Experimental unicode mathematical typesetting: The unicode-math package

Will Robertson

2009/10/11   v0.4

Abstract

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

This package is intended to be a complete implementation of unicode maths for LaTeX using the X$_\exists$TEX (and later, LuaTEX) typesetting engines. With this package, changing maths fonts will be as easy as changing text fonts — not that there are many unicode maths fonts yet.

Maths input is simplified with unicode since literal glyphs may be entered instead of control sequences.

Contents

1 Introduction

This document describes the unicode-math package, which is an *experimental* implementation of a macro to unicode glyph encoding for mathematical characters. Its intended use is for X∃TEX, although it is conjectured that some effect could be spent to create a cross-format package that would also work with LuaTEX.

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

2 Acknowledgements

Many thanks to Microsoft for developing OpenType math as part of Office 2007; Jonathan Kew for implementing unicode math support in X∃TEX; Barbara Beeton for her prodigious effort compiling the definitive list of unicode math glyphs and their LATEX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use TEX in the future. Apostolos Syropoulos, Joel Salomon, and Khaled Hosny have been fantastic beta testers.

3 Getting started

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

```
\usepackage{amsmath} % if desired
\usepackage{unicode-math}
\setmathfont{Cambria Math}
```

3.1 Package options

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

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

Table 1: Package options.

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

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

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

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

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

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

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

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

Table 2: Maths font options.

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

4.1 Using multiple fonts

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

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

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

4.1.1 Control over maths alphabets

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

- `[range=\mathbb]` to use the font for 'bb' letters only.

- `[range=\mathbfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (with `latin`, `Latin`, `num` as well for Latin lower-/upper-case and numbers).

- `[range=\mathsfit->\mathbfsfit]` to map to different output alphabet(s) (which is rather useless right now but will become less useless in the future).

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

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

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

4.2 Script and scriptscript fonts/features

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

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

5 Maths input

X_HT_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, the French (contrary again, *quelle surprise*) have been known to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

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

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

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

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

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

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

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

5.2   Bold style

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

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

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

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

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

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

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

7

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

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

5.3   Sans serif style

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

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

Therefore I reluctantly add the package options [`sans-style=upright`] and [`sans-style=italic`] to control the behaviour of \mathsf. The `upright` style sets up the command to use the seemingly-useless upright sans serif, including Greek; the `italic` style switches to using italic in both Latin and Greek alphabets. In other words, this option simply changes the meaning of \mathsf to either \mathsfup or \mathsfit, respectively. Please let me know if more granular control is necessary here.

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

5.3.1   What about bold sans serif?

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

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

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

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

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

5.4 All (the rest) of the mathematical alphabets

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

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

5.4.1 Double-struck

The double-struck alphabet (also known as 'blackboard bold') consists of upright Latin letters {a–z,A–Z}, numerals 0–9, summation symbol $\sum$, and four Greek letters only: {γ,π,Γ,Π}.

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

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

Table 6: The various forms of nabla.

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

alphabet switch, but note that only those five letters will give the expected output.

5.5 Miscellanea

5.5.1 Nabla

The symbol ∇ comes in the six forms shown in table 6. We want an individual option to specify whether we want upright or italic nabla by default (when either upright or italic nabla is used in the source). TeX classically uses an upright nabla, but ISO standards differ (I think). The package options `nabla=upright` and `nabla=italic` switch between the two choices. This is then inherited through `\mathbf`; `\mathit` and `\mathup` can be used to force one way or the other.

`nabla=italic` is implicit when using `math-style=ISO` and `nabla=upright` follows both `math-style=TeX` and `math-style=French`.

5.5.2 Partial

The same applies to the symbols U+2202: PARTIAL DIFFERENTIAL and U+1D715: MATH ITALIC PARTIAL DIFFERENTIAL.

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

Use the `partial=upright` or `partial=italic` package options to specify which one you would like. The default is (always, unless someone requests and argues otherwise) `partial=italic`.[1]

See table 7 for the variations on the partial differential symbol.

---

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

Table 7: The various forms of the partial differential. Note that in the fonts used to display these glyphs, the first upright partial is incorrectly shown in an italic style.

Description		Glyph
Regular	Upright	∂
	Italic	∂
Bold	Upright	$\boldsymbol{\partial}$
	Italic	$\boldsymbol{\partial}$
Sans bold	Upright	⍰
	Italic	⍰

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

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

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

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

U+3B5: GREEK SMALL LETTER EPSILON
U+3F5: GREEK LUNATE EPSILON SYMBOL
U+3C6: GREEK SMALL LETTER PHI
U+3D5: GREEK SMALL LETTER SCRIPT PHI

5.5.4 Primes

Primes (x') may be input in several ways. You may use any combination of ascii straight quote ('), unicode prime ('), and \prime; when multiple primes occur next to each other, they chain together to form double, triple, or quadruple primes if the font contains pre-drawn glyphs. These may also be accessed with \primedouble, \primetriple, and \primequadruple.

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

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

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

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

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

like this:

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

5.5.5 Unicode subscripts and superscripts

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

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

5.5.6 Colon

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

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

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

To preserve input compatibility, we remap the ASCII input character ':' to U+2236: RATIO. Typing a literal U+2236: RATIO 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.

Table 8: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	\solidus
U+2044	FRACTION SLASH	/	\fracslash
U+2215	DIVISION SLASH	/	\slash
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

The package option [colon=literal] forces ASCII input ':' to be printed as \mathcolon instead.

5.5.7 Slashes and backslashes

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

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

Slash Of U+2044: FRACTION SLASH, TR25 says that it is:

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

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

U+29F8: BIG SOLIDUS is a 'big operator' (like $\sum$).

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

MathML uses U+2216: SET MINUS like this: $A \setminus B$.[2] The LaTeX command name \smallsetminus is used for backwards compatibility.

---

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

Presumably, u+29f5: REVERSE SOLIDUS OPERATOR is intended to be used in a similar way, but it could also (perhaps?) be used to represent 'inverse division': $\pi \approx 7 \setminus 22$.[3] The LaTeX name for this character is `\setminus`.

Finally, u+29f9: BIG REVERSE SOLIDUS is a 'big operator' (like $\sum$).

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bigg/ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \;)$$

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

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

- `\solidus`;

- `\fracslash`;

- `\slash`; and,

- `\backslash` (the only reverse slash).

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

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

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

5.5.8   Normalising some input characters

I believe all variant forms should be used as legal input that is normalised to a consistent output glyph, because we want to be fault-tolerant in the input. Here are the duplicates:

u+251: LATIN SMALL LETTER ALPHA

u+25b: LATIN SMALL LETTER EPSILON

---

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

U+263: LATIN SMALL LETTER GAMMA

U+269: LATIN SMALL LETTER IOTA

U+278: LATIN SMALL LETTER PHI

U+28A: LATIN SMALL LETTER UPSILON

U+190: LATIN CAPITAL LETTER EPSILON

U+194: LATIN CAPITAL LETTER GAMMA

U+196: LATIN CAPITAL LETTER IOTA

U+1B1: LATIN CAPITAL LETTER UPSILON

(Not yet implemented.)

File I
The unicode-math package

This is the package.

```
1  \ProvidesPackage{unicode-math}
2    [2009/10/11 v0.4 Unicode maths in XeLaTeX]
```

6   Things we need

Packages

```
3  \RequirePackage{expl3}[2009/08/12]
4  \RequirePackage{xparse}[2009/08/31]
5  \RequirePackage{fontspec}
```

Start using LaTeX3 — finally!

```
6  \ExplSyntaxOn
```

Extras we need to define:

```
7  \cs_generate_variant:Nn \tl_put_right:Nn {cx}
8  \cs_generate_variant:Nn \seq_if_in:NnTF {NV}
9  \cs_generate_variant:Nn \prop_gput:Nnn {Nxn}
10 \cs_generate_variant:Nn \prop_get:NnN {cxN}
11 \cs_generate_variant:Nn \prop_if_in:NnTF {cx}
```

Counters and conditionals

```
12 \int_new:N \g_um_fam_int
13 \bool_new:N \l_um_fontspec_feature_bool
14 \bool_new:N \l_um_ot_math_bool
15 \bool_new:N \l_um_init_bool
```

For `math-style`:

```
16  \bool_new:N \g_um_literal_bool
17  \bool_new:N \g_um_upLatin_bool
18  \bool_new:N \g_um_uplatin_bool
19  \bool_new:N \g_um_upGreek_bool
20  \bool_new:N \g_um_upgreek_bool
```

For `bold-style`:

```
21  \bool_new:N \g_um_bfliteral_bool
22  \bool_new:N \g_um_bfupLatin_bool
23  \bool_new:N \g_um_bfuplatin_bool
24  \bool_new:N \g_um_bfupGreek_bool
25  \bool_new:N \g_um_bfupgreek_bool
```

For `nabla`:

```
26  \bool_new:N \g_um_upNabla_bool
27  \bool_new:N \g_um_uppartial_bool
28  \bool_new:N \g_um_texgreek_bool
```

6.0.9 Alphabet unicode positions

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

```
29  \def\g_um_up_num_usv{48}
30  \def\g_um_up_Latin_usv{65}
31  \def\g_um_up_latin_usv{97}
32  \def\g_um_up_Greek_usv{"391}
33  \def\g_um_up_greek_usv{"3B1}
34  \def\g_um_it_Latin_usv{"1D434}
35  \def\g_um_it_latin_usv{"1D44E}
36  \def\g_um_it_Greek_usv{"1D6E2}
37  \def\g_um_it_greek_usv{"1D6FC}
38  \def\g_um_bb_num_usv{"1D7D8}
39  \def\g_um_bb_Latin_usv{"1D538}
40  \def\g_um_bb_latin_usv{"1D552}
41  \def\g_um_scr_Latin_usv{"1D49C}
42  \def\g_um_scr_latin_usv{"1D4B6}
43  \def\g_um_frak_Latin_usv{"1D504}
44  \def\g_um_frak_latin_usv{"1D51E}
45  \def\g_um_sf_num_usv{"1D7E2}
46  \def\g_um_sfup_num_usv{"1D7E2}
47  \def\g_um_sfit_num_usv{"1D7E2}
48  \def\g_um_sfup_Latin_usv{"1D5A0}
49  \def\g_um_sf_Latin_usv   {"1D5A0}
50  \def\g_um_sfup_latin_usv{"1D5BA}
```

---

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

```
51  \def\g_um_sf_latin_usv{"1D5BA}
52  \def\g_um_sfit_Latin_usv{"1D608}
53  \def\g_um_sfit_latin_usv{"1D622}
54  \def\g_um_tt_num_usv{"1D7F6}
55  \def\g_um_tt_Latin_usv{"1D670}
56  \def\g_um_tt_latin_usv{"1D68A}
```

Bold:

```
57  \def\g_um_bf_num_usv   {"1D7CE}
58  \def\g_um_bfup_num_usv{"1D7CE}
59  \def\g_um_bfit_num_usv{"1D7CE}
60  \def\g_um_bfup_Latin_usv{"1D400}
61  \def\g_um_bfup_latin_usv{"1D41A}
62  \def\g_um_bfup_Greek_usv{"1D6A8}
63  \def\g_um_bfup_greek_usv{"1D6C2}
64  \def\g_um_bfit_Latin_usv{"1D468}
65  \def\g_um_bfit_latin_usv{"1D482}
66  \def\g_um_bfit_Greek_usv{"1D71C}
67  \def\g_um_bfit_greek_usv{"1D736}
68  \def\g_um_bffrak_Latin_usv{"1D56C}
69  \def\g_um_bffrak_latin_usv{"1D586}
70  \def\g_um_bfscr_Latin_usv{"1D4D0}
71  \def\g_um_bfscr_latin_usv{"1D4EA}
72  \def\g_um_bfsf_num_usv  {"1D7EC}
73  \def\g_um_bfsfup_num_usv{"1D7EC}
74  \def\g_um_bfsfit_num_usv{"1D7EC}
75  \def\g_um_bfsfup_Latin_usv{"1D5D4}
76  \def\g_um_bfsfup_latin_usv{"1D5EE}
77  \def\g_um_bfsfup_Greek_usv{"1D756}
78  \def\g_um_bfsfup_greek_usv{"1D770}
79  \def\g_um_bfsfit_Latin_usv{"1D63C}
80  \def\g_um_bfsfit_latin_usv{"1D656}
81  \def\g_um_bfsfit_Greek_usv{"1D790}
82  \def\g_um_bfsfit_greek_usv{"1D7AA}
83  \def\g_um_bfsf_Latin_usv { \bool_if:NTF \g_um_upLatin_bool \g_um_bfsfup_Latin_usv \g_um_bfsfi
84  \def\g_um_bfsf_latin_usv { \bool_if:NTF \g_um_uplatin_bool \g_um_bfsfup_latin_usv \g_um_bfsfi
85  \def\g_um_bfsf_Greek_usv { \bool_if:NTF \g_um_upGreek_bool \g_um_bfsfup_Greek_usv \g_um_bfsfi
86  \def\g_um_bfsf_greek_usv { \bool_if:NTF \g_um_upgreek_bool \g_um_bfsfup_greek_usv \g_um_bfsfi
87  \def\g_um_bf_Latin_usv { \bool_if:NTF \g_um_bfupLatin_bool \g_um_bfup_Latin_usv \g_um_bfit_La
88  \def\g_um_bf_latin_usv { \bool_if:NTF \g_um_bfuplatin_bool \g_um_bfup_latin_usv \g_um_bfit_la
89  \def\g_um_bf_Greek_usv { \bool_if:NTF \g_um_bfupGreek_bool \g_um_bfup_Greek_usv \g_um_bfit_Gr
90  \def\g_um_bf_greek_usv { \bool_if:NTF \g_um_bfupgreek_bool \g_um_bfup_greek_usv \g_um_bfit_gr
```

Greek variants:

```
91  \def\g_um_up_varTheta_usv{"3F4}
92  \def\g_um_up_Digamma_usv{"3DC}
93  \def\g_um_up_varepsilon_usv{"3F5}
94  \def\g_um_up_vartheta_usv{"3D1}
```

```
95  \def\g_um_up_varkappa_usv{"3F0}
96  \def\g_um_up_varphi_usv{"3D5}
97  \def\g_um_up_varrho_usv{"3F1}
98  \def\g_um_up_varpi_usv{"3D6}
99  \def\g_um_up_digamma_usv{"3DD}
```

Bold:

```
100  \def\g_um_bfup_varTheta_usv{"1D6B9}
101  \def\g_um_bfup_Digamma_usv{"1D7CA}
102  \def\g_um_bfup_varepsilon_usv{"1D6DC}
103  \def\g_um_bfup_vartheta_usv{"1D6DD}
104  \def\g_um_bfup_varkappa_usv{"1D6DE}
105  \def\g_um_bfup_varphi_usv{"1D6DF}
106  \def\g_um_bfup_varrho_usv{"1D6E0}
107  \def\g_um_bfup_varpi_usv{"1D6E1}
108  \def\g_um_bfup_digamma_usv{"1D7CB}
```

Italic Greek variants:

```
109  \def\g_um_it_varTheta_usv{"1D6F3}
110  \def\g_um_it_varepsilon_usv{"1D716}
111  \def\g_um_it_vartheta_usv{"1D717}
112  \def\g_um_it_varkappa_usv{"1D718}
113  \def\g_um_it_varphi_usv{"1D719}
114  \def\g_um_it_varrho_usv{"1D71A}
115  \def\g_um_it_varpi_usv{"1D71B}
```

Bold italic:

```
116  \def\g_um_bfit_varTheta_usv{"1D72D}
117  \def\g_um_bfit_varepsilon_usv{"1D750}
118  \def\g_um_bfit_vartheta_usv{"1D751}
119  \def\g_um_bfit_varkappa_usv{"1D752}
120  \def\g_um_bfit_varphi_usv{"1D753}
121  \def\g_um_bfit_varrho_usv{"1D754}
122  \def\g_um_bfit_varpi_usv{"1D755}
```

Bold sans:

```
123  \def\g_um_bfsfup_varTheta_usv{"1D767}
124  \def\g_um_bfsfup_varepsilon_usv{"1D78A}
125  \def\g_um_bfsfup_vartheta_usv{"1D78B}
126  \def\g_um_bfsfup_varkappa_usv{"1D78C}
127  \def\g_um_bfsfup_varphi_usv{"1D78D}
128  \def\g_um_bfsfup_varrho_usv{"1D78E}
129  \def\g_um_bfsfup_varpi_usv{"1D78F}
```

Bold sans italic:

```
130  \def\g_um_bfsfit_varTheta_usv{"1D7A1}
131  \def\g_um_bfsfit_varepsilon_usv{"1D7C4}
132  \def\g_um_bfsfit_vartheta_usv{"1D7C5}
133  \def\g_um_bfsfit_varkappa_usv{"1D7C6}
```

```
134  \def\g_um_bfsfit_varphi_usv{"1D7C7}
135  \def\g_um_bfsfit_varrho_usv{"1D7C8}
136  \def\g_um_bfsfit_varpi_usv{"1D7C9}
```

Nabla:

```
137  \def\g_um_up_Nabla_usv{"2207}
138  \def\g_um_it_Nabla_usv{"1D6FB}
139  \def\g_um_bfup_Nabla_usv{"1D6C1}
140  \def\g_um_bfit_Nabla_usv{"1D735}
141  \def\g_um_bfsfup_Nabla_usv{"1D76F}
142  \def\g_um_bfsfit_Nabla_usv{"1D7A9}
```

Partial:

```
143  \def\g_um_up_partial_usv{"2202}
144  \def\g_um_it_partial_usv{"1D715}
145  \def\g_um_bfup_partial_usv{"1D6DB}
146  \def\g_um_bfit_partial_usv{"1D74F}
147  \def\g_um_bfsfup_partial_usv{"1D789}
148  \def\g_um_bfsfit_partial_usv{"1D7C3}
```

Latin 'h':

```
149  \def\g_um_up_h_usv   {"0068}
150  \def\g_um_it_h_usv   {"210E}
151  \def\g_um_bb_h_usv   {"1D559}
152  \def\g_um_tt_h_usv   {"1D691}
153  \def\g_um_scr_h_usv  {"1D4BD}
154  \def\g_um_frak_h_usv{"1D525}
155  \def\g_um_bfup_h_usv{"1D421}
156  \def\g_um_bfit_h_usv{"1D489}
157  \def\g_um_sfup_h_usv{"1D5C1}
158  \def\g_um_sfit_h_usv{"1D629}
159  \def\g_um_bffrak_h_usv{"1D58D}
160  \def\g_um_bfscr_h_usv  {"1D4F1}
161  \def\g_um_bfsfup_h_usv {"1D5F5}
162  \def\g_um_bfsfit_h_usv {"1D65D}
```

6.1   Options

xkeyval's package support is used here. I'll switch over to l3keys2e at some stage.

