbbbit}{F}{ "E204}
 2696 \usv_set:nnn {stixbfbbbit}{G}{ "E205}
 2697 \usv_set:nnn {stixbfbbbit}{I}{ "E206}
 2698 \usv_set:nnn {stixbfbbbit}{J}{ "E207}
 2699 \usv_set:nnn {stixbfbbbit}{K}{ "E208}
 2700 \usv_set:nnn {stixbfbbbit}{L}{ "E209}
 2701 \usv_set:nnn {stixbfbbbit}{M}{ "E20A}
 2702 \usv_set:nnn {stixbfbbbit}{O}{ "E20B}

```

2703 \usv_set:nnn {stixbfbbbit}{S}{ "E20C}
2704 \usv_set:nnn {stixbfbbbit}{T}{ "E20D}
2705 \usv_set:nnn {stixbfbbbit}{U}{ "E20E}
2706 \usv_set:nnn {stixbfbbbit}{V}{ "E20F}
2707 \usv_set:nnn {stixbfbbbit}{W}{ "E210}
2708 \usv_set:nnn {stixbfbbbit}{X}{ "E211}
2709 \usv_set:nnn {stixbfbbbit}{Y}{ "E212}

2710 \usv_set:nnn {stixbfbbbit}{a}{ "E213}
2711 \usv_set:nnn {stixbfbbbit}{b}{ "E214}
2712 \usv_set:nnn {stixbfbbbit}{c}{ "E215}
2713 \usv_set:nnn {stixbfbbbit}{e}{ "E217}
2714 \usv_set:nnn {stixbfbbbit}{f}{ "E218}
2715 \usv_set:nnn {stixbfbbbit}{g}{ "E219}
2716 \usv_set:nnn {stixbfbbbit}{h}{ "E21A}
2717 \usv_set:nnn {stixbfbbbit}{k}{ "E21D}
2718 \usv_set:nnn {stixbfbbbit}{l}{ "E21E}
2719 \usv_set:nnn {stixbfbbbit}{m}{ "E21F}
2720 \usv_set:nnn {stixbfbbbit}{n}{ "E220}
2721 \usv_set:nnn {stixbfbbbit}{o}{ "E221}
2722 \usv_set:nnn {stixbfbbbit}{p}{ "E222}
2723 \usv_set:nnn {stixbfbbbit}{q}{ "E223}
2724 \usv_set:nnn {stixbfbbbit}{r}{ "E224}
2725 \usv_set:nnn {stixbfbbbit}{s}{ "E225}
2726 \usv_set:nnn {stixbfbbbit}{t}{ "E226}
2727 \usv_set:nnn {stixbfbbbit}{u}{ "E227}
2728 \usv_set:nnn {stixbfbbbit}{v}{ "E228}
2729 \usv_set:nnn {stixbfbbbit}{w}{ "E229}
2730 \usv_set:nnn {stixbfbbbit}{x}{ "E22A}
2731 \usv_set:nnn {stixbfbbbit}{y}{ "E22B}
2732 \usv_set:nnn {stixbfbbbit}{z}{ "E22C}

2733 \usv_set:nnn {stixbfcal}{Latin}{ "E247}

2734 \usv_set:nnn {stixbfitslash}{Latin}{ "E295}
2735 \usv_set:nnn {stixbfitslash}{latin}{ "E2C9}
2736 \usv_set:nnn {stixbfitslash}{greek}{ "E32D}
2737 \usv_set:nnn {stixsfitslash}{epsilon}{ "E37B}
2738 \usv_set:nnn {stixsfitslash}{vartheta}{ "E35F}
2739 \usv_set:nnn {stixsfitslash}{varkappa}{ "E375}
2740 \usv_set:nnn {stixsfitslash}{phi}{ "E361}
2741 \usv_set:nnn {stixsfitslash}{varrho}{ "E377}
2742 \usv_set:nnn {stixsfitslash}{varpi}{ "E363}
2743 \usv_set:nnn {stixsfitslash}{digamma}{ "E36B}

2744 </stix>

```

N.3 Alphabets

2745 <*alphabets>

N.3.1 Upright: up

```

2746 \@@_new_alphabet_config:nnn {up} {num}
2747 {
2748   \@@_set_normal_numbers:nn {up} {#1}
2749   \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
2750 }
2751
2752 \@@_new_alphabet_config:nnn {up} {Latin}
2753 {
2754   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
2755   {
2756     \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2757   }
2758   \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
2759   \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
2760   \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
2761 }
2762
2763 \@@_new_alphabet_config:nnn {up} {latin}
2764 {
2765   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
2766   {
2767     \bool_if:NT \g_@@_uplatin_bool
2768     {
2769       \@@_set_normal_latin:nn {up,it} {#1}
2770       \@@_set_normal_char:nnn {h} {up,it} {#1}
2771       \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
2772       \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
2773     }
2774   }
2775   \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
2776   \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
2777   \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
2778 }
2779
2780 \@@_new_alphabet_config:nnn {up} {Greek}
2781 {
2782   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
2783   {
2784     \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2785   }
2786   \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
2787   \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
2788   \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
2789 }
2790
2791 \@@_new_alphabet_config:nnn {up} {greek}
2792 {
2793   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
2794   {

