Experimental Unicode mathematical typesetting: The unicode-math package

Will Robertson
Philipp Stephani, Joseph Wright, Khaled Hosny, and others
http://github.com/wspr/unicode-math

2018/02/02 v0.81

Contents

File I
unicode-math.dtx

1 *Package metadata*

List all `dtx` files for (a) the `ins` file and (b) typesetting the code.

```
⟨*dtx⟩
\def\DTXFILES{
  \DTX{unicode-math.dtx}
  \DTX{um-code-opening.dtx}
  \DTX{um-code-variables.dtx}
  \DTX{um-code-api.dtx}
  \DTX{um-code-ui.dtx}
  \DTX{um-code-pkgopt.dtx}
  \DTX{um-code-msg.dtx}
  \DTX{um-code-usv.dtx}
  \DTX{um-code-setchar.dtx}
  \DTX{um-code-mathtext.dtx}
  \DTX{um-code-main.dtx}
  \DTX{um-code-fontopt.dtx}
  \DTX{um-code-fontparam.dtx}
  \DTX{um-code-mathmap.dtx}
  \DTX{um-code-sym-commands.dtx}
  \DTX{um-code-alphabets.dtx}
  \DTX{um-code-primes.dtx}
  \DTX{um-code-sscript.dtx}
  \DTX{um-code-compat.dtx}
  \DTX{um-code-amsmath.dtx}
  \DTX{um-code-epilogue.dtx}
}
⟨/dtx⟩
```

Now exit if we're using plain TEX when loading this file with `unicode-math.ins`.

```
⟨*dtx⟩
\ifx\plainoutput\undefined\else\expandafter\endinput\fi
⟨/dtx⟩
```

Metadata for documentation; the title and authors of the package.

```
⟨*dtx⟩
\title{
  Experimental Unicode mathematical typesetting:
  The \pkg{unicode-math} package
}
\author{
  \scshape Will Robertson\\
  \itshape Philipp Stephani, Joseph Wright, Khaled Hosny, and others\\
  \url{http://github.com/wspr/unicode-math}
}
⟨/dtx⟩
```

Declare the package version and date.

```
⟨base⟩\RequirePackage{expl3}
⟨base⟩\ProvidesExplPackage{unicode-math}
⟨package&XE⟩\ProvidesExplPackage{unicode-math-xetex}
⟨package&LU⟩\ProvidesExplPackage{unicode-math-luatex}
⟨base|package⟩  {2018/02/02} {v0.8l} {Unicode maths in XeLaTeX and LuaLaTeX}
```

Here the version and date are setup for typesetting the documentation.

```
⟨*dtx⟩
\date{
  \def\filedate{2018/02/02}
  \def\fileversion{v0.8l}
  \filedate \qquad \fileversion
}
⟨/dtx⟩
```

2 *The* `unicode-math.sty` *loading file*

The `unicode-math.sty` file is a stub which loads necessary packages and then splits into a XeTeX- or LuaTeX-specific version of the package.

```
⟨base⟩\sys_if_engine_luatex:T { \RequirePackageWithOptions{unicode-math-luatex} }
⟨base⟩\sys_if_engine_xetex:T  { \RequirePackageWithOptions{unicode-math-xetex}  }
⟨base⟩\sys_if_engine_pdftex:T
⟨base⟩  {
⟨base⟩    \msg_new:nnn {unicode-math} {not-pdftex}
⟨base⟩      { Cannot~ be~ run~ with~ pdfLaTeX!\\ Use~ XeLaTeX~ or~ LuaLaTeX~ in-
stead. }
⟨base⟩    \msg_error:nn {unicode-math} {not-pdftex}
⟨base⟩  }
⟨base⟩\endinput
```

File II
um-code-opening.dtx

3 *Start of the package code*

The prefix for unicode-math is um:

```
⟨@@=um⟩
```

```
⟨*package⟩
```

Packages Assuming people are running up-to-date packages.

```
\RequirePackage{xparse,l3keys2e}
\RequirePackage{fontspec}
\RequirePackage{filehook}
\RequirePackage{fix-cm}
⟨LU⟩\RequirePackage{lualatex-math}
⟨debug⟩\RequirePackage{l3benchmark}
```

3.1 *expl3 variants*

Variants needed from expl3:

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

For fontspec:

```
\cs_generate_variant:Nn \fontspec_set_family:Nnn {Nx}
\cs_generate_variant:Nn \prop_get:NnNTF {cx}
\cs_generate_variant:Nn \tl_if_eq:nnF {o}
```

3.2 *Low level commands*

```
\cs_set_eq:NN \@@_group_begin: \group_begin:
\cs_set_protected:Npn \@@_group_end:n #1 { #1 \group_end: }
\cs_set_eq:NN \@@_group_begin_frozen: \@@_group_begin:
\cs_set_eq:NN \@@_group_end_frozen:n  \@@_group_end:n
```

3.3 *Primitive font commands*

What might end up being provided by the kernel.

\@@_glyph_if_exist:NnTF

```
\prg_new_conditional:Nnn \@@_glyph_if_exist:Nn {p,TF,T,F}
  {
    \etex_iffontchar:D #1 #2 \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

\@@_fontface_gset_eq:NN

```
\cs_set_protected:Nn \@@_fontface_gset_eq:NN
  {
    \tex_global:D \tex_let:D #1 #2
  }
\cs_generate_variant:Nn \@@_fontface_gset_eq:NN {cN}
```

3.3.1 *Mathcode and friends*

\@@_set_mathcode:nnnn
\@@_set_mathcode:nnn

These are all wrappers for the primitive commands that take numerical input only.

```
\cs_set:Npn \@@_set_mathcode:nnnn #1#2#3#4
  {
    \Umathcode \int_eval:n {#1} =
      \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
  }
\cs_set:Npn \@@_set_mathcode:nnn #1#2#3
  {
    \Umathcode \int_eval:n {#1} =
      \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#1} \scan_stop:
  }
```

\@@_set_mathchar:NNnn
\@@_set_mathchar:cNnn

```
\cs_set:Npn \@@_set_mathchar:NNnn #1#2#3#4
  {
    \Umathchardef #1 =
      \mathchar@type#2 \csname sym#3\endcsname \int_eval:n {#4} \scan_stop:
  }
\cs_generate_variant:Nn \@@_set_mathchar:NNnn {c}
```

\@@_set_delcode:nnn

```
\cs_new:Nn \@@_set_delcode:nnn
  {
    \Udelcode#2 = \csname sym#1\endcsname #3 \scan_stop:
  }
```

\@@_radical:nn

```
\cs_new:Nn \@@_radical:nn
  {
    \Uradical \csname sym#1\endcsname #2 \scan_stop:
  }
```

\@@_delimiter:Nnn

```
\cs_new:Nn \@@_delimiter:Nnn
  {
    \Udelimiter \mathchar@type#1 \csname sym#2\endcsname #3 \scan_stop:
  }
```

\@@_accent:nnn

```
\cs_new:Nn \@@_accent:nnn
  {
   \Umathaccent #1~ \mathchar@type\mathaccent \use:c { sym #2 } #3 \scan_stop:
  }
```

\@@_char_gmake_mathactive:N
\@@_char_gmake_mathactive:n

```
\cs_new:Nn \@@_char_gmake_mathactive:N
  {
    \tex_global:D \tex_mathcode:D `#1 = "8000 \scan_stop:
  }
\cs_new:Nn \@@_char_gmake_mathactive:n
  {
    \tex_global:D \tex_mathcode:D \int_eval:n {#1} = "8000 \scan_stop:
  }
```

\@@_mathactive_remap:nn

Makes #1 math-active and defines its meaning to be #2. This is a global operation.

```
\cs_new:Nn \@@_mathactive_remap:nn
  {
    \group_begin:
      \cs_set_protected:Npn \@@_tmp: {#2}
      \@@_char_gmake_mathactive:n {#1}
      \char_gset_active_eq:nN {#1} \@@_tmp:
    \group_end:
  }
```

3.3.2 *NFSS-related interfaces*

\@@_mathgroup_set:n

Remember that \mathgroup is just \fam!

```
\cs_new_protected:Nn \@@_mathgroup_set:n
  {
    \tex_fam:D #1 \scan_stop:
  }
```

3.3.3 *Font parameters*

\@@_copy_fontdimen:nnN

```
\cs_new:Nn \@@_copy_fontdimen:nnN
  {
    \fontdimen #1 \font = \the \fontdimen #2 #3 \relax
  }
```

\@@_zero_fontdimen:n

```
\cs_new:Nn \@@_zero_fontdimen:n
  {
    \fontdimen #1 \font = 0pt\relax
  }
```

\@@_fontdimen_from_param:Nnn

This function extracts the math font dimen #3 from the font #1 and sets fontdimen #2 of the same font to that value.

Use XƎTEX's fontdimen approach because it's tidy. We don't need bells and whistles here.

```
⟨*LU⟩
\cs_new_protected:Nn \@@_fontdimen_from_param:nn
  {
    \fontdimen #1 \font =
      \lua_now:n { fontspec.mathfontdimen(font.current(),"#2") }
    \scan_stop:
  }
⟨/LU⟩
```

3.4 *Alphabet Unicode positions (USVs)*

Before we begin, let's define the positions of the various Unicode alphabets so that our code is a little more readable.[1]

\usv_set:nnn,\@@_to_usv:nn

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

```
\cs_new:Nn \usv_set:nnn  { \tl_const:cn { c_@@_#1_#2_usv } {#3} }
\cs_new:Nn \@@_to_usv:nn {       \use:c { c_@@_#1_#2_usv } }
```

\@@_usv_if_exist:nnTF

```
\prg_new_conditional:Nnn \@@_usv_if_exist:nn {T,F,TF}
  {
    \cs_if_exist:cTF { c_@@_#1_#2_usv }
      \prg_return_true: \prg_return_false:
  }
```

3.5 *Overcoming \@onlypreamble*

The requirement of only setting up the maths fonts in the preamble is lifted. (Perhaps unwisely.)

```
\tl_map_inline:nn
  {
    \new@mathgroup\cdp@list\cdp@elt\DeclareMathSizes
    \@DeclareMathSizes\newmathalphabet\newmathalphabet@@\newmathalphabet@@@
    \DeclareMathVersion\define@mathalphabet\define@mathgroup\addtoversion
    \version@list\version@elt\alpha@list\alpha@elt
    \restore@mathversion\init@restore@version\dorestore@version\process@table
    \new@mathversion\DeclareSymbolFont\group@list\group@elt
    \new@symbolfont\SetSymbolFont\SetSymbolFont@\get@cdp
    \DeclareMathAlphabet\new@mathalphabet\SetMathAlphabet\SetMathAlphabet@
    \DeclareMathAccent\set@mathaccent\DeclareMathSymbol\set@mathchar
    \set@mathsymbol\DeclareMathDelimiter\@xxDeclareMathDelimiter
    \@DeclareMathDelimiter\@xDeclareMathDelimiter\set@mathdelimiter
    \set@@mathdelimiter\DeclareMathRadical\mathchar@type
```

[1]'u.s.v.' stands for 'Unicode scalar value'.

```
    \DeclareSymbolFontAlphabet\DeclareSymbolFontAlphabet@
  }
  {
    \tl_remove_once:Nn \@preamblecmds {\do#1}
  }
```

3.6 *Wrappers for message functions*

Messages themselves are defined in section §8.

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

```
\cs_generate_variant:Nn \msg_new:nnn  {nnx}
\cs_generate_variant:Nn \msg_new:nnnn {nnxx}
\cs_new:Nn \@@_msg_new:nn  { \msg_new:nnx  {unicode-math} {#1} { \tl_trim_spaces:n {#2} } }
```

```
⟨/package⟩
```

File III

um-code-variables.dtx

4 *Variable initialisation*

```
⟨*package⟩
```

4.1 *bool*

True if using a proper OpenType font with unicode maths

```
\bool_new:N \l_@@_ot_math_bool
```

Set when \setmathfont is run to trap the problem of no main font defined.

```
\bool_new:N \g_@@_main_font_defined_bool
```

```
\bool_new:N \l_@@_init_bool
\bool_new:N \l_@@_implicit_alph_bool
```

For math-style:

```
\bool_new:N \g_@@_literal_bool
\bool_new:N \g_@@_upLatin_bool
\bool_new:N \g_@@_uplatin_bool
\bool_new:N \g_@@_upGreek_bool
\bool_new:N \g_@@_upgreek_bool
```

For bold-style:

```
\bool_new:N \g_@@_bfliteral_bool
\bool_new:N \g_@@_bfupLatin_bool
\bool_new:N \g_@@_bfuplatin_bool
\bool_new:N \g_@@_bfupGreek_bool
\bool_new:N \g_@@_bfupgreek_bool
```

For sans-style:

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

For assorted package options:

```
\bool_new:N \g_@@_upNabla_bool
\bool_new:N \g_@@_uppartial_bool
\bool_new:N \g_@@_literal_Nabla_bool
\bool_new:N \g_@@_literal_partial_bool
\bool_new:N \l_@@_smallfrac_bool
\bool_new:N \g_@@_literal_colon_bool
\bool_new:N \g_@@_mathrm_text_bool
\bool_new:N \g_@@_mathit_text_bool
\bool_new:N \g_@@_mathbf_text_bool
\bool_new:N \g_@@_mathsf_text_bool
\bool_new:N \g_@@_mathtt_text_bool
```

4.2 *int*

```
\int_new:N \g_@@_fam_int
\int_new:N \g_@@_fonts_used_int
\int_new:N \l_@@_primecount_int
```

4.3 *tl*

For displaying in warning messages, etc.:

```
\tl_const:Nn \c_@@_math_alphabet_name_latin_tl {Latin,~lowercase}
\tl_const:Nn \c_@@_math_alphabet_name_Latin_tl {Latin,~uppercase}
\tl_const:Nn \c_@@_math_alphabet_name_greek_tl {Greek,~lowercase}
\tl_const:Nn \c_@@_math_alphabet_name_Greek_tl {Greek,~uppercase}
\tl_const:Nn \c_@@_math_alphabet_name_num_tl   {Numerals}
\tl_const:Nn \c_@@_math_alphabet_name_misc_tl  {Misc.}
```

```
\tl_new:N \l_@@_mathstyle_tl
\tl_new:N \l_@@_radicals_tl
\tl_new:N \l_@@_nolimits_tl
```

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

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

```
\tl_new:N \g_@@_slash_delimiter_usv
\tl_new:N \g_@@_mathtable_tl
\tl_new:N \g_@@_fontname_tl
\tl_new:N \g_@@_mversion_tl
\tl_new:N \g_@@_symfont_tl
\tl_new:N \g_@@_font_keyval_tl
\tl_new:N \g_@@_family_tl
\tl_new:N \g_@@_style_tl
\tl_new:N \g_@@_remap_style_tl
\tl_new:N \l_@@_not_token_name_tl
\tl_new:N \g_@@_curr_font_cmd_tl
\tl_new:N \g_@@_sqrt_font_cmd_tl
\tl_new:N \g_@@_prime_font_cmd_tl
```

`\g_@@_mathparam_store_tl` Used to store and restore the math parameters used in LuaTEX. This is done to 'save' the values of the *first* (or main) maths font loaded, rather than (as per LuaTEX defaults) the last.

```
⟨*LU⟩
\tl_new:N \g_@@_mathparam_store_tl
⟨/LU⟩
```

4.4 *clist*

```
\clist_new:N \g_@@_char_nrange_clist
\clist_new:N \g_@@_unknown_keys_clist
\clist_new:N \g_@@_alphabet_clist
```

```
\clist_new:N \g_@@_bad_alpha_clist
\clist_put_right:Nx \g_@@_bad_alpha_clist { \tl_to_str:n {bf} }
```

```
\clist_put_right:Nx \g_@@_bad_alpha_clist { \tl_to_str:n {sf} }
\clist_put_right:Nx \g_@@_bad_alpha_clist { \tl_to_str:n {bfsf} }
```

4.5 *seq*

```
\seq_new:N \l_@@_missing_alph_seq
\seq_new:N \g_@@_mathalph_seq
\seq_new:N \g_@@_char_range_seq
\seq_new:N \g_@@_mclass_range_seq
```

`\g_@@_mathclasses_seq` Every math class.

```
\seq_new:N \g_@@_mathclasses_seq
\seq_set_from_clist:Nn \g_@@_mathclasses_seq
  {
    \mathord,\mathalpha,\mathbin,\mathrel,\mathpunct,
     \mathop,
    \mathopen,\mathclose,
    \mathfence,\mathover,\mathunder,
   \mathaccent,\mathaccentoverlay,\mathbotaccent,\mathaccentwide,\mathbotaccentwide
  }
```

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

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

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

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

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

4.6 *prop*

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

4.7 *muskip*

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

4.8 *fp*

```
\fp_new:N \g_@@_size_tfsf_fp
\fp_new:N \g_@@_size_sfssf_fp
```

4.9 *quark*

`\q_unicode_math` Used as a flag within control sequences to check they're recognised by the package.

```
\quark_new:N \q_unicode_math
```

```
⟨/package⟩
```

File IV

um-code-api.dtx

5 *Programmers' interface*

```
⟨*package⟩
```

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

```
\cs_new:Nn \unimath_get_mathstyle:
 {
  \tl_use:N \l_@@_mathstyle_tl
 }
```

```
⟨/package⟩
```

File V

um-code-ui.dtx

6 *The user interface commands*

```
⟨*package⟩
```

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

```
\NewDocumentCommand \unimathsetup {m} { \keys_set:nn {unicode-math} {#1} }
```

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

```
\NewDocumentCommand \setmathfont { O{} m O{} }
  {
    \@@_setmathfont:nn {#1,#3} {#2}
  }
```

\setmathfontface

```
\NewDocumentCommand \setmathfontface { m O{} m O{} }
  {
    \@@_setmathfontface:Nnn #1 {#2,#4} {#3}
  }
```

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

```
\@onlypreamble \setmathfontface
```

\setoperatorfont TODO: add check?

```
\NewDocumentCommand \setoperatorfont {m}
  {
    \tl_set:Nn \g_@@_operator_mathfont_tl {#1}
  }
\setoperatorfont{\mathrm}
```

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

```
\NewDocumentCommand \addnolimits {m}
  {
    \tl_put_right:Nn \l_@@_nolimits_tl {#1}
  }
```

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

```
\NewDocumentCommand \removenolimits {m}
  {
    \tl_remove_all:Nn \l_@@_nolimits_tl {#1}
  }
```

```
⟨/package⟩
```

File VI

um-code-pkgopt.dtx

7 *setup and package options*

```
⟨*package⟩
```

\@@_keys_choices:nn

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

```
\cs_new:Nn \@@_keys_choices:nn
  {
    \cs_set:Npn \@@_keys_choices_fn:nn { \@@_keys_choices_aux:nnn {#1} }
    \use:x
      {
        \exp_not:N \keys_define:nn {unicode-math}
          {
            #1 .choice: ,
            \@@_tl_map_dbl:nN {#2} \@@_keys_choices_fn:nn
          }
      }
  }
```

```
\cs_new:Nn \@@_keys_choices_aux:nnn { #1 / #2 .code:n = { \exp_not:n {#3} } , }
```

```
\cs_new:Nn \@@_tl_map_dbl:nN
  {
    \__@@_tl_map_dbl:Nnn #2 #1 \q_recursion_tail {}{} \q_recursion_stop
  }
```

```
\cs_new:Nn \__@@_tl_map_dbl:Nnn
  {
    \quark_if_recursion_tail_stop:n {#2}
    \quark_if_recursion_tail_stop:n {#3}
    #1 {#2} {#3}
    \__@@_tl_map_dbl:Nnn #1
 }
```

Compatibility

```
\@@_keys_choices:nn {mathup}
  {
    {sym}  { \bool_set_false:N \g_@@_mathrm_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathrm_text_bool }
  }
```

```
\@@_keys_choices:nn {mathrm}
  {
    {sym}  { \bool_set_false:N \g_@@_mathrm_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathrm_text_bool }
  }
```

```
\@@_keys_choices:nn {mathit}
  {
    {sym}  { \bool_set_false:N \g_@@_mathit_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathit_text_bool }
  }
\@@_keys_choices:nn {mathbf}
  {
    {sym}  { \bool_set_false:N \g_@@_mathbf_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathbf_text_bool }
  }
\@@_keys_choices:nn {mathsf}
  {
    {sym}  { \bool_set_false:N \g_@@_mathsf_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathsf_text_bool }
  }
\@@_keys_choices:nn {mathtt}
  {
    {sym}  { \bool_set_false:N \g_@@_mathtt_text_bool }
    {text} { \bool_set_true:N  \g_@@_mathtt_text_bool }
  }
```

math-style

```
\@@_keys_choices:nn {normal-style}
  {
       {ISO} {
              \bool_set_false:N \g_@@_literal_bool
              \bool_set_false:N \g_@@_upGreek_bool
              \bool_set_false:N \g_@@_upgreek_bool
              \bool_set_false:N \g_@@_upLatin_bool
              \bool_set_false:N \g_@@_uplatin_bool
             }
       {TeX} {
              \bool_set_false:N \g_@@_literal_bool
              \bool_set_true:N  \g_@@_upGreek_bool
              \bool_set_false:N \g_@@_upgreek_bool
              \bool_set_false:N \g_@@_upLatin_bool
              \bool_set_false:N \g_@@_uplatin_bool
             }
    {french} {
              \bool_set_false:N \g_@@_literal_bool
              \bool_set_true:N  \g_@@_upGreek_bool
              \bool_set_true:N  \g_@@_upgreek_bool
              \bool_set_true:N  \g_@@_upLatin_bool
              \bool_set_false:N \g_@@_uplatin_bool
             }
   {upright} {
              \bool_set_false:N \g_@@_literal_bool
              \bool_set_true:N  \g_@@_upGreek_bool
              \bool_set_true:N  \g_@@_upgreek_bool
              \bool_set_true:N  \g_@@_upLatin_bool
              \bool_set_true:N  \g_@@_uplatin_bool
             }
   {literal} {
              \bool_set_true:N  \g_@@_literal_bool
             }
  }
```

```
\@@_keys_choices:nn {math-style}
  {
      {ISO} {
             \unimathsetup { nabla=upright, partial=italic,
              normal-style=ISO, bold-style=ISO, sans-style=italic }
            }
      {TeX} {
             \unimathsetup { nabla=upright, partial=italic,
               normal-style=TeX, bold-style=TeX, sans-style=upright }
            }
   {french} {
             \unimathsetup { nabla=upright, partial=upright,
               normal-style=french, bold-style=upright, sans-style=upright }
            }
  {upright} {
             \unimathsetup { nabla=upright, partial=upright,
               normal-style=upright, bold-style=upright, sans-style=upright }
            }
  {literal} {
             \unimathsetup { colon=literal, nabla=literal, partial=literal,
               normal-style=literal, bold-style=literal, sans-style=literal }
            }
  }
```

bold-style

```
\@@_keys_choices:nn {bold-style}
  {
      {ISO} {
             \bool_set_false:N \g_@@_bfliteral_bool
             \bool_set_false:N \g_@@_bfupGreek_bool
             \bool_set_false:N \g_@@_bfupgreek_bool
             \bool_set_false:N \g_@@_bfupLatin_bool
             \bool_set_false:N \g_@@_bfuplatin_bool
            }
      {TeX} {
             \bool_set_false:N \g_@@_bfliteral_bool
             \bool_set_true:N  \g_@@_bfupGreek_bool
             \bool_set_false:N \g_@@_bfupgreek_bool
             \bool_set_true:N  \g_@@_bfupLatin_bool
             \bool_set_true:N  \g_@@_bfuplatin_bool
            }
  {upright} {
             \bool_set_false:N \g_@@_bfliteral_bool
             \bool_set_true:N  \g_@@_bfupGreek_bool
             \bool_set_true:N  \g_@@_bfupgreek_bool
             \bool_set_true:N  \g_@@_bfupLatin_bool
             \bool_set_true:N  \g_@@_bfuplatin_bool
            }
  {literal} {
             \bool_set_true:N  \g_@@_bfliteral_bool
            }
  }
```

sans-style