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

```
163  \DeclareDocumentCommand \unimathsetup {m} {
164    \setkeys{unicode-math.sty}{#1}
165  }
```

19

math-style

```
166  \define@choicekey*{unicode-math.sty}
167      {math-style}[\@tempa\@tempb]{iso,tex,french,upright,literal}{
168  \ifcase\@tempb\relax
169      \bool_set_false:N \g_um_upGreek_bool
170      \bool_set_false:N \g_um_upgreek_bool
171      \bool_set_false:N \g_um_upLatin_bool
172      \bool_set_false:N \g_um_uplatin_bool
173      \bool_set_false:N \g_um_bfupGreek_bool
174      \bool_set_false:N \g_um_bfupgreek_bool
175      \bool_set_false:N \g_um_bfupLatin_bool
176      \bool_set_false:N \g_um_bfuplatin_bool
177      \bool_set_false:N \g_um_upNabla_bool
178      \bool_set_false:N \g_um_uppartial_bool
179      \bool_set_false:N \g_um_upsans_bool
180      \bool_set_false:N \g_um_texgreek_bool
181      \bool_set_false:N \g_um_literal_bool
182  \or
183      \bool_set_true:N \g_um_upGreek_bool
184      \bool_set_false:N \g_um_upgreek_bool
185      \bool_set_false:N \g_um_upLatin_bool
186      \bool_set_false:N \g_um_uplatin_bool
187      \bool_set_true:N \g_um_bfupGreek_bool
188      \bool_set_false:N \g_um_bfupgreek_bool
189      \bool_set_true:N \g_um_bfupLatin_bool
190      \bool_set_true:N \g_um_bfuplatin_bool
191      \bool_set_true:N \g_um_upNabla_bool
192      \bool_set_false:N \g_um_uppartial_bool
193      \bool_set_true:N \g_um_upsans_bool
194      \bool_set_false:N \g_um_texgreek_bool
195      \bool_set_false:N \g_um_literal_bool
196  \or
197      \bool_set_true:N \g_um_upGreek_bool
198      \bool_set_true:N \g_um_upgreek_bool
199      \bool_set_true:N \g_um_upLatin_bool
200      \bool_set_false:N \g_um_uplatin_bool
201      \bool_set_true:N \g_um_bfupGreek_bool
202      \bool_set_true:N \g_um_bfupgreek_bool
203      \bool_set_true:N \g_um_bfupLatin_bool
204      \bool_set_true:N \g_um_bfuplatin_bool
205      \bool_set_true:N \g_um_upNabla_bool
206      \bool_set_true:N \g_um_uppartial_bool
207      \bool_set_true:N \g_um_upsans_bool
208      \bool_set_false:N \g_um_texgreek_bool
209      \bool_set_false:N \g_um_literal_bool
210  \or
```

```
211     \bool_set_true:N \g_um_upGreek_bool
212     \bool_set_true:N \g_um_upgreek_bool
213     \bool_set_true:N \g_um_upLatin_bool
214     \bool_set_true:N \g_um_uplatin_bool
215     \bool_set_true:N \g_um_bfupGreek_bool
216     \bool_set_true:N \g_um_bfupgreek_bool
217     \bool_set_true:N \g_um_bfupLatin_bool
218     \bool_set_true:N \g_um_bfuplatin_bool
219     \bool_set_true:N \g_um_upNabla_bool
220     \bool_set_true:N \g_um_uppartial_bool
221     \bool_set_true:N \g_um_upsans_bool
222     \bool_set_false:N \g_um_texgreek_bool
223     \bool_set_false:N \g_um_literal_bool
224   \or
225     \bool_set_true:N \g_um_literal_bool
226     \bool_set_true:N \g_um_bfliteral_bool
227     \bool_set_true:N \g_um_sfliteral_bool
228     \bool_set_false:N \g_um_texgreek_bool
229   \fi
230 }
```

bold-style

```
231 \define@choicekey*{unicode-math.sty}{bold-style}[\@tempa\@tempb]{iso,tex,upright,literal}{
232   \ifcase\@tempb\relax
233     \bool_set_false:N \g_um_bfliteral_bool
234     \bool_set_false:N \g_um_bfupGreek_bool
235     \bool_set_false:N \g_um_bfupgreek_bool
236     \bool_set_false:N \g_um_bfupLatin_bool
237     \bool_set_false:N \g_um_bfuplatin_bool
238   \or
239     \bool_set_false:N \g_um_bfliteral_bool
240     \bool_set_true:N \g_um_bfupGreek_bool
241     \bool_set_false:N \g_um_bfupgreek_bool
242     \bool_set_true:N \g_um_bfupLatin_bool
243     \bool_set_true:N \g_um_bfuplatin_bool
244   \or
245     \bool_set_false:N \g_um_bfliteral_bool
246     \bool_set_true:N \g_um_bfupGreek_bool
247     \bool_set_true:N \g_um_bfupgreek_bool
248     \bool_set_true:N \g_um_bfupLatin_bool
249     \bool_set_true:N \g_um_bfuplatin_bool
250   \or
251     \bool_set_true:N \g_um_bfliteral_bool
252   \fi
253 }
```

sans-style

```
254  \bool_new:N \g_um_upsans_bool
255  \bool_new:N \g_um_sfliteral_bool
256  \define@choicekey*{unicode-math.sty}
257      {sans-style}[\@tempa\@tempb]{italic,upright,literal}{
258    \ifcase\@tempb\relax
259      \bool_set_false:N \g_um_upsans_bool
260    \or
261      \bool_set_true:N \g_um_upsans_bool
262    \or
263      \bool_set_true:N \g_um_sfliteral_bool
264    \fi
265  }
```

Symbol obliqueness

```
266  \define@choicekey*{unicode-math.sty}{nabla}[\@tempa\@tempb]{upright,italic}{
267    \ifcase\@tempb
268      \bool_set_true:N \g_um_upNabla_bool
269    \or
270      \bool_set_false:N \g_um_upNabla_bool
271    \fi
272  }
273  \cs_set:Nn \um_setup_nabla: {
274    \bool_if:NTF \g_um_upNabla_bool {
275      \tl_set:Nn \g_um_Nabla_up_or_it_usv     { \g_um_up_Nabla_usv }
276      \tl_set:Nn \g_um_bfNabla_up_or_it_usv   { \g_um_bfup_Nabla_usv }
277      \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfup_Nabla_usv }
278    }{
279      \tl_set:Nn \g_um_Nabla_up_or_it_usv     { \g_um_it_Nabla_usv }
280      \tl_set:Nn \g_um_bfNabla_up_or_it_usv   { \g_um_bfit_Nabla_usv }
281      \tl_set:Nn \g_um_bfsfNabla_up_or_it_usv { \g_um_bfsfit_Nabla_usv }
282    }
283  }
284  \define@choicekey*{unicode-math.sty}{partial}[\@tempa\@tempb]{upright,italic}{
285    \ifcase\@tempb
286      \bool_set_true:N \g_um_uppartial_bool
287    \or
288      \bool_set_false:N \g_um_uppartial_bool
289    \fi
290  }
291  \cs_set:Nn \um_setup_partial: {
292    \bool_if:NTF \g_um_uppartial_bool {
293      \tl_set:Nn \g_um_partial_up_or_it_usv     { \g_um_up_partial_usv }
294      \tl_set:Nn \g_um_bfpartial_up_or_it_usv   { \g_um_bfup_partial_usv }
295      \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfup_partial_usv }
```

```
296   }{
297     \tl_set:Nn \g_um_partial_up_or_it_usv      { \g_um_it_partial_usv }
298     \tl_set:Nn \g_um_bfpartial_up_or_it_usv    { \g_um_bfit_partial_usv }
299     \tl_set:Nn \g_um_bfsfpartial_up_or_it_usv { \g_um_bfsfit_partial_usv }
300   }
301 }
```

Epsilon and phi shapes

```
302 \define@choicekey*{unicode-math.sty}{vargreek-shape}[\@tempa\@tempb]{unicode,TeX}{
303   \ifcase\@tempb
304     \bool_set_false:N \g_um_texgreek_bool
305   \or
306     \bool_set_true:N \g_um_texgreek_bool
307   \fi
308 }
```

Colon style

```
309 \bool_new:N \g_um_literal_colon_bool
310 \define@choicekey*{unicode-math.sty}{colon}[\@tempa\@tempb]{literal,TeX}{
311   \ifcase\@tempb
312     \bool_set_true:N \g_um_literal_colon_bool
313   \or
314     \bool_set_false:N \g_um_literal_colon_bool
315   \fi
316 }
```

Slash delimiter style

```
317 \define@choicekey*{unicode-math.sty}{slash-delimiter}[\@tempa\@tempb]{ascii,frac,div}{
318   \ifcase\@tempb
319     \tl_set:Nn \g_um_slash_delimiter_usv {"002F}
320   \or
321     \tl_set:Nn \g_um_slash_delimiter_usv {"2044}
322   \or
323     \tl_set:Nn \g_um_slash_delimiter_usv {"2215}
324   \fi
325 }
326 \ExecuteOptionsX{math-style=TeX,slash-delimiter=ascii}
327 \ProcessOptionsX
```

6.2 Overcoming \@onlypreamble

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

```
328 \tl_map_inline:nn {
```

```
329  \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
330  \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
331  \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
332  \version@list\version@elt\alpha@list\alpha@elt
333  \restore@mathversion\init@restore@version\dorestore@version\process@table
334  \new@mathversion\DeclareSymbolFont\group@list\group@elt
335  \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
336  \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
337  \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
338  \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter\@DeclareMathDelimiter
339  \@xDeclareMathDelimiter\set@mathdelimiter\set@@mathdelimiter\DeclareMathRadical
340  \mathchar@type\DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
341  }{
342    \tl_remove_in:Nn \@preamblecmds {\do#1}
343  }
```

6.3 Other things

`\um_fontdimen_to_percent:nn`

#1 : Font dimen number

\fontdimens 10, 11, and 65 aren't actually dimensions, they're percentage values given in units of sp. This macro takes a font dimension number and outputs the decimal value of the associated parameter.

```
344  \def\um_fontdimen_to_percent:nn#1#2{
345    0.\strip@pt\dimexpr\fontdimen#1#2 *65536\relax
346  }
```

`\um@scaled@apply`

#1 : A math style

#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.

```
347  \def\um@scaled@apply#1#2#3{
348    \ifx#1\scriptstyle
349      #2\um_fontdimen_to_percent:nn{10}\l_um_font#3
350    \else
351      \ifx#1\scriptscriptstyle
352        #2\um_fontdimen_to_percent:nn{11}\l_um_font#3
353      \else
354        #2#3%
355      \fi
356    \fi
357  }
```

24

7 Fundamentals

7.1 Enlarging the number of maths families

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

```
358  \def\new@mathgroup{\alloc@8\mathgroup\chardef\@cclvi}
359  \let\newfam\new@mathgroup
```

This is sufficient for LaTeX's \DeclareSymbolFont-type commands to be able to define 256 named maths fonts. Now we need a new \DeclareMathSymbol.

7.2 \DeclareMathSymbol for unicode ranges

This command is a bit funny at the moment; it doesn't define the actual macro for almost all of the symbols passed to it, but it does assign the \XeTeXmathchar.

The final macros that actually define the maths symbol with X∃TEX primitives.

\um_set_mathsymbol:nNNn #1 : Symbol font number, e.g., \symoperators
#2 : Symbol macro, *e.g.,* \alpha
#3 : Type, *e.g.,* \mathalpha
#4 : Slot, *e.g.,* "221E

If the symbol definition is for a macro. There are a bunch of tests to perform to process the various characters.

```
360  \cs_set:Nn \um_set_mathsymbol:nNNn {
```

Operators In the examples following, say we're defining for the symbol \sum(∑).

```
361    \ifx\mathop#3\relax
```

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.

The active math char is defined to expand to the macro \sum_sym.

```
362      \begingroup
363        \char_make_active:n {#4}
364        \global\mathcode#4="8000\relax
365        \um@scanactivedef #4 \@nil { \csname\cs_to_str:N #2 _sym\endcsname }
366      \endgroup
```

Some of these require a \nolimits suffix. This is controlled by the \um@nolimits macro, which contains a list of such characters. This list is checked dynamically because we're not interested in efficiency. Or something. This allows the list to be updated in the middle of a document.

Declare the plain old mathchardef for the control sequence \sumop.

```
367      \expandafter\global\expandafter\XeTeXmathchardef
368        \csname\cs_to_str:N #2 op\endcsname ="\mathchar@type#3 #1 #4\relax
```

Now define `\sum_sym` as `\sumop`, followed by `\nolimits` if necessary.

```
369      \cs_gset:cpx { \cs_to_str:N #2 _sym } {
370        \exp_not:c {\cs_to_str:N #2 op}
371        \exp_not:n {\tl_if_in:NnT \l_um_nolimits_tl {#2} \nolimits}
372      }
```

Don't forget that the actual `\sum` macro is simply defined in terms of the literal unicode symbol!

```
373      \else
```

Delimiters and radicals   Sqrt radical is defined as a csmathopen.

```
374      \ifx\mathopen#3\relax
375        \tl_if_in:NnTF \l_um_radicals_tl #2 {
376          \cs_gset:cpn {\cs_to_str:N #2 sign} { \XeTeXradical #1 #4 \relax }
377        }{
378          \cs_gset:Npn #2 {\XeTeXdelimiter "\mathchar@type#3 #1 #4\relax}
379          \global\XeTeXdelcode#4=#1 #4\relax
380          \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4\relax
381        }
382      \else
383        \ifx\mathclose#3\relax
384          \cs_gset:Npn #2 {\XeTeXdelimiter "\mathchar@type#3 #1 #4\relax}
385          \global\XeTeXdelcode#4=#1 #4\relax
386          \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4\relax
387        \else
```

Accents

```
388          \ifx\mathaccent#3\relax
389            \cs_gset:Npx #2 {\XeTeXmathaccent "\mathchar@type#3 #1 #4\relax}
390          \else
```

And finally, the general case. We define the unicode mathcode for the character. The macro is defined later on generically in terms of the unicode character.

```
391            \global\XeTeXmathcode#4="\mathchar@type#3 #1 #4\relax
392          \fi
393        \fi
394      \fi
395    \fi
396  }
```

`\um_set_mathcode:nnnn`   Note that this declaration *isn't* global so that it can be constrained by grouping inside math alphabet switches.

```
397  \cs_set:Nn \um_set_mathcode:nnnn {
398    \XeTeXmathcode#1="\mathchar@type#2 \csname sym#3\endcsname #4\relax
399  }
```

7.3 The main \setmathfont macro

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

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

```
400  \DeclareDocumentCommand \setmathfont { O{} m } {
```

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

```
401       \let\glb@currsize\relax
```

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

```
402       \bool_set_true:N \l_um_init_bool
403       \seq_clear:N \l_um_char_range_seq
404       \let\um@char@num@range\@empty
```

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

```
405       \csname S@\f@size\endcsname
```

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

```
406   \tl_set:Nn \l_um_mversion_tf {normal}
407   \DeclareMathVersion{\l_um_mversion_tf}
```

Define default font features for the script and scriptscript font.

```
408   \tl_set:Nn \l_um_script_features_tl  {ScriptStyle}
409   \tl_set:Nn \l_um_sscript_features_tl {ScriptScriptStyle}
410   \tl_set:Nn \l_um_script_font_tl      {#2}
411   \tl_set:Nn \l_um_sscript_font_tl     {#2}
```

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

```
412   \setkeys*{unicode-math.sty}{#1}
413   \cs_set:Npx \um_tmp: {
414     \exp_not:N \setkeys*[um]{options}{\exp_not:V \XKV@rm}
415   }
416   \um_tmp:
417   \cs_set:Npx \um_tmp: {
```

```
418    \exp_not:N \zf@fontspec {
419      BoldFont = {}, ItalicFont = {},
420      Script = Math,
421      SizeFeatures = {
422        {Size = \tf@size-} ,
423        {Size = \sf@size-\tf@size ,
424         Font = \l_um_script_font_tl ,
425         \l_um_script_features_tl
426        } ,
427        {Size = -\sf@size ,
428         Font = \l_um_sscript_font_tl ,
429         \l_um_sscript_features_tl
430        }
431      },
432      \XKV@rm
433    }{#2}
434  }
435  \bool_set_true:N \l_um_fontspec_feature_bool
436  \um_tmp:
437  \bool_set_false:N \l_um_fontspec_feature_bool
```

Check for the correct number of \fontdimens:

```
438    \font\l_um_font="#2"\relax
439 %%  \ifdim \dimexpr\fontdimen9\l_um_font*65536\relax =65pt\relax
440 %%    \bool_set_true:N \l_um_ot_math_bool
441 %%  \else
442 %%    \bool_set_false:N \l_um_ot_math_bool
443 %%    \PackageWarningNoLine{unicode-math}{
444 %%      The~ font~ '#2' ~is~ not~ a~ valid~ OpenType~ maths~ font.~
445 %%      Some~ maths~ features~ will~ not~ be~ available~ or~ behave~
446 %%      in~ a~ substandard~ manner
447 %%    }
448 %%  \fi
```

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

- Math symbols are defined with \UnicodeMathSymbol; see section §7.3.1 for
  the individual definitions

```
449   \bool_if:NTF \l_um_init_bool {
450     \tl_set:Nn \um_symfont_tl {um_allsym}
451   \PackageInfo{unicode-math}{Defining~ the~ default~ maths~ font~ as~ '#2'}
452     \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_noparse:nnnn
453     \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
454     \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_noparse:nnn
455     \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
456     \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_noparse:nn
```

```
457    }{
458      \int_incr:N \g_um_fam_int
459      \tl_set:Nx \um_symfont_tl {um_fam\int_use:N\g_um_fam_int}
460      \cs_set_eq:NN \UnicodeMathSymbol \um_process_symbol_parse:nnnn
461      \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_parse:Nnn
462      \cs_set_eq:NN \um_remap_symbol:nnn \um_remap_symbol_parse:nnn
463      \cs_set_eq:NN \um_maybe_init_alphabet:n \use_none:n
464      \cs_set_eq:NN \um_map_char_internal:nn \um_map_char_parse:nn
465    }
```

Now defined \um_symfont_tl as the LATEX math font to access everything:

```
466    \DeclareSymbolFont{\um_symfont_tl}
467      {\encodingdefault}{\zf@family}{\mddefault}{\updefault}
```

And now we input every single maths char. See File 12 for the source to `unicode-math.tex` which is used to create `unicode-math-table.tex`.

```
468    \@input{unicode-math-table.tex}
```

Finally,

- Set up shapes for italic/upright or ordinary/var symbols as per package options.