```

```

2795 \bool_if:NT \g_@@_upgreek_bool
2796 {
2797   \@@_set_normal_greek:nn {up,it} {#1}
2798 }
2799 }
2800 \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
2801 \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
2802 \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
2803 }
2804
2805 \@@_new_alphabet_config:nnn {up} {misc}
2806 {
2807   \bool_if:NTF \g_@@_literal_Nabla_bool
2808   {
2809     \@@_set_normal_char:nnn {Nabla}{up}{up}
2810   }
2811   {
2812     \bool_if:NT \g_@@_upNabla_bool
2813     {
2814       \@@_set_normal_char:nnn {Nabla}{up,it}{up}
2815     }
2816   }
2817   \bool_if:NTF \g_@@_literal_partial_bool
2818   {
2819     \@@_set_normal_char:nnn {partial}{up}{up}
2820   }
2821   {
2822     \bool_if:NT \g_@@_uppartial_bool
2823     {
2824       \@@_set_normal_char:nnn {partial}{up,it}{up}
2825     }
2826   }
2827   \@@_set_mathalphabet_pos:nnnn {up} {partial} {up,it} {#1}
2828   \@@_set_mathalphabet_pos:nnnn {up} {Nabla} {up,it} {#1}
2829   \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
2830   \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
2831 }

```

N.3.2 *Italic: it*

```

2832 \@@_new_alphabet_config:nnn {it} {Latin}
2833 {
2834   \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
2835   {
2836     \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
2837   }
2838   \@@_set_mathalphabet_Latin:nnn {it}{up,it}{#1}
2839 }
2840
2841 \@@_new_alphabet_config:nnn {it} {latin}

```

```

2842 {
2843   \bool_if:NTF \g_@@_literal_bool
2844   {
2845     \@@_set_normal_latin:nn {it} {#1}
2846     \@@_set_normal_char:nnn {h}{it}{#1}
2847   }
2848   {
2849     \bool_if:NF \g_@@_uplatin_bool
2850     {
2851       \@@_set_normal_latin:nn {up,it} {#1}
2852       \@@_set_normal_char:nnn {h}{up,it}{#1}
2853       \@@_set_normal_char:nnn {dotlessi}{up,it}{#1}
2854       \@@_set_normal_char:nnn {dotlessj}{up,it}{#1}
2855     }
2856   }
2857   \@@_set_mathalphabet_latin:nnn {it} {up,it} {#1}
2858   \@@_set_mathalphabet_pos:nnnn {it} {dotlessi} {up,it} {#1}
2859   \@@_set_mathalphabet_pos:nnnn {it} {dotlessj} {up,it} {#1}
2860 }
2861
2862 \@@_new_alphabet_config:nnn {it} {Greek}
2863 {
2864   \bool_if:NTF \g_@@_literal_bool
2865   {
2866     \@@_set_normal_Greek:nn {it}{#1}
2867   }
2868   {
2869     \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
2870   }
2871   \@@_set_mathalphabet_Greek:nnn {it} {up,it}{#1}
2872 }
2873
2874 \@@_new_alphabet_config:nnn {it} {greek}
2875 {
2876   \bool_if:NTF \g_@@_literal_bool
2877   {
2878     \@@_set_normal_greek:nn {it} {#1}
2879   }
2880   {
2881     \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
2882   }
2883   \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
2884 }
2885
2886 \@@_new_alphabet_config:nnn {it} {misc}
2887 {
2888   \bool_if:NTF \g_@@_literal_Nabla_bool
2889   {
2890     \@@_set_normal_char:nnn {Nabla}{it}{it}

```

```

2891 }
2892 {
2893   \bool_if:NF \g_@@_upNabla_bool
2894   {
2895     \@@_set_normal_char:nnn {Nabla}{up,it}{it}
2896   }
2897 }
2898 \bool_if:NTF \g_@@_literal_partial_bool
2899 {
2900   \@@_set_normal_char:nnn {partial}{it}{it}
2901 }
2902 {
2903   \bool_if:NF \g_@@_uppartial_bool
2904   {
2905     \@@_set_normal_char:nnn {partial}{up,it}{it}
2906   }
2907 }
2908 \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
2909 \@@_set_mathalphabet_pos:nnnn {it} {Nabla} {up,it}{#1}
2910 }

```

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

```

2911 \@@_new_alphabet_config:nnn {bb} {latin}
2912 {
2913   \@@_set_mathalphabet_latin:nnn {bb} {up,it}{#1}
2914 }
2915
2916 \@@_new_alphabet_config:nnn {bb} {Latin}
2917 {
2918   \@@_set_mathalphabet_Latin:nnn {bb} {up,it}{#1}
2919   \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
2920   \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
2921   \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
2922   \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
2923   \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
2924   \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
2925   \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
2926 }
2927
2928 \@@_new_alphabet_config:nnn {bb} {num}
2929 {
2930   \@@_set_mathalphabet_numbers:nnn {bb} {up}{#1}
2931 }
2932
2933 \@@_new_alphabet_config:nnn {bb} {misc}
2934 {
2935   \@@_set_mathalphabet_pos:nnnn {bb} {Pi} {up,it} {#1}
2936   \@@_set_mathalphabet_pos:nnnn {bb} {pi} {up,it} {#1}
2937   \@@_set_mathalphabet_pos:nnnn {bb} {Gamma} {up,it} {#1}

```

```

2938 \@@_set_mathalphabet_pos:nnnn {bb} {gamma} {up,it} {#1}
2939 \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up} {#1}
2940 }
2941
2942 \@@_new_alphabet_config:nnn {bbit} {misc}
2943 {
2944 \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
2945 \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
2946 \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
2947 \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
2948 \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
2949 }

```

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

```

2950 \@@_new_alphabet_config:nnn {scr} {Latin}
2951 {
2952 \@@_set_mathalphabet_Latin:nnn {scr} {up,it}{#1}
2953 \@@_set_mathalphabet_pos:nnnn {scr} {B}{up,it}{#1}
2954 \@@_set_mathalphabet_pos:nnnn {scr} {E}{up,it}{#1}
2955 \@@_set_mathalphabet_pos:nnnn {scr} {F}{up,it}{#1}
2956 \@@_set_mathalphabet_pos:nnnn {scr} {H}{up,it}{#1}
2957 \@@_set_mathalphabet_pos:nnnn {scr} {I}{up,it}{#1}
2958 \@@_set_mathalphabet_pos:nnnn {scr} {L}{up,it}{#1}
2959 \@@_set_mathalphabet_pos:nnnn {scr} {M}{up,it}{#1}
2960 \@@_set_mathalphabet_pos:nnnn {scr} {R}{up,it}{#1}
2961 }
2962
2963 \@@_new_alphabet_config:nnn {scr} {latin}
2964 {
2965 \@@_set_mathalphabet_latin:nnn {scr} {up,it}{#1}
2966 \@@_set_mathalphabet_pos:nnnn {scr} {e}{up,it}{#1}
2967 \@@_set_mathalphabet_pos:nnnn {scr} {g}{up,it}{#1}
2968 \@@_set_mathalphabet_pos:nnnn {scr} {o}{up,it}{#1}
2969 }