```
\@@_keys_choices:nn {sans-style}
  {
    {italic}  { \bool_set_false:N \g_@@_upsans_bool    }
    {upright} { \bool_set_true:N  \g_@@_upsans_bool    }
    {literal} { \bool_set_true:N  \g_@@_sfliteral_bool }
  }
```

Nabla and partial

```
\@@_keys_choices:nn {nabla}
  {
    {upright} {
                \bool_set_false:N \g_@@_literal_Nabla_bool
                \bool_set_true:N  \g_@@_upNabla_bool
              }
    {italic}  {
                \bool_set_false:N \g_@@_literal_Nabla_bool
                \bool_set_false:N \g_@@_upNabla_bool
              }
    {literal} {
                \bool_set_true:N  \g_@@_literal_Nabla_bool
              }
  }
```

```
\@@_keys_choices:nn {partial}
  {
   {upright} {
               \bool_set_false:N \g_@@_literal_partial_bool
               \bool_set_true:N  \g_@@_uppartial_bool
             }
   {italic}  {
               \bool_set_false:N \g_@@_literal_partial_bool
               \bool_set_false:N \g_@@_uppartial_bool
             }
   {literal} {
               \bool_set_true:N  \g_@@_literal_partial_bool
```

```
            }
  }
```

Colon style

```
\@@_keys_choices:nn {colon}
  {
    {literal} { \bool_set_true:N  \g_@@_literal_colon_bool }
    {TeX}     { \bool_set_false:N \g_@@_literal_colon_bool }
  }
```

Slash delimiter style

```
\@@_keys_choices:nn {slash-delimiter}
  {
    {ascii} { \tl_set:Nn \g_@@_slash_delimiter_usv {"002F} }
    {frac}  { \tl_set:Nn \g_@@_slash_delimiter_usv {"2044} }
    {div}   { \tl_set:Nn \g_@@_slash_delimiter_usv {"2215} }
  }
```

Active fraction style

```
\@@_keys_choices:nn {active-frac}
  {
    {small}
    {
      \cs_if_exist:NTF \tfrac
        { \bool_set_true:N \l_@@_smallfrac_bool }
        {
          \@@_warning:n {no-tfrac}
          \bool_set_false:N \l_@@_smallfrac_bool
        }
      \use:c {@@_setup_active_frac:}
    }

    {normalsize}
    {
      \bool_set_false:N \l_@@_smallfrac_bool
      \use:c {@@_setup_active_frac:}
    }
  }
```

Debug/tracing

```
\keys_define:nn {unicode-math}
  {
    warnings-off .code:n =
      {
        \clist_map_inline:nn {#1}
          { \msg_redirect_name:nnn { unicode-math } { ##1 } { none } }
      }
  }
```

```
\@@_keys_choices:nn {trace}
  {
    {on}    {} % default
    {debug} { \msg_redirect_module:nnn { unicode-math } { log } { warning } }
    {off}   { \msg_redirect_module:nnn { unicode-math } { log } { none } }
  }
```

7.1 *Defaults*

```
\unimathsetup {math-style=TeX}
\unimathsetup {slash-delimiter=ascii}
\unimathsetup {trace=off}
\unimathsetup {mathrm=text,mathit=text,mathbf=text,mathsf=text,mathtt=text}
\cs_if_exist:NT \tfrac { \unimathsetup {active-frac=small} }
\ProcessKeysOptions {unicode-math}
```

```
⟨/package⟩
```

File VII
um-code-msg.dtx

8 *Error messages*

```
⟨*package⟩
\char_set_catcode_space:n {32}
\@@_msg_new:nn {no-tfrac}
{
  Small fraction command \protect\tfrac\ not defined.\\
  Load amsmath or define it manually before loading unicode-math.
}
\@@_msg_new:nn {default-math-font}
{
  Defining the default maths font as '\l_@@_fontname_tl'.
}
\@@_msg_new:nn {setup-implicit}
{
  Setup alphabets: implicit mode.
}
\@@_msg_new:nn {setup-explicit}
{
  Setup alphabets: explicit mode.
}
\@@_msg_new:nn {alph-initialise}
{
  Initialising \@backslashchar math#1.
}
\@@_msg_new:nn {setup-alph}
{
  Setup alphabet: #1.
}
\@@_msg_new:nn {no-alphabet}
{
   I am trying to set up alphabet"#1" but there are no configuration set-
tings for it.
  (See source file "unicode-math-alphabets.dtx" to debug.)
}
\@@_msg_new:nn {no-named-range}
 {
 I am trying to define new alphabet "#2" in range "#1", but range "#1" hasn't been de-
fined yet.
 }
\@@_msg_new:nn {missing-alphabets}
 {
  Missingmathalphabetsinfont "\fontname\g_@@_curr_font_cmd_tl" \\ \\
  \seq_map_function:NN \l_@@_missing_alph_seq \@@_print_indent:n
 }
\cs_new:Nn \@@_print_indent:n { \space\space\space\space #1 \\ }
\@@_msg_new:nn {macro-expected}
{
  I've expected that #1 is a macro, but it isn't.
}
\@@_msg_new:nn {wrong-meaning}
{
  I've expected #1 to have the meaning #3, but it has the meaning #2.
}
\@@_msg_new:nn {patch-macro}
{
  I'm going to patch macro #1.
}
\@@_msg_new:nn {mathtools-overbracket} {
  Using \token_to_str:N \overbracket\ and
         \token_to_str:N \underbracket\ from
 `mathtools' package.\\
  \\
  Use \token_to_str:N \Uoverbracket\ and
       \token_to_str:N \Uunderbracket\ for
       original `unicode-math' definition.
}
\@@_msg_new:nn {mathtools-colon} {
  I'm going to overwrite the following commands from
  the `mathtools' package: \\ \\
  \ \ \ \ \token_to_str:N \dblcolon,
  \token_to_str:N \coloneqq,
  \token_to_str:N \Coloneqq,
  \token_to_str:N \eqqcolon. \\ \\
  Note that since I won't overwrite the other colon-like
  commands, using them will lead to inconsistencies.
}
\@@_msg_new:nn {colonequals} {
  I'm going to overwrite the following commands from
  the `colonequals' package: \\ \\
  \ \ \ \ \token_to_str:N \ratio,
          \token_to_str:N \coloncolon,
          \token_to_str:N \minuscolon, \\
  \ \ \ \ \token_to_str:N \colonequals,
          \token_to_str:N \equalscolon,
          \token_to_str:N \coloncolonequals. \\ \\
  Note that since I won't overwrite the other colon-like
  commands, using them will lead to inconsistencies.
  Furthermore, changing \token_to_str:N \colonsep \c_space_tl
  or \token_to_str:N \doublecolonsep \c_space_tl won't have
  any effect on the re-defined commands.
}
\@@_msg_new:nn {bad-cs-in-range}
```

```
  {
    Command `#1` in math range is not recognised as a maths symbol.
    Check file "unicode-math-table.tex" for allowable commands.
  }
\@@_msg_new:nn {legacy-char-not-supported}
  {
   Command `#1` is a legacy maths symbol that is not supported by unicode-math.
  }
\@@_msg_new:nn {range-not-bf-sf}
  {
   Range alphabets cannot include alphabets referring to `bf`, `sf`, or `bfsf`
    since they relate to input commands not output glyphs.
    Use `bfit` or `bfup` (etc.) to specify which.
  }
\@@_msg_new:nn {no-main-font}
  {
     No main maths font has been set up yet.\\If you simply want ‘the de-
fault’, use: \\
    \iow_indent:n {\token_to_str:N\setmathfont{latinmodern-math.otf}}
  }
\char_set_catcode_ignore:n {32}
⟨/package⟩
```

File VIII

um-code-usv.dtx

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.[2]

```
⟨*package⟩
```

Alphabets 'Normal':

```
\usv_set:nnn {normal} {num}      {48}
\usv_set:nnn {normal} {Latin}    {"1D434}
\usv_set:nnn {normal} {latin}    {"1D44E}
\usv_set:nnn {normal} {Greek}    {"1D6E2}
\usv_set:nnn {normal} {greek}    {"1D6FC}
\usv_set:nnn {normal} {varTheta} {"1D6F3}
\usv_set:nnn {normal} {epsilon}  {"1D716}
\usv_set:nnn {normal} {vartheta} {"1D717}
\usv_set:nnn {normal} {varkappa} {"1D718}
\usv_set:nnn {normal} {phi}      {"1D719}
\usv_set:nnn {normal} {varrho}   {"1D71A}
\usv_set:nnn {normal} {varpi}    {"1D71B}
\usv_set:nnn {normal} {Nabla}    {"1D6FB}
\usv_set:nnn {normal} {partial}  {"1D715}
```

Regular weights:

```
\usv_set:nnn {up}   {num}   {48}
\usv_set:nnn {up}   {Latin} {65}
\usv_set:nnn {up}   {latin} {97}
\usv_set:nnn {up}   {Greek} {"391}
\usv_set:nnn {up}   {greek} {"3B1}
\usv_set:nnn {it}   {Latin} {"1D434}
\usv_set:nnn {it}   {latin} {"1D44E}
\usv_set:nnn {it}   {Greek} {"1D6E2}
\usv_set:nnn {it}   {greek} {"1D6FC}
\usv_set:nnn {bb}   {num}   {"1D7D8}
\usv_set:nnn {bb}   {Latin} {"1D538}
\usv_set:nnn {bb}   {latin} {"1D552}
\usv_set:nnn {scr}  {Latin} {"1D49C}
\usv_set:nnn {cal}  {Latin} {"1D49C}
\usv_set:nnn {scr}  {latin} {"1D4B6}
\usv_set:nnn {frak} {Latin} {"1D504}
\usv_set:nnn {frak} {latin} {"1D51E}
\usv_set:nnn {sf}   {num}   {"1D7E2}
\usv_set:nnn {sfup} {num}   {"1D7E2}
```

[2] 'u.s.v.' stands for 'Unicode scalar value'.

```
\usv_set:nnn {sfit} {num}   {"1D7E2}
\usv_set:nnn {sfup} {Latin} {"1D5A0}
\usv_set:nnn {sf}   {Latin} {"1D5A0}
\usv_set:nnn {sfup} {latin} {"1D5BA}
\usv_set:nnn {sf}   {latin} {"1D5BA}
\usv_set:nnn {sfit} {Latin} {"1D608}
\usv_set:nnn {sfit} {latin} {"1D622}
\usv_set:nnn {tt}   {num}   {"1D7F6}
\usv_set:nnn {tt}   {Latin} {"1D670}
\usv_set:nnn {tt}   {latin} {"1D68A}
```

Bold weights:

```
\usv_set:nnn {bf}     {num}   {"1D7CE}
\usv_set:nnn {bfup}   {num}   {"1D7CE}
\usv_set:nnn {bfit}   {num}   {"1D7CE}
\usv_set:nnn {bfup}   {Latin} {"1D400}
\usv_set:nnn {bfup}   {latin} {"1D41A}
\usv_set:nnn {bfup}   {Greek} {"1D6A8}
\usv_set:nnn {bfup}   {greek} {"1D6C2}
\usv_set:nnn {bfit}   {Latin} {"1D468}
\usv_set:nnn {bfit}   {latin} {"1D482}
\usv_set:nnn {bfit}   {Greek} {"1D71C}
\usv_set:nnn {bfit}   {greek} {"1D736}
\usv_set:nnn {bffrak} {Latin} {"1D56C}
\usv_set:nnn {bffrak} {latin} {"1D586}
\usv_set:nnn {bfscr}  {Latin} {"1D4D0}
\usv_set:nnn {bfcal}  {Latin} {"1D4D0}
\usv_set:nnn {bfscr}  {latin} {"1D4EA}
\usv_set:nnn {bfsf}   {num}   {"1D7EC}
\usv_set:nnn {bfsfup} {num}   {"1D7EC}
\usv_set:nnn {bfsfit} {num}   {"1D7EC}
\usv_set:nnn {bfsfup} {Latin} {"1D5D4}
\usv_set:nnn {bfsfup} {latin} {"1D5EE}
\usv_set:nnn {bfsfup} {Greek} {"1D756}
\usv_set:nnn {bfsfup} {greek} {"1D770}
\usv_set:nnn {bfsfit} {Latin} {"1D63C}
\usv_set:nnn {bfsfit} {latin} {"1D656}
\usv_set:nnn {bfsfit} {Greek} {"1D790}
\usv_set:nnn {bfsfit} {greek} {"1D7AA}
```

The ‘auto’ bolds:

```
\usv_set:nnn {bfsf} {Latin} { \bool_if:NTF \g_@@_upLatin_bool   \g_@@_bfsfup_Latin_usv \g_@@_bfsfit_Lat
\usv_set:nnn {bfsf} {latin} { \bool_if:NTF \g_@@_uplatin_bool   \g_@@_bfsfup_latin_usv \g_@@_bfsfit_lat
\usv_set:nnn {bfsf} {Greek} { \bool_if:NTF \g_@@_upGreek_bool   \g_@@_bfsfup_Greek_usv \g_@@_bfsfit_Gre
\usv_set:nnn {bfsf} {greek} { \bool_if:NTF \g_@@_upgreek_bool   \g_@@_bfsfup_greek_usv \g_@@_bfsfit_gre
\usv_set:nnn {bf}   {Latin} { \bool_if:NTF \g_@@_bfupLatin_bool \g_@@_bfup_Latin_usv   \g_@@_bfit_Latin
\usv_set:nnn {bf}   {latin} { \bool_if:NTF \g_@@_bfuplatin_bool \g_@@_bfup_latin_usv   \g_@@_bfit_latin
\usv_set:nnn {bf}   {Greek} { \bool_if:NTF \g_@@_bfupGreek_bool \g_@@_bfup_Greek_usv   \g_@@_bfit_Greek
\usv_set:nnn {bf}   {greek} { \bool_if:NTF \g_@@_bfupgreek_bool \g_@@_bfup_greek_usv   \g_@@_bfit_greek
```

Greek variants Upright:

```
\usv_set:nnn {up} {varTheta} {"3F4}
\usv_set:nnn {up} {Digamma}  {"3DC}
\usv_set:nnn {up} {epsilon}  {"3F5}
\usv_set:nnn {up} {vartheta} {"3D1}
\usv_set:nnn {up} {varkappa} {"3F0}
\usv_set:nnn {up} {phi}      {"3D5}
\usv_set:nnn {up} {varrho}   {"3F1}
\usv_set:nnn {up} {varpi}    {"3D6}
\usv_set:nnn {up} {digamma}  {"3DD}
```

Bold:

```
\usv_set:nnn {bfup} {varTheta} {"1D6B9}
\usv_set:nnn {bfup} {Digamma}  {"1D7CA}
\usv_set:nnn {bfup} {epsilon}  {"1D6DC}
\usv_set:nnn {bfup} {vartheta} {"1D6DD}
\usv_set:nnn {bfup} {varkappa} {"1D6DE}
\usv_set:nnn {bfup} {phi}      {"1D6DF}
\usv_set:nnn {bfup} {varrho}   {"1D6E0}
\usv_set:nnn {bfup} {varpi}    {"1D6E1}
\usv_set:nnn {bfup} {digamma}  {"1D7CB}
```

Italic:

```
\usv_set:nnn {it} {varTheta} {"1D6F3}
\usv_set:nnn {it} {epsilon}  {"1D716}
\usv_set:nnn {it} {vartheta} {"1D717}
\usv_set:nnn {it} {varkappa} {"1D718}
\usv_set:nnn {it} {phi}      {"1D719}
\usv_set:nnn {it} {varrho}   {"1D71A}
\usv_set:nnn {it} {varpi}    {"1D71B}
```

Bold italic:

```
\usv_set:nnn {bfit} {varTheta} {"1D72D}
\usv_set:nnn {bfit} {epsilon}  {"1D750}
\usv_set:nnn {bfit} {vartheta} {"1D751}
\usv_set:nnn {bfit} {varkappa} {"1D752}
\usv_set:nnn {bfit} {phi}      {"1D753}
\usv_set:nnn {bfit} {varrho}   {"1D754}
\usv_set:nnn {bfit} {varpi}    {"1D755}
```

Bold sans:

```
\usv_set:nnn {bfsfup} {varTheta} {"1D767}
\usv_set:nnn {bfsfup} {epsilon}  {"1D78A}
\usv_set:nnn {bfsfup} {vartheta} {"1D78B}
\usv_set:nnn {bfsfup} {varkappa} {"1D78C}
\usv_set:nnn {bfsfup} {phi}      {"1D78D}
\usv_set:nnn {bfsfup} {varrho}   {"1D78E}
\usv_set:nnn {bfsfup} {varpi}    {"1D78F}
```

Bold sans italic:

```
\usv_set:nnn {bfsfit} {varTheta} {"1D7A1}
```

```
\usv_set:nnn {bfsfit} {epsilon}  {"1D7C4}
\usv_set:nnn {bfsfit} {vartheta} {"1D7C5}
\usv_set:nnn {bfsfit} {varkappa} {"1D7C6}
\usv_set:nnn {bfsfit} {phi}      {"1D7C7}
\usv_set:nnn {bfsfit} {varrho}   {"1D7C8}
\usv_set:nnn {bfsfit} {varpi}    {"1D7C9}
```

Nabla:

```
\usv_set:nnn {up}     {Nabla} {"02207}
\usv_set:nnn {it}     {Nabla} {"1D6FB}
\usv_set:nnn {bfup}   {Nabla} {"1D6C1}
\usv_set:nnn {bfit}   {Nabla} {"1D735}
\usv_set:nnn {bfsfup} {Nabla} {"1D76F}
\usv_set:nnn {bfsfit} {Nabla} {"1D7A9}
```

Partial:

```
\usv_set:nnn {up}     {partial} {"02202}
\usv_set:nnn {it}     {partial} {"1D715}
\usv_set:nnn {bfup}   {partial} {"1D6DB}
\usv_set:nnn {bfit}   {partial} {"1D74F}
\usv_set:nnn {bfsfup} {partial} {"1D789}
\usv_set:nnn {bfsfit} {partial} {"1D7C3}
```

Exceptions Upright uppercase:

```
\usv_set:nnn {up} {B} {`\B}
\usv_set:nnn {up} {C} {`\C}
\usv_set:nnn {up} {D} {`\D}
\usv_set:nnn {up} {E} {`\E}
\usv_set:nnn {up} {F} {`\F}
\usv_set:nnn {up} {H} {`\H}
\usv_set:nnn {up} {I} {`\I}
\usv_set:nnn {up} {L} {`\L}
\usv_set:nnn {up} {M} {`\M}
\usv_set:nnn {up} {N} {`\N}
\usv_set:nnn {up} {P} {`\P}
\usv_set:nnn {up} {Q} {`\Q}
\usv_set:nnn {up} {R} {`\R}
\usv_set:nnn {up} {Z} {`\Z}
```

Italic uppercase:

```
\usv_set:nnn {it} {B} {"1D435}
\usv_set:nnn {it} {C} {"1D436}
\usv_set:nnn {it} {D} {"1D437}
\usv_set:nnn {it} {E} {"1D438}
\usv_set:nnn {it} {F} {"1D439}
\usv_set:nnn {it} {H} {"1D43B}
\usv_set:nnn {it} {I} {"1D43C}
\usv_set:nnn {it} {L} {"1D43F}
\usv_set:nnn {it} {M} {"1D440}
\usv_set:nnn {it} {N} {"1D441}
```

```
\usv_set:nnn {it} {P} {"1D443}
\usv_set:nnn {it} {Q} {"1D444}
\usv_set:nnn {it} {R} {"1D445}
\usv_set:nnn {it} {Z} {"1D44D}
```

Upright lowercase (needed for later mappings):

```
\usv_set:nnn {up} {d} {`\d}
\usv_set:nnn {up} {e} {`\e}
\usv_set:nnn {up} {g} {`\g}
\usv_set:nnn {up} {h} {`\h}
\usv_set:nnn {up} {i} {`\i}
\usv_set:nnn {up} {j} {`\j}
\usv_set:nnn {up} {o} {`\o}
```

Italic lowercase:

```
\usv_set:nnn {it} {d} {"1D451}
\usv_set:nnn {it} {e} {"1D452}
\usv_set:nnn {it} {g} {"1D454}
\usv_set:nnn {it} {h} {"0210E}
\usv_set:nnn {it} {i} {"1D456}
\usv_set:nnn {it} {j} {"1D457}
\usv_set:nnn {it} {o} {"1D45C}
```

Latin ‘h’:

```
\usv_set:nnn {bb}     {h} {"1D559}
\usv_set:nnn {tt}     {h} {"1D691}
\usv_set:nnn {scr}    {h} {"1D4BD}
\usv_set:nnn {frak}   {h} {"1D525}
\usv_set:nnn {bfup}   {h} {"1D421}
\usv_set:nnn {bfit}   {h} {"1D489}
\usv_set:nnn {sfup}   {h} {"1D5C1}
\usv_set:nnn {sfit}   {h} {"1D629}
\usv_set:nnn {bffrak} {h} {"1D58D}
\usv_set:nnn {bfscr}  {h} {"1D4F1}
\usv_set:nnn {bfsfup} {h} {"1D5F5}
\usv_set:nnn {bfsfit} {h} {"1D65D}
```

Dotless ‘i’ and ‘j’:

```
\usv_set:nnn {up} {dotlessi} {"00131}
\usv_set:nnn {up} {dotlessj} {"00237}
\usv_set:nnn {it} {dotlessi} {"1D6A4}
\usv_set:nnn {it} {dotlessj} {"1D6A5}
```

Blackboard:

```
\usv_set:nnn {bb} {C}         {"2102}
\usv_set:nnn {bb} {H}         {"210D}
\usv_set:nnn {bb} {N}         {"2115}
\usv_set:nnn {bb} {P}         {"2119}
\usv_set:nnn {bb} {Q}         {"211A}
\usv_set:nnn {bb} {R}         {"211D}
\usv_set:nnn {bb} {Z}         {"2124}
```

```
\usv_set:nnn {up} {Pi}        {"003A0}
\usv_set:nnn {up} {pi}        {"003C0}
\usv_set:nnn {up} {Gamma}     {"00393}
\usv_set:nnn {up} {gamma}     {"003B3}
\usv_set:nnn {up} {summation} {"02211}
\usv_set:nnn {it} {Pi}        {"1D6F1}
\usv_set:nnn {it} {pi}        {"1D70B}
\usv_set:nnn {it} {Gamma}     {"1D6E4}
\usv_set:nnn {it} {gamma}     {"1D6FE}
\usv_set:nnn {bb} {Pi}        {"0213F}
\usv_set:nnn {bb} {pi}        {"0213C}
\usv_set:nnn {bb} {Gamma}     {"0213E}
\usv_set:nnn {bb} {gamma}     {"0213D}
\usv_set:nnn {bb} {summation} {"02140}
```

Italic blackboard:

```
\usv_set:nnn {bbit} {D} {"2145}
\usv_set:nnn {bbit} {d} {"2146}
\usv_set:nnn {bbit} {e} {"2147}
\usv_set:nnn {bbit} {i} {"2148}
\usv_set:nnn {bbit} {j} {"2149}
```

Script:

```
\usv_set:nnn {scr} {B} {"212C}
\usv_set:nnn {scr} {E} {"2130}
\usv_set:nnn {scr} {F} {"2131}
\usv_set:nnn {scr} {H} {"210B}
\usv_set:nnn {scr} {I} {"2110}
\usv_set:nnn {scr} {L} {"2112}
\usv_set:nnn {scr} {M} {"2133}
\usv_set:nnn {scr} {R} {"211B}
\usv_set:nnn {scr} {e} {"212F}
\usv_set:nnn {scr} {g} {"210A}
\usv_set:nnn {scr} {o} {"2134}
```

Caligraphic:

```
\usv_set:nnn {cal} {B} {"212C}
\usv_set:nnn {cal} {E} {"2130}
\usv_set:nnn {cal} {F} {"2131}
\usv_set:nnn {cal} {H} {"210B}
\usv_set:nnn {cal} {I} {"2110}
\usv_set:nnn {cal} {L} {"2112}
\usv_set:nnn {cal} {M} {"2133}
\usv_set:nnn {cal} {R} {"211B}
```

Fractur:

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

```
⟨/package⟩
```

9.1 STIX fonts

Version 1.0.0 of the STIX fonts contains a number of alphabets in the private use area of Unicode; i.e., it contains many math glyphs that have not (yet or if ever) been accepted into the Unicode standard.

But we still want to be able to use them if possible.