- Remap symbols that don't take their natural mathcode

- Activate any symbols that need to be math-active

- Assign delimiter codes for symbols that need to grow

- Setup the maths alphabets (\mathbf etc.)

```
469    \um_setup_nabla:
470    \um_setup_partial:
471    \um_remap_symbols:
472    \um_setup_mathactives:
473    \um_setup_delcodes:
474    \um_setup_alphabets:
475  }
```

7.3.1 Functions for setting up symbols with mathcodes

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

```
476  \cs_set:Nn \um_process_symbol_noparse:nnnn {
477    \exp_args:Nc \um_set_mathsymbol:nNNn {sym\um_symfont_tl}#2#3{#1}
478  }
479  \cs_set:Nn \um_process_symbol_parse:nnnn {
480    \um@parse@term{#1}{#2}{#3}{
481      \um_process_symbol_noparse:nnnn{#1}{#2}{#3}{#4}
```

```
482     }
483   }
```

`\um_remap_symbols:`
`\um_remap_symbol_noparse:nnn`
`\um_remap_symbol_parse:nnn`

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

```
484 \cs_new:Nn \um_remap_symbols: {
485   \um_remap_symbol:nnn{`\-}{\mathbin}{"02212}% hyphen to minus
486    \um_remap_symbol:nnn{`\*}{\mathbin}{"02217}% text asterisk to "cen-
   tred asterisk"
487   \bool_if:NF \g_um_literal_colon_bool {
488    \um_remap_symbol:nnn{`\:}{\mathrel}{"02236}% colon to ratio (i.e., punct to rel)
489   }
490   \bool_if:NTF \g_um_literal_bool {
491    \um_remap_symbol:nnn {\g_um_up_Nabla_usv}{\mathord}{\g_um_up_Nabla_usv}
492    \um_remap_symbol:nnn {\g_um_it_Nabla_usv}{\mathord}{\g_um_it_Nabla_usv}
493    \um_remap_symbol:nnn {\g_um_up_partial_usv}{\mathord}{\g_um_up_partial_usv}
494    \um_remap_symbol:nnn {\g_um_it_partial_usv}{\mathord}{\g_um_it_partial_usv}
495   }{
496    \um_remap_symbol:nnn {\g_um_up_Nabla_usv,\g_um_it_Nabla_usv}{\mathord}{\g_um_Nabla_up_or_
497    \um_remap_symbol:nnn {\g_um_up_partial_usv,\g_um_it_partial_usv}{\mathord}{\g_um_partial_
498   }
```

Some of these in the `bfliteral` block may be redundant, but that's okay:

```
499   \bool_if:NTF \g_um_bfliteral_bool {
500    \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv   }{\mathord}{\g_um_bfup_Nabla_usv}
501    \um_remap_symbol:nnn {\g_um_bfit_Nabla_usv   }{\mathord}{\g_um_bfit_Nabla_usv}
502    \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv }{\mathord}{\g_um_bfsfup_Nabla_usv}
503    \um_remap_symbol:nnn {\g_um_bfsfit_Nabla_usv }{\mathord}{\g_um_bfsfit_Nabla_usv}
504    \um_remap_symbol:nnn {\g_um_bfup_partial_usv }{\mathord}{\g_um_bfup_partial_usv}
505    \um_remap_symbol:nnn {\g_um_bfit_partial_usv }{\mathord}{\g_um_bfit_partial_usv}
506    \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv}{\mathord}{\g_um_bfsfup_partial_usv}
507    \um_remap_symbol:nnn {\g_um_bfsfit_partial_usv}{\mathord}{\g_um_bfsfit_partial_usv}
508   }{
509    \um_remap_symbol:nnn {\g_um_bfup_Nabla_usv,\g_um_bfit_Nabla_usv}{\mathord}{\g_um_bfNabla_
510    \um_remap_symbol:nnn {\g_um_bfsfup_Nabla_usv,\g_um_bfsfit_Nabla_usv}{\mathord}{\g_um_bfsf
511    \um_remap_symbol:nnn {\g_um_bfup_partial_usv,\g_um_bfit_partial_usv}{\mathord}{\g_um_bfpa
512    \um_remap_symbol:nnn {\g_um_bfsfup_partial_usv,\g_um_bfsfit_partial_usv}{\mathord}{\g_um_
513   }
514 }
```

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

```
515 \cs_new:Nn \um_remap_symbol_parse:nnn {
516   \um@parse@term {#3} {\@nil} {#2} {
517     \um_remap_symbol_noparse:nnn {#1} {#2} {#3}
518   }
519 }
```

```
520 \cs_new:Nn \um_remap_symbol_noparse:nnn {
521   \clist_map_inline:nn {#1} {
522     \um_set_mathcode:nnnn {##1} {#2} {\um_symfont_tl} {#3}
523   }
524 }
```

7.3.2 Active math characters

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

`\um_setup_mathactives:`

```
525 \cs_new:Nn \um_setup_mathactives: {
526   \um_make_mathactive:nNN {"2032} \primesingle \mathord
527 }
```

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

```
528 \cs_new:Nn \um_make_mathactive:nNN {
529   \XeTeXmathchardef #2 = "\mathchar@type #3
530                           \csname sym\um_symfont_tl\endcsname
531                           #1 \scan_stop:
532   \XeTeXmathcodenum #1 = "1FFFFF \scan_stop:
533 }
```

7.3.3 Delimiter codes

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.

`\um_setup_delcodes:`

```
534 \cs_new:Nn \um_setup_delcodes: {
535   \um_set_delcode:nn {`\/}   {\g_um_slash_delimiter_usv}
536   \um_set_delcode:nn {"2044} {\g_um_slash_delimiter_usv} % fracslash
537   \um_set_delcode:nn {"2215} {\g_um_slash_delimiter_usv} % divslash
538   \um_set_delcode:n {"005C} % backslash
539   \um_set_delcode:nn {`\<} {"27E8} % angle brackets with ascii notation
540   \um_set_delcode:nn {`\>} {"27E9} % angle brackets with ascii notation
541   \um_set_delcode:n {"2191} % up arrow
542   \um_set_delcode:n {"2193} % down arrow
543   \um_set_delcode:n {"2195} % updown arrow
544   \um_set_delcode:n {"219F} % up arrow twohead
545   \um_set_delcode:n {"21A1} % down arrow twohead
546   \um_set_delcode:n {"21A5} % up arrow from bar
```

```
547   \um_set_delcode:n {"21A7} % down arrow from bar
548   \um_set_delcode:n {"21A8} % updown arrow from bar
549   \um_set_delcode:n {"21BE} % up harpoon right
550   \um_set_delcode:n {"21BF} % up harpoon left
551   \um_set_delcode:n {"21C2} % down harpoon right
552   \um_set_delcode:n {"21C3} % down harpoon left
553   \um_set_delcode:n {"21C5} % arrows up down
554   \um_set_delcode:n {"21F5} % arrows down up
555   \um_set_delcode:n {"21C8} % arrows up up
556   \um_set_delcode:n {"21CA} % arrows down down
557   \um_set_delcode:n {"21D1} % double up arrow
558   \um_set_delcode:n {"21D3} % double down arrow
559   \um_set_delcode:n {"21D5} % double updown arrow
560   \um_set_delcode:n {"21DE} % up arrow double stroke
561   \um_set_delcode:n {"21DF} % down arrow double stroke
562   \um_set_delcode:n {"21E1} % up arrow dashed
563   \um_set_delcode:n {"21E3} % down arrow dashed
564 }
```

\um_setup_delcodes: : TODO : hook into range feature

```
565 \cs_new:Nn \um_set_delcode:nn {
566   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #2
567 }
568 \cs_new:Nn \um_set_delcode:n {
569   \XeTeXdelcode#1 = \csname sym\um_symfont_tl\endcsname #1
570 }
```

7.3.4 Maths alphabets' character mapping

7.3.5 Functions for setting up the maths alphabets

\um_mathmap_noparse:Nnn   #1 : Maths alphabet, *e.g.*, \mathbb
                          #2 : Input slot(s), *e.g.*, the slot for 'A' (comma separated)
                          #3 : Output slot, *e.g.*, the slot for 'A'
Adds \um_set_mathcode:nnnn declarations to the specified maths alphabet's def-
inition.

```
571 \cs_set:Nn \um_mathmap_noparse:Nnn {
572   \clist_map_inline:nn {#2} {
573     \tl_put_right:cx {um_setup_\cs_to_str:N #1:} {
574     \exp_not:N\um_set_mathcode:nnnn{##1}{\exp_not:N\mathalpha}{\um_symfont_tl}{#3}
575     }
576   }
577 }
```

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

#3 : Output slot, *e.g.*, the slot for '𝔸'

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

```
578  \cs_set:Nn \um_mathmap_parse:Nnn {
579    \clist_map_inline:Nn \um@char@num@range {
580      \ifnum##1=#3\relax
581        \um_mathmap_noparse:Nnn {#1}{#2}{#3}
582      \fi
583    }
584  }
```

7.4 (Big) operators

Turns out that X⅁TEX is clever enough to deal with big operators for us automatically with \XeTeXmathchardef. 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 \UnicodeMathSymbol in the appropriate contexts.

Following is a table of every math operator (\mathop) defined in unicode-math-table.tex, from which a subset need to be flagged for \nolimits adjustments. The limits behaviour as specified by unicode-math are shown (with grey 'scripts).

USV	Ex.	Macro	Description
U+02140	$\sum$	\Bbbsum	DOUBLE-STRUCK N-ARY SUMMATION
U+0220F	$\prod$	\prod	PRODUCT OPERATOR
U+02210	$\coprod$	\coprod	COPRODUCT OPERATOR
U+02211	$\sum$	\sum	SUMMATION OPERATOR
U+0222B	$\int$	\int	INTEGRAL OPERATOR
U+0222C	$\iint$	\iint	DOUBLE INTEGRAL OPERATOR
U+0222D	$\iiint$	\iiint	TRIPLE INTEGRAL OPERATOR
U+0222E	$\oint$	\oint	CONTOUR INTEGRAL OPERATOR
U+0222F	$\oiint$	\oiint	DOUBLE CONTOUR INTEGRAL OPERATOR
U+02230	$\oiiint$	\oiiint	TRIPLE CONTOUR INTEGRAL OPERATOR
U+02231	$\intclockwise$	\intclockwise	CLOCKWISE INTEGRAL

33

U+02232	$\oint$	\varointclockwise	CONTOUR INTEGRAL, CLOCKWISE
U+02233	$\oint$	\ointctrclockwise	CONTOUR INTEGRAL, ANTICLOCKWISE
U+022C0	$\bigwedge$	\bigwedge	LOGICAL OR OPERATOR
U+022C1	$\bigvee$	\bigvee	LOGICAL AND OPERATOR
U+022C2	$\bigcap$	\bigcap	INTERSECTION OPERATOR
U+022C3	$\bigcup$	\bigcup	UNION OPERATOR
U+027D5	$\bowtie$	\leftouterjoin	LEFT OUTER JOIN
U+027D6	$\bowtie$	\rightouterjoin	RIGHT OUTER JOIN
U+027D7	$\bowtie$	\fullouterjoin	FULL OUTER JOIN
U+027D8	$\bot$	\bigbot	LARGE UP TACK
U+027D9	$\top$	\bigtop	LARGE DOWN TACK
U+029F8	$/$	\xsol	BIG SOLIDUS
U+029F9	$\backslash$	\xbsol	BIG REVERSE SOLIDUS
U+02A00	$\bigodot$	\bigodot	N-ARY CIRCLED DOT OPERATOR
U+02A01	$\bigoplus$	\bigoplus	N-ARY CIRCLED PLUS OPERATOR
U+02A02	$\bigotimes$	\bigotimes	N-ARY CIRCLED TIMES OPERATOR
U+02A03	$\biguplus$	\bigcupdot	N-ARY UNION OPERATOR WITH DOT
U+02A04	$\biguplus$	\biguplus	N-ARY UNION OPERATOR WITH PLUS
U+02A05	$\bigsqcap$	\bigsqcap	N-ARY SQUARE INTERSECTION OPERATOR
U+02A06	$\bigsqcup$	\bigsqcup	N-ARY SQUARE UNION OPERATOR
U+02A07	$\bigwedge$	\conjquant	TWO LOGICAL AND OPERATOR
U+02A08	$\bigvee$	\disjquant	TWO LOGICAL OR OPERATOR
U+02A09	$\bigtimes$	\bigtimes	N-ARY TIMES OPERATOR

U+02A0B	$\int_0^1$	\sumint	SUMMATION WITH INTEGRAL
U+02A0C	$\iiiint_0^1$	\iiiint	QUADRUPLE INTEGRAL OPERATOR
U+02A0D	$\int_0^1$	\intbar	FINITE PART INTEGRAL
U+02A0E	$\int_0^1$	\intBar	INTEGRAL WITH DOUBLE STROKE
U+02A0F	$\int_0^1$	\fint	INTEGRAL AVERAGE WITH SLASH
U+02A10	$\oint_0^1$	\cirfnint	CIRCULATION FUNCTION
U+02A11	$\oint_0^1$	\awint	ANTICLOCKWISE INTEGRATION
U+02A12	$\oint_0^1$	\rppolint	LINE INTEGRATION WITH RECTANGULAR PATH AROUND POLE
U+02A13	$\oint_0^1$	\scpolint	LINE INTEGRATION WITH SEMICIRCULAR PATH AROUND POLE
U+02A14	$\oint_0^1$	\npolint	LINE INTEGRATION NOT INCLUDING THE POLE
U+02A15	$\oint_0^1$	\pointint	INTEGRAL AROUND A POINT OPERATOR
U+02A16	$\oint_0^1$	\sqint	QUATERNION INTEGRAL OPERATOR
U+02A17	$\int_0^1$	\intlarhk	INTEGRAL WITH LEFTWARDS ARROW WITH HOOK
U+02A18	$\int_0^1$	\intx	INTEGRAL WITH TIMES SIGN
U+02A19	$\int_0^1$	\intcap	INTEGRAL WITH INTERSECTION
U+02A1A	$\int_0^1$	\intcup	INTEGRAL WITH UNION
U+02A1B	$\int_0^1$	\upint	INTEGRAL WITH OVERBAR
U+02A1C	$\int_0^1$	\lowint	INTEGRAL WITH UNDERBAR
U+02A1D	$\bowtie_0^1$	\Join	JOIN
U+02A1E	$\triangleleft_0^1$	\bigtriangleleft	LARGE LEFT TRIANGLE OPERATOR
U+02A1F	$\S_0^1$	\zcmp	Z NOTATION SCHEMA COMPOSITION
U+02A20	$\gg_0^1$	\zpipe	Z NOTATION SCHEMA PIPING
U+02A21	$\upharpoonleft_0^1$	\zproject	Z NOTATION SCHEMA PROJECTION
U+02AFC	$\|_0^1$	\biginterleave	LARGE TRIPLE VERTICAL BAR OPERATOR
U+02AFF	$\|_0^1$	\bigtalloblong	N-ARY WHITE VERTICAL BAR

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

585 \tl_new:Nn \l_um_nolimits_tl {

```
586    \int\iint\iiint\iiiint\oint\oiint\oiiint
587    \intclockwise\varointclockwise\ointctrclockwise\sumint
588    \intbar\intBar\fint\cirfnint\awint\rppolint
589    \scpolint\npolint\pointint\sqint\intlarhk\intx
590    \intcap\intcup\upint\lowint
591  }
```

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

```
592  \DeclareDocumentCommand \addnolimits {m} {
593    \tl_put_right:Nn \l_um_nolimits_tl {#1}
594  }
```

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

```
595  \DeclareDocumentCommand \removenolimits {m} {
596    \tl_remove_all_in:Nn \l_um_nolimits_tl {#1}
597  }
```

7.5 Radicals

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

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

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

```
598  \tl_new:Nn \l_um_radicals_tl {\sqrt}
```

$$\sqrt[2]{1+\sqrt[3]{1+x}}$$

```
\setmathfont{Cambria Math}
\[ \sqrt[2]{1+\sqrt[3]{1+x}} \]
```