```

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

```

2970 \@@_new_alphabet_config:nnn {cal} {Latin}
2971 {
2972 \@@_set_mathalphabet_Latin:nnn {cal} {up,it}{#1}
2973 \@@_set_mathalphabet_pos:nnnn {cal} {B}{up,it}{#1}
2974 \@@_set_mathalphabet_pos:nnnn {cal} {E}{up,it}{#1}
2975 \@@_set_mathalphabet_pos:nnnn {cal} {F}{up,it}{#1}
2976 \@@_set_mathalphabet_pos:nnnn {cal} {H}{up,it}{#1}
2977 \@@_set_mathalphabet_pos:nnnn {cal} {I}{up,it}{#1}
2978 \@@_set_mathalphabet_pos:nnnn {cal} {L}{up,it}{#1}
2979 \@@_set_mathalphabet_pos:nnnn {cal} {M}{up,it}{#1}
2980 \@@_set_mathalphabet_pos:nnnn {cal} {R}{up,it}{#1}
2981 }

```

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

```

2982 \@@_new_alphabet_config:nnn {frak} {Latin}
2983 {
2984   \@@_set_mathalphabet_Latin:nnn {frak} {up,it}{#1}
2985   \@@_set_mathalphabet_pos:nnnn {frak} {C}{up,it}{#1}
2986   \@@_set_mathalphabet_pos:nnnn {frak} {H}{up,it}{#1}
2987   \@@_set_mathalphabet_pos:nnnn {frak} {I}{up,it}{#1}
2988   \@@_set_mathalphabet_pos:nnnn {frak} {R}{up,it}{#1}
2989   \@@_set_mathalphabet_pos:nnnn {frak} {Z}{up,it}{#1}
2990 }
2991 \@@_new_alphabet_config:nnn {frak} {latin}
2992 {
2993   \@@_set_mathalphabet_latin:nnn {frak} {up,it}{#1}
2994 }

```

N.3.6 *Sans serif upright: sfup*

```

2995 \@@_new_alphabet_config:nnn {sfup} {num}
2996 {
2997   \@@_set_mathalphabet_numbers:nnn {sf} {up}{#1}
2998   \@@_set_mathalphabet_numbers:nnn {sfup} {up}{#1}
2999 }
3000 \@@_new_alphabet_config:nnn {sfup} {Latin}
3001 {
3002   \bool_if:NTF \g_@@_sfliteral_bool
3003   {
3004     \@@_set_normal_Latin:nn {sfup} {#1}
3005     \@@_set_mathalphabet_Latin:nnn {sf} {up}{#1}
3006   }
3007   {
3008     \bool_if:NT \g_@@_upsans_bool
3009     {
3010       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3011       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3012     }
3013   }
3014   \@@_set_mathalphabet_Latin:nnn {sfup} {up,it}{#1}
3015 }
3016 \@@_new_alphabet_config:nnn {sfup} {latin}
3017 {
3018   \bool_if:NTF \g_@@_sfliteral_bool
3019   {
3020     \@@_set_normal_latin:nn {sfup} {#1}
3021     \@@_set_mathalphabet_latin:nnn {sf} {up}{#1}
3022   }
3023   {
3024     \bool_if:NT \g_@@_upsans_bool
3025     {
3026       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3027       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}

```



```

3028     }
3029   }
3030   \@@_set_mathalphabet_latin:nnn {sfup} {up,it}{#1}
3031 }

```

N.3.7 *Sans serif italic: sfit*

```

3032 \@@_new_alphabet_config:nnn {sfit} {Latin}
3033 {
3034   \bool_if:NTF \g_@@_sfliteral_bool
3035   {
3036     \@@_set_normal_Latin:nn {sfit} {#1}
3037     \@@_set_mathalphabet_Latin:nnn {sf} {it}{#1}
3038   }
3039   {
3040     \bool_if:NF \g_@@_upsans_bool
3041     {
3042       \@@_set_normal_Latin:nn {sfup,sfit} {#1}
3043       \@@_set_mathalphabet_Latin:nnn {sf} {up,it}{#1}
3044     }
3045   }
3046   \@@_set_mathalphabet_Latin:nnn {sfit} {up,it}{#1}
3047 }
3048 \@@_new_alphabet_config:nnn {sfit} {latin}
3049 {
3050   \bool_if:NTF \g_@@_sfliteral_bool
3051   {
3052     \@@_set_normal_latin:nn {sfit} {#1}
3053     \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
3054   }
3055   {
3056     \bool_if:NF \g_@@_upsans_bool
3057     {
3058       \@@_set_normal_latin:nn {sfup,sfit} {#1}
3059       \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
3060     }
3061   }
3062   \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
3063 }

```

N.3.8 *Typewriter or monospaced: tt*

```

3064 \@@_new_alphabet_config:nnn {tt} {num}
3065 {
3066   \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
3067 }
3068 \@@_new_alphabet_config:nnn {tt} {Latin}
3069 {
3070   \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
3071 }
3072 \@@_new_alphabet_config:nnn {tt} {latin}

```

```

3073 {
3074 \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
3075 }