```
⟨*stix⟩
```

Upright

```
\usv_set:nnn {stixsfup}{partial}{"E17C}
\usv_set:nnn {stixsfup}{Greek}{"E17D}
\usv_set:nnn {stixsfup}{greek}{"E196}
\usv_set:nnn {stixsfup}{varTheta}{"E18E}
\usv_set:nnn {stixsfup}{epsilon}{"E1AF}
\usv_set:nnn {stixsfup}{vartheta}{"E1B0}
\usv_set:nnn {stixsfup}{varkappa}{0000} % ???
\usv_set:nnn {stixsfup}{phi}{"E1B1}
\usv_set:nnn {stixsfup}{varrho}{"E1B2}
\usv_set:nnn {stixsfup}{varpi}{"E1B3}
\usv_set:nnn {stixupslash}{Greek}{"E2FC}
```

Italic

```
\usv_set:nnn {stixbbit}{A}{"E154}
\usv_set:nnn {stixbbit}{B}{"E155}
\usv_set:nnn {stixbbit}{E}{"E156}
\usv_set:nnn {stixbbit}{F}{"E157}
\usv_set:nnn {stixbbit}{G}{"E158}
\usv_set:nnn {stixbbit}{I}{"E159}
\usv_set:nnn {stixbbit}{J}{"E15A}
\usv_set:nnn {stixbbit}{K}{"E15B}
\usv_set:nnn {stixbbit}{L}{"E15C}
\usv_set:nnn {stixbbit}{M}{"E15D}
\usv_set:nnn {stixbbit}{O}{"E15E}
\usv_set:nnn {stixbbit}{S}{"E15F}
\usv_set:nnn {stixbbit}{T}{"E160}
\usv_set:nnn {stixbbit}{U}{"E161}
\usv_set:nnn {stixbbit}{V}{"E162}
\usv_set:nnn {stixbbit}{W}{"E163}
\usv_set:nnn {stixbbit}{X}{"E164}
\usv_set:nnn {stixbbit}{Y}{"E165}

\usv_set:nnn {stixbbit}{a}{"E166}
\usv_set:nnn {stixbbit}{b}{"E167}
\usv_set:nnn {stixbbit}{c}{"E168}
\usv_set:nnn {stixbbit}{f}{"E169}
\usv_set:nnn {stixbbit}{g}{"E16A}
```

```
\usv_set:nnn {stixbbit}{h}{"E16B}
\usv_set:nnn {stixbbit}{k}{"E16C}
\usv_set:nnn {stixbbit}{l}{"E16D}
\usv_set:nnn {stixbbit}{m}{"E16E}
\usv_set:nnn {stixbbit}{n}{"E16F}
\usv_set:nnn {stixbbit}{o}{"E170}
\usv_set:nnn {stixbbit}{p}{"E171}
\usv_set:nnn {stixbbit}{q}{"E172}
\usv_set:nnn {stixbbit}{r}{"E173}
\usv_set:nnn {stixbbit}{s}{"E174}
\usv_set:nnn {stixbbit}{t}{"E175}
\usv_set:nnn {stixbbit}{u}{"E176}
\usv_set:nnn {stixbbit}{v}{"E177}
\usv_set:nnn {stixbbit}{w}{"E178}
\usv_set:nnn {stixbbit}{x}{"E179}
\usv_set:nnn {stixbbit}{y}{"E17A}
\usv_set:nnn {stixbbit}{z}{"E17B}

\usv_set:nnn {stixsfit}{Numerals}{"E1B4}
\usv_set:nnn {stixsfit}{partial}{"E1BE}
\usv_set:nnn {stixsfit}{Greek}{"E1BF}
\usv_set:nnn {stixsfit}{greek}{"E1D8}
\usv_set:nnn {stixsfit}{varTheta}{"E1D0}
\usv_set:nnn {stixsfit}{epsilon}{"E1F1}
\usv_set:nnn {stixsfit}{vartheta}{"E1F2}
\usv_set:nnn {stixsfit}{varkappa}{0000} % ???
\usv_set:nnn {stixsfit}{phi}{"E1F3}
\usv_set:nnn {stixsfit}{varrho}{"E1F4}
\usv_set:nnn {stixsfit}{varpi}{"E1F5}

\usv_set:nnn {stixcal}{Latin}{"E22D}
\usv_set:nnn {stixcal}{num}{"E262}
\usv_set:nnn {scr}{num}{48}
\usv_set:nnn {it}{num}{48}

\usv_set:nnn {stixsfitslash}{Latin}{"E294}
\usv_set:nnn {stixsfitslash}{latin}{"E2C8}
\usv_set:nnn {stixsfitslash}{greek}{"E32C}
\usv_set:nnn {stixsfitslash}{epsilon}{"E37A}
\usv_set:nnn {stixsfitslash}{vartheta}{"E35E}
\usv_set:nnn {stixsfitslash}{varkappa}{"E374}
\usv_set:nnn {stixsfitslash}{phi}{"E360}
\usv_set:nnn {stixsfitslash}{varrho}{"E376}
\usv_set:nnn {stixsfitslash}{varpi}{"E362}
\usv_set:nnn {stixsfitslash}{digamma}{"E36A}
```

Bold

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

\usv_set:nnn {stixbfbb}{A}{"E38A}
```

```
\usv_set:nnn {stixbfbb}{B}{"E38B}
\usv_set:nnn {stixbfbb}{E}{"E38D}
\usv_set:nnn {stixbfbb}{F}{"E38E}
\usv_set:nnn {stixbfbb}{G}{"E38F}
\usv_set:nnn {stixbfbb}{I}{"E390}
\usv_set:nnn {stixbfbb}{J}{"E391}
\usv_set:nnn {stixbfbb}{K}{"E392}
\usv_set:nnn {stixbfbb}{L}{"E393}
\usv_set:nnn {stixbfbb}{M}{"E394}
\usv_set:nnn {stixbfbb}{O}{"E395}
\usv_set:nnn {stixbfbb}{S}{"E396}
\usv_set:nnn {stixbfbb}{T}{"E397}
\usv_set:nnn {stixbfbb}{U}{"E398}
\usv_set:nnn {stixbfbb}{V}{"E399}
\usv_set:nnn {stixbfbb}{W}{"E39A}
\usv_set:nnn {stixbfbb}{X}{"E39B}
\usv_set:nnn {stixbfbb}{Y}{"E39C}

\usv_set:nnn {stixbfbb}{a}{"E39D}
\usv_set:nnn {stixbfbb}{b}{"E39E}
\usv_set:nnn {stixbfbb}{c}{"E39F}
\usv_set:nnn {stixbfbb}{f}{"E3A2}
\usv_set:nnn {stixbfbb}{g}{"E3A3}
\usv_set:nnn {stixbfbb}{h}{"E3A4}
\usv_set:nnn {stixbfbb}{k}{"E3A7}
\usv_set:nnn {stixbfbb}{l}{"E3A8}
\usv_set:nnn {stixbfbb}{m}{"E3A9}
\usv_set:nnn {stixbfbb}{n}{"E3AA}
\usv_set:nnn {stixbfbb}{o}{"E3AB}
\usv_set:nnn {stixbfbb}{p}{"E3AC}
\usv_set:nnn {stixbfbb}{q}{"E3AD}
\usv_set:nnn {stixbfbb}{r}{"E3AE}
\usv_set:nnn {stixbfbb}{s}{"E3AF}
\usv_set:nnn {stixbfbb}{t}{"E3B0}
\usv_set:nnn {stixbfbb}{u}{"E3B1}
\usv_set:nnn {stixbfbb}{v}{"E3B2}
\usv_set:nnn {stixbfbb}{w}{"E3B3}
\usv_set:nnn {stixbfbb}{x}{"E3B4}
\usv_set:nnn {stixbfbb}{y}{"E3B5}
\usv_set:nnn {stixbfbb}{z}{"E3B6}

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

Bold Italic

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

\usv_set:nnn {stixbfbbit}{A}{"E200}
\usv_set:nnn {stixbfbbit}{B}{"E201}
\usv_set:nnn {stixbfbbit}{E}{"E203}
\usv_set:nnn {stixbfbbit}{F}{"E204}
\usv_set:nnn {stixbfbbit}{G}{"E205}
```

```
\usv_set:nnn {stixbfbbit}{I}{"E206}
\usv_set:nnn {stixbfbbit}{J}{"E207}
\usv_set:nnn {stixbfbbit}{K}{"E208}
\usv_set:nnn {stixbfbbit}{L}{"E209}
\usv_set:nnn {stixbfbbit}{M}{"E20A}
\usv_set:nnn {stixbfbbit}{O}{"E20B}
\usv_set:nnn {stixbfbbit}{S}{"E20C}
\usv_set:nnn {stixbfbbit}{T}{"E20D}
\usv_set:nnn {stixbfbbit}{U}{"E20E}
\usv_set:nnn {stixbfbbit}{V}{"E20F}
\usv_set:nnn {stixbfbbit}{W}{"E210}
\usv_set:nnn {stixbfbbit}{X}{"E211}
\usv_set:nnn {stixbfbbit}{Y}{"E212}

\usv_set:nnn {stixbfbbit}{a}{"E213}
\usv_set:nnn {stixbfbbit}{b}{"E214}
\usv_set:nnn {stixbfbbit}{c}{"E215}
\usv_set:nnn {stixbfbbit}{e}{"E217}
\usv_set:nnn {stixbfbbit}{f}{"E218}
\usv_set:nnn {stixbfbbit}{g}{"E219}
\usv_set:nnn {stixbfbbit}{h}{"E21A}
\usv_set:nnn {stixbfbbit}{k}{"E21D}
\usv_set:nnn {stixbfbbit}{l}{"E21E}
\usv_set:nnn {stixbfbbit}{m}{"E21F}
\usv_set:nnn {stixbfbbit}{n}{"E220}
\usv_set:nnn {stixbfbbit}{o}{"E221}
\usv_set:nnn {stixbfbbit}{p}{"E222}
\usv_set:nnn {stixbfbbit}{q}{"E223}
\usv_set:nnn {stixbfbbit}{r}{"E224}
\usv_set:nnn {stixbfbbit}{s}{"E225}
\usv_set:nnn {stixbfbbit}{t}{"E226}
\usv_set:nnn {stixbfbbit}{u}{"E227}
\usv_set:nnn {stixbfbbit}{v}{"E228}
\usv_set:nnn {stixbfbbit}{w}{"E229}
\usv_set:nnn {stixbfbbit}{x}{"E22A}
\usv_set:nnn {stixbfbbit}{y}{"E22B}
\usv_set:nnn {stixbfbbit}{z}{"E22C}

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

\usv_set:nnn {stixbfitslash}{Latin}{"E295}
\usv_set:nnn {stixbfitslash}{latin}{"E2C9}
\usv_set:nnn {stixbfitslash}{greek}{"E32D}
\usv_set:nnn {stixsfitslash}{epsilon}{"E37B}
\usv_set:nnn {stixsfitslash}{vartheta}{"E35F}
\usv_set:nnn {stixsfitslash}{varkappa}{"E375}
\usv_set:nnn {stixsfitslash}{phi}{"E361}
\usv_set:nnn {stixsfitslash}{varrho}{"E377}
\usv_set:nnn {stixsfitslash}{varpi}{"E363}
\usv_set:nnn {stixsfitslash}{digamma}{"E36B}

⟨/stix⟩
```

File IX

um-code-setchar.dtx

10 *Setting up maths chars*

```
⟨*package⟩
```

10.1 *A token list to contain the data of the math table*

Instead of `\input`-ing the unicode math table every time we want to re-read its data, we save it within a macro. This has two advantages: 1. it should be slightly faster, at the expense of memory; 2. we don't need to worry about catcodes later, since they're frozen at this point.

In time, the case statement inside `set_mathsymbol` will be moved in here to avoid re-running it every time.

```
\cs_new:Npn \@@_symbol_setup:
 {
  \cs_set:Npn \UnicodeMathSymbol ##1##2##3##4
   {
    \exp_not:n { \_@@_sym:nnn {##1} {##2} {##3} }
   }
 }
\tl_set_from_file_x:Nnn \g_@@_mathtable_tl {\@@_symbol_setup:} {unicode-math-
table.tex}
```

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

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

10.2 *Definitions of the active math characters*

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

```
\AtBeginDocument{\@@_define_math_chars:}
\cs_new:Nn \@@_define_math_chars:
 {
  \group_begin:
    \cs_set:Npn \_@@_sym:nnn ##1##2##3
     {
      \tl_if_in:nnT
       { \mathord \mathalpha \mathbin \mathrel \mathpunct \mathop \mathfence }
       {##3}
      {
      \exp_last_unbraced:NNx \cs_gset_eq:NN ##2 { \char_generate:nn {##1} {12} }
      }
     }
    \@@_input_math_symbol_table:
  \group_end:
}
```

10.3 *Commands for each symbol/glyph/char*

`\@@_set_mathsymbol:nNNn`

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

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

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

```
\cs_set:Nn \@@_set_mathsymbol:nNNn
 {
  \bool_lazy_and:nnT
   {
    \int_compare_p:nNn {#4} > {127}
   }
   {
    \int_compare_p:nNn { \char_value_catcode:n {#4} } = {11}
   }
   { \char_set_catcode_other:n {#4} }

  \tl_case:Nn #3
   {
    \mathord   { \@@_set_mathcode:nnn {#4} {#3} {#1} }
    \mathalpha { \@@_set_mathcode:nnn {#4} {#3} {#1} }
    \mathbin   { \@@_set_mathcode:nnn {#4} {#3} {#1} }
    \mathrel   { \@@_set_mathcode:nnn {#4} {#3} {#1} }
    \mathpunct { \@@_set_mathcode:nnn {#4} {#3} {#1} }
    \mathop    { \@@_set_big_operator:nnn {#1} {#2} {#4} }
    \mathopen  { \@@_set_math_open:nnn    {#1} {#2} {#4} }
    \mathclose { \@@_set_math_close:nnn   {#1} {#2} {#4} }
    \mathfence { \@@_set_math_fence:nnnn  {#1} {#2} {#3} {#4} }
    \mathaccent
     { \@@_set_math_accent:Nnnn #2 {fixed} {#1} {#4} }
    \mathbotaccent
     { \@@_set_math_accent:Nnnn #2 {bottom~ fixed} {#1} {#4} }
    \mathaccentwide
     { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
    \mathbotaccentwide
     { \@@_set_math_accent:Nnnn #2 {bottom} {#1} {#4} }
    \mathover
     { \@@_set_math_overunder:Nnnn #2 {} {#1} {#4} }
    \mathunder
     { \@@_set_math_overunder:Nnnn #2 {bottom} {#1} {#4} }
    \mathaccentoverlay
⟨LU⟩     { \@@_set_math_accent:Nnnn #2 {overlay~ fixed} {#1} {#4} }
⟨XE⟩     { \@@_set_math_accent:Nnnn #2 {} {#1} {#4} }
   }
 }
```

```
\edef\mathfence{\string\mathfence}
\edef\mathover{\string\mathover}
\edef\mathunder{\string\mathunder}
\edef\mathbotaccent{\string\mathbotaccent}
\edef\mathaccentwide{\string\mathaccentwide}
\edef\mathaccentoverlay{\string\mathaccentoverlay}
\edef\mathbotaccentwide{\string\mathbotaccentwide}
```

`\@@_set_big_operator:nnn`

#1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot

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

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

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

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

(`\sum` →) $\sum$ → `\sum_sym` → `\sumop\nolimits`

(`\int` →) $\int$ → `\int_sym` → `\intop`

```
\cs_new:Nn \@@_set_big_operator:nnn
 {
  \@@_char_gmake_mathactive:n {#3}
  \cs_set_protected_nopar:Npx \@@_tmpa: { \exp_not:c { \cs_to_str:N #2 _sym } }
  \char_gset_active_eq:nN {#3} \@@_tmpa:

  \@@_set_mathchar:cNnn {\cs_to_str:N #2 op} \mathop {#1} {#3}

  \cs_gset:cpx { \cs_to_str:N #2 _sym }
   {
    \exp_not:c { \cs_to_str:N #2 op   }
    \exp_not:n { \tl_if_in:NnT \l_@@_nolimits_tl {#2} \nolimits }
   }
 }
```

\@@_set_math_open:nnn

#1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot

```
\cs_new:Nn \@@_set_math_open:nnn
 {
  \tl_if_in:NnTF \l_@@_radicals_tl {#2}
   {
     \cs_gset_protected_nopar:cpx {\cs_to_str:N #2 sign}
       { \@@_radical:nn {#1} {#3} }
     \tl_set:cn {l_@@_radical_\cs_to_str:N #2_tl} {\use:c{sym #1}~ #3}
   }
   {
     \@@_set_delcode:nnn {#1} {#3} {#3}
     \@@_set_mathcode:nnn {#3} \mathopen {#1}
     \cs_gset_protected_nopar:Npx #2
       { \@@_delimiter:Nnn \mathopen {#1} {#3} }
   }
 }
```

\@@_set_math_close:nnn

#1 : Symbol font name
#2 : Macro to assign
#3 : Glyph slot

```
\cs_new:Nn \@@_set_math_close:nnn
 {
  \@@_set_delcode:nnn {#1} {#3} {#3}
  \@@_set_mathcode:nnn {#3} \mathclose {#1}
  \cs_gset_protected_nopar:Npx #2
    { \@@_delimiter:Nnn \mathclose {#1} {#3} }
 }
```

\@@_set_math_fence:nnnn

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

```
\cs_new:Nn \@@_set_math_fence:nnnn
 {
  \@@_set_mathcode:nnn {#4} {#3} {#1}
  \@@_set_delcode:nnn  {#1} {#4} {#4}
  \cs_gset_protected_nopar:cpx {l \cs_to_str:N #2}
    { \@@_delimiter:Nnn \mathopen  {#1} {#4} }
  \cs_gset_protected_nopar:cpx {r \cs_to_str:N #2}
    { \@@_delimiter:Nnn \mathclose {#1} {#4} }
 }
```

\@@_set_math_accent:Nnnn

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

```
\cs_new:Nn \@@_set_math_accent:Nnnn
 {
  \cs_gset_protected_nopar:Npx #1
   { \@@_accent:nnn {#2} {#3} {#4} }
 }
```

\@@_set_math_overunder:Nnnn

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

```
\cs_new:Nn \@@_set_math_overunder:Nnnn
 {
  \cs_gset_protected_nopar:Npx #1 ##1
   {
    \mathop
     { \@@_accent:nnn {#2} {#3} {#4} {{}##1} }
     %      TODO: remove braces above ^^ which work around a LuaTeX bug
    \limits
   }
 }
```

```
⟨/package⟩
```

File X
um-code-mathtext.dtx

11 *Maths text commands*

```
⟨*package⟩
```

11.1 \setmathfontface

\@@_setmathfontface:Nnn

Interface around \SetMathAlphabet.

```
\keys_define:nn {@@_mathface}
  {
    version .tl_set:N = \l_@@_mversion_tl
  }
```

```
\cs_set:Nn \@@_setmathfontface:Nnn
  {
    \tl_clear:N \l_@@_mversion_tl

    \keys_set_known:nnN {@@_mathface} {#2} \l_@@_keyval_clist

    \exp_args:Nnx \fontspec_set_family:Nxn \l_@@_tmpa_tl
      { ItalicFont={}, BoldFont={}, \exp_not:V \l_@@_keyval_clist } {#3}

    \tl_if_empty:NT \l_@@_mversion_tl
      {
        \tl_set:Nn \l_@@_mversion_tl {normal}
       \DeclareMathAlphabet #1 {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\md-
default} {\updefault}
      }

   \SetMathAlphabet #1 {\l_@@_mversion_tl} {\g_fontspec_encoding_tl} {\l_@@_tmpa_tl} {\md-
default} {\updefault}

    % integrate with fontspec's \setmathrm etc:
    \tl_case:Nn #1
      {
        \mathrm { \cs_set_eq:NN \g__fontspec_mathrm_tl \l_@@_tmpa_tl }
        \mathsf { \cs_set_eq:NN \g__fontspec_mathsf_tl \l_@@_tmpa_tl }
        \mathtt { \cs_set_eq:NN \g__fontspec_mathtt_tl \l_@@_tmpa_tl }
      }
  }
```

11.2 *Hooks into LaTeX 2_ε*

Switching to a different style of alphabetic symbols was traditionally performed with commands like \mathbf, which literally changes fonts to access alternate symbols. This is not as simple with Unicode fonts.

In traditional TEX maths font setups, you simply switch between different 'families' (`\fam`), which is analogous to changing from one font to another—a symbol such as 'a' will be upright in one font, bold in another, and so on. In pkgunicode-math, a different mechanism is used to switch between styles. For every letter (start with ascii a-zA-Z and numbers to keep things simple for now), they are assigned a 'mathcode' with `\Umathcode` that maps from input letter to output font glyph slot. This is done with the equivalent of

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

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

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

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

`\use@mathgroup` To overcome this, we patch `\use@mathgroup`, which is only used inside of commands such as `\mathXYZ`, so this shouldn't have any side-effects. Omit the test for math mode because this is only called *inside* `\mathrm` or similar, which already has a math mode check.

```
\cs_set:Npn \use@mathgroup #1 #2
  {
    \@@_group_begin:
      \cs_if_eq:cNF {M@\f@encoding} #1 {#1}
      \@@_switchto_literal:
      \@@_mathgroup_set:n {#2}
    \@@_group_end:n
  }
```

In LaTeX maths, the command `\operator@font` is defined that switches to the `operator` mathgroup. The classic example is the `\sin` in `$\sin{x}$`; essentially we're using `\mathrm` to typeset the upright symbols, but the syntax is `{\operator@font sin}`. I thought that hooking into `\operator@font` would be hard because all other maths font selection in 2e uses `\mathrm{...}` style. Then reading source2e a little more I stumbled upon `\@fontswitch`. Reimplement that here to avoid `\bgroup`/`\egroup`.