7.6 Delimiters

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

http://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3853&prlatex/
3754

```
599  \let\left@primitive\left
600  \def\left{\mathopen{}\left@primitive}
```

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

Here are all \mathopen characters:

USV	Ex.	Macro	Description
U+00028	(	\lparen	LEFT PARENTHESIS
U+0005B	[	\lbrack	LEFT SQUARE BRACKET
U+0007B	{	\lbrace	LEFT CURLY BRACKET
U+0007C	|	\lvert	VERTICAL BAR
U+02016	‖	\lVert	DOUBLE VERTICAL BAR
U+0221A	$\sqrt{}$	\sqrt	RADICAL
U+0221B	$\sqrt[3]{}$	\cuberoot	CUBE ROOT
U+0221C	$\sqrt[4]{}$	\fourthroot	FOURTH ROOT
U+02308	⌈	\lceil	LEFT CEILING
U+0230A	⌊	\lfloor	LEFT FLOOR
U+0231C	⌜	\ulcorner	UPPER LEFT CORNER
U+0231E	⌞	\llcorner	LOWER LEFT CORNER
U+02772		\lbrbrak	LIGHT LEFT TORTOISE SHELL BRACKET ORNAMENT
U+027C5	⟅	\lbag	LEFT S-SHAPED BAG DELIMITER
U+027CC	⟌	\longdivision	LONG DIVISION
U+027E6	⟦	\lBrack	MATHEMATICAL LEFT WHITE SQUARE BRACKET
U+027E8	⟨	\langle	MATHEMATICAL LEFT ANGLE BRACKET
U+027EA	⟪	\lAngle	MATHEMATICAL LEFT DOUBLE ANGLE BRACKET
U+027EC		\Lbrbrak	MATHEMATICAL LEFT WHITE TORTOISE SHELL BRACKET
U+02983	⦃	\lBrace	LEFT WHITE CURLY BRACKET
U+02985	⦅	\lParen	LEFT WHITE PARENTHESIS
U+02987	⦇	\llparenthesis	Z NOTATION LEFT IMAGE BRACKET
U+02989	⦉	\llangle	Z NOTATION LEFT BINDING BRACKET
U+0298B	⦋	\lbrackubar	LEFT SQUARE BRACKET WITH UNDERBAR
U+0298D	⦍	\lbrackultick	LEFT SQUARE BRACKET WITH TICK IN TOP CORNER
U+0298F	⦏	\lbracklltick	LEFT SQUARE BRACKET WITH TICK IN BOTTOM CORNER
U+02991	⦑	\langledot	LEFT ANGLE BRACKET WITH DOT
U+02993	⦓	\lparenless	LEFT ARC LESS-THAN BRACKET
U+02997	⦗	\lblkbrbrak	LEFT BLACK TORTOISE SHELL BRACKET
U+029D8	⧘	\lvzigzag	LEFT WIGGLY FENCE
U+029DA	⧚	\Lvzigzag	LEFT DOUBLE WIGGLY FENCE
U+029FC	⧼	\lcurvyangle	LEFT POINTING CURVED ANGLE BRACKET
U+03014		\lbrbrak	LEFT BROKEN BRACKET
U+03018		\Lbrbrak	LEFT WHITE TORTOISE SHELL BRACKET

And \mathclose:

USV	Ex.	Macro	Description
U+00029	)	\rparen	RIGHT PARENTHESIS
U+0005D	]	\rbrack	RIGHT SQUARE BRACKET
U+0007C	|	\rvert	VERTICAL BAR
U+0007D	}	\rbrace	RIGHT CURLY BRACKET
U+02016	‖	\rVert	DOUBLE VERTICAL BAR
U+02309	⌉	\rceil	RIGHT CEILING
U+0230B	⌋	\rfloor	RIGHT FLOOR
U+0231D	⌝	\urcorner	UPPER RIGHT CORNER
U+0231F	⌟	\lrcorner	LOWER RIGHT CORNER
U+02773		\rbrbrak	LIGHT RIGHT TORTOISE SHELL BRACKET ORNAMENT
U+027C6	⟆	\rbag	RIGHT S-SHAPED BAG DELIMITER
U+027E7	⟧	\rBrack	MATHEMATICAL RIGHT WHITE SQUARE BRACKET
U+027E9	⟩	\rangle	MATHEMATICAL RIGHT ANGLE BRACKET
U+027EB	⟫	\rAngle	MATHEMATICAL RIGHT DOUBLE ANGLE BRACKET
U+027ED		\Rbrbrak	MATHEMATICAL RIGHT WHITE TORTOISE SHELL BRACKET
U+02984	⦄	\rBrace	RIGHT WHITE CURLY BRACKET
U+02986	⦆	\rParen	RIGHT WHITE PARENTHESIS
U+02988	⦈	\rrparenthesis	Z NOTATION RIGHT IMAGE BRACKET
U+0298A	⦊	\rrangle	Z NOTATION RIGHT BINDING BRACKET
U+0298C	⦌	\rbrackubar	RIGHT SQUARE BRACKET WITH UNDERBAR
U+0298E	⦎	\rbracklrtick	RIGHT SQUARE BRACKET WITH TICK IN BOTTOM CORNER
U+02990	⦐	\rbrackurtick	RIGHT SQUARE BRACKET WITH TICK IN TOP CORNER
U+02992	⦒	\rangledot	RIGHT ANGLE BRACKET WITH DOT
U+02994	⦔	\rparengtr	RIGHT ARC GREATER-THAN BRACKET
U+02998	⦘	\rblkbrbrak	RIGHT BLACK TORTOISE SHELL BRACKET
U+029D9	⧙	\rvzigzag	RIGHT WIGGLY FENCE
U+029DB	⧛	\Rvzigzag	RIGHT DOUBLE WIGGLY FENCE
U+029FD	⧽	\rcurvyangle	RIGHT POINTING CURVED ANGLE BRACKET
U+03015		\rbrbrak	RIGHT BROKEN BRACKET
U+03019		\Rbrbrak	RIGHT WHITE TORTOISE SHELL BRACKET

7.7   Maths accents

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

USV	Ex.	Macro	Description

U+00300	$\grave{x}$	\grave	GRAVE ACCENT
U+00301	$\acute{x}$	\acute	ACUTE ACCENT
U+00302	$\hat{x}$	\hat	CIRCUMFLEX ACCENT
U+00303	$\tilde{x}$	\tilde	TILDE
U+00304	$\bar{x}$	\bar	MACRON
U+00305	$\overline{x}$	\overbar	OVERBAR EMBELLISHMENT
U+00306	$\breve{x}$	\breve	BREVE
U+00307	$\dot{x}$	\dot	DOT ABOVE
U+00308	$\ddot{x}$	\ddot	DIERESIS
U+00309	$\overset{\text{ʾ}}{x}$	\ovhook	COMBINING HOOK ABOVE
U+0030A	$\mathring{x}$	\ocirc	RING
U+0030C	$\check{x}$	\check	CARON
U+00310	$\overset{\smile}{x}$	\candra	CANDRABINDU (NON-SPACING)
U+00312	$\acute{x}$	\oturnedcomma	COMBINING TURNED COMMA ABOVE
U+00313	$\dot{x}$	\osmooth	GREEK PSILI (SMOOTH BREATHING) (NON-SPACING)
U+00314	$\grave{x}$	\orough	GREEK DASIA (ROUGH BREATHING) (NON-SPACING)
U+00315	$\acute{x}$	\ocommatopright	COMBINING COMMA ABOVE RIGHT
U+0031A	$\overset{\langle}{x}$	\droang	LEFT ANGLE ABOVE (NON-SPACING)
U+00338	$\not{x}$	\not	COMBINING LONG SOLIDUS OVERLAY
U+020D0	$\overset{\leftharpoonup}{x}$	\leftharpoonaccent	COMBINING LEFT HARPOON ABOVE
U+020D1	$\overset{\rightharpoonup}{x}$	\rightharpoonaccent	COMBINING RIGHT HARPOON ABOVE
U+020D2	$\vert{x}$	\vertoverlay	COMBINING LONG VERTICAL LINE OVERLAY
U+020D6	$\overleftarrow{x}$	\overleftarrow	COMBINING LEFT ARROW ABOVE
U+020D7	$\vec{x}$	\overrightarrow	COMBINING RIGHT ARROW ABOVE
U+020DB	$\dddot{x}$	\dddot	COMBINING THREE DOTS ABOVE
U+020DC	$\ddddot{x}$	\ddddot	COMBINING FOUR DOTS ABOVE
U+020E1	$\overleftrightarrow{x}$	\overleftrightarrow	COMBINING LEFT RIGHT ARROW ABOVE
U+020E7	$\boxtimes$	\annuity	COMBINING ANNUITY SYMBOL
U+020E8	$\underset{\dots}{x}$	\threeunderdot	COMBINING TRIPLE UNDERDOT
U+020E9	$\overline{x}$	\widebridgeabove	COMBINING WIDE BRIDGE ABOVE
U+020EC	☒	\underrightharpoondown	COMBINING RIGHTWARDS HARPOON WITH BARB DOWNWARDS
U+020ED	☒	\underleftharpoondown	COMBINING LEFTWARDS HARPOON WITH BARB DOWNWARDS
U+020EE	☒	\underleftarrow	COMBINING LEFT ARROW BELOW
U+020EF	☒	\underrightarrow	COMBINING RIGHT ARROW BELOW
U+020F0	☒	\asteraccent	COMBINING ASTERISK ABOVE

8 Font features

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

```
601 \newcommand\um@zf@feature[2]{
602   \define@key[zf]{options}{#1}[]{
603     \bool_if:NTF \l_um_fontspec_feature_bool {
604       #2
605     }{
606       \PackageError{fontspec/unicode-math}
607         {The '#1' font feature can only be used for maths fonts}
608         {The feature you tried to use can only be in commands
609           like \protect\setmathfont}
610     }
611   }
612 }
```

8.1 OpenType maths font features

```
613 \um@zf@feature{ScriptStyle}{
614   \zf@update@ff{+ssty=0}
615 }
616 \um@zf@feature{ScriptScriptStyle}{
617   \zf@update@ff{+ssty=1}
618 }
```

8.2 Script and scriptscript font options

```
619 \define@cmdkey[um]{options}[um@]{script-features}{}
620 \define@cmdkey[um]{options}[um@]{sscript-features}{}
621 \define@cmdkey[um]{options}[um@]{script-font}{}
622 \define@cmdkey[um]{options}[um@]{sscript-font}{}
```

8.3 Range processing

The 'ALL' branch here is deprecated and happens automatically.

```
623 \seq_new:N \g_um_mathalph_seq
624 \seq_new:N \l_um_mathalph_seq
625 \seq_new:N \l_um_char_range_seq
626 \define@choicekey+[um]{options}{range}[\@tempa\@tempb]{ALL}{
627   \ifcase\@tempb\relax
628     \bool_set_true:N \l_um_init_bool
629   \fi
630 }{
631   \bool_set_false:N \l_um_init_bool
632   \seq_clear:N \l_um_char_range_seq
```

41

```
633    \seq_clear:N \l_um_mathalph_seq
634    \clist_map_inline:nn {#1} {
635      \um_if_mathalph_decl:nTF {##1} {
636      \seq_put_right:Nx \l_um_mathalph_seq { {\exp_not:V\l_um_tmpa_tl} {\exp_not:V\l_um_tmpb_tl
637      }{
638        \seq_put_right:Nn \l_um_char_range_seq {##1}
639      }
640    }
641  }
642  \prg_new_conditional:Nnn \um_if_mathalph_decl:n {TF} {
643    \tl_set:Nn \l_um_tmpa_tl {#1}
644    \tl_set:Nn \l_um_tmpb_tl {}
645    \tl_set:Nn \l_um_tmpc_tl {}
646    \tl_if_in:NnT \l_um_tmpa_tl {->} {
647      \exp_after:wN \um_split_arrow:w \l_um_tmpa_tl \q_nil
648    }
649    \tl_if_in:NnT \l_um_tmpa_tl {/} {
650      \exp_after:wN \um_split_slash:w \l_um_tmpa_tl \q_nil
651    }
652    \seq_if_in:NVTF \g_um_mathalph_seq \l_um_tmpa_tl {
653      \prg_return_true:
654    }{
655      \prg_return_false:
656    }
657  }
658  \cs_set:Npn \um_split_arrow:w #1->#2 \q_nil {
659    \tl_set:Nn \l_um_tmpa_tl {#1}
660    \tl_set:Nn \l_um_tmpc_tl {#2}
661  }
662  \cs_set:Npn \um_split_slash:w #1/#2 \q_nil {
663    \tl_set:Nn \l_um_tmpa_tl {#1}
664    \tl_set:Nn \l_um_tmpb_tl {#2}
665  }
```

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

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

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

Character ranges are passed to \um@parse@range, which accepts input in the form shown in table 13.

Table 13: Ranges accepted by \um@parse@range.

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

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

```
666 \newcommand\um@parse@term[4]{
667   \seq_map_variable:NNn \l_um_char_range_seq \@ii {
668     \unless\ifx\@ii\@empty
669       \@tempswafalse
```

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

```
670       \expandafter\um@firstchar\expandafter{\@ii}
671       \ifx\@tempa\um@backslash
672         \expandafter\ifx\@ii#2\relax
673           \@tempswatrue
674         \else
675           \expandafter\ifx\@ii#3\relax
676             \@tempswatrue
677           \fi
678         \fi
```

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

```
679       \else
680         \expandafter\um@parse@range\@ii-\@marker-\@nil#1\@nil
681       \fi
```

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

```
682       \if@tempswa
683         \ifx\um@char@num@range\@empty
684           \g@addto@macro\um@char@num@range{#1}
685         \else
686           \g@addto@macro\um@char@num@range{,#1}
687         \fi
688         #4%
689       \fi
690     \fi
691   }
692 }
693 \def\um@firstof#1#2\@nil{#1}
694 \edef\um@backslash{\expandafter\um@firstof\string\string\@nil}
```

```
695  \def\um@firstchar#1{\edef\@tempa{\expandafter\um@firstof\string#1\@nil}}
```

\um@parse@range Weird syntax. As shown previously in table 13, this macro can be passed four different input types via \um@parse@term.

```
696  \def\um@parse@range#1-#2-#3\@nil#4\@nil{
697    \def\@tempa{#1}
698    \def\@tempb{#2}
```

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

```
699    \expandafter\ifx\expandafter\@marker\@tempb\relax
700      \ifnum#4=#1\relax
701        \@tempswatrue
702      \fi
703    \else
```

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

```
704      \ifx\@empty\@tempb
705        \ifnum#4>\numexpr#1-1\relax
706          \@tempswatrue
707        \fi
708      \else
```

Range	r ≤ y
C-list input	\@ii=-Y
Macro input	\um@parse@range -Y-\@marker-\@nil#1\@nil
Arguments	#1-#2-#3 = {}-Y-\@marker-

```
709        \ifx\@empty\@tempa
710          \ifnum#4<\numexpr#2+1\relax
711            \@tempswatrue
712          \fi
```

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

```
713        \else
714          \ifnum#4>\numexpr#1-1\relax
715            \ifnum#4<\numexpr#2+1\relax
716              \@tempswatrue
717            \fi
718          \fi
719        \fi
```

44

```
720       \fi
721     \fi
722   }
```

\um_map_char:nn   #1 : Number of iterations
\um_map_chars_xxvi:nn   #2 : Starting input char(s)
\um_map_chars_xxiii:nn   #3 : Starting output char

Loops through character ranges setting \mathcode.

```
723 \cs_set:Nn \um_map_chars_range:nnn {
724   \clist_map_inline:nn {#2} {
725     \prg_stepwise_inline:nnnn {0}{1}{#1} {
726       \um_map_char_internal:nn {##1+####1}{#3+####1}
727     }
728   }
729 }
730 \cs_new:Nn \um_map_char_noparse:nn {
731   \um_set_mathcode:nnnn
732     {\numexpr #1 \relax}{\mathalpha}{\um_symfont_tl}{\numexpr #2 \relax}
733 }
734 \cs_new:Nn \um_map_char_parse:nn {
735   \um@parse@term {#1} {\@nil} {\mathalpha} {
736     \um_map_char_noparse:nn {#1}{#2}
737   }
738 }
739 \cs_set:Nn \um_map_chars_xxvi:nn {
740   \um_map_chars_range:nnn {25}{#1}{#2}
741 }
742 \cs_set:Nn \um_map_chars_xxiii:nn {
743   \um_map_chars_range:nnn {24}{#1}{#2}
744 }
745 \cs_set:Nn \um_map_chars_Latin:nn {
746   \clist_map_inline:nn {#1} {
747     \um_map_chars_xxvi:cc {g_um_ ##1 _Latin_usv}{g_um_ #2 _Latin_usv}
748   }
749 }
750 \cs_set:Nn \um_map_chars_latin:nn {
751   \clist_map_inline:nn {#1} {
752     \um_map_chars_xxvi:cc {g_um_ ##1 _latin_usv}{g_um_ #2 _latin_usv}
753   }
754 }
755 \cs_set:Nn \um_map_chars_greek:nn {
756   \clist_map_inline:nn {#1} {
757     \um_map_chars_xxiii:cc {g_um_ ##1 _greek_usv}{g_um_ #2 _greek_usv}
758     \um_map_char:cc {g_um_ ##1 _varepsilon_usv}{g_um_ #2 _varepsilon_usv}
759     \um_map_char:cc {g_um_ ##1 _vartheta_usv  }{g_um_ #2 _vartheta_usv  }
760     \um_map_char:cc {g_um_ ##1 _varkappa_usv  }{g_um_ #2 _varkappa_usv  }
```