```

N.3.9 *Bold Italic: bfit*

```

3076 \@@_new_alphabet_config:nnn {bfit} {Latin}
3077 {
3078 \bool_if:NF \g_@@_bfupLatin_bool
3079 {
3080 \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3081 }
3082 \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
3083 \bool_if:NTF \g_@@_bfliteral_bool
3084 {
3085 \@@_set_normal_Latin:nn {bfit} {#1}
3086 \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
3087 }
3088 {
3089 \bool_if:NF \g_@@_bfupLatin_bool
3090 {
3091 \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3092 \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3093 }
3094 }
3095 }
3096
3097 \@@_new_alphabet_config:nnn {bfit} {latin}
3098 {
3099 \bool_if:NF \g_@@_bfuplatin_bool
3100 {
3101 \@@_set_normal_latin:nn {bfup,bfit} {#1}
3102 }
3103 \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
3104 \bool_if:NTF \g_@@_bfliteral_bool
3105 {
3106 \@@_set_normal_latin:nn {bfit} {#1}
3107 \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
3108 }
3109 {
3110 \bool_if:NF \g_@@_bfuplatin_bool
3111 {
3112 \@@_set_normal_latin:nn {bfup,bfit} {#1}
3113 \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3114 }
3115 }
3116 }
3117
3118 \@@_new_alphabet_config:nnn {bfit} {Greek}
3119 {

```

```

3120 \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
3121 \bool_if:NTF \g_@@_bfliteral_bool
3122 {
3123   \@@_set_normal_Greek:nn {bfit}{#1}
3124   \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
3125 }
3126 {
3127   \bool_if:NF \g_@@_bfupGreek_bool
3128   {
3129     \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3130     \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3131   }
3132 }
3133 }
3134
3135 \@@_new_alphabet_config:nnn {bfit} {greek}
3136 {
3137   \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
3138   \bool_if:NTF \g_@@_bfliteral_bool
3139   {
3140     \@@_set_normal_greek:nn {bfit} {#1}
3141     \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
3142   }
3143   {
3144     \bool_if:NF \g_@@_bfupgreek_bool
3145     {
3146       \@@_set_normal_greek:nn {bfit,bfup} {#1}
3147       \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3148     }
3149   }
3150 }
3151
3152 \@@_new_alphabet_config:nnn {bfit} {misc}
3153 {
3154   \bool_if:NTF \g_@@_literal_Nabla_bool
3155   { \@@_set_normal_char:nnn {Nabla}{bfit}{#1} }
3156   {
3157     \bool_if:NF \g_@@_upNabla_bool
3158     { \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1} }
3159   }
3160   \bool_if:NTF \g_@@_literal_partial_bool
3161   { \@@_set_normal_char:nnn {partial}{bfit}{#1} }
3162   {
3163     \bool_if:NF \g_@@_uppartial_bool
3164     { \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1} }
3165   }
3166   \@@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it}{#1}
3167   \@@_set_mathalphabet_pos:nnnn {bfit} {Nabla} {up,it}{#1}
3168   \bool_if:NTF \g_@@_literal_partial_bool

```

```

3169 {
3170   \@@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
3171 }
3172 {
3173   \bool_if:NF \g_@@_uppartial_bool
3174   {
3175     \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3176   }
3177 }
3178 \bool_if:NTF \g_@@_literal_Nabla_bool
3179 {
3180   \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {it}{#1}
3181 }
3182 {
3183   \bool_if:NF \g_@@_upNabla_bool
3184   {
3185     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3186   }
3187 }
3188 }

```

N.3.10 *Bold Upright: bfup*

```

3189 \@@_new_alphabet_config:nnn {bfup} {num}
3190 {
3191   \@@_set_mathalphabet_numbers:nnn {bf} {up}{#1}
3192   \@@_set_mathalphabet_numbers:nnn {bfup} {up}{#1}
3193 }
3194
3195 \@@_new_alphabet_config:nnn {bfup} {Latin}
3196 {
3197   \bool_if:NT \g_@@_bfupLatin_bool
3198   {
3199     \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3200   }
3201   \@@_set_mathalphabet_Latin:nnn {bfup} {up,it}{#1}
3202   \bool_if:NTF \g_@@_bfliteral_bool
3203   {
3204     \@@_set_normal_Latin:nn {bfup} {#1}
3205     \@@_set_mathalphabet_Latin:nnn {bf} {up}{#1}
3206   }
3207   {
3208     \bool_if:NT \g_@@_bfupLatin_bool
3209     {
3210       \@@_set_normal_Latin:nn {bfup,bfit} {#1}
3211       \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
3212     }
3213   }
3214 }
3215

```

```

3216 \@@_new_alphabet_config:nnn {bfup} {latin}
3217 {
3218   \bool_if:NT \g_@@_bfuplatin_bool
3219   {
3220     \@@_set_normal_latin:nn {bfup,bfit} {#1}
3221   }
3222   \@@_set_mathalphabet_latin:nnn {bfup} {up,it}{#1}
3223   \bool_if:NTF \g_@@_bfliteral_bool
3224   {
3225     \@@_set_normal_latin:nn {bfup} {#1}
3226     \@@_set_mathalphabet_latin:nnn {bf} {up}{#1}
3227   }
3228   {
3229     \bool_if:NT \g_@@_bfuplatin_bool
3230     {
3231       \@@_set_normal_latin:nn {bfup,bfit} {#1}
3232       \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
3233     }
3234   }
3235 }
3236 \@@_new_alphabet_config:nnn {bfup} {Greek}
3237 {
3238   \@@_set_mathalphabet_Greek:nnn {bfup} {up,it}{#1}
3239   \bool_if:NTF \g_@@_bfliteral_bool
3240   {
3241     \@@_set_normal_Greek:nn {bfup}{#1}
3242     \@@_set_mathalphabet_Greek:nnn {bf} {up}{#1}
3243   }
3244   {
3245     \bool_if:NT \g_@@_bfupGreek_bool
3246     {
3247       \@@_set_normal_Greek:nn {bfup,bfit}{#1}
3248       \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
3249     }
3250   }
3251 }
3252
3253 \@@_new_alphabet_config:nnn {bfup} {greek}
3254 {
3255   \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
3256   \bool_if:NTF \g_@@_bfliteral_bool
3257   {
3258     \@@_set_normal_greek:nn {bfup} {#1}
3259     \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
3260   }
3261   {
3262     \bool_if:NT \g_@@_bfupgreek_bool
3263     {
3264       \@@_set_normal_greek:nn {bfup,bfit} {#1}