`\operator@font`

```
\cs_set:Npn \operator@font
```

```
  {
    \@@_switchto_literal:
    \@@_fontswitch:n { \g_@@_operator_mathfont_tl }
  }
```

\@@_fontswitch:n Omit the check for math mode as #1 should do that for us.

```
\cs_set:Nn \@@_fontswitch:n
  {
    \cs_set_eq:NN \math@bgroup     \scan_stop:
    \cs_set_eq:NN \@@_group_begin: \scan_stop:
    \cs_set:Npn \@@_group_end:n % takes no argument in this case
      {
        \cs_set_eq:NN \@@_group_begin:  \@@_group_begin_frozen:
        \cs_set_eq:NN \@@_group_end:n   \@@_group_end_frozen:n
        \cs_set_eq:NN \math@bgroup \@@math@bgroup
        \cs_set_eq:NN \math@egroup \@@math@egroup
      }
    \cs_set_eq:NN \math@egroup \@@_group_end:n
    #1 \scan_stop:
  }
```

11.3 *Hooks into fontspec*

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

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

11.3.1 *Text font*

```
\cs_generate_variant:Nn \tl_if_eq:nnT {o}
\cs_set:Nn \__fontspec_setmainfont_hook:nn
  {
    \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault}
      {
⟨XE⟩    \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g__fontspec_mathrm_tl {Renderer=Basic,#1} {#2}
        \__fontspec_setmathrm_hook:nn {#1} {#2}
      }
  }
\cs_set:Nn \__fontspec_setsansfont_hook:nn
  {
    \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault}
      {
⟨XE⟩    \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g__fontspec_mathsf_tl {Renderer=Basic,#1} {#2}
        \__fontspec_setmathsf_hook:nn {#1} {#2}
      }
  }
\cs_set:Nn \__fontspec_setmonofont_hook:nn
  {
    \tl_if_eq:onT {\g__fontspec_mathtt_tl} {\ttdefault}
      {
⟨XE⟩   \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {#1} {#2}
⟨LU⟩   \fontspec_set_family:Nnn \g__fontspec_mathtt_tl {Renderer=Basic,#1} {#2}
        \__fontspec_setmathtt_hook:nn {#1} {#2}
      }
  }
```

11.3.2 *Maths font*

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

```
\cs_set:Nn \__fontspec_setmathrm_hook:nn
  {
   \SetMathAlphabet\mathrm{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\updefault
   \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\mddefault\itdefault
   \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g__fontspec_mathrm_tl\bfdefault\updefault
  }
\cs_set:Nn \__fontspec_setboldmathrm_hook:nn
  {
   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\updefault
   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\bfdefault\updefault
   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g__fontspec_bfmathrm_tl\mddefault\itdefault
  }
\cs_set:Nn \__fontspec_setmathsf_hook:nn
  {
   \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g__fontspec_mathsf_tl\mddefault\updefault
   \SetMathAlphabet\mathsf{bold} \g_fontspec_encoding_tl\g__fontspec_mathsf_tl\bfdefault\updefault
  }
\cs_set:Nn \__fontspec_setmathtt_hook:nn
  {
   \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g__fontspec_mathtt_tl\mddefault\updefault
   \SetMathAlphabet\mathtt{bold} \g_fontspec_encoding_tl\g__fontspec_mathtt_tl\bfdefault\updefault
  }
```

I can't quite remember the logic behind the following two.
If fontspec has been loaded and `\setmathsf` (etc) run, this syncs things up:

```
\tl_if_eq:onF {\g__fontspec_mathrm_tl} {\rmdefault} { \__fontspec_setmathrm_hook:nn {} {} }
\tl_if_eq:onF {\g__fontspec_mathsf_tl} {\sfdefault} { \__fontspec_setmathsf_hook:nn {} {} }
\tl_if_eq:onF {\g__fontspec_mathtt_tl} {\ttdefault} { \__fontspec_setmathtt_hook:nn {} {} }
```

I suppose this is to make things work if neither fontspec or unicode-math load any fonts: (I should check that)

```
\AtBeginDocument
  {
   \tl_if_eq:onT {\g__fontspec_mathrm_tl} {\rmdefault} { \__fontspec_setmathrm_hook:nn {} {} }
   \tl_if_eq:onT {\g__fontspec_mathsf_tl} {\sfdefault} { \__fontspec_setmathsf_hook:nn {} {} }
   \tl_if_eq:onT {\g__fontspec_mathtt_tl} {\ttdefault} { \__fontspec_setmathtt_hook:nn {} {} }
  }
⟨/package⟩
```

File XI

um-code-main.dtx

12 *The main* `\setmathfont` *macro*

```
⟨*package⟩
```

`\@@_setmathfont:nn`

#1 : keyval options
#2 : font name/file

```
\cs_set:Nn \@@_setmathfont:nn
  {
```

- Initialise all local variables.
- Erase any conception LaTeX has of previously defined math symbol fonts; this allows `\DeclareSymbolFont` at any point in the document.
- Grab the current size information: (is this robust enough? Maybe it should be preceded by `\normalsize`). The macro `\S@`⟨*size*⟩ contains the definitions of the sizes used for maths letters, subscripts and subsubscripts in `\tf@size`, `\sf@size`, and `\ssf@size`, respectively.

```

    \@@_init:n {#2}
    \cs_set_eq:NN \glb@currsize \scan_stop:
    \cs_if_exist:cF { S@ \f@size } { \calculate@math@sizes }
    \use:c { S@ \f@size }

    \keys_set_known:nnN {unicode-math} {#1} \l_@@_unknown_keys_clist

    \bool_if:NT \l_@@_init_bool
      {
        \@@_fontspec_trial_font:
        \@@_declare_math_sizes:
      }
    \@@_fontspec_select_font:
    \@@_setup_math_fam:
    \bool_if:NT \l_@@_init_bool
      {
        \@@_setup_legacy_fam_two:
        \@@_setup_legacy_fam_three:
      }

    \@@_input_math_symbol_table:

```

- the 'once-off' setup that doesn't need to be per-font
- 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 (\symbf etc.) — this is an extensive part of the code; see Section 15;

```
    \bool_if:NT \l_@@_init_bool \@@_onceoff_setup:
    \@@_remap_symbols:
    \@@_setup_mathactives:
    \@@_setup_delcodes:
    \@@_setup_alphabets:

   %% TODO: what of the above should only be run for the "default"/"main" font?

    \bool_if:NTF \l_@@_init_bool
      {
        \bool_gset_true:N \g_@@_main_font_defined_bool
⟨LU⟩    \@@_mathparam_store:
        \@@_log:n {default-math-font}
      }
      {
⟨LU⟩    \@@_mathparam_restore:
      }
  }
```

Fall-back font Want to load Latin Modern Math if nothing else. This needs to happen early so that all of the font-loading machinery executes before the other 'AtBeginDocument' code.

```
\AtBeginDocument { \@@_load_lm_if_necessary: }
\cs_new:Nn \@@_load_lm_if_necessary:
  {
    \bool_if:NF \g_@@_main_font_defined_bool
      {
⟨debug⟩\typeout{SETTING~ DEFAULT~ FONT~(latinmodern-math)}
        \setmathfont{latinmodern-math.otf}
          [BoldFont={latinmodern-math.otf}]
⟨debug⟩\typeout{END~ DEFAULT~ FONT)}
      }
  }
```

\@@_init:n Reset local variables. Default to defining the font for every math symbol character.

```
\cs_new:Nn \@@_init:n
  {
    \tl_set:Nn \l_@@_fontname_tl {#1}
    \bool_set_true:N  \l_@@_ot_math_bool
    \tl_set:Nn \l_@@_mversion_tl      {normal}
    \tl_set:Nn \l_@@_symfont_label_tl {operators}

    \tl_set:Nn    \l_@@_script_features_tl  {Style=MathScript}
    \tl_set:Nn    \l_@@_sscript_features_tl {Style=MathScriptScript}
    \tl_set_eq:NN \l_@@_script_font_tl      \l_@@_fontname_tl
    \tl_set_eq:NN \l_@@_sscript_font_tl     \l_@@_fontname_tl

    \bool_set_true:N \l_@@_init_bool
    \seq_gclear:N    \g_@@_char_range_seq
    \clist_clear:N   \l_@@_mathmap_charints_clist
    \seq_gclear:N    \g_@@_mathalph_seq
    \seq_clear:N     \l_@@_missing_alph_seq

    \cs_set_eq:NN \_@@_sym:nnn                  \@@_process_symbol_noparse:nnn
    \cs_set_eq:NN \@@_remap_symbol:nnn          \@@_remap_symbol_noparse:nnn
    \cs_set_eq:NN \@@_maybe_init_alphabet:n     \@@_init_alphabet:n
    \cs_set_eq:NN \@@_assign_delcode:nn         \@@_assign_delcode_noparse:nn
   \cs_set_eq:NN \@@_make_mathactive:nNN      \@@_make_mathactive_noparse:nNN
  }
```

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

```
\tl_new:N \g_@@_main_font_cmd_tl
\cs_new:Nn \@@_declare_math_sizes:
  {
    \dim_compare:nF { \fontdimen 10 \g_@@_trial_font == 0pt }
      {
        \DeclareMathSizes { \f@size } { \f@size }
          { \@@_fontdimen_to_scale:nN {10} \g_@@_trial_font }
          { \@@_fontdimen_to_scale:nN {11} \g_@@_trial_font }
      }
  }
```

\@@_fontspec_trial_font:

```
\cs_new:Nn \@@_fontspec_trial_font:
  {
    \tl_set:Nx \l_@@_font_keyval_tl
      {
⟨LU⟩   Renderer = Basic,
        BoldItalicFont = {}, ItalicFont = {},
        Script = Math,
⟨LU⟩   FontAdjustment = { \@@_luatex_copy_fontdimens: },
        \l_@@_unknown_keys_clist
      }

   \fontspec_set_family:Nxn \l_@@_trial_family_tl {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

    \group_begin:
      \fontfamily { \l_@@_trial_family_tl } \selectfont
    \exp_last_unbraced:NNo \@@_fontface_gset_eq:NN \g_@@_trial_font \font@name
      \fontspec_if_script:nF {math}
        {
          \bool_gset_false:N \l_@@_ot_math_bool
          \bool_gset_false:N \l_@@_init_bool
        }
    \group_end:

  }
```

\@@_fontspec_select_font:

```
\cs_new:Nn \@@_fontspec_select_font:
  {
    \fp_gset:Nn \g_@@_size_tfsf_fp   { (\f@size +\sf@size )/2 }
    \fp_gset:Nn \g_@@_size_sfssf_fp  { (\sf@size+\ssf@size)/2 }

    \tl_set:Nx \l_@@_font_keyval_tl
      {
⟨LU⟩  Renderer = Basic,
        BoldItalicFont = {}, ItalicFont = {},
        Script = Math,
        SizeFeatures =
          {
            {
              Size = \fp_use:N \g_@@_size_tfsf_fp -
            } ,
            {
          Size = \fp_use:N \g_@@_size_sfssf_fp - \fp_use:N \g_@@_size_tfsf_fp ,
              Font = \l_@@_script_font_tl ,
              \l_@@_script_features_tl
            } ,
            {
              Size = - \fp_use:N \g_@@_size_sfssf_fp ,
              Font = \l_@@_sscript_font_tl ,
              \l_@@_sscript_features_tl
            }
          } ,
⟨LU⟩  FontAdjustment = { \@@_luatex_copy_fontdimens: },
        \l_@@_unknown_keys_clist
      }

   \fontspec_set_family:Nxn \l_@@_family_tl {\l_@@_font_keyval_tl} {\l_@@_fontname_tl}

    \int_gincr:N \g_@@_fonts_used_int
    \group_begin:
      \fontfamily { \l_@@_family_tl } \selectfont
    \exp_last_unbraced:Nno \@@_fontface_gset_eq:cN {g_@@_mathfont_ \int_use:N \g_@@_fonts_used_int _f
    \tl_gset:Nx \g_@@_curr_font_cmd_tl { \exp_not:c {g_@@_mathfont_ \int_use:N \g_@@_fonts_used_int _f
      \bool_if:NT \l_@@_init_bool
{
\exp_last_unbraced:NNo \@@_fontface_gset_eq:NN \l_@@_font \font@name
}
```

```
      \fontspec_if_script:nF {math}
        {
          \bool_gset_false:N \l_@@_ot_math_bool
          \bool_gset_false:N \l_@@_init_bool
        }
    \group_end:
  }
```

```
\tl_set:Nn \g_@@_main_font_cmd_tl  { \l_@@_font }
\tl_set:Nn \g_@@_sqrt_font_cmd_tl  { \l_@@_font }
\tl_set:Nn \g_@@_prime_font_cmd_tl { \l_@@_font }
```

\@@_luatex_copy_fontdimens:

```
⟨*LU⟩
\cs_new_protected:Nn \@@_luatex_copy_fontdimens:
  {
    \@@_fontdimen_from_param:nn {10} {ScriptPercentScaleDown}
    \@@_fontdimen_from_param:nn {11} {ScriptScriptPercentScaleDown}
    \@@_fontdimen_from_param:nn {15} {AxisHeight}
    \@@_fontdimen_from_param:nn {18} {SubscriptShiftDown}
    \@@_fontdimen_from_param:nn {20} {SubscriptBaselineDropMin}
    \@@_fontdimen_from_param:nn {21} {SuperscriptShiftUp}
    \@@_fontdimen_from_param:nn {22} {SuperscriptShiftUpCramped}
    \@@_fontdimen_from_param:nn {24} {SuperscriptBaselineDropMax}
    \@@_fontdimen_from_param:nn {28} {UpperLimitGapMin}
    \@@_fontdimen_from_param:nn {29} {UpperLimitBaselineRiseMin}
    \@@_fontdimen_from_param:nn {30} {LowerLimitGapMin}
    \@@_fontdimen_from_param:nn {31} {LowerLimitBaselineDropMin}
    \@@_fontdimen_from_param:nn {32} {StackTopShiftUp}
    \@@_fontdimen_from_param:nn {42} {FractionNumeratorShiftUp}
    \@@_fontdimen_from_param:nn {43} {FractionNumeratorDisplayStyleShiftUp}
    \@@_fontdimen_from_param:nn {44} {FractionDenominatorShiftDown}
   \@@_fontdimen_from_param:nn {45} {FractionDenominatorDisplayStyleShiftDown}
    \@@_fontdimen_from_param:nn {48} {FractionRuleThickness}
  }
⟨/LU⟩
```

\@@_setup_math_fam:

```
\cs_new:Nn \@@_setup_math_fam:
  {
    \cs_if_exist:cF { sym \l_@@_symfont_label_tl }
      {
        \DeclareSymbolFont{\l_@@_symfont_label_tl}
          {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
      }
    \SetSymbolFont{\l_@@_symfont_label_tl}{\l_@@_mversion_tl}
      {\encodingdefault}{\l_@@_family_tl}{\mddefault}{\updefault}
```

Set the bold math version.

```
    \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
    {
      \SetSymbolFont{\l_@@_symfont_label_tl}{bold}
        {\encodingdefault}{\l_@@_family_tl}{\bfdefault}{\updefault}
    }
}
```

\@@_setup_legacy_fam_two: TeX won't load the same font twice at the same scale, so we need to magnify this one by an imperceptable amount.

```
\cs_new:Nn \@@_setup_legacy_fam_two:
  {
    \fontspec_set_family:Nxn \l_@@_fam_two_tl
      {
        \l_@@_font_keyval_tl,
        Scale=1.00001,
        FontAdjustment =
          {
            \@@_copy_fontdimen:nnN { 8} {43} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN { 9} {42} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {10} {32} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {11} {45} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {12} {44} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {13} {21} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {14} {21} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {15} {22} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {16} {18} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {17} {18} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {18} {24} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {19} {20} \g_@@_main_font_cmd_tl
            \@@_copy_fontdimen:nnN {22} {15} \g_@@_main_font_cmd_tl
           \@@_zero_fontdimen:n   {20} % delim1 = FractionDelimiterDisplaySize
            \@@_zero_fontdimen:n   {21} % delim2 = FractionDelimiterSize
          }
      } {\l_@@_fontname_tl}

    \SetSymbolFont{symbols}{\l_@@_mversion_tl}
      {\encodingdefault}{\l_@@_fam_two_tl}{\mddefault}{\updefault}

    \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
      {
        \SetSymbolFont{symbols}{bold}
          {\encodingdefault}{\l_@@_fam_two_tl}{\bfdefault}{\updefault}
      }
  }
```

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

```
\cs_new:Nn \@@_setup_legacy_fam_three:
  {
    \fontspec_set_family:Nxn \l_@@_fam_three_tl
      {
        \l_@@_font_keyval_tl,
        Scale=0.99999,
        FontAdjustment = {
          \@@_copy_fontdimen:nnN { 8} {48} \g_@@_main_font_cmd_tl
          \@@_copy_fontdimen:nnN { 9} {28} \g_@@_main_font_cmd_tl
          \@@_copy_fontdimen:nnN {10} {30} \g_@@_main_font_cmd_tl
          \@@_copy_fontdimen:nnN {11} {29} \g_@@_main_font_cmd_tl
          \@@_copy_fontdimen:nnN {12} {31} \g_@@_main_font_cmd_tl
          \@@_zero_fontdimen:n   {13}
       }
      } {\l_@@_fontname_tl}

    \SetSymbolFont{largesymbols}{\l_@@_mversion_tl}
      {\encodingdefault}{\l_@@_fam_three_tl}{\mddefault}{\updefault}

    \str_if_eq_x:nnT {\l_@@_mversion_tl} {normal}
      {
        \SetSymbolFont{largesymbols}{bold}
          {\encodingdefault}{\l_@@_fam_three_tl}{\bfdefault}{\updefault}
      }
  }
```

\@@_onceoff_setup:

```
\cs_new:Nn \@@_onceoff_setup:
  {
    \@@_set_delcode:nnn {operators} {`\.} {\c_zero}
  }
```

12.1 *Functions for setting up symbols with mathcodes*

\@@_process_symbol_noparse:nnn
\@@_process_symbol_parse:nnn

If the range font feature has been used, then only a subset of the Unicode glyphs are to be defined. See section §13.3 for the code that enables this.

```
\cs_set:Nn \@@_process_symbol_noparse:nnn
  {
    \@@_set_mathsymbol:nNNn {\l_@@_symfont_label_tl} #2 #3 {#1}
  }
\cs_set:Nn \@@_process_symbol_parse:nnn
  {
    \@@_if_char_spec:nNT {#1} {#3}
      {
        \@@_process_symbol_noparse:nnn {#1} {#2} {#3}
      }
  }
```

\@@_remap_symbols:

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

```
\cs_new:Nn \@@_remap_symbols:
  {
    \@@_remap_symbol:nnn {`\-} {\mathbin} {"2212}
     \@@_remap_symbol:nnn {`\*} {\mathbin} {"02217}% text asterisk to "cen-
tred asterisk"
    \bool_if:NF \g_@@_literal_colon_bool
      {
              \@@_remap_symbol:nnn {`\:} {\mathrel} {"02236}% colon to ra-
tio (i.e., punct to rel)
      }
  }
```

`\@@_remap_symbol_noparse:nnn`
`\@@_remap_symbol_parse:nnn`

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

```
\cs_new:Nn \@@_remap_symbol_parse:nnn
  {
    \@@_if_char_spec:nNT {#3} {#2}
      { \@@_remap_symbol_noparse:nnn {#1} {#2} {#3} }
  }
\cs_new:Nn \@@_remap_symbol_noparse:nnn
  {
    \clist_map_inline:nn {#1}
      { \@@_set_mathcode:nnnn {##1} {#2} {\l_@@_symfont_label_tl} {#3} }
  }
```

12.2 *Active math characters*

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

`\@@_setup_mathactives:`

TODO: if not an OpenType math font, we should ignore doing anything with primes. This needs a revamped 'range' feature, I think.

```
\cs_new:Npn \@@_setup_mathactives:
  {
\@@_make_mathactive:nNN {"2032} \@@_prime_single_mchar \mathord
\@@_make_mathactive:nNN {"2033} \@@_prime_double_mchar \mathord
\@@_make_mathactive:nNN {"2034} \@@_prime_triple_mchar \mathord
\@@_make_mathactive:nNN {"2057} \@@_prime_quad_mchar   \mathord
\@@_make_mathactive:nNN {"2035} \@@_backprime_single_mchar \mathord
\@@_make_mathactive:nNN {"2036} \@@_backprime_double_mchar \mathord
\@@_make_mathactive:nNN {"2037} \@@_backprime_triple_mchar \mathord
    \@@_make_mathactive:nNN {`\'} \mathstraightquote \mathord
    \@@_make_mathactive:nNN {`\`} \mathbacktick      \mathord
  }
```

`\@@_make_mathactive:nNN`

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

```
\cs_new:Nn \@@_make_mathactive_parse:nNN
  {
    \@@_if_char_spec:nNT {#1} #3
      { \@@_make_mathactive_noparse:nNN {#1} #2 #3 }
  }
\cs_new:Nn \@@_make_mathactive_noparse:nNN
  {
    \@@_set_mathchar:NNnn #2 #3 {\l_@@_symfont_label_tl} {#1}
    \@@_char_gmake_mathactive:n {#1}
  }
```

12.3 Delimiter codes

\@@_assign_delcode:nn

```
\cs_new:Nn \@@_assign_delcode_noparse:nn
  {
    \@@_set_delcode:nnn \l_@@_symfont_label_tl {#1} {#2}
  }
\cs_new:Nn \@@_assign_delcode_parse:nn
  {
    \@@_if_char_spec:nNT {#2} \@nil
      {
        \@@_assign_delcode_noparse:nn {#1} {#2}
      }
  }
```

\@@_assign_delcode:n

Shorthand.

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

\@@_setup_delcodes:

Some symbols that aren't mathopen/mathclose still need to have delimiter codes assigned. The list of vertical arrows may be incomplete. On the other hand, many fonts won't support them all being stretchy. And some of them are probably not meant to stretch, either. But adding them here doesn't hurt.

```
\cs_new:Npn \@@_setup_delcodes:
  {
    \@@_assign_delcode:nn {`\/}   {\g_@@_slash_delimiter_usv}
    \@@_assign_delcode:nn {"2044} {\g_@@_slash_delimiter_usv} % fracslash
    \@@_assign_delcode:nn {"2215} {\g_@@_slash_delimiter_usv} % divslash
    \@@_assign_delcode:n {"005C} % backslash
    \@@_assign_delcode:nn {`\<} {"27E8} % angle brackets with ascii notation
    \@@_assign_delcode:nn {`\>} {"27E9} % angle brackets with ascii notation
    \@@_assign_delcode:n {"2191} % up arrow
    \@@_assign_delcode:n {"2193} % down arrow
    \@@_assign_delcode:n {"2195} % updown arrow
    \@@_assign_delcode:n {"219F} % up arrow twohead
    \@@_assign_delcode:n {"21A1} % down arrow twohead
    \@@_assign_delcode:n {"21A5} % up arrow from bar
    \@@_assign_delcode:n {"21A7} % down arrow from bar
    \@@_assign_delcode:n {"21A8} % updown arrow from bar
    \@@_assign_delcode:n {"21BE} % up harpoon right
    \@@_assign_delcode:n {"21BF} % up harpoon left
    \@@_assign_delcode:n {"21C2} % down harpoon right
    \@@_assign_delcode:n {"21C3} % down harpoon left
    \@@_assign_delcode:n {"21C5} % arrows up down
    \@@_assign_delcode:n {"21F5} % arrows down up
    \@@_assign_delcode:n {"21C8} % arrows up up
    \@@_assign_delcode:n {"21CA} % arrows down down
    \@@_assign_delcode:n {"21D1} % double up arrow
    \@@_assign_delcode:n {"21D3} % double down arrow
    \@@_assign_delcode:n {"21D5} % double updown arrow
    \@@_assign_delcode:n {"21DE} % up arrow double stroke
    \@@_assign_delcode:n {"21DF} % down arrow double stroke
    \@@_assign_delcode:n {"21E1} % up arrow dashed
    \@@_assign_delcode:n {"21E3} % down arrow dashed
    \@@_assign_delcode:n {"21E7} % up white arrow
    \@@_assign_delcode:n {"21E9} % down white arrow
    \@@_assign_delcode:n {"21EA} % up white arrow from bar
    \@@_assign_delcode:n {"21F3} % updown white arrow
  }
```

12.4 *(Big) operators*

The engine does what is necessary to deal with big operators for us automatically with `\Umathchardef`. However, the limits aren't set automatically; that is, we want to define, a la Plain TEX *etc.*, `\def\int{\intop\nolimits}`, so there needs to be a transformation from `\int` to `\intop` during the expansion of _@@_sym:nnn in the appropriate contexts.