45

```
761      \um_map_char:cc {g_um_ ##1 _varphi_usv    }{g_um_ #2 _varphi_usv    }
762      \um_map_char:cc {g_um_ ##1 _varrho_usv    }{g_um_ #2 _varrho_usv    }
763      \um_map_char:cc {g_um_ ##1 _varpi_usv     }{g_um_ #2 _varpi_usv     }
764    }
765  }
766  \cs_set:Nn \um_map_chars_Greek:nn {
767    \clist_map_inline:nn {#1} {
768      \um_map_chars_xxiii:cc {g_um_ ##1 _Greek_usv}{g_um_ #2 _Greek_usv}
769      \um_map_char:cc {g_um_ ##1 _varTheta_usv}{g_um_ #2 _varTheta_usv}
770    }
771  }
772  \cs_set:Nn \um_map_chars_numbers:nn {
773    \um_map_chars_range:nnn {9}{#1}{#2}
774  }
775  \cs_set:Nn \um_map_char:nn {
776    \um_map_chars_range:nnn {0}{#1}{#2}
777  }
778  \cs_generate_variant:Nn \um_map_char:nn {cc}
779  \cs_generate_variant:Nn \um_map_chars_xxiii:nn {cc}
780  \cs_generate_variant:Nn \um_map_chars_xxvi:nn {cc}
```

`\um_set_mathalphabet_char:Nnn` #1 : Maths alphabet
#2 : Input char(s)
#3 : Output char
Loops through character ranges setting `\mathcode`.

```
781  \cs_set:Npn \exp_args:Nnff {\::n\::f\::f\:::}
782  \cs_new:Nn \um_set_mathalphabet_char:Nnn {
783    \clist_map_variable:nNn {#2} \l_um_input_num {
784      \exp_args:Nnff \um_mathmap:Nnn {#1}
785        {\number\numexpr\l_um_input_num\relax} {\number\numexpr#3\relax}
786    }
787  }
```

`\um_set_mathalph_range:Nnn` [⟨*Number of iterations*⟩] #1 : Maths alphabet
#2 : Starting input char(s)
#3 : Starting output char
Loops through character ranges setting `\mathcode`.

```
788  \cs_new:Nn \um_set_mathalph_range:nNnn {
789    \clist_map_variable:nNn {#3} \l_um_input_num {
790      \errorcontextlines=999
791      \prg_stepwise_variable:nnnNn {0}{1}{#1} \l_um_inc_num {
792        \exp_args:Nnff \um_mathmap:Nnn {#2}
793          {\number\numexpr \l_um_inc_num + \l_um_input_num \relax}
794          {\number\numexpr \l_um_inc_num + #4 \relax}
795      }
796    }
```

```
797  }
798  \cs_new:Nn \um_set_mathalphabet_x:Nnn {
799    \um_set_mathalph_range:nNnn {9}{#1}{#2}{#3}
800  }
801  \cs_new:Nn \um_set_mathalphabet_xxvi:Nnn {
802    \um_set_mathalph_range:nNnn {25}{#1}{#2}{#3}
803  }
804  \cs_new:Nn \um_set_mathalphabet_xxiii:Nnn {
805    \um_set_mathalph_range:nNnn {24}{#1}{#2}{#3}
806  }
807
808  \cs_new:Nn \um_set_mathalphabet_pos:Nnnn {
809    \clist_map_inline:nn {#3} {
810      \um_set_mathalphabet_char:Ncc  #1 {g_um_##1_#2_usv}{g_um_#4_#2_usv}
811    }
812  }
813  \cs_new:Nn \um_set_mathalphabet_numbers:Nnn {
814    \clist_map_inline:nn {#2} {
815      \um_set_mathalphabet_x:Ncc  #1 {g_um_##1_num_usv}{g_um_#3_num_usv}
816    }
817  }
818  \cs_new:Nn \um_set_mathalphabet_Latin:Nnn {
819    \clist_map_inline:nn {#2} {
820     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_Latin_usv}{g_um_#3_Latin_usv}
821    }
822  }
823  \cs_new:Nn \um_set_mathalphabet_latin:Nnn {
824    \clist_map_inline:nn {#2} {
825     \um_set_mathalphabet_xxvi:Ncc #1 {g_um_##1_latin_usv}{g_um_#3_latin_usv}
826      \um_set_mathalphabet_char:Ncc #1 {g_um_##1_h_usv}     {g_um_#3_h_usv}
827    }
828  }
829  \cs_new:Nn \um_set_mathalphabet_Greek:Nnn {
830    \clist_map_inline:nn {#2} {
831     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_Greek_usv}  {g_um_#3_Greek_usv}
832     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varTheta_usv}{g_um_#3_varTheta_usv}
833    }
834  }
835  \cs_new:Nn \um_set_mathalphabet_greek:Nnn {
836    \clist_map_inline:nn {#2} {
837     \um_set_mathalphabet_xxiii:Ncc #1 {g_um_##1_greek_usv}     {g_um_#3_greek_usv}
838     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varepsilon_usv}{g_um_#3_varepsilon_usv}
839     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_vartheta_usv} {g_um_#3_vartheta_usv}
840     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varkappa_usv} {g_um_#3_varkappa_usv}
841     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varphi_usv}   {g_um_#3_varphi_usv}
842     \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varrho_usv}   {g_um_#3_varrho_usv}
```

```
843    \um_set_mathalphabet_char:Ncc #1 {g_um_##1_varpi_usv}    {g_um_#3_varpi_usv}
844    }
845  }
846  \cs_generate_variant:Nn \um_set_mathalphabet_char:Nnn {Ncc}
847  \cs_generate_variant:Nn \um_set_mathalphabet_xxiii:Nnn {Ncc}
848  \cs_generate_variant:Nn \um_set_mathalphabet_xxvi:Nnn {Ncc}
849  \cs_generate_variant:Nn \um_set_mathalphabet_x:Nnn {Ncc}
```

8.4 Resolving Greek symbol name control sequences

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

```
850  \AtBeginDocument{\um_resolve_greek:}
851  \cs_new:Nn \um_resolve_greek: {
852    \clist_map_inline:nn {
853      Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
854      alpha,beta,gamma,delta,          zeta,eta,theta,ioto,kappa,lambda,
855      Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
856      mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,    chi,psi,omega,
857      varTheta,
858      varsigma,vartheta,varkappa,varrho,varpi
859    }{
860      \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
861    }
862    \tl_set:Nn \epsilon {
863      \bool_if:NTF \g_um_texgreek_bool \mitvarepsilon \mitepsilon
864    }
865    \tl_set:Nn \phi {
866      \bool_if:NTF \g_um_texgreek_bool \mitvarphi \mitphi
867    }
868    \tl_set:Nn \varepsilon {
869      \bool_if:NTF \g_um_texgreek_bool \mitepsilon \mitvarepsilon
870    }
871    \tl_set:Nn \varphi {
872      \bool_if:NTF \g_um_texgreek_bool \mitphi \mitvarphi
873    }
874  }
```

9   Maths alphabets mapping definitions

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

Implicit mode:

- Try and set all of the alphabet shapes.

- Check for the first glyph of each alphabet to detect if the font supports each alphabet shape.

- For alphabets that do exist, overwrite whatever's already there.

- For alphabets that are not supported, *do nothing*. (This includes leaving the old alphabet definition in place.)

Explicit mode:

- Only set the alphabets specified.

- Check for the first glyph of the alphabet to detect if the font contains the alphabet shape in the unicode math plane.

- For unicode math alphabets, overwrite whatever's already there.

- Otherwise, use the ᴀꜱᴄɪɪ letters instead.

9.0.1 Macros

This is every math alphabet known to unicode-math:

\g_um_mathalph_seq

```
875 \seq_clear:N \g_um_mathalph_seq
876 \tl_map_inline:nn {
877   \mathup\mathit
878   \mathbb\mathscr\mathfrak\mathtt
879   \mathsf\mathsfup\mathsfit
880   \mathbf\mathbfup\mathbfit
881   \mathbfscr\mathbffrak
882   \mathbfsf\mathbfsfup\mathbfsfit
883 }{
884   \seq_put_right:Nn \g_um_mathalph_seq {#1}
885 }
```

\um_setup_alphabets:

```
886
887 \tl_new:Nn \g_um_mathup_alph_clist {latin,Latin,greek,Greek}
888 \tl_new:Nn \g_um_mathit_alph_clist {latin,Latin,greek,Greek}
889 \tl_new:Nn \g_um_mathscr_alph_clist   {latin,Latin}
890 \tl_new:Nn \g_um_mathfrak_alph_clist  {latin,Latin}
891 \tl_new:Nn \g_um_mathbfscr_alph_clist {latin,Latin}
892 \tl_new:Nn \g_um_mathbffrak_alph_clist {latin,Latin}
893 \tl_new:Nn \g_um_mathbb_alph_clist    {latin,Latin,num}
```

```
894  \tl_new:Nn \g_um_mathtt_alph_clist      {latin,Latin,num}
895  \tl_new:Nn \g_um_mathsf_alph_clist       {latin,Latin,num}
896  \tl_new:Nn \g_um_mathsfup_alph_clist     {latin,Latin,num}
897  \tl_new:Nn \g_um_mathsfit_alph_clist     {latin,Latin}
898  \tl_new:Nn \g_um_mathbf_alph_clist       {latin,Latin,greek,Greek,num}
899  \tl_new:Nn \g_um_mathbfup_alph_clist     {latin,Latin,greek,Greek,num}
900  \tl_new:Nn \g_um_mathbfit_alph_clist     {latin,Latin,greek,Greek,num}
901  \tl_new:Nn \g_um_mathbfsf_alph_clist     {latin,Latin,greek,Greek,num}
902  \tl_new:Nn \g_um_mathbfsfup_alph_clist {latin,Latin,greek,Greek,num}
903  \tl_new:Nn \g_um_mathbfsfit_alph_clist {latin,Latin,greek,Greek}
904
905  \tl_new:Nn \g_um_mathup_latin_usv {`\a-`\z}
906  \tl_new:Nn \g_um_mathup_Latin_usv {`\A-`\Z}
907  \tl_new:Nn \g_um_mathup_greek_usv {"3B1-"3C9,"3F5,"3D1,"3F0,"3D5,"3F1,"3D6,"3DD}
908  \tl_new:Nn \g_um_mathup_Greek_usv {"391-"3A9,"3F4,"3DC}
909  \tl_new:Nn \g_um_mathup_num_usv    {`\0-`\9}
910
911  \tl_new:Nn \g_um_mathit_latin_usv {"1D44E-"1D467,\g_um_it_h_usv}
912  \tl_new:Nn \g_um_mathit_Latin_usv {"1D434-"1D44C}
913  \tl_new:Nn \g_um_mathit_greek_usv {"1D6FC-"1D714,"1D716-"1D71B}
914  \tl_new:Nn \g_um_mathit_Greek_usv {"1D6E2-"1D6FA}
915
916  \seq_new:N \l_um_missing_alph_seq
917  \cs_new:Nn \um_setup_alphabets: {
918    \seq_clear:N \l_um_missing_alph_seq
919    \seq_if_empty:NTF \l_um_mathalph_seq {
920      \um_setup_math_alphabet:NV \mathup      \g_um_mathup_alph_clist
921      \um_setup_math_alphabet:NV \mathit      \g_um_mathit_alph_clist
922      \um_setup_math_alphabet:NV \mathbb      \g_um_mathbb_alph_clist
923      \um_setup_math_alphabet:NV \mathscr     \g_um_mathscr_alph_clist
924      \um_setup_math_alphabet:NV \mathfrak    \g_um_mathfrak_alph_clist
925      \um_setup_math_alphabet:NV \mathsf      \g_um_mathsf_alph_clist
926      \um_setup_math_alphabet:NV \mathsfup    \g_um_mathsfup_alph_clist
927      \um_setup_math_alphabet:NV \mathsfit    \g_um_mathsfit_alph_clist
928      \um_setup_math_alphabet:NV \mathtt      \g_um_mathtt_alph_clist
929      \um_setup_math_alphabet:NV \mathbf      \g_um_mathbf_alph_clist
930      \um_setup_math_alphabet:NV \mathbfup    \g_um_mathbfup_alph_clist
931      \um_setup_math_alphabet:NV \mathbfit    \g_um_mathbfit_alph_clist
932      \um_setup_math_alphabet:NV \mathbfscr  \g_um_mathbfscr_alph_clist
933      \um_setup_math_alphabet:NV \mathbffrak \g_um_mathbffrak_alph_clist
934      \um_setup_math_alphabet:NV \mathbfsf   \g_um_mathbfsf_alph_clist
935      \um_setup_math_alphabet:NV \mathbfsfup \g_um_mathbfsfup_alph_clist
936      \um_setup_math_alphabet:NV \mathbfsfit \g_um_mathbfsfit_alph_clist
937      \um_setup_math_mapping:n   {up    }
938      \um_setup_math_mapping:n   {it    }
939      \um_setup_math_mapping:n   {bb    }
```

```
940      \um_maybe_init_alphabet:n  {bbit  }
941      \um_setup_math_mapping:n   {bbit  }
942      \um_setup_math_mapping:n   {bfup  }
943      \um_setup_math_mapping:n   {bfit  }
944      \um_setup_math_mapping:n   {bfsfup}
945      \um_setup_math_mapping:n   {bfsfit}
946      \seq_if_empty:NF \l_um_missing_alph_seq {
947        \typeout{
948          Package~unicode-math~Warning:~
949          missing~math~alphabets~in~font~ \fontname\l_um_font
950        }
951        \seq_map_inline:Nn \l_um_missing_alph_seq {
952          \typeout{\space\space\space\space##1}
953        }
954      }
955    }{
956      \cs_set_eq:NN \um_mathmap:Nnn \um_mathmap_noparse:Nnn
957      \seq_map_inline:Nn \l_um_mathalph_seq {
958        \tl_set:No \l_um_tmpa_tl { \use_i:nnn   ##1 }
959        \tl_set:No \l_um_tmpb_tl { \use_ii:nnn  ##1 }
960        \tl_set:No \l_um_tmpc_tl { \use_iii:nnn ##1 }
961        \tl_if_empty:NF \l_um_tmpc_tl {
962      \PackageWarning{unicode-math}{alphabet~remapping~not~yet~implemented}
963        }
964        \tl_if_empty:NT \l_um_tmpb_tl {
965          \cs_set_eq:NN \um_maybe_init_alphabet:n \um_init_alphabet:n
966      \tl_set:Nv \l_um_tmpb_tl { g_um_ \exp_after:wN \cs_to_str:N \l_um_tmpa_tl _alph_clist }
967        }
968        \um_setup_math_alphabet:VV \l_um_tmpa_tl \l_um_tmpb_tl
969      }
970    }
971 }
```

\um_setup_math_alphabet:Nn   #1 : Math font family name (e.g., \mathbb)
#2 : Math alphabets, comma separated of {latin,Latin,greek,Greek,num}
First check that at least one of the alphabets for the font shape is defined, and then
loop through them defining the individual ranges.

```
972 \cs_new:Nn \um_setup_math_alphabet:Nn {
973   \tl_set:Nx \l_um_tmpa_tl {\cs_to_str:N #1}
974   \tl_set:Nx \l_um_tmpb_tl {\exp_after:wN \use_none:nnnn \l_um_tmpa_tl}
975   \clist_map_inline:nn {#2} {
976     \um_glyph_if_exist:cT {g_um_ \l_um_tmpb_tl _##1_usv}{
977       \exp_args:NV \um_maybe_init_alphabet:n \l_um_tmpb_tl
978       \clist_map_break:
979     }
980   }
```

```
981      \clist_map_inline:nn {#2} {
982        \um_glyph_if_exist:cTF {g_um_ \l_um_tmpb_tl _##1_usv}{
983          \use:c {um_config_ \l_um_tmpa_tl _##1:}
984        }{
985          \seq_put_right:Nx \l_um_missing_alph_seq {
986            \@backslashchar
987            \l_um_tmpa_tl\space(\tl_use:c{g_um_math_alphabet_name_##1_tl})
988          }
989        }
990      }
991    }
992    \cs_generate_variant:Nn \um_setup_math_alphabet:Nn {NV,VV}
993
994    \tl_set:Nn \g_um_math_alphabet_name_latin_tl {Latin,~lowercase}
995    \tl_set:Nn \g_um_math_alphabet_name_Latin_tl {Latin,~uppercase}
996    \tl_set:Nn \g_um_math_alphabet_name_greek_tl {Greek,~lowercase}
997    \tl_set:Nn \g_um_math_alphabet_name_Greek_tl {Greek,~uppercase}
998    \tl_set:Nn \g_um_math_alphabet_name_num_tl    {Numerals}
999    \cs_new:Nn \um_setup_math_mapping:n {
1000     \cs_if_exist:cT {um_setup_math#1:} {
1001       \use:c {um_config_math#1_misc:}
1002     }
1003   }
1004   \cs_set:Nn \um_init_alphabet:n {
1005     \wlog{unicode-math:~Initialiasing~\@backslashchar math#1}
1006     \um_prepare_alph:n {#1}
1007     \cs_set_eq:cN {um_setup_math#1:} \prg_do_nothing:
1008   }
```

\um_glyph_if_exist:nTF : TODO: Generalise for arbitrary fonts! \um@font is not always the one used for a specific glyph!!

```
1009   \prg_new_conditional:Nnn \um_glyph_if_exist:n {p,TF,T,F} {
1010     \etex_iffontchar:D \l_um_font #1 \scan_stop: \prg_return_true: \else: \prg_return_false: \fi:
1011   }
1012   \cs_generate_variant:Nn \um_glyph_if_exist_p:n {c}
1013   \cs_generate_variant:Nn \um_glyph_if_exist:nTF {c}
1014   \cs_generate_variant:Nn \um_glyph_if_exist:nT  {c}
1015   \cs_generate_variant:Nn \um_glyph_if_exist:nF  {c}
```