```

```

3265 \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
3266 }
3267 }
3268 }
3269
3270 \@@_new_alphabet_config:nnn {bfup} {misc}
3271 {
3272 \bool_if:NTF \g_@@_literal_Nabla_bool
3273 {
3274 \@@_set_normal_char:nnn {Nabla}{bfup}{#1}
3275 }
3276 {
3277 \bool_if:NT \g_@@_upNabla_bool
3278 {
3279 \@@_set_normal_char:nnn {Nabla}{bfup,bfit}{#1}
3280 }
3281 }
3282 \bool_if:NTF \g_@@_literal_partial_bool
3283 {
3284 \@@_set_normal_char:nnn {partial}{bfup}{#1}
3285 }
3286 {
3287 \bool_if:NT \g_@@_uppartial_bool
3288 {
3289 \@@_set_normal_char:nnn {partial}{bfup,bfit}{#1}
3290 }
3291 }
3292 \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it}{#1}
3293 \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla} {up,it}{#1}
3294 \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up}{#1}
3295 \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up}{#1}
3296 \@@_set_mathalphabet_pos:nnnn {bf} {digamma} {up}{#1}
3297 \@@_set_mathalphabet_pos:nnnn {bf} {Digamma} {up}{#1}
3298 \bool_if:NTF \g_@@_literal_partial_bool
3299 {
3300 \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up}{#1}
3301 }
3302 {
3303 \bool_if:NT \g_@@_uppartial_bool
3304 {
3305 \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
3306 }
3307 }
3308 \bool_if:NTF \g_@@_literal_Nabla_bool
3309 {
3310 \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}
3311 }
3312 {
3313 \bool_if:NT \g_@@_upNabla_bool

```

```

3314     {
3315     \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it}{#1}
3316     }
3317   }
3318 }

```

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

```

3319 \@@_new_alphabet_config:nnn {bffrak} {Latin}
3320 {
3321 \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
3322 }
3323
3324 \@@_new_alphabet_config:nnn {bffrak} {latin}
3325 {
3326 \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
3327 }

```

N.3.12 *Bold script or calligraphic: bfscr*

```

3328 \@@_new_alphabet_config:nnn {bfscr} {Latin}
3329 {
3330 \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
3331 }
3332 \@@_new_alphabet_config:nnn {bfscr} {latin}
3333 {
3334 \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
3335 }
3336 \@@_new_alphabet_config:nnn {bfcal} {Latin}
3337 {
3338 \@@_set_mathalphabet_Latin:nnn {bfcal} {up,it}{#1}
3339 }

```

N.3.13 *Bold upright sans serif: bfsfup*

```

3340 \@@_new_alphabet_config:nnn {bfsfup} {num}
3341 {
3342 \@@_set_mathalphabet_numbers:nnn {bfsf} {up}{#1}
3343 \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
3344 }
3345 \@@_new_alphabet_config:nnn {bfsfup} {Latin}
3346 {
3347 \bool_if:NTF \g_@@_sfliteral_bool
3348 {
3349 \@@_set_normal_Latin:nn {bfsfup} {#1}
3350 \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
3351 }
3352 {
3353 \bool_if:NT \g_@@_upsans_bool
3354 {
3355 \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
3356 \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}

```

```

3357     }
3358   }
3359   \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
3360 }
3361
3362 \@@_new_alphabet_config:nnn {bfsfup} {latin}
3363 {
3364   \bool_if:NTF \g_@@_sfliteral_bool
3365   {
3366     \@@_set_normal_latin:nn {bfsfup} {#1}
3367     \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
3368   }
3369   {
3370     \bool_if:NT \g_@@_upsans_bool
3371     {
3372       \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
3373       \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
3374     }
3375   }
3376   \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
3377 }
3378
3379 \@@_new_alphabet_config:nnn {bfsfup} {Greek}
3380 {
3381   \bool_if:NTF \g_@@_sfliteral_bool
3382   {
3383     \@@_set_normal_Greek:nn {bfsfup}{#1}
3384     \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
3385   }
3386   {
3387     \bool_if:NT \g_@@_upsans_bool
3388     {
3389       \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
3390       \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3391     }
3392   }
3393   \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
3394 }
3395
3396 \@@_new_alphabet_config:nnn {bfsfup} {greek}
3397 {
3398   \bool_if:NTF \g_@@_sfliteral_bool
3399   {
3400     \@@_set_normal_greek:nn {bfsfup} {#1}
3401     \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
3402   }
3403   {
3404     \bool_if:NT \g_@@_upsans_bool
3405     {

```



```

3406 \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3407 \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3408 }
3409 }
3410 \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
3411 }
3412 \@@_new_alphabet_config:nnn {bfsfup} {misc}
3413 {
3414 \bool_if:NTF \g_@@_literal_Nabla_bool
3415 {
3416 \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
3417 }
3418 {
3419 \bool_if:NT \g_@@_upNabla_bool
3420 {
3421 \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3422 }
3423 }
3424 \bool_if:NTF \g_@@_literal_partial_bool
3425 {
3426 \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
3427 }
3428 {
3429 \bool_if:NT \g_@@_uppartial_bool
3430 {
3431 \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3432 }
3433 }
3434 \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
3435 \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla} {up,it}{#1}
3436 \bool_if:NTF \g_@@_literal_partial_bool
3437 {
3438 \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
3439 }
3440 {
3441 \bool_if:NT \g_@@_uppartial_bool
3442 {
3443 \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3444 }
3445 }
3446 \bool_if:NTF \g_@@_literal_Nabla_bool
3447 {
3448 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up}{#1}
3449 }
3450 {
3451 \bool_if:NT \g_@@_upNabla_bool
3452 {
3453 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3454 }