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

```
\tl_set:Nn \l_@@_nolimits_tl
  {
    \int\iint\iiint\iiiint\oint\oiint\oiiint
    \intclockwise\varointclockwise\ointctrclockwise\sumint
    \intbar\intBar\fint\cirfnint\awint\rppolint
    \scpolint\npolint\pointint\sqint\intlarhk\intx
    \intcap\intcup\upint\lowint
  }
```

12.5 *Radicals*

`\l_@@_radicals_tl` The radicals are organised in `\@@_set_mathsymbol:nNNn`. We organise radicals in the same way as nolimits-operators. (`\cuberoot` and `\fourthroot`, don't seem to behave as proper radicals.)

```
\tl_set:Nn \l_@@_radicals_tl {\sqrt \longdivision}
```

12.6 Fontdimens

```
⟨*LU⟩
```

\@@_mathparam_restore:

```
\cs_set:Nn \@@_mathparam_restore:
  {
    \check@mathfonts
    \tl_use:N \g_@@_mathparam_store_tl
  }
```

\@@_mathparam_store:

```
\cs_set:Nn \@@_mathparam_store:
  {
    \check@mathfonts
    \tl_gset:Nx \g_@@_mathparam_store_tl
      {
        \@@_mathparam_store_aux:N \displaystyle
        \@@_mathparam_store_aux:N \textstyle
        \@@_mathparam_store_aux:N \scriptstyle
        \@@_mathparam_store_aux:N \scriptscriptstyle
      }
  }
\cs_set:Nn \@@_mathparam_store_aux:N
  {
    \Umathquad                 #1 = \the \Umathquad                 #1
    \Umathaxis                 #1 = \the \Umathaxis                 #1
    \Umathoperatorsize         #1 = \the \Umathoperatorsize         #1
    \Umathoverbarkern          #1 = \the \Umathoverbarkern          #1
    \Umathoverbarrule          #1 = \the \Umathoverbarrule          #1
    \Umathoverbarvgap          #1 = \the \Umathoverbarvgap          #1
    \Umathunderbarkern         #1 = \the \Umathunderbarkern         #1
    \Umathunderbarrule         #1 = \the \Umathunderbarrule         #1
    \Umathunderbarvgap         #1 = \the \Umathunderbarvgap         #1
    \Umathradicalkern          #1 = \the \Umathradicalkern          #1
    \Umathradicalrule          #1 = \the \Umathradicalrule          #1
    \Umathradicalvgap          #1 = \the \Umathradicalvgap          #1
    \Umathradicaldegreebefore #1 = \the \Umathradicaldegreebefore #1
    \Umathradicaldegreeafter  #1 = \the \Umathradicaldegreeafter  #1
    \Umathradicaldegreeraise  #1 = \the \Umathradicaldegreeraise  #1
    \Umathstackvgap            #1 = \the \Umathstackvgap            #1
    \Umathstacknumup           #1 = \the \Umathstacknumup           #1
    \Umathstackdenomdown       #1 = \the \Umathstackdenomdown       #1
    \Umathfractionrule         #1 = \the \Umathfractionrule         #1
    \Umathfractionnumvgap      #1 = \the \Umathfractionnumvgap      #1
    \Umathfractionnumup        #1 = \the \Umathfractionnumup        #1
    \Umathfractiondenomvgap    #1 = \the \Umathfractiondenomvgap    #1
    \Umathfractiondenomdown    #1 = \the \Umathfractiondenomdown    #1
    \Umathfractiondelsize      #1 = \the \Umathfractiondelsize      #1
    \Umathlimitabovevgap       #1 = \the \Umathlimitabovevgap       #1
    \Umathlimitabovebgap      #1 = \the \Umathlimitabovebgap      #1
    \Umathlimitabovekern      #1 = \the \Umathlimitabovekern      #1
    \Umathlimitbelowvgap      #1 = \the \Umathlimitbelowvgap      #1
    \Umathlimitbelowbgap      #1 = \the \Umathlimitbelowbgap      #1
    \Umathlimitbelowkern      #1 = \the \Umathlimitbelowkern      #1
    \Umathoverdelimitervgap   #1 = \the \Umathoverdelimitervgap   #1
    \Umathoverdelimiterbgap   #1 = \the \Umathoverdelimiterbgap   #1
    \Umathunderdelimitervgap  #1 = \the \Umathunderdelimitervgap  #1
    \Umathunderdelimiterbgap  #1 = \the \Umathunderdelimiterbgap  #1
    \Umathsubshiftdrop        #1 = \the \Umathsubshiftdrop        #1
    \Umathsubshiftdown        #1 = \the \Umathsubshiftdown        #1
    \Umathsupshiftdrop        #1 = \the \Umathsupshiftdrop        #1
    \Umathsupshiftup          #1 = \the \Umathsupshiftup          #1
    \Umathsubsupshiftdown     #1 = \the \Umathsubsupshiftdown     #1
    \Umathsubtopmax           #1 = \the \Umathsubtopmax           #1
    \Umathsupbottommin        #1 = \the \Umathsupbottommin        #1
    \Umathsupsubbottommax     #1 = \the \Umathsupsubbottommax     #1
    \Umathsubsupvgap          #1 = \the \Umathsubsupvgap          #1
    \Umathspaceafterscript    #1 = \the \Umathspaceafterscript    #1
    \Umathconnectoroverlapmin #1 = \the \Umathconnectoroverlapmin #1
  }
```

⟨/LU⟩

⟨/package⟩

File XII
um-code-fontopt.dtx

13 Font loading options

```
⟨*package⟩
```

13.1 Math version

```
\keys_define:nn {unicode-math}
  {
    version .code:n =
      {
        \tl_set:Nn \l_@@_mversion_tl {#1}
        \DeclareMathVersion {\l_@@_mversion_tl}
      }
  }
```

13.2 Script and scriptscript font options

```
\keys_define:nn {unicode-math}
  {
    script-features  .tl_set:N =  \l_@@_script_features_tl ,
    sscript-features .tl_set:N = \l_@@_sscript_features_tl ,
         script-font .tl_set:N =      \l_@@_script_font_tl ,
        sscript-font .tl_set:N =     \l_@@_sscript_font_tl ,
  }
```

13.3 Range processing

Locally redefined all math symbol commands to their slot number prefixed by a quark. Similary for the math classes.

```
\keys_define:nn {unicode-math}
  {
    range .code:n =
      {
       \bool_if:NF \g_@@_main_font_defined_bool { \@@_error:n {no-main-font} }
        \bool_set_false:N \l_@@_init_bool
        \@@_range_init:
        \group_begin:
          \seq_map_inline:Nn \g_@@_mathclasses_seq
            {
              \cs_set:Npn ##1 { \use_none:n \q_unicode_math \exp_not:N ##1 }
            }
          \cs_set:Npn \_@@_sym:nnn ##1 ##2 ##3
            {
              \cs_set:Npn ##2 { \use_none:n \q_unicode_math ##1 }
            }
          \@@_input_math_symbol_table:
```

```
          \@@_range_process:n {#1}
        \group_end:
      }
  }
```

`\@@_range_init:` Set processing functions if we're not defining the full Unicode math repetoire. Math symbols are defined with _@@_sym:nnn; see section §12.1 for the individual definitions

```
\cs_new:Nn \@@_range_init:
  {
    \int_incr:N \g_@@_fam_int
    \tl_set:Nx \l_@@_symfont_label_tl {@@_fam\int_use:N\g_@@_fam_int}
    \cs_set_eq:NN \_@@_sym:nnn \@@_process_symbol_parse:nnn
    \cs_set_eq:NN \@@_remap_symbol:nnn \@@_remap_symbol_parse:nnn
    \cs_set_eq:NN \@@_maybe_init_alphabet:n \use_none:n
    \cs_set_eq:NN \@@_assign_delcode:nn \@@_assign_delcode_parse:nn
    \cs_set_eq:NN \@@_make_mathactive:nNN \@@_make_mathactive_parse:nNN
```

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

```
    \seq_gclear:N \g_@@_char_range_seq
    \seq_gclear:N \g_@@_mclass_range_seq
    \seq_gclear:N \g_@@_mathalph_seq
  }
```

`\@@_range_process:`

```
\cs_new:Nn \@@_range_process:n
  {
    \clist_map_inline:nn {#1}
      {
        \@@_mathalph_decl:nF {##1} { \@@_range_decl:n {##1} }
      }
}
```

`\@@_mathalph_decl:nF` Possible forms of input:
`\mathscr`
`\mathscr->\mathup`
`\mathscr/{Latin}`
`\mathscr/{Latin}->\mathup`
Outputs:
tmpa: math style (*e.g.*, `\mathscr`)
tmpb: alphabets (*e.g.*, `Latin`)
tmpc: remap style (*e.g.*, `\mathup`). Defaults to `tmpa`.

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

```
\cs_new:Nn \@@_mathalph_decl:nF
  {

    \tl_set:Nn  \l_@@_tmpa_tl {#1}
    \tl_clear:N \l_@@_tmpb_tl
    \tl_clear:N \l_@@_tmpc_tl

    \tl_if_in:NnT \l_@@_tmpa_tl {->}
      { \exp_after:wN \@@_split_arrow:w \l_@@_tmpa_tl \q_nil }

    \tl_if_in:NnT \l_@@_tmpa_tl {/}
      { \exp_after:wN \@@_split_slash:w \l_@@_tmpa_tl \q_nil }

    \tl_set:Nx \l_@@_tmpa_tl { \tl_to_str:N \l_@@_tmpa_tl }
    \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \math }
    \exp_args:NNx \tl_remove_all:Nn \l_@@_tmpa_tl { \token_to_str:N \sym }
    \tl_trim_spaces:N \l_@@_tmpa_tl

    \tl_if_empty:NT \l_@@_tmpc_tl
      { \tl_set_eq:NN \l_@@_tmpc_tl \l_@@_tmpa_tl }

    \clist_if_in:NVT \g_@@_bad_alpha_clist \l_@@_tmpa_tl { \@@_error:n {range-not-bf-sf} }

    \prop_if_exist:cTF {g_@@_named_range_ \l_@@_tmpa_tl _prop}
      {
        \seq_gput_right:Nx \g_@@_mathalph_seq
          {
            { \exp_not:V \l_@@_tmpa_tl }
            { \exp_not:V \l_@@_tmpb_tl }
            { \exp_not:V \l_@@_tmpc_tl }
          }
      }
      {#2}
  }
\cs_set:Npn \@@_split_arrow:w #1->#2 \q_nil
  {
    \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
    \tl_set:Nx \l_@@_tmpc_tl { \tl_trim_spaces:n {#2} }
  }
\cs_set:Npn \@@_split_slash:w #1/#2 \q_nil
  {
    \tl_set:Nx \l_@@_tmpa_tl { \tl_trim_spaces:n {#1} }
    \tl_set:Nx \l_@@_tmpb_tl { \tl_trim_spaces:n {#2} }
  }
```

\@@_range_decl:n

```
\cs_new_protected:Nn \@@_range_decl:n
  {
    \bool_lazy_and:nnTF { \tl_if_single_p:n {#1} } { \token_if_cs_p:N #1 }
      % IF A CSNAME:
      {
        \tl_if_in:VnTF #1 { \q_unicode_math }
          {
            \seq_if_in:NnTF \g__um_mathclasses_seq {#1}
              { \seq_gput_right:Nn \g_@@_mclass_range_seq {#1} }
              { \seq_gput_right:Nx \g_@@_char_range_seq   { #1 } }
          }
          { \@@_error:nx {bad-cs-in-range} { \tl_to_str:n {#1} } }
      }
      % ELSE ASSUME NUMERIC INPUT:
      {
        \seq_gput_right:Nx \g_@@_char_range_seq { #1 }
      }
  }
```

\@@_if_char_spec:nNT

#1	:	Unicode character slot
#2	:	control sequence (math class)
#3	:	code to execute

This macro expands to #3 if any of its arguments are contained in \g_@@_char_range_seq. This list can contain either character ranges (for checking with #1) or control sequences. These latter can either be the command name of a specific character, *or* the math type of one (*e.g.*, \mathbin).

Character ranges are passed to \@@_if_char_spec:nNT, which accepts input in the form shown in table 1.

Table 1: Ranges accepted by \@@_if_char_spec:nNT.

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

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

```
\cs_new:Nn \@@_if_char_spec:nNT
  {
    % math class:
    \seq_if_in:NnT \g_@@_mclass_range_seq {#2}
      { \use_none_delimit_by_q_nil:w }

    % character slot:
    \seq_map_inline:Nn \g_@@_char_range_seq
      {
        \@@_int_if_slot_is_last_in_range:nnT {#1} {##1}
          { \seq_gremove_all:Nn \g_@@_char_range_seq {##1} }

        \@@_int_if_slot_in_range:nnT {#1} {##1}
          { \seq_map_break:n { \use_none_delimit_by_q_nil:w } }
      }

    % the following expands to nil if no match was found:
    \use_none:nnn
    \q_nil
    \use:n
      {
        \cs_if_eq:NNT #2 \mathalpha
          {
          \clist_put_right:Nx \l_@@_mathmap_charints_clist { \int_eval:n {#1} }
          }
        #3
      }
  }
```

\@@_int_if_slot_in_range:nnT Pretty basic comma separated range processing. Donald Arseneau's selectp package has a cleverer technique.

A 'numrange' is like `-2,5-8,12,17-` (can be unsorted).

Four cases, four argument types:

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

```
\cs_new:Nn \@@_int_if_slot_in_range:nnT
  {
    \@@_numrange_parse:nwT {#1} #2 - \q_nil - \q_stop {#3}
  }
\cs_set:Npn \@@_numrange_parse:nwT #1 #2 - #3 - #4 \q_stop #5
  {
    \tl_if_empty:nTF {#4} { \int_compare:nT {#1=#2} {#5} }
      {
    \tl_if_empty:nTF {#3} { \int_compare:nT {#1>=#2} {#5} }
      {
    \tl_if_empty:nTF {#2} { \int_compare:nT {#1<=#3} {#5} }
      {
    \int_compare:nT {#1>=#2} { \int_compare:nT {#1<=#3} {#5} }
      } } }
  }
\cs_new:Nn \@@_int_if_slot_is_last_in_range:nnT
  {
    \@@_numrange_last_parse:nwT {#1} #2 - \q_nil - \q_stop {#3}
  }
\cs_set:Npn \@@_numrange_last_parse:nwT #1 #2 - #3 - #4 \q_stop #5
  {
    \tl_if_empty:nTF {#4} { \int_compare:nT {#1==#2} {#5} }
      {
    \tl_if_empty:nTF {#2} { \int_compare:nT {#1==#3} {#5} }
      {
```

```
                                      \int_compare:nT {#1==#3} {#5}
      } }
  }
⟨/package⟩
```

File XIII
um-code-fontparam.dtx

14 *Cross-platform interface for font parameters*

```
⟨*package⟩
```

XeTeX and LuaTeX have different interfaces for math font parameters. We use LuaTeX's interface because it's much better, but rename the primitives to be more LaTeX3-like. There are getter and setter commands for each font parameter. The names of the parameters is derived from the LuaTeX names, with underscores inserted between words. For every parameter `\Umath⟨LuaTeX name⟩`, we define an expandable getter command `\@@_⟨LaTeX3 name⟩:N` and a protected setter command `\@@_set_⟨LaTeX3 name⟩:Nn`. The getter command takes one of the style primitives (`\displaystyle` etc.) and expands to the font parameter, which is a ⟨*dimension*⟩. The setter command takes a style primitive and a dimension expression, which is parsed with `\dim_eval:n`.

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

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

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

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

Define ⟨*command*⟩ as a new cramped style switch. For LuaTeX, simply rename the corresponding primitive if it is not already defined. For XeTeX, define ⟨*command*⟩ as a new quark.

```
\cs_new_protected_nopar:Nn \@@_new_cramped_style:N
⟨XE⟩  { \quark_new:N #1 }
⟨LU⟩  {
⟨LU⟩    \cs_if_exist:NF #1
⟨LU⟩      { \cs_new_eq:Nc #1 { luatex \cs_to_str:N #1 } }
⟨LU⟩  }
```

\crampeddisplaystyle
\crampedtextstyle
\crampedscriptstyle
\crampedscriptscriptstyle

The cramped style commands.

```
\@@_new_cramped_style:N \crampeddisplaystyle
\@@_new_cramped_style:N \crampedtextstyle
\@@_new_cramped_style:N \crampedscriptstyle
\@@_new_cramped_style:N \crampedscriptscriptstyle
```

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

\@@_font_dimen:Nnnnn

#1 : style token
#2 : font dimen for display style
#3 : font dimen for cramped display style
#4 : font dimen for non-display styles
#5 : font dimen for cramped non-display styles

Map math style to XƎTEX math font dimension. ⟨*style token*⟩ must be one of the style switches (\displaystyle, \crampeddisplaystyle, ...). The other parameters are integer constants referring to font dimension numbers. The macro expands to a dimension which contains the appropriate font dimension.

```
⟨*XE⟩
  \cs_new_nopar:Npn \@@_font_dimen:Nnnnn #1 #2 #3 #4 #5 {
    \fontdimen
    \cs_if_eq:NNTF #1 \displaystyle {
      #2 \textfont
    } {
      \cs_if_eq:NNTF #1 \crampeddisplaystyle {
        #3 \textfont
      } {
        \cs_if_eq:NNTF #1 \textstyle {
          #4 \textfont
        } {
          \cs_if_eq:NNTF #1 \crampedtextstyle {
            #5 \textfont
          } {
            \cs_if_eq:NNTF #1 \scriptstyle {
              #4 \scriptfont
            } {
              \cs_if_eq:NNTF #1 \crampedscriptstyle {
                #5 \scriptfont
              } {
                \cs_if_eq:NNTF #1 \scriptscriptstyle {
                  #4 \scriptscriptfont
                } {
```

Should we check here if the style is invalid?

```
                  #5 \scriptscriptfont
                }
                }
              }
            }
          }
        }
      }
```

Which family to use?

```
      \c_two
    }
⟨/XE⟩
```

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

\@@_font_param:nnnnn

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

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTEX font parameter name is produced by removing all underscores and prefixing the result with `Umath`. The XƎTEX font dimension numbers must be integer constants.

```
\cs_new_protected_nopar:Nn \@@_font_param:nnnnn
⟨*XE⟩
  {
    \@@_font_param_aux:ccnnnn { @@_ #1 :N } { @@_set_ #1 :Nn }
      { #2 } { #3 } { #4 } { #5 }
  }
⟨/XE⟩
⟨*LU⟩
  {
    \tl_set:Nn \l_@@_tmpa_tl { #1 }
    \tl_remove_all:Nn \l_@@_tmpa_tl { _ }
    \@@_font_param_aux:ccc { @@_ #1 :N } { @@_set_ #1 :Nn }
      { Umath \l_@@_tmpa_tl }
  }
⟨/LU⟩
```

\@@_font_param:nnn

#1 : name
#2 : font dimension for display style
#3 : font dimension for non-display styles

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTEX font parameter name is produced by removing all underscores and prefixing the result with `Umath`. The XƎTEX font dimension numbers must be integer constants.

```
\cs_new_protected_nopar:Nn \@@_font_param:nnn
```

```
  {
    \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #3 } { #3 }
  }
```

\@@_font_param:nn

#1 : name
#2 : font dimension

This macro defines getter and setter functions for the font parameter ⟨*name*⟩. The LuaTEX font parameter name is produced by removing all underscores and prefixing the result with `Umath`. The XƎTEX font dimension number must be an integer constant.

```
\cs_new_protected_nopar:Nn \@@_font_param:nn
  {
    \@@_font_param:nnnnn { #1 } { #2 } { #2 } { #2 } { #2 }
  }
```

\@@_font_param:n

#1 : name

This macro defines getter and setter functions for the font parameter ⟨*name*⟩, which is considered unavailable in XƎTEX. The LuaTEX font parameter name is produced by removing all underscores and prefixing the result with `Umath`.

```
\cs_new_protected_nopar:Nn \@@_font_param:n
⟨XE⟩  { }
⟨LU⟩  { \@@_font_param:nnnnn { #1 } { 0 } { 0 } { 0 } { 0 } }
```

\@@_font_param_aux:NNnnnn
\@@_font_param_aux:NNN

Auxiliary macros for generating font parameter accessor macros.

```
⟨*XE⟩
\cs_new_protected_nopar:Nn \@@_font_param_aux:NNnnnn
  {
    \cs_new_nopar:Npn #1 ##1
      {
        \@@_font_dimen:Nnnnn ##1 { #3 } { #4 } { #5 } { #6 }
      }
    \cs_new_protected_nopar:Npn #2 ##1 ##2
      {
        #1 ##1 \dim_eval:n { ##2 }
      }
  }
\cs_generate_variant:Nn \@@_font_param_aux:NNnnnn { cc }
⟨/XE⟩
⟨*LU⟩
\cs_new_protected_nopar:Nn \@@_font_param_aux:NNN
  {
    \cs_new_nopar:Npn #1 ##1
      {
        #3 ##1
      }
    \cs_new_protected_nopar:Npn #2 ##1 ##2
      {
        #3 ##1 \dim_eval:n { ##2 }
      }
  }
\cs_generate_variant:Nn \@@_font_param_aux:NNN { ccc }
⟨/LU⟩
```

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

```
\@@_font_param:nn { axis } { 15 }
\@@_font_param:nn { operator_size } { 13 }
\@@_font_param:n { fraction_del_size }
\@@_font_param:nnn { fraction_denom_down } { 45 } { 44 }
\@@_font_param:nnn { fraction_denom_vgap } { 50 } { 49 }
\@@_font_param:nnn { fraction_num_up } { 43 } { 42 }
\@@_font_param:nnn { fraction_num_vgap } { 47 } { 46 }
\@@_font_param:nn { fraction_rule } { 48 }
\@@_font_param:nn { limit_above_bgap } { 29 }
\@@_font_param:n { limit_above_kern }
\@@_font_param:nn { limit_above_vgap } { 28 }
\@@_font_param:nn { limit_below_bgap } { 31 }
\@@_font_param:n { limit_below_kern }
\@@_font_param:nn { limit_below_vgap } { 30 }
\@@_font_param:nn { over_delimiter_vgap } { 41 }
\@@_font_param:nn { over_delimiter_bgap } { 38 }
\@@_font_param:nn { under_delimiter_vgap } { 40 }
\@@_font_param:nn { under_delimiter_bgap } { 39 }
\@@_font_param:nn { overbar_kern } { 55 }
\@@_font_param:nn { overbar_rule } { 54 }
\@@_font_param:nn { overbar_vgap } { 53 }
\@@_font_param:n { quad }
\@@_font_param:nn { radical_kern } { 62 }
\@@_font_param:nn { radical_rule } { 61 }
\@@_font_param:nnn { radical_vgap } { 60 } { 59 }
\@@_font_param:nn { radical_degree_before } { 63 }
\@@_font_param:nn { radical_degree_after } { 64 }
\@@_font_param:nn { radical_degree_raise } { 65 }
\@@_font_param:nn { space_after_script } { 27 }
\@@_font_param:nnn { stack_denom_down } { 35 } { 34 }
\@@_font_param:nnn { stack_num_up } { 33 } { 32 }
\@@_font_param:nnn { stack_vgap } { 37 } { 36 }
\@@_font_param:nn { sub_shift_down } { 18 }
\@@_font_param:nn { sub_shift_drop } { 20 }
\@@_font_param:n { subsup_shift_down }
\@@_font_param:nn { sub_top_max } { 19 }
\@@_font_param:nn { subsup_vgap } { 25 }
\@@_font_param:nn { sup_bottom_min } { 23 }
\@@_font_param:nn { sup_shift_drop } { 24 }
\@@_font_param:nnnnn { sup_shift_up } { 21 } { 22 } { 21 } { 22 }
\@@_font_param:nn { supsub_bottom_max } { 26 }
\@@_font_param:nn { underbar_kern } { 58 }
\@@_font_param:nn { underbar_rule } { 57 }
```

```
\@@_font_param:nn { underbar_vgap } { 56 }
\@@_font_param:n { connector_overlap_min }
```