\um_prepare_alph:n If \mathXY hasn't been (re-)declared yet, then define it in terms of unicode-math defintions. Use \bgroup/\egroup so s'scripts scan the whole thing.

```
1016   \cs_new:Nn \um_prepare_alph:n {
1017     \cs_if_exist:cF {um_math#1:n} {
1018       \cs_set:cpn {um_math#1:n} ##1 {
1019         \use:c {um_setup_math#1:} ##1 \egroup
1020       }
```

```
1021    \cs_set_protected:cpn {math#1} {
1022        \bgroup
1023        \mode_if_math:F {
1024            \egroup\expandafter
1025            \non@alpherr\expandafter{\csname math#1\endcsname\space}
1026        }
1027        \use:c {um_math#1:n}
1028    }
1029    }
1030 }
```

9.1 Alphabets

9.1.1 Upright: \mathup

```
1031 \cs_new:Npn \um_config_mathup_Latin: {
1032    \bool_if:NTF \g_um_literal_bool {
1033        \um_map_chars_latin:nn {up} {up}
1034    }{
1035        \bool_if:NT \g_um_upLatin_bool {
1036            \um_map_chars_Latin:nn {up,it} {up}
1037        }
1038    }
1039    \um_set_mathalphabet_Latin:Nnn \mathup {up,it}{up}
1040 }
1041 \cs_new:Npn \um_config_mathup_latin: {
1042    \bool_if:NTF \g_um_literal_bool {
1043        \um_map_chars_latin:nn {up} {up}
1044    }{
1045        \bool_if:NT \g_um_uplatin_bool {
1046            \um_map_chars_latin:nn {up,it} {up}
1047        \um_map_char:nn {\g_um_up_h_usv,\g_um_it_h_usv}{\g_um_up_h_usv}% KEEP
1048        }
1049    }
1050    \um_set_mathalphabet_latin:Nnn \mathup {up,it}{up}
1051 }
1052 \cs_new:Npn \um_config_mathup_Greek: {
1053    \bool_if:NTF \g_um_literal_bool {
1054        \um_map_chars_Greek:nn {up}{up}
1055    }{
1056        \bool_if:NT \g_um_upGreek_bool {
1057            \um_map_chars_Greek:nn {up,it}{up}
1058        }
1059    }
1060    \um_set_mathalphabet_Greek:Nnn \mathup {up,it}{up}
1061 }
1062 \cs_new:Npn \um_config_mathup_greek: {
```

```
1063    \bool_if:NTF \g_um_literal_bool {
1064      \um_map_chars_greek:nn {up} {up}
1065    }{
1066      \bool_if:NT \g_um_upgreek_bool {
1067        \um_map_chars_greek:nn {up,it} {up}
1068      }
1069    }
1070    \um_set_mathalphabet_greek:Nnn \mathup {up,it} {up}
1071  }
1072  \cs_new:Npn \um_config_mathup_misc: {
1073    \um_set_mathalphabet_pos:Nnnn \mathup {partial} {up,it}{up}
1074    \um_set_mathalphabet_pos:Nnnn \mathup {Nabla}   {up,it}{up}
1075  }
```

9.1.2   Italic: `\mathit`

```
1076  \cs_new:Npn \um_config_mathit_Latin: {
1077    \bool_if:NTF \g_um_literal_bool {
1078      \um_map_chars_Latin:nn {it} {it}
1079    }{
1080      \bool_if:NF \g_um_upLatin_bool {
1081        \um_map_chars_Latin:nn {up,it} {it}
1082      }
1083    }
1084    \um_set_mathalphabet_Latin:Nnn \mathit {up,it}{it}
1085  }
1086  \cs_new:Npn \um_config_mathit_latin: {
1087    \bool_if:NTF \g_um_literal_bool {
1088      \um_map_chars_latin:nn {it} {it}
1089      \um_map_char:nn {\g_um_it_h_usv}{\g_um_it_h_usv}% KEEP
1090    }{
1091      \bool_if:NF \g_um_uplatin_bool {
1092        \um_map_chars_latin:nn {up,it} {it}
1093      \um_map_char:nn {\g_um_up_h_usv,\g_um_it_h_usv}{\g_um_it_h_usv}% KEEP
1094      }
1095    }
1096    \um_set_mathalphabet_latin:Nnn \mathit {up,it}{it}
1097  }
1098  \cs_new:Npn \um_config_mathit_Greek: {
1099    \bool_if:NTF \g_um_literal_bool {
1100      \um_map_chars_Greek:nn {it}{it}
1101    }{
1102      \bool_if:NF \g_um_upGreek_bool {
1103        \um_map_chars_Greek:nn {up,it}{it}
1104      }
1105    }
1106    \um_set_mathalphabet_Greek:Nnn \mathit {up,it}{it}
```

```
1107 }
1108 \cs_new:Npn \um_config_mathit_greek: {
1109   \bool_if:NTF \g_um_literal_bool {
1110     \um_map_chars_greek:nn {it} {it}
1111   }{
1112     \bool_if:NF \g_um_upgreek_bool {
1113       \um_map_chars_greek:nn {it,up} {it}
1114     }
1115   }
1116   \um_set_mathalphabet_greek:Nnn \mathit {up,it} {it}
1117 }
1118 \cs_new:Npn \um_config_mathit_misc: {
1119   \um_set_mathalphabet_pos:Nnnn \mathit {partial} {up,it}{it}
1120   \um_set_mathalphabet_pos:Nnnn \mathit {Nabla}   {up,it}{it}
1121 }
```

9.1.3   Blackboard or double-struck: `\mathbb` and `\mathbbit`

```
1122 \cs_new:Npn \um_config_mathbb_latin: {
1123   \um_set_mathalphabet_latin:Nnn \mathbb {up,it}{bb}
1124 }
1125 \cs_new:Npn \um_config_mathbb_Latin: {
1126   \um_set_mathalphabet_Latin:Nnn \mathbb {up,it}{bb}
1127   \um_set_mathalphabet_char:Nnn{\mathbb}{`\C,"1D436}{"2102}
1128   \um_set_mathalphabet_char:Nnn{\mathbb}{`\H,"1D43B}{"210D}
1129   \um_set_mathalphabet_char:Nnn{\mathbb}{`\N,"1D441}{"2115}
1130   \um_set_mathalphabet_char:Nnn{\mathbb}{`\P,"1D443}{"2119}
1131   \um_set_mathalphabet_char:Nnn{\mathbb}{`\Q,"1D444}{"211A}
1132   \um_set_mathalphabet_char:Nnn{\mathbb}{`\R,"1D445}{"211D}
1133   \um_set_mathalphabet_char:Nnn{\mathbb}{`\Z,"1D44D} {"2124}
1134 }
1135 \cs_new:Npn \um_config_mathbb_num: {
1136   \um_set_mathalphabet_numbers:Nnn \mathbb {up}{bb}
1137 }
1138 \cs_new:Npn \um_config_mathbb_misc: {
1139   \um_set_mathalphabet_char:Nnn \mathbb {"03A0,"1D6F1}{"213F} % Pi
1140   \um_set_mathalphabet_char:Nnn \mathbb {"03C0,"1D70B}{"213C} % pi
1141   \um_set_mathalphabet_char:Nnn \mathbb {"0393,"1D6E4}{"213E} % Gamma
1142   \um_set_mathalphabet_char:Nnn \mathbb {"03B3,"1D6FE}{"213D} % gamma
1143   \um_set_mathalphabet_char:Nnn \mathbb {"2211}{"2140} % summation
1144 }
1145 \cs_new:Npn \um_config_mathbbit_misc: {
1146   \um_set_mathalphabet_char:Nnn \mathbbit {`\D,"1D437}{"2145}
1147   \um_set_mathalphabet_char:Nnn \mathbbit {`\d,"1D451}{"2146}
1148   \um_set_mathalphabet_char:Nnn \mathbbit {`\e,"1D452}{"2147}
1149   \um_set_mathalphabet_char:Nnn \mathbbit {`\i,"1D456}{"2148}
1150   \um_set_mathalphabet_char:Nnn \mathbbit {`\j,"1D457}{"2149}
```

```
1151 }
```

9.1.4 Script or caligraphic: `\mathscr` and `\mathcal`

```
1152 \cs_new:Npn \um_config_mathscr_Latin: {
1153   \um_set_mathalphabet_Latin:Nnn \mathscr {up,it}{scr}
1154   \um_set_mathalphabet_char:Nnn  \mathscr {`\B,"1D435}{"212C}
1155   \um_set_mathalphabet_char:Nnn  \mathscr {`\E,"1D438}{"2130}
1156   \um_set_mathalphabet_char:Nnn  \mathscr {`\F,"1D439}{"2131}
1157   \um_set_mathalphabet_char:Nnn  \mathscr {`\H,"1D43B}{"210B}
1158   \um_set_mathalphabet_char:Nnn  \mathscr {`\I,"1D43C}{"2110}
1159   \um_set_mathalphabet_char:Nnn  \mathscr {`\L,"1D43F}{"2112}
1160   \um_set_mathalphabet_char:Nnn  \mathscr {`\M,"1D440}{"2133}
1161   \um_set_mathalphabet_char:Nnn  \mathscr {`\R,"1D445}{"211B}
1162 }
1163 \cs_new:Npn \um_config_mathscr_latin: {
1164   \um_set_mathalphabet_latin:Nnn \mathscr {up,it}{scr}
1165   \um_set_mathalphabet_char:Nnn \mathscr {`\e,"1D452}{"212F}
1166   \um_set_mathalphabet_char:Nnn \mathscr {`\g,"1D454}{"210A}
1167   \um_set_mathalphabet_char:Nnn \mathscr {`\o,"1D45C}{"2134}
1168 }
```

9.1.5 Fractur or fraktur or blackletter: `\mathfrak`

```
1169 \cs_new:Npn \um_config_mathfrak_Latin: {
1170   \um_set_mathalphabet_Latin:Nnn \mathfrak {up,it}{frak}
1171   \um_set_mathalphabet_char:Nnn  \mathfrak {`\C,"1D436}{"212D}
1172   \um_set_mathalphabet_char:Nnn  \mathfrak {`\H,"1D43B}{"210C}
1173   \um_set_mathalphabet_char:Nnn  \mathfrak {`\I,"1D43C}{"2111}
1174   \um_set_mathalphabet_char:Nnn  \mathfrak {`\R,"1D445}{"211C}
1175   \um_set_mathalphabet_char:Nnn  \mathfrak {`\Z,"1D44D}{"2128}
1176 }
1177 \cs_new:Npn \um_config_mathfrak_latin: {
1178   \um_set_mathalphabet_latin:Nnn \mathfrak {up,it}{frak}
1179 }
```

9.1.6 Sans serif upright: `\mathsfup`

```
1180 \cs_new:Npn \um_config_mathsfup_num: {
1181   \um_set_mathalphabet_numbers:Nnn \mathsf   {up}{sf}
1182   \um_set_mathalphabet_numbers:Nnn \mathsfup {up}{sf}
1183 }
1184 \cs_new:Npn \um_config_mathsfup_Latin: {
1185   \bool_if:NTF \g_um_sfliteral_bool {
1186     \um_map_chars_Latin:nn {sfup} {sfup}
1187     \um_set_mathalphabet_Latin:Nnn \mathsf {up}{sfup}
1188   }{
1189     \bool_if:NT \g_um_upsans_bool {
1190       \um_map_chars_Latin:nn {sfup,sfit} {sfup}
```

```
1191        \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{sfup}
1192      }
1193    }
1194    \um_set_mathalphabet_Latin:Nnn \mathsfup {up,it}{sfup}
1195  }
1196  \cs_new:Npn \um_config_mathsfup_latin: {
1197    \bool_if:NTF \g_um_sfliteral_bool {
1198      \um_map_chars_latin:nn {sfup} {sfup}
1199      \um_set_mathalphabet_latin:Nnn \mathsf {up}{sfup}
1200    }{
1201      \bool_if:NT \g_um_upsans_bool {
1202        \um_map_chars_latin:nn {sfup,sfit} {sfup}
1203        \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{sfup}
1204      }
1205    }
1206    \um_set_mathalphabet_latin:Nnn \mathsfup {up,it}{sfup}
1207  }
```

9.1.7   Sans serif italic: `\mathsfit`

```
1208  \cs_new:Npn \um_config_mathsfit_Latin: {
1209    \bool_if:NTF \g_um_sfliteral_bool {
1210      \um_map_chars_Latin:nn {sfit} {sfit}
1211      \um_set_mathalphabet_Latin:Nnn \mathsf {it}{sfit}
1212    }{
1213      \bool_if:NF \g_um_upsans_bool {
1214        \um_map_chars_Latin:nn {sfup,sfit} {sfit}
1215        \um_set_mathalphabet_Latin:Nnn \mathsf {up,it}{sfit}
1216      }
1217    }
1218    \um_set_mathalphabet_Latin:Nnn \mathsfit {up,it}{sfit}
1219  }
1220  \cs_new:Npn \um_config_mathsfit_latin: {
1221    \bool_if:NTF \g_um_sfliteral_bool {
1222      \um_map_chars_latin:nn {sfit} {sfit}
1223      \um_set_mathalphabet_latin:Nnn \mathsf {it}{sfit}
1224    }{
1225      \bool_if:NF \g_um_upsans_bool {
1226        \um_map_chars_latin:nn {sfup,sfit} {sfit}
1227        \um_set_mathalphabet_latin:Nnn \mathsf {up,it}{sfit}
1228      }
1229    }
1230    \um_set_mathalphabet_latin:Nnn \mathsfit {up,it}{sfit}
1231  }
```

9.1.8   Typewriter or monospaced: `\mathtt`

```
1232  \cs_new:Npn \um_config_mathtt_num: {
1233    \um_set_mathalphabet_numbers:Nnn \mathtt {up}{tt}
```

```
1234 }
1235 \cs_new:Npn \um_config_mathtt_Latin: {
1236   \um_set_mathalphabet_Latin:Nnn \mathtt {up,it}{tt}
1237 }
1238 \cs_new:Npn \um_config_mathtt_latin: {
1239   \um_set_mathalphabet_latin:Nnn \mathtt {up,it}{tt}
1240 }
```

9.1.9 Bold Italic: `\mathbfit`

```
1241 \cs_new:Npn \um_config_mathbfit_Latin: {
1242   \bool_if:NF \g_um_bfupLatin_bool {
1243     \um_map_chars_Latin:nn {bfup,bfit} {bfup}
1244   }
1245   \um_set_mathalphabet_Latin:Nnn \mathbfit {up,it}{bfit}
1246   \bool_if:NTF \g_um_bfliteral_bool {
1247     \um_map_chars_Latin:nn {bfit} {bfit}
1248     \um_set_mathalphabet_Latin:Nnn \mathbf {it}{bfit}
1249   }{
1250     \bool_if:NF \g_um_bfupLatin_bool {
1251       \um_map_chars_Latin:nn {bfup,bfit} {bfit}
1252       \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{bfit}
1253     }
1254   }
1255 }
1256 \cs_new:Npn \um_config_mathbfit_latin: {
1257   \bool_if:NF \g_um_bfuplatin_bool {
1258     \um_map_chars_latin:nn {bfup,bfit} {bfit}
1259   }
1260   \um_set_mathalphabet_latin:Nnn \mathbfit {up,it}{bfit}
1261   \bool_if:NTF \g_um_bfliteral_bool {
1262     \um_map_chars_latin:nn {bfit} {bfit}
1263     \um_set_mathalphabet_latin:Nnn \mathbf {it}{bfit}
1264   }{
1265     \bool_if:NF \g_um_bfuplatin_bool {
1266       \um_map_chars_latin:nn {bfup,bfit} {bfit}
1267       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{bfit}
1268     }
1269   }
1270 }
1271 \cs_new:Npn \um_config_mathbfit_Greek: {
1272   \um_set_mathalphabet_Greek:Nnn \mathbfit {up,it}{bfit}
1273   \bool_if:NTF \g_um_bfliteral_bool {
1274     \um_map_chars_Greek:nn {bfit}{bfit}
1275     \um_set_mathalphabet_Greek:Nnn \mathbf {it}{bfit}
1276   }{
1277     \bool_if:NF \g_um_bfupGreek_bool {
```

```
1278        \um_map_chars_Greek:nn {bfup,bfit}{bfit}
1279        \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{bfit}
1280      }
1281    }
1282 }
1283 \cs_new:Npn \um_config_mathbfit_greek: {
1284   \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {bfit}
1285   \bool_if:NTF \g_um_bfliteral_bool {
1286     \um_map_chars_greek:nn {bfit} {bfit}
1287     \um_set_mathalphabet_greek:Nnn \mathbfit {it} {bfit}
1288   }{
1289     \bool_if:NF \g_um_bfupgreek_bool {
1290       \um_map_chars_greek:nn {bfit,bfup} {bfit}
1291     }
1292     \bool_if:NF \g_um_bfupgreek_bool {
1293       \um_set_mathalphabet_greek:Nnn \mathbfit {up,it} {bfit}
1294     }
1295   }
1296 }
1297 \cs_new:Npn \um_config_mathbfit_misc: {
1298   \um_set_mathalphabet_pos:Nnnn  \mathbfit {partial} {up,it}{bfit}
1299   \um_set_mathalphabet_pos:Nnnn  \mathbfit {Nabla}   {up,it}{bfit}
1300   \bool_if:NTF \g_um_bfliteral_bool {
1301     \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {it}{bfit}
1302     \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {it}{bfit}
1303   }{
1304     \bool_if:NF \g_um_upNabla_bool {
1305       \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up,it}{bfit}
1306     }
1307     \bool_if:NF \g_um_uppartial_bool {
1308       \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up,it}{bfit}
1309     }
1310   }
1311 }
```