```

3455 }

3456 }

N.3.14 *Bold italic sans serif: bfsfit*

3457 \@@_new_alphabet_config:nnn {bfsfit} {Latin}

3458 {

3459 \bool_if:NTF \g_@@_sfliteral_bool

3460 {

3461 \@@_set_normal_Latin:nn {bfsfit} {#1}

3462 \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}

3463 }

3464 {

3465 \bool_if:NF \g_@@_upsans_bool

3466 {

3467 \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}

3468 \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}

3469 }

3470 }

3471 \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}

3472 }

3473

3474 \@@_new_alphabet_config:nnn {bfsfit} {latin}

3475 {

3476 \bool_if:NTF \g_@@_sfliteral_bool

3477 {

3478 \@@_set_normal_latin:nn {bfsfit} {#1}

3479 \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}

3480 }

3481 {

3482 \bool_if:NF \g_@@_upsans_bool

3483 {

3484 \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}

3485 \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}

3486 }

3487 }

3488 \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}

3489 }

3490

3491 \@@_new_alphabet_config:nnn {bfsfit} {Greek}

3492 {

3493 \bool_if:NTF \g_@@_sfliteral_bool

3494 {

3495 \@@_set_normal_Greek:nn {bfsfit}{#1}

3496 \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}

3497 }

3498 {

3499 \bool_if:NF \g_@@_upsans_bool

3500 {

3501 \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}

```

3502     \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
3503   }
3504 }
3505 \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
3506 }
3507
3508 \@@_new_alphabet_config:nnn {bfsfit} {greek}
3509 {
3510   \bool_if:NTF \g_@@_sfliteral_bool
3511   {
3512     \@@_set_normal_greek:nn {bfsfit} {#1}
3513     \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
3514   }
3515   {
3516     \bool_if:NF \g_@@_upsans_bool
3517     {
3518       \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
3519       \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
3520     }
3521   }
3522   \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
3523 }
3524
3525 \@@_new_alphabet_config:nnn {bfsfit} {misc}
3526 {
3527   \bool_if:NTF \g_@@_literal_Nabla_bool
3528   {
3529     \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
3530   }
3531   {
3532     \bool_if:NF \g_@@_upNabla_bool
3533     {
3534       \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
3535     }
3536   }
3537   \bool_if:NTF \g_@@_literal_partial_bool
3538   {
3539     \@@_set_normal_char:nnn {partial}{bfsfit}{#1}
3540   }
3541   {
3542     \bool_if:NF \g_@@_uppartial_bool
3543     {
3544       \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
3545     }
3546   }
3547   \@@_set_mathalphabet_pos:nnnn {bfsfit} {partial} {up,it}{#1}
3548   \@@_set_mathalphabet_pos:nnnn {bfsfit} {Nabla} {up,it}{#1}
3549   \bool_if:NTF \g_@@_literal_partial_bool
3550   {

```

```

3551 \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {it}{#1}
3552 }
3553 {
3554 \bool_if:NF \g_@@_uppartial_bool
3555 {
3556 \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
3557 }
3558 }
3559 \bool_if:NTF \g_@@_literal_Nabla_bool
3560 {
3561 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {it}{#1}
3562 }
3563 {
3564 \bool_if:NF \g_@@_upNabla_bool
3565 {
3566 \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla} {up,it}{#1}
3567 }
3568 }
3569 }
3570 </alphabets>

```

N.4 Compatibility

```

3571 <*compat>

```

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

Tries to patch $\langle command \rangle$. 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 $\langle new replacement text for \text{LuaTeX} \rangle$ or the $\langle new replacement text for \text{XeTeX} \rangle$, depending on the engine. Otherwise issue a warning and don't overwrite.

```

3572 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnnn
3573 {
3574 \cs_if_exist:NT #1
3575 {
3576 \token_if_macro:NTF #1
3577 {
3578 \group_begin:
3579 #2 \@@_tmpa:w #3 { #4 }
3580 \cs_if_eq:NNTF #1 \@@_tmpa:w
3581 {
3582 \msg_info:nx { unicode-math } { patch-macro }
3583 { \token_to_str:N #1 }

```

```

3584     \group_end:
3585     #2 #1 #3
3586     <XE>     { #6 }
3587     <LU>     { #5 }
3588   }
3589   {
3590     \msg_warning:nnxxx { unicode-math } { wrong-meaning }
3591     { \token_to_str:N #1 } { \token_to_meaning:N #1 }
3592     { \token_to_meaning:N \@_tmpa:w }
3593   \group_end:
3594   }
3595 }
3596 {
3597   \msg_warning:nnx { unicode-math } { macro-expected }
3598   { \token_to_str:N #1 }
3599 }
3600 }
3601 }

```

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

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

```

3602 \cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
3603 {
3604   \@@_check_and_fix:NNnnnn #1 #2 { #3 } { #4 } { #5 } { #5 }
3605 }

```

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

```

3606 \cs_new_protected_nopar:Nn \@@_check_and_fix_luatex:NNnnn
3607 {
3608   <LU> \@@_check_and_fix:NNnnn #1 #2 { #3 } { #4 } { #5 }
3609 }

```

```
3610 \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.