14.1 *Historical commands*

`\@@_fontdimen_to_percent:nN`
`\@@_fontdimen_to_scale:nN`

#1 : Font dimen number
#2 : Font 'variable'

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

```
\cs_new:Nn \@@_fontdimen_to_percent:nN
  {
    \fp_eval:n { \dim_to_decimal:n { \fontdimen #1 #2 } * 65536 / 100 }
  }
\cs_new:Nn \@@_fontdimen_to_scale:nN
  {
    \fp_eval:n { \@@_fontdimen_to_percent:nN {#1} #2 * \f@size } pt
  }
```

`\@@_mathstyle_scale:NnnN`

#1 : A math style (`\scriptstyle`, say)
#2 : Macro that takes a non-delimited length argument (like `\kern`)
#3 : Length control sequence to be scaled according to the math style
#4 : Math font face to use for the lookups

This macro is used to scale the lengths reported by `\fontdimen` according to the scale factor for script- and scriptscript-size objects.

```
\cs_new:Nn \@@_mathstyle_scale:NnnN
  {
    \ifx#1\scriptstyle
      #2 \@@_fontdimen_to_percent:nN {10} #4 #3
    \else
      \ifx#1\scriptscriptstyle
        #2 \@@_fontdimen_to_percent:nN {11} #4 #3
      \else
        #2 #3
      \fi
    \fi
  }
```

```
⟨/package⟩
```

File XIV

um-code-mathmap.dtx

15 *Defining the math alphabets per style*

```
⟨*package⟩
```

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

IMPLICIT No ranges specified, set up everything

EXPLICIT Some ranges specified, set up what is requested only

INHERIT Of the slots in the ranges specified, compare against slots in each styled alphabet and only set up those needed

The INHERIT mode saves less time than I was hoping for but is still beneficial in simple cases.

```
\cs_new:Npn \@@_setup_alphabets:
  {
    \bool_if:NTF \l_@@_init_bool { \@@_setup_alphabets_implicit: }
      {
        \seq_if_empty:NF \g_@@_mathalph_seq { \@@_setup_alphabets_explicit: }
     \clist_if_empty:NF \l_@@_mathmap_charints_clist { \@@_setup_alphabets_inherit: }
      }
  }
```

\@@_setup_alphabets_implicit:

```
\cs_new:Nn \@@_setup_alphabets_implicit:
  {
    \@@_log:n {setup-implicit}
    \seq_gset_eq:NN \g_@@_mathalph_seq \g_@@_default_mathalph_seq
    \bool_set_true:N \l_@@_implicit_alph_bool
    \@@_maybe_init_alphabet:n  {sf}
    \@@_maybe_init_alphabet:n  {bf}
    \@@_maybe_init_alphabet:n  {bfsf}
    \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
    \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
    \@@_mathalph_map:
   \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
  }
```

\@@_setup_alphabets_explicit:

```
\cs_new:Nn \@@_setup_alphabets_explicit:
  {
    \@@_log:n {setup-explicit}
    \bool_set_false:N \l_@@_implicit_alph_bool
    \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_noparse:nnn
    \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_noparse:nn
    \@@_mathalph_map:
    \seq_if_empty:NF \l_@@_missing_alph_seq { \@@_log:n { missing-alphabets } }
  }
```

\@@_setup_alphabets_inherit:

```
\cs_new:Nn \@@_setup_alphabets_inherit:
  {
    \seq_gclear:N \g_@@_mathalph_seq
    \seq_map_inline:Nn \g_@@_default_mathalph_seq
      {
        \tl_set:No    \l_@@_style_tl       { \use_i:nnn   ##1 }
        \clist_set:No \l_@@_alphabet_clist { \use_ii:nnn  ##1 }

        \clist_map_inline:Nn \l_@@_alphabet_clist
          {
          \clist_if_exist:cT {g_@@_named_slots_ \l_@@_style_tl _ ####1 _clist}
              {
           \clist_map_inline:cn {g_@@_named_slots_ \l_@@_style_tl _ ####1 _clist}
                  {
                    \clist_map_inline:Nn \l_@@_mathmap_charints_clist
                      {
                  \@@_int_if_slot_in_range:nnT {################1} {########1}
                          {
                            \seq_gput_right:Nn \g_@@_mathalph_seq {##1}
                   \clist_map_break:n { \clist_map_break:n { \clist_map_break: } }
                          }
                      }
                  }
              }
          }
      }

    \cs_set_eq:NN \@@_set_mathalphabet_char:nnn \@@_mathmap_parse:nnn
    \cs_set_eq:NN \@@_map_char_single:nn \@@_map_char_parse:nn
    \@@_mathalph_map:
  }
```

\@@_mathalph_map:

```
\cs_set:Nn \@@_mathalph_map:
  {
   \seq_map_inline:Nn \g_@@_mathalph_seq
      {
        \tl_set:No    \l_@@_style_tl       { \use_i:nnn   ##1 }
        \clist_set:No \l_@@_alphabet_clist { \use_ii:nnn  ##1 }
        \tl_set:No    \l_@@_remap_style_tl { \use_iii:nnn ##1 }

        % If no set of alphabets is defined:
```

```
      \clist_if_empty:NT \l_@@_alphabet_clist
        {
          \cs_set_eq:NN \@@_maybe_init_alphabet:n \@@_init_alphabet:n
          \prop_get:cnN { g_@@_named_range_ \l_@@_style_tl _prop }
            { default-alpha } \l_@@_alphabet_clist
        }

      \@@_check_math_alphabet:
      \@@_setup_math_alphabet:
    }
  }
```

\@@_check_math_alphabet: First check that at least one of the alphabets for the font shape is defined (this process is fast) …

```
\cs_new:Nn \@@_check_math_alphabet:
  {
    \clist_map_inline:Nn \l_@@_alphabet_clist
      {
        \tl_set:Nn \l_@@_alphabet_tl {##1}
        \@@_if_alphabet_exists:nnTF \l_@@_style_tl \l_@@_alphabet_tl
          {
            \str_if_eq_x:nnTF {\l_@@_alphabet_tl} {misc}
              {
                \@@_maybe_init_alphabet:n \l_@@_style_tl
                \clist_map_break:
              }
              {
                \@@_glyph_if_exist:NnT \g_@@_curr_font_cmd_tl
                  { \@@_to_usv:nn {\l_@@_style_tl} {\l_@@_alphabet_tl} }
                  {
                    \@@_maybe_init_alphabet:n \l_@@_style_tl
                    \clist_map_break:
                  }
              }
          }
          {
            \msg_warning:nnx {unicode-math} {no-alphabet}
              { \l_@@_style_tl / \l_@@_alphabet_tl }
          }
      }
  }
```

\@@_setup_math_alphabet: …and then loop through them defining the individual ranges: (currently this process is slow)

```
\cs_new:Nn \@@_setup_math_alphabet:
  {
    \clist_map_inline:Nn \l_@@_alphabet_clist
      {
        \tl_set:Nx \l_@@_alphabet_tl { \tl_trim_spaces:n {##1} }
```

```

⟨debug⟩\typeout{_setup_math_alphabet:~\l_@@_style_tl/\l_@@_alphabet_tl}

        \@@_if_alphabet_exists:nnT {\l_@@_style_tl} {\l_@@_alphabet_tl}
          {
            \exp_args:No \tl_if_eq:nnTF \l_@@_alphabet_tl {misc}
              {
              \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
          \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
              }
              {
          \@@_glyph_if_exist:NnTF \g_@@_curr_font_cmd_tl { \@@_to_usv:nn {\l_@@_remap_style_tl} {\l_@
                  {
              \@@_log:nx {setup-alph} {sym \l_@@_style_tl~(\l_@@_alphabet_tl)}
             \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {\l_@@_remap_style_tl}
                  }
                  {
                    \bool_if:NTF \l_@@_implicit_alph_bool
                      {
                        \seq_put_right:Nx \l_@@_missing_alph_seq
                          {
                            \@backslashchar sym \l_@@_style_tl \space
                   (\tl_use:c{c_@@_math_alphabet_name_ \l_@@_alphabet_tl _tl})
                          }
                      }
                      {
                \@@_alphabet_config:nnn {\l_@@_style_tl} {\l_@@_alphabet_tl} {up}
                      }
                  }
              }
          }
      }
  }
```

Each alphabet style needs to be configured. This happens in Section 17.

```
\cs_new:Nn \@@_new_alphabet_config:nnn
  {
    \prop_if_exist:cF {g_@@_named_range_#1_prop}
      { \@@_warning:nnn {no-named-range} {#1} {#2} }


    \prop_gput:cnn {g_@@_named_range_#1_prop} { alpha_tl }
      {
        \prop_item:cn {g_@@_named_range_#1_prop} { alpha_tl } {#2}
      }
    % Q: do I need to bother removing duplicates?

```

Create list of all chars defined in this named range:

```
    \cs_new:cn { @@_config_#1_#2:n }
    {
      \clist_gclear_new:c {g_@@_named_slots_#1_#2_clist}
      \tl_set:Nn \l_@@_curr_named_slot { g_@@_named_slots_#1_#2_clist }
      #3
      \clist_gremove_duplicates:c {g_@@_named_slots_#1_#2_clist}
    }

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

15.1 *Mapping 'naked' math characters*

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

15.1.1 *Functions*

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

`\@@_map_char_noparse:nn`
`\@@_map_char_parse:nn`

```
\cs_new:Nn \@@_map_char_noparse:nn
  {
    \@@_set_mathcode:nnnn {#1} {\mathalpha} {\l_@@_symfont_label_tl} {#2}
  }
\cs_new:Nn \@@_map_char_parse:nn
  {
    \@@_if_char_spec:nNT {#1} {\mathalpha}
      { \@@_map_char_noparse:nn {#1}{#2} }
  }
```

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

```
\cs_new:Nn \@@_map_char_single:nnn
  {
    \@@_map_char_single:nn { \@@_to_usv:nn {#1} {#3} }
                           { \@@_to_usv:nn {#2} {#3} }
  }
```

\@@_map_chars_range:nnnn

#1 : Number of chars (26)
#2 : From style, one or more (it)
#3 : To style (up)
#4 : Alphabet name (Latin)

First the function with numbers:

```
\cs_set:Nn \@@_map_chars_range:nnn
  {
    \int_step_inline:nnnn {0} {1} {#1-1}
      { \@@_map_char_single:nn {#2+##1} {#3+##1} }

    \clist_gput_right:cx { \l_@@_curr_named_slot }
      { \int_eval:n { #3 } - \int_eval:n { #3 + #1-1 } }
  }
```

And the wrapper with names:

```
\cs_new:Nn \@@_map_chars_range:nnnn
  {
    \@@_map_chars_range:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
                                 { \@@_to_usv:nn {#3} {#4} }
  }
```

15.1.2 *Functions for 'normal' alphabet symbols*

\@@_set_normal_char:nnn

```
\cs_set:Nn \@@_set_normal_char:nnn
  {
    \@@_usv_if_exist:nnT {#3} {#1}
      {
        \clist_map_inline:nn {#2}
          {
            \@@_set_mathalphabet_pos:nnnn {normal} {#1} {##1} {#3}
            \@@_map_char_single:nnn {##1} {#3} {#1}

            \clist_gput_right:cx {\l_@@_curr_named_slot}
              { \int_eval:n { \@@_to_usv:nn {#3} {#1} } }
          }
      }
  }

\cs_new:Nn \@@_set_normal_Latin:nn
  {
    \clist_map_inline:nn {#1}
      {
        \@@_set_mathalphabet_Latin:nnn {normal} {##1} {#2}
        \@@_map_chars_range:nnnn {26} {##1} {#2} {Latin}
      }
  }
```

```
\cs_new:Nn \@@_set_normal_latin:nn
  {
    \clist_map_inline:nn {#1}
      {
        \@@_set_mathalphabet_latin:nnn {normal} {##1} {#2}
        \@@_map_chars_range:nnnn {26} {##1} {#2} {latin}
      }
  }
\cs_new:Nn \@@_set_normal_greek:nn
  {
    \clist_map_inline:nn {#1}
      {
        \@@_set_mathalphabet_greek:nnn {normal} {##1} {#2}
        \@@_map_chars_range:nnnn {25} {##1} {#2} {greek}
        \@@_map_char_single:nnn {##1} {#2} {epsilon}
        \@@_map_char_single:nnn {##1} {#2} {vartheta}
        \@@_map_char_single:nnn {##1} {#2} {varkappa}
        \@@_map_char_single:nnn {##1} {#2} {phi}
        \@@_map_char_single:nnn {##1} {#2} {varrho}
        \@@_map_char_single:nnn {##1} {#2} {varpi}
        \@@_set_mathalphabet_pos:nnnn {normal} {epsilon} {##1} {#2}
        \@@_set_mathalphabet_pos:nnnn {normal} {vartheta} {##1} {#2}
        \@@_set_mathalphabet_pos:nnnn {normal} {varkappa} {##1} {#2}
        \@@_set_mathalphabet_pos:nnnn {normal} {phi} {##1} {#2}
        \@@_set_mathalphabet_pos:nnnn {normal} {varrho} {##1} {#2}
        \@@_set_mathalphabet_pos:nnnn {normal} {varpi} {##1} {#2}
      }
  }
\cs_new:Nn \@@_set_normal_Greek:nn
  {
    \clist_map_inline:nn {#1}
      {
        \@@_set_mathalphabet_Greek:nnn {normal} {##1} {#2}
        \@@_map_chars_range:nnnn {25} {##1} {#2} {Greek}
        \@@_map_char_single:nnn {##1} {#2} {varTheta}
        \@@_set_mathalphabet_pos:nnnn {normal} {varTheta} {##1} {#2}
      }
  }
\cs_new:Nn \@@_set_normal_numbers:nn
  {
    \@@_set_mathalphabet_numbers:nnn {normal} {#1} {#2}
    \@@_map_chars_range:nnnn {10} {#1} {#2} {num}
  }
```

15.2 *Mapping chars inside a math style*

15.2.1 *Functions for setting up the maths alphabets*

\@@_set_mathalphabet_char:nnn

#1 : Maths alphabet, *e.g.*, 'bb'

`#2` : Input slot, *e.g.*, the slot for 'A' (comma separated)
`#3` : Output slot, *e.g.*, the slot for '𝔸'
This is a wrapper for either `\@@_mathmap_noparse:nnn` or `\@@_mathmap_parse:nnn`, depending on the context.

`\@@_mathmap_noparse:nnn`

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

```
\cs_new:Nn \@@_mathmap_noparse:nnn
  {
    \tl_put_right:cx { @@_switchto_#1: }
      {
        \@@_set_mathcode:nnnn {#2} {\mathalpha} {\l_@@_symfont_label_tl} {#3}
      }
  }
```

`\@@_mathmap_parse:nnn`

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

```
\cs_new:Nn \@@_mathmap_parse:nnn
  {
   \exp_args:NNx \clist_if_in:NnT \l_@@_mathmap_charints_clist { \int_eval:n {#3} }
      {
        \@@_mathmap_noparse:nnn {#1} {#2} {#3}
      }
  }
```

`\@@_set_mathalphabet_char:nnnn`

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

```
\cs_new:Nn \@@_set_mathalphabet_char:nnnn
  {
    \@@_set_mathalphabet_char:nnn {#1} { \@@_to_usv:nn {#2} {#4} }
                                       { \@@_to_usv:nn {#3} {#4} }
  }
```

`\@@_set_mathalph_range:nnnn`

`#1` : Number of iterations
`#2` : Sym command suffix
`#3` : Starting input char
`#4` : Starting output char

Loops through character ranges setting `\mathcode`. First the version that uses numbers:

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

`\@@_set_mathalph_range:nnnn`

#1 : Number of iterations
#2 : Sym command suffix
#3 : input style
#4 : output style
#5 : alphabet

Then the wrapper version that uses names:

```
\cs_new:Nn \@@_set_mathalph_range:nnnnn
  {
    \clist_gput_right:cx { \l_@@_curr_named_slot }
         { \int_eval:n { \@@_to_usv:nn {#4} {#5} } - \int_eval:n { (#1-
1)+\@@_to_usv:nn {#4} {#5} } }

    \@@_set_mathalph_range:nnnn {#1} {#2} { \@@_to_usv:nn {#3} {#5} }
                                            { \@@_to_usv:nn {#4} {#5} }
  }
```

15.2.2 *Individual mapping functions for different alphabets*

```
\cs_new:Nn \@@_set_mathalphabet_pos:nnnn
  {
    \@@_usv_if_exist:nnT {#4} {#2}
      {
        \clist_map_inline:nn {#3}
          { \@@_set_mathalphabet_char:nnnn {#1} {##1} {#4} {#2} }

        \clist_gput_right:cx {\l_@@_curr_named_slot}
          { \int_eval:n { \@@_to_usv:nn {#4} {#2} } }
      }
  }
```

```
\cs_new:Nn \@@_set_mathalphabet_numbers:nnn
  {
    \clist_map_inline:nn {#2}
      { \@@_set_mathalph_range:nnnnn {10} {#1} {##1} {#3} {num} }
  }
```

```
\cs_new:Nn \@@_set_mathalphabet_Latin:nnn
  {
    \clist_map_inline:nn {#2}
      { \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {Latin} }
  }
```

```
\cs_new:Nn \@@_set_mathalphabet_latin:nnn
  {
    \clist_map_inline:nn {#2}
      {
        \@@_set_mathalph_range:nnnnn {26} {#1} {##1} {#3} {latin}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {h}
      }
  }
\cs_new:Nn \@@_set_mathalphabet_Greek:nnn
  {
    \clist_map_inline:nn {#2}
      {
        \@@_set_mathalph_range:nnnnn {25} {#1} {##1} {#3} {Greek}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {varTheta}
      }
  }
\cs_new:Nn \@@_set_mathalphabet_greek:nnn
  {
    \clist_map_inline:nn {#2}
      {
        \@@_set_mathalph_range:nnnnn {25} {#1} {##1} {#3} {greek}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {epsilon}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {vartheta}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {varkappa}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {phi}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {varrho}
        \@@_set_mathalphabet_char:nnnn    {#1} {##1} {#3} {varpi}
      }
  }
⟨/package⟩
```

File XV

um-code-sym-commands.dtx

16 *Mapping in maths alphabets*

```
⟨*package⟩
```

16.1 *Setting styles*

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

Implicit mode:

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

Explicit mode:

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

16.2 *Defining the math style macros*

We call the different shapes that a math alphabet can be a 'math style'. Note that different alphabets can exist within the same math style. E.g., we call 'bold' the math style `bf` and within it there are upper and lower case Greek and Roman alphabets and Arabic numerals.

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

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

```
\cs_new:Nn \@@_prepare_mathstyle:n
  {
    \seq_gput_right:Nn \g_@@_mathstyles_seq {#1}
    \@@_init_alphabet:n {#1}
    \cs_set_protected:cpx {sym#1}
      {
        \@@_group_begin:
          \exp_not:n
            {
              \mode_if_math:F { \exp_args:Nc \non@alpherr {sym#1} }
              \tl_set:Nn \l_@@_mathstyle_tl {#1}
            }
          \exp_not:c {@@_switchto_#1:}
          \@@_mathgroup_set:n {-1}
        \@@_group_end:n
      }
  }
```

`\@@_init_alphabet:n` #1 : math alphabet name (e.g., `it` or bb)

This macro initialises the macros used to set up a math alphabet. First used when the math alphabet macro is first defined, but then used later when redefining a particular maths alphabet.

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

16.3 *Definition of alphabets and styles*

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

```
\cs_new:Nn \@@_default_mathalph:nnn
  {
    \prop_new:c {g_@@_named_range_#1_prop}
    \seq_gput_right:Nn \g_@@_default_mathalph_seq {{#1}{#2}{#3}}
    \prop_gput:cnn { g_@@_named_range_#1_prop } { default-alpha } {#2}
  }
```

```
\@@_default_mathalph:nnn {up    } {latin,Latin,greek,Greek,num,misc} {up    }
\@@_default_mathalph:nnn {it    } {latin,Latin,greek,Greek,misc}     {it    }
\@@_default_mathalph:nnn {bb    } {latin,Latin,num,misc}             {bb    }
\@@_default_mathalph:nnn {bbit  } {misc}                             {bbit  }
\@@_default_mathalph:nnn {scr   } {latin,Latin}                      {scr   }
\@@_default_mathalph:nnn {cal   } {Latin}                            {scr   }
\@@_default_mathalph:nnn {bfcal } {Latin}                            {bfscr }
\@@_default_mathalph:nnn {frak  } {latin,Latin}                      {frak  }
\@@_default_mathalph:nnn {tt    } {latin,Latin,num}                  {tt    }
\@@_default_mathalph:nnn {sfup  } {latin,Latin,num}                  {sfup  }
\@@_default_mathalph:nnn {sfit  } {latin,Latin}                      {sfit  }
```

```
\@@_default_mathalph:nnn {bfup  } {latin,Latin,greek,Greek,num,misc} {bfup  }
\@@_default_mathalph:nnn {bfit  } {latin,Latin,greek,Greek,misc}     {bfit  }
\@@_default_mathalph:nnn {bfscr } {latin,Latin}                      {bfscr }
\@@_default_mathalph:nnn {bffrak} {latin,Latin}                      {bffrak}
\@@_default_mathalph:nnn {bfsfup} {latin,Latin,greek,Greek,num,misc} {bfsfup}
\@@_default_mathalph:nnn {bfsfit} {latin,Latin,greek,Greek,misc}     {bfsfit}
```

16.3.1 *Define symbol style commands*

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

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

16.3.2 *New names for legacy textmath alphabet selection*

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

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

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

16.3.3 *Replacing legacy pure-maths alphabets*

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

```
\clist_map_inline:nn
  {
    normal, bb , bbit, scr, bfscr, cal, bfcal, frak, bffrak, tt,
    bfup, bfit, sfup, sfit, bfsfup, bfsfit, bfsf
  }
  {
    \cs_set:cpx { math #1 } { \exp_not:c { sym #1 } }
  }
```

16.3.4 *New commands for ambiguous alphabets*

```
\AtBeginDocument
  {
    \clist_map_inline:nn
      { rm, it, bf, sf, tt }
      {
        \cs_set_protected:cpx { math #1 }
        {
          \exp_not:n { \bool_if:NTF  } \exp_not:c { g_@@_ math #1 _text_bool}
            { \exp_not:c { mathtext #1 } }
            { \exp_not:c { sym #1 } }
        }
      }
  }
```

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

```
\cs_set_protected:Npn \mathup { \mathrm }
\cs_set_protected:Npn \symrm  { \symup  }
```

```
⟨/package⟩
```

File XVI

um-code-alphabets.dtx

17 *Setting up alphabets*

```
⟨*package⟩
```

17.1 *Upright: up*

```
\@@_new_alphabet_config:nnn {up} {num}
  {
    \@@_set_normal_numbers:nn {up} {#1}
    \@@_set_mathalphabet_numbers:nnn {up} {up} {#1}
  }

\@@_new_alphabet_config:nnn {up} {Latin}
  {
    \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {up} {#1} }
      {
       \bool_if:NT \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
      }
    \@@_set_mathalphabet_Latin:nnn {up} {up,it} {#1}
    \@@_set_mathalphabet_Latin:nnn {literal} {up} {up}
    \@@_set_mathalphabet_Latin:nnn {literal} {it} {it}
  }