9.1.10 Bold Upright: `\mathbfup`

```
1312 \cs_new:Npn \um_config_mathbfup_num: {
1313   \um_set_mathalphabet_numbers:Nnn \mathbf   {up}{bfup}
1314   \um_set_mathalphabet_numbers:Nnn \mathbfup {up}{bfup}
1315 }
1316 \cs_new:Npn \um_config_mathbfup_Latin: {
1317   \bool_if:NT \g_um_bfupLatin_bool {
1318     \um_map_chars_Latin:nn {bfup,bfit} {bfit}
1319   }
1320   \um_set_mathalphabet_Latin:Nnn \mathbfup {up,it}{bfup}
1321   \bool_if:NTF \g_um_bfliteral_bool {
```

```
1322    \um_map_chars_Latin:nn {bfup} {bfup}
1323    \um_set_mathalphabet_Latin:Nnn \mathbf {up}{bfup}
1324  }{
1325    \bool_if:NT \g_um_bfupLatin_bool {
1326      \um_map_chars_Latin:nn {bfup,bfit} {bfup}
1327      \um_set_mathalphabet_Latin:Nnn \mathbf {up,it}{bfup}
1328    }
1329  }
1330 }
1331 \cs_new:Npn \um_config_mathbfup_latin: {
1332   \bool_if:NT \g_um_bfuplatin_bool {
1333     \um_map_chars_latin:nn {bfup,bfit} {bfup}
1334   }
1335   \um_set_mathalphabet_latin:Nnn \mathbfup {up,it}{bfup}
1336   \bool_if:NTF \g_um_bfliteral_bool {
1337     \um_map_chars_latin:nn {bfup} {bfup}
1338     \um_set_mathalphabet_latin:Nnn \mathbf {up}{bfup}
1339   }{
1340     \bool_if:NT \g_um_bfuplatin_bool {
1341       \um_map_chars_latin:nn {bfup,bfit} {bfup}
1342       \um_set_mathalphabet_latin:Nnn \mathbf {up,it}{bfup}
1343     }
1344   }
1345 }
1346 \cs_new:Npn \um_config_mathbfup_Greek: {
1347   \um_set_mathalphabet_Greek:Nnn \mathbfup {up,it}{bfup}
1348   \bool_if:NTF \g_um_bfliteral_bool {
1349     \um_map_chars_Greek:nn {bfup}{bfup}
1350     \um_set_mathalphabet_Greek:Nnn \mathbf {up}{bfup}
1351   }{
1352     \bool_if:NF \g_um_bfupGreek_bool {
1353       \um_map_chars_Greek:nn {bfup,bfit}{bfup}
1354       \um_set_mathalphabet_Greek:Nnn \mathbf {up,it}{bfup}
1355     }
1356   }
1357 }
1358 \cs_new:Npn \um_config_mathbfup_greek: {
1359   \um_set_mathalphabet_greek:Nnn \mathbfup {up,it} {bfup}
1360   \bool_if:NTF \g_um_bfliteral_bool {
1361     \um_map_chars_greek:nn {bfup} {bfup}
1362     \um_set_mathalphabet_greek:Nnn \mathbf {up} {bfup}
1363   }{
1364     \bool_if:NT \g_um_bfupgreek_bool {
1365       \um_map_chars_greek:nn {bfup,bfit} {bfup}
1366     }
1367     \bool_if:NT \g_um_bfupgreek_bool {
```

```
1368          \um_set_mathalphabet_greek:Nnn \mathbf {up,it} {bfup}
1369       }
1370    }
1371 }
1372 \cs_new:Npn \um_config_mathbfup_misc: {
1373    \um_set_mathalphabet_pos:Nnnn  \mathbfup {partial} {up,it}{bfup}
1374    \um_set_mathalphabet_pos:Nnnn  \mathbfup {Nabla}   {up,it}{bfup}
1375    \um_set_mathalphabet_pos:Nnnn  \mathbfup {digamma} {up}{bfup}
1376    \um_set_mathalphabet_pos:Nnnn  \mathbfup {Digamma} {up}{bfup}
1377    \um_set_mathalphabet_pos:Nnnn  \mathbf   {digamma} {up}{bfup}
1378    \um_set_mathalphabet_pos:Nnnn  \mathbf   {Digamma} {up}{bfup}
1379    \bool_if:NTF \g_um_bfliteral_bool {
1380       \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up}{bfup}
1381       \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up}{bfup}
1382    }{
1383       \bool_if:NT \g_um_upNabla_bool {
1384          \um_set_mathalphabet_pos:Nnnn  \mathbf {Nabla}   {up,it}{bfup}
1385       }
1386       \bool_if:NT \g_um_uppartial_bool {
1387          \um_set_mathalphabet_pos:Nnnn  \mathbf {partial} {up,it}{bfup}
1388       }
1389    }
1390 }
```

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

```
1391 \cs_new:Npn \um_config_mathbffrak_Latin: {
1392    \um_set_mathalphabet_Latin:Nnn \mathbffrak {up,it}{bffrak}
1393 }
1394 \cs_new:Npn \um_config_mathbffrak_latin: {
1395    \um_set_mathalphabet_latin:Nnn \mathbffrak {up,it}{bffrak}
1396 }
```

9.1.12 Bold script or calligraphic: `\mathbfscr`

```
1397 \cs_new:Npn \um_config_mathbfscr_Latin: {
1398    \um_set_mathalphabet_Latin:Nnn \mathbfscr {up,it}{bfscr}
1399 }
1400 \cs_new:Npn \um_config_mathbfscr_latin: {
1401    \um_set_mathalphabet_latin:Nnn \mathbfscr {up,it}{bfscr}
1402 }
```

9.1.13 Bold upright sans serif: `\mathbfsfup`

```
1403 \cs_new:Npn \um_config_mathbfsfup_num: {
1404    \um_set_mathalphabet_numbers:Nnn \mathbfsf   {up}{bfsfup}
1405    \um_set_mathalphabet_numbers:Nnn \mathbfsfup {up}{bfsfup}
1406 }
1407 \cs_new:Npn \um_config_mathbfsfup_Latin: {
```

```
1408    \bool_if:NTF \g_um_sfliteral_bool {
1409      \um_map_chars_Latin:nn {bfsfup} {bfsfup}
1410      \um_set_mathalphabet_Latin:Nnn \mathbfsf {up}{bfsfup}
1411    }{
1412      \bool_if:NT \g_um_upsans_bool {
1413        \um_map_chars_Latin:nn {bfsfup,bfsfit} {bfsfup}
1414        \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{bfsfup}
1415      }
1416    }
1417    \um_set_mathalphabet_Latin:Nnn \mathbfsfup {up,it}{bfsfup}
1418  }
1419  \cs_new:Npn \um_config_mathbfsfup_latin: {
1420    \bool_if:NTF \g_um_sfliteral_bool {
1421      \um_map_chars_latin:nn {bfsfup} {bfsfup}
1422      \um_set_mathalphabet_latin:Nnn \mathbfsf {up}{bfsfup}
1423    }{
1424      \bool_if:NT \g_um_upsans_bool {
1425        \um_map_chars_latin:nn {bfsfup,bfsfit} {bfsfup}
1426        \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{bfsfup}
1427      }
1428    }
1429    \um_set_mathalphabet_latin:Nnn \mathbfsfup {up,it}{bfsfup}
1430  }
1431  \cs_new:Npn \um_config_mathbfsfup_Greek: {
1432    \bool_if:NTF \g_um_sfliteral_bool {
1433      \um_map_chars_Greek:nn {bfsfup}{bfsfup}
1434      \um_set_mathalphabet_Greek:Nnn \mathbfsf {up}{bfsfup}
1435    }{
1436      \bool_if:NT \g_um_upsans_bool {
1437        \um_map_chars_Greek:nn {bfsfup,bfsfit}{bfsfup}
1438        \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{bfsfup}
1439      }
1440    }
1441    \um_set_mathalphabet_Greek:Nnn \mathbfsfup {up,it}{bfsfup}
1442  }
1443  \cs_new:Npn \um_config_mathbfsfup_greek: {
1444    \bool_if:NTF \g_um_sfliteral_bool {
1445      \um_map_chars_greek:nn {bfsfup} {bfsfup}
1446      \um_set_mathalphabet_greek:Nnn \mathbfsf {up} {bfsfup}
1447    }{
1448      \bool_if:NT \g_um_upsans_bool {
1449        \um_map_chars_greek:nn {bfsfup,bfsfit} {bfsfup}
1450        \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {bfsfup}
1451      }
1452    }
1453    \um_set_mathalphabet_greek:Nnn \mathbfsfup {up,it} {bfsfup}
```

```
1454 }
1455 \cs_new:Npn \um_config_mathbfsfup_misc: {
1456   \um_set_mathalphabet_pos:Nnnn  \mathbfsfup {partial} {up,it}{bfsfup}
1457   \um_set_mathalphabet_pos:Nnnn  \mathbfsfup {Nabla}   {up,it}{bfsfup}
1458   \bool_if:NTF \g_um_sfliteral_bool {
1459     \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up}{bfsfup}
1460     \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up}{bfsfup}
1461   }{
1462     \bool_if:NT \g_um_upNabla_bool {
1463       \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up,it}{bfsfup}
1464     }
1465     \bool_if:NT \g_um_uppartial_bool {
1466       \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up,it}{bfsfup}
1467     }
1468   }
1469 }
```

9.1.14 Bold italic sans serif: `\mathbfsfit`

```
1470 \cs_new:Npn \um_config_mathbfsfit_Latin: {
1471   \bool_if:NTF \g_um_sfliteral_bool {
1472     \um_map_chars_Latin:nn {bfsfit} {bfsfit}
1473     \um_set_mathalphabet_Latin:Nnn \mathbfsf {it}{bfsfit}
1474   }{
1475     \bool_if:NF \g_um_upsans_bool {
1476       \um_map_chars_Latin:nn {bfsfup,bfsfit} {bfsfit}
1477       \um_set_mathalphabet_Latin:Nnn \mathbfsf {up,it}{bfsfit}
1478     }
1479   }
1480   \um_set_mathalphabet_Latin:Nnn \mathbfsfit {up,it}{bfsfit}
1481 }
1482 \cs_new:Npn \um_config_mathbfsfit_latin: {
1483   \bool_if:NTF \g_um_sfliteral_bool {
1484     \um_map_chars_latin:nn {bfsfit} {bfsfit}
1485     \um_set_mathalphabet_latin:Nnn \mathbfsf {it}{bfsfit}
1486   }{
1487     \bool_if:NF \g_um_upsans_bool {
1488       \um_map_chars_latin:nn {bfsfup,bfsfit} {bfsfit}
1489       \um_set_mathalphabet_latin:Nnn \mathbfsf {up,it}{bfsfit}
1490     }
1491   }
1492   \um_set_mathalphabet_latin:Nnn \mathbfsfit {up,it}{bfsfit}
1493 }
1494 \cs_new:Npn \um_config_mathbfsfit_Greek: {
1495   \bool_if:NTF \g_um_sfliteral_bool {
1496     \um_map_chars_Greek:nn {bfsfit}{bfsfit}
1497     \um_set_mathalphabet_Greek:Nnn \mathbfsf {it}{bfsfit}
```

```
1498  }{
1499    \bool_if:NF \g_um_upsans_bool {
1500      \um_map_chars_Greek:nn {bfsfup,bfsfit}{bfsfit}
1501      \um_set_mathalphabet_Greek:Nnn \mathbfsf {up,it}{bfsfit}
1502    }
1503  }
1504  \um_set_mathalphabet_Greek:Nnn \mathbfsfit {up,it}{bfsfit}
1505 }
1506 \cs_new:Npn \um_config_mathbfsfit_greek: {
1507  \bool_if:NTF \g_um_sfliteral_bool {
1508    \um_map_chars_greek:nn {bfsfit} {bfsfit}
1509    \um_set_mathalphabet_greek:Nnn \mathbfsf {it} {bfsfit}
1510  }{
1511    \bool_if:NF \g_um_upsans_bool {
1512      \um_map_chars_greek:nn {bfsfup,bfsfit} {bfsfit}
1513      \um_set_mathalphabet_greek:Nnn \mathbfsf {up,it} {bfsfit}
1514    }
1515  }
1516  \um_set_mathalphabet_greek:Nnn \mathbfsfit {up,it} {bfsfit}
1517 }
1518 \cs_new:Npn \um_config_mathbfsfit_misc: {
1519  \um_set_mathalphabet_pos:Nnnn  \mathbfsfit {partial} {up,it}{bfsfit}
1520  \um_set_mathalphabet_pos:Nnnn  \mathbfsfit {Nabla}   {up,it}{bfsfit}
1521  \bool_if:NTF \g_um_sfliteral_bool {
1522    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {it}{bfsfit}
1523    \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {it}{bfsfit}
1524  }{
1525    \bool_if:NF \g_um_upNabla_bool {
1526      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {Nabla}   {up,it}{bfsfit}
1527    }
1528    \bool_if:NF \g_um_uppartial_bool {
1529      \um_set_mathalphabet_pos:Nnnn  \mathbfsf {partial} {up,it}{bfsfit}
1530    }
1531  }
1532 }
```

10 Definitions of the math symbols

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

\um@scancharlet We need to do some trickery to transform the \UnicodeMathSymbol argument
\um@scanactivedef "ABCDEF into the X$_{\textrm{E}}$TEX 'caret input' form ^^^^^abcdef. It is *very important* that the argument has five characters. Otherwise we need to change the number of ^ chars.

To do this, turn ^ into a regular 'other' character and define the macro to

64

perform the lowercasing and `\let`. `\scantokens` changes the carets back into their original meaning after the group has ended and ^'s catcode returns to normal.

```
1533 \begingroup
1534   \char_make_other:N \^
1535   \cs_gset:Npn \um@scancharlet#1="#2\@nil {
1536     \lowercase{
1537       \scantokens{\global\let#1=^^^^#2}
1538     }
1539   }
```

Making ^ the right catcode isn't strictly necessary right now but it helps to future proof us with, e.g., breqn.

```
1540   \gdef\um@scanactivedef"#1\@nil#2{
1541     \lowercase{
1542       \tl_rescan:nn{
1543         \ExplSyntaxOn
1544         \char_make_math_superscript:N\^
1545       }{
1546         \global\def^^^^#1{#2}
1547       }
1548     }
1549   }
1550 \endgroup
```

Now give `\UnicodeMathSymbol` a definition in terms of `\um@scancharlet` and we're good to go. Make sure # is an 'other' so that we don't get confused with `\mathoctothorpe`.

```
1551 \begingroup
1552   \def\UnicodeMathSymbol#1#2#3#4{
1553     \um@scancharlet#2=#1\@nil
1554   }
1555   \char_make_other:N \#
1556   \@input{unicode-math-table.tex}
1557 \endgroup
```

Fix `\backslash`:

```
1558 \group_begin:
1559   \lccode`\*=`\\
1560   \char_make_escape:N \|
1561   \char_make_other:N \\
1562   |lowercase{
1563 |group_end:|let|backslash=*}
```

11   Epilogue

Lots of little things to tidy up.

11.0.15 Primes

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

> U+2032: PRIME (\primesingle): x'
>
> U+2033: DOUBLE PRIME (\primedouble): x''
>
> U+2034: TRIPLE PRIME (\primetriple): x'''
>
> U+2057: QUADRUPLE PRIME (\primequadruple): x''''

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

> U+2032: PRIME in the 'scriptstyle' font: x'

The shrinking and offsetting is done as it is turned into a superscript. This means, luckily, that by default things work nicely for single primes. We can write x\primesingle or x^\primesingle and get: x' and x'. To support single primes, then, things are easier than in LaTeX; we can just map ' to \prime and not worry about it.

However, it would be nice to use the pre-composed primes above if they exist in the font; consider x''' vs. x'''. 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 \primedouble; if it exists, use it, end; if not, goto last step.

- Ditto pcount=3 & \primetriple.

- Ditto pcount=4 & \primequadruple.

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

```
1564 \muskip_new:N \g_um_primekern_muskip
1565 \muskip_gset:Nn \g_um_primekern_muskip { -\thinmuskip/2 }% arbitrary
1566 \num_new:N \l_um_primecount_num
1567 \cs_new:Nn \um_nprimes:n {
1568   ^{
1569     \primesingle
1570     \prg_replicate:nn {#1-1} { \mskip \g_um_primekern_muskip \primesingle }
```

```
1571    }
1572 }
1573 \cs_new:Nn \um_nprimes_select:n {
1574   \prg_case_int:nnn {#1}{
1575     {1} { ^{\primesingle} }
1576     {2} {
1577     \um_glyph_if_exist:nTF {"2033} { ^{\primedouble} } {\um_nprimes:n {#1}}
1578     }
1579     {3} {
1580     \um_glyph_if_exist:nTF {"2034} {^{\primetriple} } {\um_nprimes:n {#1}}
1581     }
1582     {4} {
1583     \um_glyph_if_exist:nTF {"2057} { ^{\primequadruple} } {\um_nprimes:n {#1}}
1584     }
1585   }{
1586     \um_nprimes:n {#1}
1587   }
1588 }
```

Scanning is more annoying than you'd think because we want to support all three of \prime, ', and the unicode prime. And \ifx doesn't work with mathactive chars.

```
1589 \cs_new:Nn \um_scanprime: {
1590   \num_zero:N \l_um_primecount_num
1591   \um_scanprime_collect:
1592 }
1593 \cs_new:Nn \um_scanprime_collect: {
1594   \num_incr:N \l_um_primecount_num
1595   \peek_meaning_remove:NTF ' {
1596     \um_scanprime_collect:
1597   }{
1598     \peek_meaning_remove:NTF \um_scanprime: {
1599       \um_scanprime_collect:
1600     }{
1601       \peek_meaning_remove:NTF ^^^^2032 {
1602         \um_scanprime_collect:
1603       }{
1604         \um_nprimes_select:n {\l_um_primecount_num}
1605       }
1606     }
1607   }
1608 }
1609 \cs_set_eq:NN \prime \um_scanprime:
1610 \group_begin:
1611   \char_make_active:N \'
1612   \char_make_active:n {"2032}
```

```
1613    \cs_gset_eq:NN ' \um_scanprime:
1614    \cs_gset_eq:NN ^^^^2032 \um_scanprime:
1615 \group_end:
```

11.0.16   Unicode radicals

Undo the damage made to \sqrt:

```
1616 \DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
```

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

```
1617 \def\r@@t#1#2{
1618    \setbox\z@\hbox{$\m@th #1\sqrtsign{#2}$}
1619    \um@scaled@apply{#1}{\kern}{\fontdimen63\l_um_font}
1620    \raise \dimexpr(
1621        \um_fontdimen_to_percent:nn{65}{\l_um_font}\ht\z@-
1622        \um_fontdimen_to_percent:nn{65}{\l_um_font}\dp\z@
1623      )\relax
1624      \copy \rootbox
1625    \um@scaled@apply{#1}{\kern}{\fontdimen64\l_um_font}
1626    \box \z@
1627 }
```

11.0.17   Unicode sub- and super-scripts

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

Open question: should the superscript-like 'modifiers' (U+1D2C: MODIFIER CAP-
ITAL LETTER A and on) be included here?