```
3611 \AtEndOfPackageFile * {url}
3612 {
3613   \tl_put_left:Nn \Url@FormatString { \@@_switchto_up: }
3614   \tl_put_right:Nn \Url@Specials
3615   {
3616     \do{\mathchar`\'}
3617     \do{\mathchar`\'}
3618     \do{\mathchar`\$}
3619     \do{\mathchar`\&}
3620   }
3621 }
```

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:

```
3622 \AtEndOfPackageFile * {amsmath}
3623 {
3624   (*XE)
3625   \tl_remove_once:Nn \@begindocumenthook
3626   {
3627     \mathchardef\std@minus\mathcode`\-\relax
3628     \mathchardef\std@equal\mathcode`\=\relax
3629   }
3630   \def\std@minus{\Umathcharnum\Umathcodenum`\-\relax}
3631   \def\std@equal{\Umathcharnum\Umathcodenum`\=\relax}
3632   (/XE)
3633   \cs_set:Npn \@cdots {\mathinner{\cdots}}
3634   \cs_set_eq:NN \dotsof \cdots
```

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

```
3635   (*XE)
3636   \def \resetMathstrut@
3637   {%
3638     \setbox\z@\hbox{$(\$)%}
3639     \ht\Mathstrutbox@ \ht\z@ \dp\Mathstrutbox@ \dp\z@
3640   }
```

The subarray environment uses inappropriate font dimensions.

```
3641   \@@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
3642   {
```

```

3643 \vcenter
3644 \bgroup
3645 \Let@
3646 \restore@math@cr
3647 \default@tag
3648 \baselineskip \fontdimen 10~ \scriptfont \tw@
3649 \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
3650 \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
3651 \lineskiplimit \lineskip
3652 \ialign
3653 \bgroup
3654 \ifx c #1 \hfil \fi
3655 $ \m@th \scriptstyle ## $
3656 \hfil
3657 \crr
3658 }
3659 {
3660 \vcenter
3661 \c_group_begin_token
3662 \Let@
3663 \restore@math@cr
3664 \default@tag
3665 \skip_set:Nn \baselineskip
3666 {

```

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

```

3667 \@@_stack_num_up:N \scriptstyle
3668 + \@@_stack_denom_down:N \scriptstyle
3669 }

```

Here we use the minimum stack gap.

```

3670 \lineskip \@@_stack_vgap:N \scriptstyle
3671 \lineskiplimit \lineskip
3672 \ialign
3673 \c_group_begin_token
3674 \token_if_eq_meaning:NNT c #1 { \hfil }
3675 \c_math_toggle_token
3676 \m@th
3677 \scriptstyle
3678 \c_parameter_token \c_parameter_token
3679 \c_math_toggle_token
3680 \hfil
3681 \crr
3682 }
3683 (/XE)

```

The roots need a complete rework.

```

3684 \@@_check_and_fix luatex:Nnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
3685 {
3686 \setbox \rootbox \hbox
3687 {

```

```

3688     $ \m@th \scriptscriptstyle { #1 } $
3689   }
3690   \mathchoice
3691     { \r@@@t \displaystyle      { #2 } }
3692     { \r@@@t \textstyle        { #2 } }~
3693     { \r@@@t \scriptstyle      { #2 } }
3694     { \r@@@t \scriptscriptstyle { #2 } }
3695   \egroup
3696 }
3697 {
3698   \bool_if:nTF
3699   {
3700     \int_compare_p:nNn { \uproot@ } = { \c_zero }
3701     && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
3702   }
3703   {
3704     \Uroot \l_@_radical_sqrt_tl { #1 } { #2 }
3705   }
3706   {
3707     \hbox_set:Nn \rootbox
3708     {
3709       \c_math_toggle_token
3710       \m@th
3711       \scriptscriptstyle { #1 }
3712       \c_math_toggle_token
3713     }
3714     \mathchoice
3715       { \r@@@t \displaystyle      { #2 } }
3716       { \r@@@t \textstyle        { #2 } }
3717       { \r@@@t \scriptstyle      { #2 } }
3718       { \r@@@t \scriptscriptstyle { #2 } }
3719   }
3720   \c_group_end_token
3721 }
3722 \@_check_and_fix:NNnnn \r@@@t \cs_set_nopar:Npn { #1 #2 }
3723 {
3724   \setboxz@h { $ \m@th #1 \sqrt{sign { #2 } } $ }
3725   \dimen@ \ht\z@
3726   \advance \dimen@ -\dp\z@
3727   \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
3728   \advance \dimen@ by 1.667 \wd\@ne
3729   \mkern -\leftroot@ mu
3730   \mkern 5mu
3731   \raise .6\dimen@ \copy\rootbox
3732   \mkern -10mu
3733   \mkern \leftroot@ mu
3734   \boxz@
3735 }
3736 {

```



```

3737 \hbox_set:Nn \l_tmpa_box
3738 {
3739   \c_math_toggle_token
3740   \m@th
3741   #1
3742   \mskip \uproot@ mu
3743   \c_math_toggle_token
3744 }
3745 \Uroot \l_@@_radical_sqrt_tl
3746 {
3747   \box_move_up:nn { \box_wd:N \l_tmpa_box }
3748   {
3749     \hbox:n
3750     {
3751       \c_math_toggle_token
3752       \m@th
3753       \mkern -\leftroot@ mu
3754       \box_use:N \rootbox
3755       \mkern \leftroot@ mu
3756       \c_math_toggle_token
3757     }
3758   }
3759 }
3760 { #2 }
3761 }
3762 {
3763   \hbox_set:Nn \l_tmpa_box
3764   {
3765     \c_math_toggle_token
3766     \m@th
3767     #1
3768     \sqrtsign { #2 }
3769     \c_math_toggle_token
3770   }
3771   \hbox_set:Nn \l_tmpb_box
3772   {
3773     \c_math_toggle_token
3774     \m@th
3775     #1
3776     \mskip \uproot@ mu
3777     \c_math_toggle_token
3778   }
3779   \mkern -\leftroot@ mu
3780   \@@_mathstyle_scale:Nnn #1 { \kern }
3781   {
3782     \fontdimen 63 \l_@@_font
3783   }
3784   \box_move_up:nn
3785   {

```