\@@_new_alphabet_config:nnn {up} {latin}
  {
    \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_latin:nn {up} {#1} }
      {
        \bool_if:NT \g_@@_uplatin_bool
          {
            \@@_set_normal_latin:nn       {up,it} {#1}
            \@@_set_normal_char:nnn       {h} {up,it} {#1}
            \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
            \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_latin:nnn {up} {up,it}{#1}
    \@@_set_mathalphabet_latin:nnn {literal} {up} {up}
    \@@_set_mathalphabet_latin:nnn {literal} {it} {it}
  }

\@@_new_alphabet_config:nnn {up} {Greek}
  {
    \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Greek:nn {up}{#1} }
      {
        \bool_if:NT \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it}{#1} }
      }
    \@@_set_mathalphabet_Greek:nnn {up} {up,it}{#1}
    \@@_set_mathalphabet_Greek:nnn {literal} {up} {up}
    \@@_set_mathalphabet_Greek:nnn {literal} {it} {it}
  }

\@@_new_alphabet_config:nnn {up} {greek}
  {
    \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_greek:nn {up} {#1} }
      {
        \bool_if:NT \g_@@_upgreek_bool
          {
            \@@_set_normal_greek:nn {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_greek:nnn {up} {up,it} {#1}
    \@@_set_mathalphabet_greek:nnn {literal} {up} {up}
    \@@_set_mathalphabet_greek:nnn {literal} {it} {it}
  }

\@@_new_alphabet_config:nnn {up} {misc}
  {
    \bool_if:NTF \g_@@_literal_Nabla_bool
      {
        \@@_set_normal_char:nnn {Nabla}{up}{up}
      }
      {
        \bool_if:NT \g_@@_upNabla_bool
          {
            \@@_set_normal_char:nnn {Nabla}{up,it}{up}
          }
      }
    \bool_if:NTF \g_@@_literal_partial_bool
      {
        \@@_set_normal_char:nnn {partial}{up}{up}
      }
      {
        \bool_if:NT \g_@@_uppartial_bool
          {
            \@@_set_normal_char:nnn {partial}{up,it}{up}
          }
      }
    \@@_set_mathalphabet_pos:nnnn {up}  {partial} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {up}    {Nabla} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {up} {dotlessi} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {up} {dotlessj} {up,it} {#1}
  }
```

17.2 Italic: it

```
\@@_new_alphabet_config:nnn {it} {Latin}
  {
    \bool_if:NTF \g_@@_literal_bool { \@@_set_normal_Latin:nn {it} {#1} }
      {
       \bool_if:NF \g_@@_upLatin_bool { \@@_set_normal_Latin:nn {up,it} {#1} }
      }
    \@@_set_mathalphabet_Latin:nnn {it} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {it} {latin}
  {
    \bool_if:NTF \g_@@_literal_bool
      {
        \@@_set_normal_latin:nn    {it}{#1}
        \@@_set_normal_char:nnn {h}{it}{#1}
      }
      {
        \bool_if:NF \g_@@_uplatin_bool
          {
            \@@_set_normal_latin:nn            {up,it} {#1}
            \@@_set_normal_char:nnn {h}        {up,it} {#1}
            \@@_set_normal_char:nnn {dotlessi} {up,it} {#1}
            \@@_set_normal_char:nnn {dotlessj} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_latin:nnn {it}            {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {it} {dotlessi} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {it} {dotlessj} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {it} {Greek}
  {
    \bool_if:NTF \g_@@_literal_bool
      {
        \@@_set_normal_Greek:nn {it} {#1}
      }
      {
       \bool_if:NF \g_@@_upGreek_bool { \@@_set_normal_Greek:nn {up,it} {#1} }
      }
    \@@_set_mathalphabet_Greek:nnn {it} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {it} {greek}
  {
    \bool_if:NTF \g_@@_literal_bool
      {
        \@@_set_normal_greek:nn {it} {#1}
      }
      {
```

```
      \bool_if:NF \g_@@_upgreek_bool { \@@_set_normal_greek:nn {it,up} {#1} }
    }
  \@@_set_mathalphabet_greek:nnn {it} {up,it} {#1}
}

\@@_new_alphabet_config:nnn {it} {misc}
{
  \bool_if:NTF \g_@@_literal_Nabla_bool
    {
      \@@_set_normal_char:nnn {Nabla} {it} {it}
    }
    {
      \bool_if:NF \g_@@_upNabla_bool
        {
          \@@_set_normal_char:nnn {Nabla} {up,it} {it}
        }
    }
  \bool_if:NTF \g_@@_literal_partial_bool
    {
      \@@_set_normal_char:nnn {partial} {it} {it}
    }
    {
      \bool_if:NF \g_@@_uppartial_bool
        {
          \@@_set_normal_char:nnn {partial} {up,it} {it}
        }
    }
  \@@_set_mathalphabet_pos:nnnn {it} {partial} {up,it}{#1}
  \@@_set_mathalphabet_pos:nnnn {it} {Nabla}   {up,it}{#1}
}
```

17.3 *Blackboard or double-struck: bb and bbit*

```
\@@_new_alphabet_config:nnn {bb} {latin}
{
  \@@_set_mathalphabet_latin:nnn {bb} {up,it} {#1}
}

\@@_new_alphabet_config:nnn {bb} {Latin}
{
  \@@_set_mathalphabet_Latin:nnn {bb}    {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {C} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {H} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {N} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {P} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {Q} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {R} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bb} {Z} {up,it} {#1}
}

```

```
\@@_new_alphabet_config:nnn {bb} {num}
  {
    \@@_set_mathalphabet_numbers:nnn {bb} {up} {#1}
  }

\@@_new_alphabet_config:nnn {bb} {misc}
  {
    \@@_set_mathalphabet_pos:nnnn {bb}        {Pi} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bb}        {pi} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bb}     {Gamma} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bb}     {gamma} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bb} {summation} {up}    {#1}
  }

\@@_new_alphabet_config:nnn {bbit} {misc}
  {
    \@@_set_mathalphabet_pos:nnnn {bbit} {D} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bbit} {d} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bbit} {e} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bbit} {i} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bbit} {j} {up,it} {#1}
  }
```

17.4 *Script and caligraphic: scr and cal*

```
\@@_new_alphabet_config:nnn {scr} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {scr}     {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {B} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {E} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {F} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {H} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {I} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {L} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {M} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {R} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {scr} {latin}
  {
    \@@_set_mathalphabet_latin:nnn {scr}     {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {e} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {g} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {scr} {o} {up,it} {#1}
  }
```

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

```
\@@_new_alphabet_config:nnn {cal} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {cal}     {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {B} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {E} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {F} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {H} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {I} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {L} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {M} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {cal} {R} {up,it} {#1}
  }
```

17.5 *Fractur or fraktur or blackletter: frak*

```
\@@_new_alphabet_config:nnn {frak} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {frak}     {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {frak} {C} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {frak} {H} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {frak} {I} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {frak} {R} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn  {frak} {Z} {up,it} {#1}
  }
\@@_new_alphabet_config:nnn {frak} {latin}
  {
    \@@_set_mathalphabet_latin:nnn {frak} {up,it} {#1}
  }
```

17.6 *Sans serif upright: sfup*

```
\@@_new_alphabet_config:nnn {sfup} {num}
  {
    \@@_set_mathalphabet_numbers:nnn {sf}   {up} {#1}
    \@@_set_mathalphabet_numbers:nnn {sfup} {up} {#1}
  }
\@@_new_alphabet_config:nnn {sfup} {Latin}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_Latin:nn {sfup} {#1}
        \@@_set_mathalphabet_Latin:nnn {sf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_Latin:nn {sfup,sfit} {#1}
            \@@_set_mathalphabet_Latin:nnn {sf} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_Latin:nnn {sfup} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {sfup} {latin}
```

```
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_latin:nn {sfup} {#1}
        \@@_set_mathalphabet_latin:nnn {sf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_latin:nn {sfup,sfit} {#1}
            \@@_set_mathalphabet_latin:nnn {sf} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_latin:nnn {sfup} {up,it} {#1}
  }
```

17.7 *Sans serif italic: sfit*

```
\@@_new_alphabet_config:nnn {sfit} {Latin}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_Latin:nn {sfit} {#1}
        \@@_set_mathalphabet_Latin:nnn {sf} {it} {#1}
      }
      {
        \bool_if:NF \g_@@_upsans_bool
          {
            \@@_set_normal_Latin:nn {sfup,sfit} {#1}
            \@@_set_mathalphabet_Latin:nnn {sf} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_Latin:nnn {sfit} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {sfit} {latin}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_latin:nn {sfit} {#1}
        \@@_set_mathalphabet_latin:nnn {sf} {it}{#1}
      }
      {
        \bool_if:NF \g_@@_upsans_bool
          {
            \@@_set_normal_latin:nn {sfup,sfit} {#1}
            \@@_set_mathalphabet_latin:nnn {sf} {up,it}{#1}
          }
      }
    \@@_set_mathalphabet_latin:nnn {sfit} {up,it}{#1}
  }
```

17.8 *Typewriter or monospaced: tt*

```
\@@_new_alphabet_config:nnn {tt} {num}
  {
    \@@_set_mathalphabet_numbers:nnn {tt} {up}{#1}
  }
\@@_new_alphabet_config:nnn {tt} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {tt} {up,it}{#1}
  }
\@@_new_alphabet_config:nnn {tt} {latin}
  {
    \@@_set_mathalphabet_latin:nnn {tt} {up,it}{#1}
  }
```

17.9 *Bold Italic: bfit*

```
\@@_new_alphabet_config:nnn {bfit} {Latin}
  {
    \bool_if:NF \g_@@_bfupLatin_bool
      {
        \@@_set_normal_Latin:nn {bfup,bfit} {#1}
      }
    \@@_set_mathalphabet_Latin:nnn {bfit} {up,it}{#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_Latin:nn {bfit} {#1}
        \@@_set_mathalphabet_Latin:nnn {bf} {it}{#1}
      }
      {
        \bool_if:NF \g_@@_bfupLatin_bool
          {
            \@@_set_normal_Latin:nn {bfup,bfit} {#1}
            \@@_set_mathalphabet_Latin:nnn {bf} {up,it}{#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfit} {latin}
  {
    \bool_if:NF \g_@@_bfuplatin_bool
      {
        \@@_set_normal_latin:nn {bfup,bfit} {#1}
      }
    \@@_set_mathalphabet_latin:nnn {bfit} {up,it}{#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_latin:nn {bfit} {#1}
        \@@_set_mathalphabet_latin:nnn {bf} {it}{#1}
      }
      {
        \bool_if:NF \g_@@_bfuplatin_bool
          {
            \@@_set_normal_latin:nn {bfup,bfit} {#1}
            \@@_set_mathalphabet_latin:nnn {bf} {up,it}{#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfit} {Greek}
  {
    \@@_set_mathalphabet_Greek:nnn {bfit} {up,it}{#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_Greek:nn {bfit}{#1}
        \@@_set_mathalphabet_Greek:nnn {bf} {it}{#1}
      }
      {
        \bool_if:NF \g_@@_bfupGreek_bool
          {
            \@@_set_normal_Greek:nn {bfup,bfit}{#1}
            \@@_set_mathalphabet_Greek:nnn {bf} {up,it}{#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfit} {greek}
  {
    \@@_set_mathalphabet_greek:nnn {bfit} {up,it} {#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_greek:nn {bfit} {#1}
        \@@_set_mathalphabet_greek:nnn {bf} {it} {#1}
      }
      {
        \bool_if:NF \g_@@_bfupgreek_bool
          {
            \@@_set_normal_greek:nn {bfit,bfup} {#1}
            \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfit} {misc}
  {
    \bool_if:NTF \g_@@_literal_Nabla_bool
      { \@@_set_normal_char:nnn {Nabla} {bfit} {#1} }
      {
```

```
      \bool_if:NF \g_@@_upNabla_bool
        { \@@_set_normal_char:nnn {Nabla} {bfup,bfit} {#1} }
    }

  \bool_if:NTF \g_@@_literal_partial_bool
    { \@@_set_normal_char:nnn {partial} {bfit} {#1} }
    {
      \bool_if:NF \g_@@_uppartial_bool
        { \@@_set_normal_char:nnn {partial} {bfup,bfit} {#1} }
    }

  \@@_set_mathalphabet_pos:nnnn {bfit} {partial} {up,it} {#1}
  \@@_set_mathalphabet_pos:nnnn {bfit} {Nabla}   {up,it} {#1}

  \bool_if:NTF \g_@@_literal_partial_bool
    {
      \@@_set_mathalphabet_pos:nnnn {bf} {partial} {it}{#1}
    }
    {
      \bool_if:NF \g_@@_uppartial_bool
        {
          \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it}{#1}
        }
    }

  \bool_if:NTF \g_@@_literal_Nabla_bool
    {
      \@@_set_mathalphabet_pos:nnnn {bf} {Nabla}   {it}{#1}
    }
    {
      \bool_if:NF \g_@@_upNabla_bool
        {
          \@@_set_mathalphabet_pos:nnnn {bf} {Nabla}   {up,it}{#1}
        }
    }
}
```

17.10 *Bold Upright: bfup*

```
\@@_new_alphabet_config:nnn {bfup} {num}
  {
    \@@_set_mathalphabet_numbers:nnn {bf}   {up} {#1}
    \@@_set_mathalphabet_numbers:nnn {bfup} {up} {#1}
  }

\@@_new_alphabet_config:nnn {bfup} {Latin}
  {
    \bool_if:NT \g_@@_bfupLatin_bool
      {
        \@@_set_normal_Latin:nn {bfup,bfit} {#1}
      }
    \@@_set_mathalphabet_Latin:nnn {bfup} {up,it} {#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_Latin:nn {bfup} {#1}
        \@@_set_mathalphabet_Latin:nnn {bf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_bfupLatin_bool
          {
            \@@_set_normal_Latin:nn {bfup,bfit} {#1}
            \@@_set_mathalphabet_Latin:nnn {bf} {up,it} {#1}
          }
      }
 }

\@@_new_alphabet_config:nnn {bfup} {latin}
  {
    \bool_if:NT \g_@@_bfuplatin_bool
      {
        \@@_set_normal_latin:nn {bfup,bfit} {#1}
      }
    \@@_set_mathalphabet_latin:nnn {bfup} {up,it} {#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_latin:nn {bfup} {#1}
        \@@_set_mathalphabet_latin:nnn {bf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_bfuplatin_bool
          {
            \@@_set_normal_latin:nn {bfup,bfit} {#1}
            \@@_set_mathalphabet_latin:nnn {bf} {up,it} {#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfup} {Greek}
  {
    \@@_set_mathalphabet_Greek:nnn {bfup} {up,it} {#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_Greek:nn {bfup} {#1}
        \@@_set_mathalphabet_Greek:nnn {bf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_bfupGreek_bool
          {
            \@@_set_normal_Greek:nn {bfup,bfit} {#1}
        \@@_set_mathalphabet_Greek:nnn {bf} {up,it} {#1}
      }
    }
  }

\@@_new_alphabet_config:nnn {bfup} {greek}
  {
    \@@_set_mathalphabet_greek:nnn {bfup} {up,it} {#1}
    \bool_if:NTF \g_@@_bfliteral_bool
      {
        \@@_set_normal_greek:nn {bfup} {#1}
        \@@_set_mathalphabet_greek:nnn {bf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_bfupgreek_bool
          {
            \@@_set_normal_greek:nn {bfup,bfit} {#1}
            \@@_set_mathalphabet_greek:nnn {bf} {up,it} {#1}
          }
      }
  }

\@@_new_alphabet_config:nnn {bfup} {misc}
  {
    \bool_if:NTF \g_@@_literal_Nabla_bool
      {
        \@@_set_normal_char:nnn {Nabla} {bfup} {#1}
      }
      {
        \bool_if:NT \g_@@_upNabla_bool
          {
            \@@_set_normal_char:nnn {Nabla} {bfup,bfit} {#1}
          }
      }
    \bool_if:NTF \g_@@_literal_partial_bool
      {
        \@@_set_normal_char:nnn {partial} {bfup} {#1}
      }
      {
        \bool_if:NT \g_@@_uppartial_bool
          {
            \@@_set_normal_char:nnn {partial} {bfup,bfit} {#1}
          }
      }
    \@@_set_mathalphabet_pos:nnnn {bfup} {partial} {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bfup} {Nabla}   {up,it} {#1}
    \@@_set_mathalphabet_pos:nnnn {bfup} {digamma} {up} {#1}
    \@@_set_mathalphabet_pos:nnnn {bfup} {Digamma} {up} {#1}
    \@@_set_mathalphabet_pos:nnnn {bf}   {digamma} {up} {#1}
    \@@_set_mathalphabet_pos:nnnn {bf}   {Digamma} {up} {#1}
    \bool_if:NTF \g_@@_literal_partial_bool
      {
        \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_uppartial_bool
          {
            \@@_set_mathalphabet_pos:nnnn {bf} {partial} {up,it} {#1}
          }
      }
    \bool_if:NTF \g_@@_literal_Nabla_bool
      {
        \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up}{#1}
      }
      {
        \bool_if:NT \g_@@_upNabla_bool
          {
            \@@_set_mathalphabet_pos:nnnn {bf} {Nabla} {up,it} {#1}
          }
      }
  }
```

17.11 *Bold fractur or fraktur or blackletter: bffrak*

```
\@@_new_alphabet_config:nnn {bffrak} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {bffrak} {up,it}{#1}
  }

\@@_new_alphabet_config:nnn {bffrak} {latin}
  {
    \@@_set_mathalphabet_latin:nnn {bffrak} {up,it}{#1}
  }
```

17.12 *Bold script or calligraphic: bfscr*

```
\@@_new_alphabet_config:nnn {bfscr} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {bfscr} {up,it}{#1}
  }
\@@_new_alphabet_config:nnn {bfscr} {latin}
  {
    \@@_set_mathalphabet_latin:nnn {bfscr} {up,it}{#1}
  }
\@@_new_alphabet_config:nnn {bfcal} {Latin}
  {
    \@@_set_mathalphabet_Latin:nnn {bfcal}  {up,it}{#1}
  }
```

17.13 *Bold upright sans serif: bfsfup*

```
\@@_new_alphabet_config:nnn {bfsfup} {num}
  {
    \@@_set_mathalphabet_numbers:nnn {bfsf}   {up}{#1}
    \@@_set_mathalphabet_numbers:nnn {bfsfup} {up}{#1}
  }
\@@_new_alphabet_config:nnn {bfsfup} {Latin}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_Latin:nn {bfsfup} {#1}
        \@@_set_mathalphabet_Latin:nnn {bfsf} {up}{#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
            \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
          }
      }
    \@@_set_mathalphabet_Latin:nnn {bfsfup} {up,it}{#1}
  }

\@@_new_alphabet_config:nnn {bfsfup} {latin}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_latin:nn {bfsfup} {#1}
        \@@_set_mathalphabet_latin:nnn {bfsf} {up}{#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
            \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
          }
      }
    \@@_set_mathalphabet_latin:nnn {bfsfup} {up,it}{#1}
  }

\@@_new_alphabet_config:nnn {bfsfup} {Greek}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_Greek:nn {bfsfup}{#1}
        \@@_set_mathalphabet_Greek:nnn {bfsf} {up}{#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
            \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
          }
      }
    \@@_set_mathalphabet_Greek:nnn {bfsfup} {up,it}{#1}
  }

\@@_new_alphabet_config:nnn {bfsfup} {greek}
  {
    \bool_if:NTF \g_@@_sfliteral_bool
      {
        \@@_set_normal_greek:nn {bfsfup} {#1}
        \@@_set_mathalphabet_greek:nnn {bfsf} {up} {#1}
      }
      {
        \bool_if:NT \g_@@_upsans_bool
          {
            \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
            \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
          }
      }
    \@@_set_mathalphabet_greek:nnn {bfsfup} {up,it} {#1}
  }

\@@_new_alphabet_config:nnn {bfsfup} {misc}
 {
  \bool_if:NTF \g_@@_literal_Nabla_bool
   {
    \@@_set_normal_char:nnn {Nabla}{bfsfup}{#1}
   }
   {
    \bool_if:NT \g_@@_upNabla_bool
     {
      \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
     }
   }
  \bool_if:NTF \g_@@_literal_partial_bool
   {
    \@@_set_normal_char:nnn {partial}{bfsfup}{#1}
   }
   {
    \bool_if:NT \g_@@_uppartial_bool
     {
      \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
     }
   }
  \@@_set_mathalphabet_pos:nnnn {bfsfup} {partial} {up,it}{#1}
  \@@_set_mathalphabet_pos:nnnn {bfsfup} {Nabla}   {up,it}{#1}
  \bool_if:NTF \g_@@_literal_partial_bool
```

```
    {
     \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up}{#1}
    }
    {
     \bool_if:NT \g_@@_uppartial_bool
      {
       \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
      }
    }
   \bool_if:NTF \g_@@_literal_Nabla_bool
    {
     \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}   {up}{#1}
    }
    {
     \bool_if:NT \g_@@_upNabla_bool
      {
       \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}   {up,it}{#1}
      }
    }
  }
```

17.14 *Bold italic sans serif: bfsfit*

```
\@@_new_alphabet_config:nnn {bfsfit} {Latin}
 {
  \bool_if:NTF \g_@@_sfliteral_bool
   {
    \@@_set_normal_Latin:nn {bfsfit} {#1}
    \@@_set_mathalphabet_Latin:nnn {bfsf} {it}{#1}
   }
   {
    \bool_if:NF \g_@@_upsans_bool
     {
      \@@_set_normal_Latin:nn {bfsfup,bfsfit} {#1}
      \@@_set_mathalphabet_Latin:nnn {bfsf} {up,it}{#1}
     }
   }
  \@@_set_mathalphabet_Latin:nnn {bfsfit} {up,it}{#1}
 }

\@@_new_alphabet_config:nnn {bfsfit} {latin}
 {
  \bool_if:NTF \g_@@_sfliteral_bool
   {
    \@@_set_normal_latin:nn {bfsfit} {#1}
    \@@_set_mathalphabet_latin:nnn {bfsf} {it}{#1}
   }
   {
    \bool_if:NF \g_@@_upsans_bool
     {
      \@@_set_normal_latin:nn {bfsfup,bfsfit} {#1}
      \@@_set_mathalphabet_latin:nnn {bfsf} {up,it}{#1}
     }
   }
  \@@_set_mathalphabet_latin:nnn {bfsfit} {up,it}{#1}
 }

\@@_new_alphabet_config:nnn {bfsfit} {Greek}
 {
  \bool_if:NTF \g_@@_sfliteral_bool
   {
    \@@_set_normal_Greek:nn {bfsfit}{#1}
    \@@_set_mathalphabet_Greek:nnn {bfsf} {it}{#1}
   }
   {
    \bool_if:NF \g_@@_upsans_bool
     {
      \@@_set_normal_Greek:nn {bfsfup,bfsfit}{#1}
      \@@_set_mathalphabet_Greek:nnn {bfsf} {up,it}{#1}
     }
   }
  \@@_set_mathalphabet_Greek:nnn {bfsfit} {up,it}{#1}
 }

\@@_new_alphabet_config:nnn {bfsfit} {greek}
 {
  \bool_if:NTF \g_@@_sfliteral_bool
   {
    \@@_set_normal_greek:nn {bfsfit} {#1}
    \@@_set_mathalphabet_greek:nnn {bfsf} {it} {#1}
   }
   {
    \bool_if:NF \g_@@_upsans_bool
     {
      \@@_set_normal_greek:nn {bfsfup,bfsfit} {#1}
      \@@_set_mathalphabet_greek:nnn {bfsf} {up,it} {#1}
     }
   }
  \@@_set_mathalphabet_greek:nnn {bfsfit} {up,it} {#1}
 }