First, the setup of each mathactive char:

```
1628 \prop_new:N \g_um_supers_prop
1629 \prop_new:N \g_um_subs_prop
1630
1631 \group_begin:
1632
1633 % Populate a property list with superscript characters; their mean-
       ing as their key,
1634 % for reasons that will become apparent soon, and their replace-
       ment as each key's value.
```

```
1635  % Then make the superscript active and bind it to the scanning function.
1636  %
1637  % \cs{scantokens} makes  this  process  much  simpler  since  we  can  acti-
      vate the char
1638  % and assign its meaning in one step.
1639  \cs_set:Nn \um_setup_active_superscript:nn {
1640    \prop_gput:Nxn \g_um_supers_prop   {\meaning #1} {#2}
1641    \char_make_active:n {`#1}
1642    \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1643    \scantokens{
1644      \cs_gset:Npn #1 {
1645        \tl_set:Nn \l_um_ss_chain_tl {#2}
1646        \cs_set_eq:NN \um_sub_or_super:n \sp
1647        \tl_set:Nn \l_um_tmpa_tl {supers}
1648        \um_scan_sscript:
1649      }
1650    }
1651  }
1652
1653  \um_setup_active_superscript:nn {^^^^2070} {0}
1654  \um_setup_active_superscript:nn {^^^^00b9} {1}
1655  \um_setup_active_superscript:nn {^^^^00b2} {2}
1656  \um_setup_active_superscript:nn {^^^^00b3} {3}
1657  \um_setup_active_superscript:nn {^^^^2074} {4}
1658  \um_setup_active_superscript:nn {^^^^2075} {5}
1659  \um_setup_active_superscript:nn {^^^^2076} {6}
1660  \um_setup_active_superscript:nn {^^^^2077} {7}
1661  \um_setup_active_superscript:nn {^^^^2078} {8}
1662  \um_setup_active_superscript:nn {^^^^2079} {9}
1663  \um_setup_active_superscript:nn {^^^^207a} {+}
1664  \um_setup_active_superscript:nn {^^^^207b} {-}
1665  \um_setup_active_superscript:nn {^^^^207c} {=}
1666  \um_setup_active_superscript:nn {^^^^207d} {(}
1667  \um_setup_active_superscript:nn {^^^^207e} {)}
1668  \um_setup_active_superscript:nn {^^^^2071} {i}
1669  \um_setup_active_superscript:nn {^^^^207f} {n}
1670
1671  % Ditto above.
1672  \cs_set:Nn \um_setup_active_subscript:nn {
1673    \prop_gput:Nxn \g_um_subs_prop    {\meaning #1} {#2}
1674    \char_make_active:n {`#1}
1675    \global\XeTeXmathcodenum `#1 = "1FFFFF \scan_stop:
1676    \scantokens{
1677      \cs_gset:Npn #1 {
1678        \tl_set:Nn \l_um_ss_chain_tl {#2}
1679        \cs_set_eq:NN \um_sub_or_super:n \sb
```

```
1680        \tl_set:Nn \l_um_tmpa_tl {subs}
1681        \um_scan_sscript:
1682      }
1683    }
1684  }
1685
1686  \um_setup_active_subscript:nn {^^^^2080} {0}
1687  \um_setup_active_subscript:nn {^^^^2081} {1}
1688  \um_setup_active_subscript:nn {^^^^2082} {2}
1689  \um_setup_active_subscript:nn {^^^^2083} {3}
1690  \um_setup_active_subscript:nn {^^^^2084} {4}
1691  \um_setup_active_subscript:nn {^^^^2085} {5}
1692  \um_setup_active_subscript:nn {^^^^2086} {6}
1693  \um_setup_active_subscript:nn {^^^^2087} {7}
1694  \um_setup_active_subscript:nn {^^^^2088} {8}
1695  \um_setup_active_subscript:nn {^^^^2089} {9}
1696  \um_setup_active_subscript:nn {^^^^208a} {+}
1697  \um_setup_active_subscript:nn {^^^^208b} {-}
1698  \um_setup_active_subscript:nn {^^^^208c} {=}
1699  \um_setup_active_subscript:nn {^^^^208d} {(}
1700  \um_setup_active_subscript:nn {^^^^208e} {)}
1701  \um_setup_active_subscript:nn {^^^^2090} {a}
1702  \um_setup_active_subscript:nn {^^^^2091} {e}
1703  \um_setup_active_subscript:nn {^^^^1d62} {i}
1704  \um_setup_active_subscript:nn {^^^^2092} {o}
1705  \um_setup_active_subscript:nn {^^^^1d63} {r}
1706  \um_setup_active_subscript:nn {^^^^1d64} {u}
1707  \um_setup_active_subscript:nn {^^^^1d65} {v}
1708  \um_setup_active_subscript:nn {^^^^2093} {x}
1709  \um_setup_active_subscript:nn {^^^^1d66} {\beta}
1710  \um_setup_active_subscript:nn {^^^^1d67} {\gamma}
1711  \um_setup_active_subscript:nn {^^^^1d68} {\rho}
1712  \um_setup_active_subscript:nn {^^^^1d69} {\phi}
1713  \um_setup_active_subscript:nn {^^^^1d6a} {\chi}
1714
1715  \group_end:
1716
1717  % The scanning command, evident in its purpose:
1718  \cs_new:Nn \um_scan_sscript: {
1719    \um_scan_sscript:TF {
1720      \um_scan_sscript:
1721    }{
1722      \um_sub_or_super:n {\l_um_ss_chain_tl}
1723    }
1724  }
1725
```

```
1726  % The  main  theme  here  is  stolen  from  the  source  to  the  vari-
      ous \cs{peek_} functions.
1727  % Consider this function as simply boilerplate:
1728  \cs_new:Nn \um_scan_sscript:TF {
1729    \tl_set:Nx \l_peek_true_aux_tl { \exp_not:n{ #1 } }
1730    \tl_set_eq:NN \l_peek_true_tl \c_peek_true_remove_next_tl
1731    \tl_set:Nx \l_peek_false_tl {\exp_not:n{\group_align_safe_end: #2}}
1732    \group_align_safe_begin:
1733      \peek_after:NN \um_peek_execute_branches_ss:
1734  }
1735
1736  % We do not skip spaces when scanning ahead, and we explicitly wish to
1737  % bail out on encountering a space or a brace.
1738  \cs_new:Npn \um_peek_execute_branches_ss: {
1739    \bool_if:nTF {
1740      \token_if_eq_catcode_p:NN \l_peek_token \c_group_begin_token ||
1741      \token_if_eq_catcode_p:NN \l_peek_token \c_group_end_token ||
1742      \token_if_eq_meaning_p:NN \l_peek_token \c_space_token
1743    }
1744    { \l_peek_false_tl  }
1745    { \um_peek_execute_branches_ss_aux: }
1746  }
1747
1748  % This is the actual comparison code.
1749  % Because the peeking has already tokenised the next token,
1750  % it's too late to extract its charcode directly. Instead,
1751  % we look at its meaning, which remains a `character' even
1752  % though it is itself math-active. If the character is ever
1753  % made fully active, this will break our assumptions!
1754  %
1755  % If the char's meaning exists as a property list key, we
1756  % build up a chain of sub-/superscripts and iterate. (If not, exit and
1757  % typeset what we've already collected.)
1758  \cs_new:Nn \um_peek_execute_branches_ss_aux: {
1759    \prop_if_in:cxTF
1760      {g_um_\l_um_tmpa_tl _prop}
1761      {\meaning\l_peek_token}
1762      {
1763        \prop_get:cxN
1764          {g_um_\l_um_tmpa_tl _prop}
1765          {\meaning\l_peek_token}
1766          \l_um_tmpb_tl
1767        \tl_put_right:NV \l_um_ss_chain_tl \l_um_tmpb_tl
1768        \l_peek_true_tl
1769      }
1770      {\l_peek_false_tl}
```

```
1771 }
```

11.0.18   Synonyms and all the rest

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

```
1772 \def\operator@font{\um_setup_mathup:}
```

```
1773 \def\to{\rightarrow}
1774 \def\vec{\overrightarrow}
1775 \def\le{\leq}
1776 \def\ge{\geq}
1777 \def\neq{\ne}
```

Define `\colon` as a mathpunct ':'. This is wrong: it should be u+003a: colon instead!

```
1778 \@ifpackageloaded{amsmath}{
1779   % define their own colon, perhaps I should just steal it.
1780 }{
1781   \cs_set_protected:Npn \colon {
1782     \bool_if:NTF \g_um_literal_colon_bool {:} { \mathpunct{:} }
1783   }
1784 }
```

`\mathcal`

```
1785 \def\mathcal{\mathscr}
```

`\mathrm`

```
1786 \def\mathrm{\mathup}
```

11.0.19   Compatibility

Note that amsmath will always be loaded before unicode-math. (Conflicts occur if you try it the other way around.)

- 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:

```
1787   \@ifpackageloaded{amsmath}{
1788     \tl_remove_in:Nn \@begindocumenthook {
1789       \mathchardef\std@minus\mathcode`\-\relax
1790       \mathchardef\std@equal\mathcode`\=\relax
1791     }
1792   }{}
```

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

```
1793    \@ifpackageloaded{amsopn}{
1794      \cs_set:Npn \newmcodes@ {
1795        \mathcode`\'39
1796        \mathcode`\*42
1797        \mathcode`\."613A%
1798 %  \ifnum\mathcode`\-=45 \else
1799 %    \mathchardef\std@minus\mathcode`\-\relax
1800 %  \fi
1801        \mathcode`\-45
1802        \mathcode`\/47
1803        \mathcode`\:"603A\relax
1804      }
1805    }{}
```

Octothorpe is an odd one:

```
1806 \AtBeginDocument{
1807   \def\#{\mode_if_math:TF{\mathoctothorpe}{\char`\#}}
1808   \def\widehat{\hat}
1809   \def\widetilde{\tilde}
1810 }
```

\digamma   I might end up just changing these in the table.
\Digamma
```
1811 \def\digamma{\updigamma}
1812 \def\Digamma{\upDigamma}
```

Overriding amsmath definitions:

```
1813 \AtBeginDocument{
1814   \def\@cdots{\mathinner{\cdots}}
1815 }
```

Interaction with beamer:

```
1816 \@ifclassloaded{beamer}{
1817   \ifbeamer@suppressreplacements\else
1818     \PackageWarningNoLine{unicode-math}{
1819       Disabling~ beamer's~ math~ setup.^^J
1820       Please~ load~ beamer~ with~ the~ [professionalfonts]~ class~ option
1821     }
1822     \beamer@suppressreplacementstrue
1823   \fi
1824 }{}
```

The end.

```
1825 \ExplSyntaxOff
```

12 STIX table data extraction

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

This table is converted into a form suitable for reading by XƎTeX, and then
hand-edited by the author; the result is unicode-math-table.tex.

A single file is produced containing all (more than 3298) symbols. Future op-
timisations might include generating various (possibly overlapping) subsets so
not all definitions must be read just to redefine a small range of symbols. Perfor-
mance for now seems to be acceptable without such measures.

```
1826  #!/bin/sh
1827
1828  cat stix-tbl.txt |
1829  awk '
```

If the USV isn't repeated (TODO: check this is valid!) and the entry isn't one of the
weird ones in the big block at the end of the STIX table (TODO: check that out!)…

```
1830  {if (usv != substr($0,2,5) && substr($0,2,1) != " ")
1831    {usv = substr($0,2,5);
1832     texname = substr($0,84,25);
1833     class = substr($0,57,1);
1834     description = tolower(substr($0,233,350));
```

If the USV has a macro name, which isn't \text..., and isn't a single character
macro (e.g., \#, \S, …), and has a class, and it isn't reserved (*i.e.,* doubled up with
a previously assigned glyph):

```
1835     if (texname      ~ /[\\]/ &&
1836        substr(texname,0,5) != "\\text"    &&
1837        substr(texname,0,4) != "\\ipa"    &&
1838        substr(texname,0,5) != "\\tone"    &&
1839        substr(texname,3,1) != " "    &&
1840        class       != " "    &&
1841        description !~ /<reserved>/ )
```

Print the actual entry corresponding to the unicode character:

```
1842     print "\\UnicodeMathSymbol{\"" \
1843        usv "}{" \
1844        texname "}{" \
1845        class "}{" \
1846        description "}%";
1847    }}' - |
```

Now replace the STIX class abbreviations with their TeX macro names.

```
1848  sed -e ' s/{N}/{\\mathord}/    ' \
```

74

A 'fence' defined by the stix table is something like \vert; in X∃TEX this is just a
\mathord that will grow with the magic of \XeTeXmathchardef.

```
1849      -e ' s/{F}/{\\mathord}/    ' \
1850      -e ' s/{A}/{\\mathalpha}/ ' \
1851      -e ' s/{D}/{\\mathaccent}/ ' \
1852      -e ' s/{P}/{\\mathpunct}/ ' \
1853      -e ' s/{B}/{\\mathbin}/    ' \
1854      -e ' s/{R}/{\\mathrel}/    ' \
1855      -e ' s/{L}/{\\mathop}/     ' \
1856      -e ' s/{O}/{\\mathopen}/   ' \
1857      -e ' s/{C}/{\\mathclose}/ ' \
```

Fixing up a couple of things in the STIX table.

```
1858      -e ' s/\^/\\string^/    ' > unicode-math.tex
```

A Documenting maths support in the NFSS

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

Maths symbol fonts Fonts for symbols: ∝, ≤, →

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

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

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

For commands such as \mathbf, accessed through maths mode that are un-
affected by the current text font, and which are used for alphabetic symbols
in the ascii range.

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

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

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

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

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

75

Delimiters and radicals use wrappers around TₑX's \delimiter/\radical primitives, which are re-designed in X∃TₑX. The syntax used in LATₑX's NFSS is therefore not so relevant here.

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

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

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

In those cases, glyph slots in *two* symbol fonts are required; one for the small ('regular') case, the other for situations when the glyph is larger. This is not the case in X∃TₑX.

Accents are not included yet.

Summary For symbols, something like:

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

For characters, something like:

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

B   X∃TₑX math font dimensions

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

\fontdimen	Dimension name	Description
10	SCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 1. Suggested value: 80%.
11	SCRIPTSCRIPTPERCENTSCALE-DOWN	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.

\fontdimen	Dimension name	Description
12	DELIMITEDSUBFORMULAMIN-HEIGHT	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height × 1.5.
13	DISPLAYOPERATORMINHEIGHT	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.
14	MATHLEADING	White space to be left between math formulas to ensure proper line spacing. For example, for applications that treat line gap as a part of line ascender, formulas with ink going above (os2.sTypoAscender + os2.sTypoLineGap − MathLeading) or with ink going below os2.sTypoDescender will result in increasing line height.
15	AXISHEIGHT	Axis height of the font.
16	ACCENTBASEHEIGHT	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (os2.sxHeight) plus any possible overshots.
17	FLATTENEDACCENTBASE-HEIGHT	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font (os2.sCapHeight).
18	SUBSCRIPTSHIFTDOWN	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: os2.ySubscriptYOffset.
19	SUBSCRIPTTOPMAX	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: /5 x-height.
20	SUBSCRIPTBASELINEDROPMIN	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.
21	SUPERSCRIPTSHIFTUP	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.

\fontdimen	Dimension name	Description
22	SUPERSCRIPTSHIFTUPCRAMPED	Standard shift of superscripts relative to the base, in cramped style.
23	SUPERSCRIPTBOTTOMMIN	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: ¼ x-height.
24	SUPERSCRIPTBASELINEDROP-MAX	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SUBSUPERSCRIPTGAPMIN	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	SUPERSCRIPTBOTTOMMAX-WITHSUBSCRIPT	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: /5 x-height.
27	SPACEAFTERSCRIPT	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	UPPERLIMITGAPMIN	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	UPPERLIMITBASELINERISEMIN	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	LOWERLIMITGAPMIN	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LOWERLIMITBASELINEDROP-MIN	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	STACKTOPSHIFTUP	Standard shift up applied to the top element of a stack.
33	STACKTOPDISPLAYSTYLESHIFT-UP	Standard shift up applied to the top element of a stack in display style.
34	STACKBOTTOMSHIFTDOWN	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.

\fontdimen	Dimension name	Description
35	STACKBOTTOMDISPLAYSTYLE-SHIFTDOWN	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	STACKGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.
37	STACKDISPLAYSTYLEGAPMIN	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	STRETCHSTACKTOPSHIFTUP	Standard shift up applied to the top element of the stretch stack.
39	STRETCHSTACKBOTTOMSHIFT-DOWN	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	STRETCHSTACKGAPABOVEMIN	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: UpperLimitGapMin
41	STRETCHSTACKGAPBELOWMIN	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: LowerLimitGapMin.
42	FRACTIONNUMERATORSHIFTUP	Standard shift up applied to the numerator.
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.

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

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

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

87