```

3786 \box_wd:N \l_tmpb_box
3787 + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
3788 * \number \fontdimen 65 \l_@@_font / 100
3789 }
3790 {
3791 \box_use:N \rootbox
3792 }
3793 \@@_mathstyle_scale:Nnn #1 { \kern }
3794 {
3795 \fontdimen 64 \l_@@_font
3796 }
3797 \mkern \leftroot@ mu
3798 \box_use_clear:N \l_tmpa_box
3799 }
3800 }

```

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:

```

3801 (*XE)
3802 \AtEndOfPackageFile * {amsopn}
3803 {
3804 \cs_set:Npn \newmcodes@
3805 {
3806 \mathcode'\`39\scan_stop:
3807 \mathcode'\`42\scan_stop:
3808 \mathcode'\`613A\scan_stop:
3809 %% \ifnum\mathcode'\`-=45 \else
3810 %% \mathchardef\std@minus\mathcode'\`-\relax
3811 %% \fi
3812 \mathcode'\`-45\scan_stop:
3813 \mathcode'\`/47\scan_stop:
3814 \mathcode'\`:"603A\scan_stop:
3815 }
3816 }
3817 (/XE)

```

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

```

3818 (*XE)
3819 \AtEndOfPackageFile * { mathtools }
3820 {
3821 \@@_check_and_fix:NNnnn
3822 \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
3823 {
3824 \sbox \z@
3825 {
3826 $
3827 \m@th

```

```

3828     #1
3829     \nulldelimiterspace = \z@
3830     \radical \z@ { #2 }
3831     $
3832   }
3833   \ifx #1 \displaystyle
3834     \dimen@ = \fontdimen 8 \textfont 3
3835     \advance \dimen@ .25 \fontdimen 5 \textfont 2
3836   \else
3837     \dimen@ = 1.25 \fontdimen 8
3838     \ifx #1 \textstyle
3839       \textfont
3840     \else
3841       \ifx #1 \scriptstyle
3842         \scriptfont
3843       \else
3844         \scriptscriptfont
3845       \fi
3846     \fi
3847     3
3848   \fi
3849   \advance \dimen@ -\ht\z@
3850   \ht\z@ = -\dimen@
3851   \box\z@
3852 }

```

The Xe_{La}TeX version is pretty similar to the legacy version, only using the correct font dimensions. Note we used ‘\XeTeXradical’ with the family 255 to be almost sure that the radical rule width is not set. Former use of ‘\newfam’ had an upsetting effect on legacy math alphabets.

```

3853 {
3854   \hbox_set:Nn \l_tmpa_box
3855   {
3856     \color@setgroup
3857     \c_math_toggle_token
3858     \m@th
3859     #1
3860     \dim_zero:N \nulldelimiterspace
3861     \XeTeXradical \c_two_hundred_fifty_five \c_zero { #2 }
3862     \c_math_toggle_token
3863     \color@endgroup
3864   }
3865   \box_set_ht:Nn \l_tmpa_box
3866   {
3867     \box_ht:N \l_tmpa_box

```

Here we use the radical vertical gap.

```

3868     - \@@_radical_vgap:N #1
3869   }
3870   \box_use_clear:N \l_tmpa_box

```

```

3871     }
3872   }
3873 </XE>

```

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

```

3874 \AtEndOfPackageFile * { mathtools }
3875 {
3876   \cs_set_eq:NN \MToverbracket \overbracket
3877   \cs_set_eq:NN \MTunderbracket \underbracket
3878
3879   \AtBeginDocument
3880   {
3881     \msg_warning:nn { unicode-math } { mathtools-overbracket }
3882
3883     \def\downbracketfill#1#2
3884     {%

```

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.

```

3885         \edef\l_MT_bracketheight_fdim{.27ex}%
3886         \downbracketend{#1}{#2}
3887         \leaders \vrule \@height #1 \@depth \z@ \hfill
3888         \downbracketend{#1}{#2}%
3889     }
3890 \def\upbracketfill#1#2
3891 {%
3892         \edef\l_MT_bracketheight_fdim{.27ex}%
3893         \upbracketend{#1}{#2}
3894         \leaders \vrule \@height \z@ \@depth #1 \hfill
3895         \upbracketend{#1}{#2}%
3896     }
3897 \let\Uoverbracket =\overbracket
3898 \let\Underbracket=\underbracket
3899     \let\overbracket =\MToverbracket
3900     \let\underbracket =\MTunderbracket
3901 }% end of AtBeginDocument

```

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

```

3902 \msg_warning:nn { unicode-math } { mathtools-colon }
3903 \NewDocumentCommand \dblcolon { } { \Colon }
3904 \NewDocumentCommand \coloneqq { } { \coloneq }
3905 \NewDocumentCommand \Coloneqq { } { \Coloneq }
3906 \NewDocumentCommand \eqqcolon { } { \eqqcolon }
3907 }

```

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`

`\minuscolon` 3908 `\AtEndOfPackageFile * { colonequals }`

`\colonequals` 3909 `{`

`\equalscolon` 3910 `\msg_warning:nn { unicode-math } { colonequals }`

`\coloncolonequals` 3911 `\RenewDocumentCommand \ratio { } { \mathratio }`

3912 `\RenewDocumentCommand \coloncolon { } { \Colon }`

3913 `\RenewDocumentCommand \minuscolon { } { \dashcolon }`

3914 `\RenewDocumentCommand \colonequals { } { \coloneq }`

3915 `\RenewDocumentCommand \equalscolon { } { \eqcolon }`

3916 `\RenewDocumentCommand \coloncolonequals { } { \Coloneq }`

3917 `}`

3918 `</compat>`