\@@_new_alphabet_config:nnn {bfsfit} {misc}
 {
  \bool_if:NTF \g_@@_literal_Nabla_bool
   {
    \@@_set_normal_char:nnn {Nabla}{bfsfit}{#1}
   }
   {
    \bool_if:NF \g_@@_upNabla_bool
```

```
     {
      \@@_set_normal_char:nnn {Nabla}{bfsfup,bfsfit}{#1}
     }
   }
  \bool_if:NTF \g_@@_literal_partial_bool
   {
    \@@_set_normal_char:nnn {partial}{bfsfit}{#1}
   }
   {
    \bool_if:NF \g_@@_uppartial_bool
     {
      \@@_set_normal_char:nnn {partial}{bfsfup,bfsfit}{#1}
     }
   }
  \@@_set_mathalphabet_pos:nnnn {bfsfit} {partial} {up,it}{#1}
  \@@_set_mathalphabet_pos:nnnn {bfsfit} {Nabla}   {up,it}{#1}
  \bool_if:NTF \g_@@_literal_partial_bool
   {
    \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {it}{#1}
   }
   {
    \bool_if:NF \g_@@_uppartial_bool
     {
      \@@_set_mathalphabet_pos:nnnn {bfsf} {partial} {up,it}{#1}
     }
   }
  \bool_if:NTF \g_@@_literal_Nabla_bool
   {
    \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}   {it}{#1}
   }
   {
    \bool_if:NF \g_@@_upNabla_bool
     {
      \@@_set_mathalphabet_pos:nnnn {bfsf} {Nabla}   {up,it}{#1}
     }
   }
 }
```

⟨/package⟩

File XVII

um-code-primes.dtx

18 Primes

```
⟨*package⟩
```

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

U+2032 prime (\prime): x'

U+2033 double prime (\dprime): x''

U+2034 triple prime (\trprime): x'''

U+2057 quadruple prime (\qprime): x''''

As you can see, they're all drawn at the correct height without being superscripted. However, in a correctly behaving OpenType font, we also see different behaviour after the ssty feature is applied:

$x′ \quad x″ \quad x‴ \quad x⁗$

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

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

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

Our algorithm is:

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

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

```
\cs_new:Nn \@@_arg_i_before_egroup:n {#1\egroup}
```

```
\cs_new:Nn \@@_superscript:n
 {
  ^\bgroup #1
  \peek_meaning_remove:NTF ^ \@@_arg_i_before_egroup:n \egroup
 }
\cs_new:Nn \@@_nprimes:Nn
 {
  \@@_superscript:n
   {
    #1
    \prg_replicate:nn {#2-1} { \mskip \g_@@_primekern_muskip #1 }
   }
 }
\cs_new:Nn \@@_nprimes_select:nn
 {
  \int_case:nnF {#2}
   {
    {1} { \@@_superscript:n {#1} }
    {2} {
      \@@_glyph_if_exist:NnTF \g_@@_prime_font_cmd_tl {"2033}
        { \@@_superscript:n {\@@_prime_double_mchar} }
        { \@@_nprimes:Nn #1 {#2} }
    }
    {3} {
      \@@_glyph_if_exist:NnTF \g_@@_prime_font_cmd_tl {"2034}
        { \@@_superscript:n {\@@_prime_triple_mchar} }
        { \@@_nprimes:Nn #1 {#2} }
    }
    {4} {
      \@@_glyph_if_exist:NnTF \g_@@_prime_font_cmd_tl {"2057}
        { \@@_superscript:n {\@@_prime_quad_mchar} }
        { \@@_nprimes:Nn #1 {#2} }
    }
   }
   {
    \@@_nprimes:Nn #1 {#2}
   }
 }
\cs_new:Nn \@@_nbackprimes_select:nn
 {
  \int_case:nnF {#2}
   {
    {1} { \@@_superscript:n {#1} }
    {2} {
      \@@_glyph_if_exist:NnTF \g_@@_prime_font_cmd_tl {"2036}
        { \@@_superscript:n {\@@_backprime_double_mchar} }
        { \@@_nprimes:Nn #1 {#2} }
    }
    {3} {
      \@@_glyph_if_exist:NnTF \g_@@_prime_font_cmd_tl {"2037}
        { \@@_superscript:n {\@@_backprime_triple_mchar} }
        { \@@_nprimes:Nn #1 {#2} }
    }
   }
   {
    \@@_nprimes:Nn #1 {#2}
   }
 }
```

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

```
\cs_new:Npn \@@_scan_prime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_zero:N \l_@@_primecount_int
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_dprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_set:Nn \l_@@_primecount_int {1}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_trprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_set:Nn \l_@@_primecount_int {2}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_qprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_set:Nn \l_@@_primecount_int {3}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_prime:
 {
  \int_zero:N \l_@@_primecount_int
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_dprime:
 {
  \int_set:Nn \l_@@_primecount_int {1}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_trprime:
 {
  \int_set:Nn \l_@@_primecount_int {2}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_qprime:
 {
  \int_set:Nn \l_@@_primecount_int {3}
  \@@_scanprime_collect:N \@@_prime_single_mchar
 }
\cs_new:Nn \@@_scanprime_collect:N
 {
  \int_incr:N \l_@@_primecount_int
  \peek_meaning_remove:NTF '
   { \@@_scanprime_collect:N #1 }
   {
    \peek_meaning_remove:NTF \@@_scan_prime:
     { \@@_scanprime_collect:N #1 }
     {
      \peek_meaning_remove:NTF ^^^^2032
       { \@@_scanprime_collect:N #1 }
       {
        \peek_meaning_remove:NTF \@@_scan_dprime:
         {
          \int_incr:N \l_@@_primecount_int
          \@@_scanprime_collect:N #1
         }
         {
          \peek_meaning_remove:NTF ^^^^2033
           {
            \int_incr:N \l_@@_primecount_int
            \@@_scanprime_collect:N #1
           }
           {
            \peek_meaning_remove:NTF \@@_scan_trprime:
             {
              \int_add:Nn \l_@@_primecount_int {2}
              \@@_scanprime_collect:N #1
             }
             {
              \peek_meaning_remove:NTF ^^^^2034
               {
                \int_add:Nn \l_@@_primecount_int {2}
                \@@_scanprime_collect:N #1
               }
               {
                \peek_meaning_remove:NTF \@@_scan_qprime:
                 {
                  \int_add:Nn \l_@@_primecount_int {3}
                  \@@_scanprime_collect:N #1
                 }
                 {
                  \peek_meaning_remove:NTF ^^^^2057
                  {
                   \int_add:Nn \l_@@_primecount_int {3}
                   \@@_scanprime_collect:N #1
                  }
                  {
                   \@@_nprimes_select:nn {#1} {\l_@@_primecount_int}
                  }
                }
              }
            }
          }
        }
      }
    }
  }
 }
```

```
\cs_new:Npn \@@_scan_backprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_zero:N \l_@@_primecount_int
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
\cs_new:Npn \@@_scan_backdprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_set:Nn \l_@@_primecount_int {1}
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
\cs_new:Npn \@@_scan_backtrprime:
 {
  \cs_set_eq:NN \@@_superscript:n \use:n
  \int_set:Nn \l_@@_primecount_int {2}
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_backprime:
 {
  \int_zero:N \l_@@_primecount_int
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_backdprime:
 {
  \int_set:Nn \l_@@_primecount_int {1}
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
\cs_new:Npn \@@_scan_sup_backtrprime:
 {
  \int_set:Nn \l_@@_primecount_int {2}
  \@@_scanbackprime_collect:N \@@_backprime_single_mchar
 }
```

```
\cs_new:Nn \@@_scanbackprime_collect:N
 {
  \int_incr:N \l_@@_primecount_int
  \peek_meaning_remove:NTF `
   {
    \@@_scanbackprime_collect:N #1
   }
   {
    \peek_meaning_remove:NTF \@@_scan_backprime:
     {
      \@@_scanbackprime_collect:N #1
     }
     {
      \peek_meaning_remove:NTF ^^^^2035
       {
        \@@_scanbackprime_collect:N #1
       }
       {
        \peek_meaning_remove:NTF \@@_scan_backdprime:
         {
          \int_incr:N \l_@@_primecount_int
          \@@_scanbackprime_collect:N #1
         }
         {
          \peek_meaning_remove:NTF ^^^^2036
           {
            \int_incr:N \l_@@_primecount_int
            \@@_scanbackprime_collect:N #1
           }
           {
            \peek_meaning_remove:NTF \@@_scan_backtrprime:
             {
              \int_add:Nn \l_@@_primecount_int {2}
              \@@_scanbackprime_collect:N #1
             }
             {
              \peek_meaning_remove:NTF ^^^^2037
               {
                \int_add:Nn \l_@@_primecount_int {2}
                \@@_scanbackprime_collect:N #1
               }
               {
                \@@_nbackprimes_select:nn {#1} {\l_@@_primecount_int}
               }
             }
           }
         }
       }
     }
    }
  }
\AtBeginDocument { \@@_define_prime_commands: \@@_define_prime_chars: }
\cs_new:Nn \@@_define_prime_commands:
 {
  \cs_set_eq:NN \prime       \@@_prime_single_mchar
  \cs_set_eq:NN \dprime      \@@_prime_double_mchar
  \cs_set_eq:NN \trprime     \@@_prime_triple_mchar
  \cs_set_eq:NN \qprime      \@@_prime_quad_mchar
  \cs_set_eq:NN \backprime   \@@_backprime_single_mchar
  \cs_set_eq:NN \backdprime  \@@_backprime_double_mchar
  \cs_set_eq:NN \backtrprime \@@_backprime_triple_mchar
 }
\group_begin:
  \char_set_catcode_active:N \'
  \char_set_catcode_active:N \`
  \char_set_catcode_active:n {"2032}
  \char_set_catcode_active:n {"2033}
  \char_set_catcode_active:n {"2034}
  \char_set_catcode_active:n {"2057}
  \char_set_catcode_active:n {"2035}
  \char_set_catcode_active:n {"2036}
  \char_set_catcode_active:n {"2037}
  \cs_gset:Nn \@@_define_prime_chars:
   {
    \cs_set_eq:NN '        \@@_scan_sup_prime:
    \cs_set_eq:NN ^^^^2032 \@@_scan_sup_prime:
    \cs_set_eq:NN ^^^^2033 \@@_scan_sup_dprime:
    \cs_set_eq:NN ^^^^2034 \@@_scan_sup_trprime:
    \cs_set_eq:NN ^^^^2057 \@@_scan_sup_qprime:
    \cs_set_eq:NN `        \@@_scan_sup_backprime:
    \cs_set_eq:NN ^^^^2035 \@@_scan_sup_backprime:
    \cs_set_eq:NN ^^^^2036 \@@_scan_sup_backdprime:
    \cs_set_eq:NN ^^^^2037 \@@_scan_sup_backtrprime:
   }
\group_end:
⟨/package⟩
```

File XVIII
um-code-sscript.dtx

19 *Unicode sub- and super-scripts*

```
⟨*package⟩
```

The idea here is to enter a scanning state after a superscript or subscript is encountered. If subsequent superscripts or subscripts (resp.) are found, they are lumped together. Each sub/super has a corresponding regular size glyph which is used by XƎTEX to typeset the results; this means that the actual subscript/superscript glyphs are never seen in the output document — they are only used as input characters.

Open question: should the superscript-like 'modifiers' (U+1D2C modifier capital letter a and on) be included here?

Superscripts Populate a property list with superscript characters; themselves as their key, and their replacement as each key's value. Then make the superscript active and bind it to the scanning function.

\scantokens makes this process much simpler since we can activate the char and assign its meaning in one step.

```
\cs_new:Nn \@@_setup_active_superscript:nn
  {
    \prop_gput:Nxn \g_@@_supers_prop { \int_eval:n {#1} } {#2}
    \@@_mathactive_remap:nn {#1}
       {
        \tl_set:Nn \l_@@_ss_chain_tl {#2}
        \cs_set_eq:NN \@@_sub_or_super:n \sp
        \tl_set:Nn \l_@@_tmpa_tl {supers}
        \@@_scan_sscript:
       }
  }
```

Subscripts

```
\cs_new:Nn \@@_setup_active_subscript:nn
  {
    \prop_gput:Nxn \g_@@_subs_prop { \int_eval:n {#1} } {#2}
    \@@_mathactive_remap:nn {#1}
      {
        \tl_set:Nn \l_@@_ss_chain_tl {#2}
        \cs_set_eq:NN \@@_sub_or_super:n \sb
        \tl_set:Nn \l_@@_tmpa_tl {subs}
        \@@_scan_sscript:
      }
  }
```

The scanning command Collects a chain of subscripts or a chain of superscripts and then typesets what it has collected.

```
\cs_new:Nn \@@_scan_sscript:
  {
    \@@_scan_sscript:TF
      { \@@_scan_sscript: }
      { \@@_sub_or_super:n {\l_@@_ss_chain_tl} }
  }
```

We do not skip spaces when scanning ahead, and we explicitly wish to bail out on encountering a space or a brace. These cases are filtered using `\peek_N_type:TF`. Otherwise the token can be taken as an `N`-type argument. Then we search for it in the appropriate property list (`\l_@@_tmpa_tl` is `subs` or `supers`). If found, add the value to the current chain of sub/superscripts. Remember to put the character back in the input otherwise. The `\group_align_safe_begin:` and `\group_align_-safe_end:` are needed in case `#3` is `&`.

```
\cs_new:Nn \@@_scan_sscript:TF
  {
    \peek_N_type:TF
      {
        \group_align_safe_begin:
        \@@_scan_sscript_aux:nnN {#1} {#2}
      }
      {#2}
  }
```

The look-ahead for the sscripts doesn't try to peek inside the lookahead.

```
\cs_new:Nn \@@_scan_sscript_aux:nnN
  {
    \tl_set:Nx \l_@@_tmpa_key_tl { \tl_to_str:n {#3} }
    \prop_get:cxNTF {g_@@_\l_@@_tmpa_tl _prop}
      { \int_eval:n { \exp_after:wN ` \l_@@_tmpa_key_tl } }
      \l_@@_tmpb_tl
      {
        \tl_put_right:NV \l_@@_ss_chain_tl \l_@@_tmpb_tl
        \group_align_safe_end:
        #1
      }
      { \group_align_safe_end: #2 #3 }
  }
```

Definitions Superscripts.

```
\@@_setup_active_superscript:nn {"2070} {0}
\@@_setup_active_superscript:nn {"00B9} {1}
\@@_setup_active_superscript:nn {"00B2} {2}
\@@_setup_active_superscript:nn {"00B3} {3}
\@@_setup_active_superscript:nn {"2074} {4}
\@@_setup_active_superscript:nn {"2075} {5}
\@@_setup_active_superscript:nn {"2076} {6}
```

```
\@@_setup_active_superscript:nn {"2077} {7}
\@@_setup_active_superscript:nn {"2078} {8}
\@@_setup_active_superscript:nn {"2079} {9}
\@@_setup_active_superscript:nn {"207A} {+}
\@@_setup_active_superscript:nn {"207B} {-}
\@@_setup_active_superscript:nn {"207C} {=}
\@@_setup_active_superscript:nn {"207D} {(}
\@@_setup_active_superscript:nn {"207E} {)}
\@@_setup_active_superscript:nn {"2071} {i}
\@@_setup_active_superscript:nn {"207F} {n}
\@@_setup_active_superscript:nn {"02B0} {h}
\@@_setup_active_superscript:nn {"02B2} {j}
\@@_setup_active_superscript:nn {"02B3} {r}
\@@_setup_active_superscript:nn {"02B7} {w}
\@@_setup_active_superscript:nn {"02B8} {y}
```

A few more subscripts than superscripts:

```
\@@_setup_active_subscript:nn {"2080} {0}
\@@_setup_active_subscript:nn {"2081} {1}
\@@_setup_active_subscript:nn {"2082} {2}
\@@_setup_active_subscript:nn {"2083} {3}
\@@_setup_active_subscript:nn {"2084} {4}
\@@_setup_active_subscript:nn {"2085} {5}
\@@_setup_active_subscript:nn {"2086} {6}
\@@_setup_active_subscript:nn {"2087} {7}
\@@_setup_active_subscript:nn {"2088} {8}
\@@_setup_active_subscript:nn {"2089} {9}
\@@_setup_active_subscript:nn {"208A} {+}
\@@_setup_active_subscript:nn {"208B} {-}
\@@_setup_active_subscript:nn {"208C} {=}
\@@_setup_active_subscript:nn {"208D} {(}
\@@_setup_active_subscript:nn {"208E} {)}
\@@_setup_active_subscript:nn {"2090} {a}
\@@_setup_active_subscript:nn {"2091} {e}
\@@_setup_active_subscript:nn {"2095} {h}
\@@_setup_active_subscript:nn {"1D62} {i}
\@@_setup_active_subscript:nn {"2C7C} {j}
\@@_setup_active_subscript:nn {"2096} {k}
\@@_setup_active_subscript:nn {"2097} {l}
\@@_setup_active_subscript:nn {"2098} {m}
\@@_setup_active_subscript:nn {"2099} {n}
\@@_setup_active_subscript:nn {"2092} {o}
\@@_setup_active_subscript:nn {"209A} {p}
\@@_setup_active_subscript:nn {"1D63} {r}
\@@_setup_active_subscript:nn {"209B} {s}
\@@_setup_active_subscript:nn {"209C} {t}
\@@_setup_active_subscript:nn {"1D64} {u}
\@@_setup_active_subscript:nn {"1D65} {v}
\@@_setup_active_subscript:nn {"2093} {x}
\@@_setup_active_subscript:nn {"1D66} {\beta}
```

```
\@@_setup_active_subscript:nn {"1D67} {\gamma}
\@@_setup_active_subscript:nn {"1D68} {\rho}
\@@_setup_active_subscript:nn {"1D69} {\phi}
\@@_setup_active_subscript:nn {"1D6A} {\chi}
```

⟨/package⟩

File XIX

um-code-compat.dtx

20 Compatibility

```
⟨*package⟩
```

\@@_check_and_fix:NNnnn

#1 : command
#2 : factory command
#3 : parameter text
#4 : expected replacement text
#5 : new replacement text

Tries to patch ⟨*command*⟩. If ⟨*command*⟩ is undefined, do nothing. Otherwise it must be a macro with the given ⟨*parameter text*⟩ and ⟨*expected replacement text*⟩, created by the given ⟨*factory command*⟩ or equivalent. In this case it will be overwritten using the ⟨*parameter text*⟩ and the ⟨*new replacement text for LuaTEX*⟩ or the ⟨*new replacement text for XƎTEX*⟩, depending on the engine. Otherwise issue a warning and don't overwrite.

```
\cs_new_protected_nopar:Nn \@@_check_and_fix:NNnnn
  {
    \cs_if_exist:NT #1
      {
        \token_if_macro:NTF #1
          {
            \group_begin:
            #2 \@@_tmpa:w #3 { #4 }
            \cs_if_eq:NNTF #1 \@@_tmpa:w
              {
           \msg_info:nnx { unicode-math } { patch-macro } { \token_to_str:N #1 }
                \group_end:
                #2 #1 #3 { #5 }
              }
              {
                \msg_warning:nnxxx { unicode-math } { wrong-meaning }
                  { \token_to_str:N #1 } { \token_to_meaning:N #1 }
                  { \token_to_meaning:N \@@_tmpa:w }
                \group_end:
              }
          }
          {
            \msg_warning:nnx { unicode-math } { macro-expected }
              { \token_to_str:N #1 }
          }
      }
  }
```

21 Patching/augmenting 3rd-party packages

21.1 url

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

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

```
\AtEndOfPackageFile * {url}
  {
    \tl_put_left:Nn \Url@FormatString { \@@_switchto_up: }
    \tl_put_right:Nn \UrlSpecials
      {
        \do \` { \mathchar`\` }
        \do \' { \mathchar`\' }
        \do \$ { \mathchar`\$ }
        \do \& { \mathchar`\& }
      }
  }
```

21.2 mathtools

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

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

```
⟨*XE⟩
\AtEndOfPackageFile * { mathtools }
  {
   \@@_check_and_fix:NNnnn \MT_cramped_internal:Nn \cs_set_nopar:Npn { #1 #2 }
     {
      \sbox \z@ { $ \m@th #1 \nulldelimiterspace = \z@ \radical \z@ { #2 } $ }
       \ifx #1 \displaystyle
         \dimen@ = \fontdimen 8 \textfont 3
         \advance \dimen@ .25 \fontdimen 5 \textfont 2
       \else
         \dimen@ = 1.25 \fontdimen 8
         \ifx #1 \textstyle
           \textfont
         \else
           \ifx #1 \scriptstyle
             \scriptfont
           \else
             \scriptscriptfont
```

```
          \fi
        \fi
        3
      \fi
      \advance \dimen@ -\ht\z@
      \ht\z@ = -\dimen@
      \box\z@
    }
    {
      \hbox_set:Nn \l_tmpa_box
        {
          \color@setgroup \c_math_toggle_token \m@th
            #1
            \dim_zero:N \nulldelimiterspace
            \XeTeXradical \c_two_hundred_fifty_five \c_zero { #2 }
          \c_math_toggle_token \color@endgroup
        }
      \box_set_ht:Nn \l_tmpa_box
        {
          \box_ht:N \l_tmpa_box - \@@_radical_vgap:N #1
        }
      \box_use_drop:N \l_tmpa_box
    }
  }
⟨/XE⟩
```

\overbracket
\underbracket

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

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

```
\AtEndOfPackageFile * { mathtools }
  {
    \cs_set_eq:NN \MToverbracket  \overbracket
    \cs_set_eq:NN \MTunderbracket \underbracket

    \AtBeginDocument
      {
        \msg_warning:nn { unicode-math } { mathtools-overbracket }

        \cs_set:Npn \downbracketfill #1 #2
         {
             \tl_set:Nn \l_MT_bracketheight_fdim {.27ex}
             \downbracketend {#1} {#2}
             \leaders \vrule \@height #1 \@depth \z@ \hfill
             \downbracketend {#1} {#2}
           }

      \cs_set:Npn \upbracketfill #1 #2
        {
            \tl_set:Nn \l_MT_bracketheight_fdim {.27ex}
            \upbracketend {#1} {#2}
            \leaders \vrule \@height \z@ \@depth #1 \hfill
            \upbracketend {#1} {#2}
          }

      \cs_set_eq:NN \Uoverbracket  \overbracket
      \cs_set_eq:NN \Uunderbracket \underbracket
        \cs_set_eq:NN \overbracket   \MToverbracket
        \cs_set_eq:NN \underbracket  \MTunderbracket
    }
}
```

\dblcolon \coloneqq \Coloneqq \eqqcolon

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

```
\AtEndOfPackageFile * { mathtools }
  {
    \msg_warning:nn { unicode-math } { mathtools-colon }
    \NewDocumentCommand \dblcolon { } { \Colon }
    \NewDocumentCommand \coloneqq { } { \coloneq }
    \NewDocumentCommand \Coloneqq { } { \Coloneq }
    \NewDocumentCommand \eqqcolon { } { \eqcolon }
  }
```

21.3 *colonequals*

\ratio \coloncolon \minuscolon \colonequals \equalscolon \coloncolonequals

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

```
\AtEndOfPackageFile * { colonequals }
  {
    \msg_warning:nn { unicode-math } { colonequals }
    \RenewDocumentCommand \ratio { } { \mathratio }
    \RenewDocumentCommand \coloncolon { } { \Colon }
    \RenewDocumentCommand \minuscolon { } { \dashcolon }
    \RenewDocumentCommand \colonequals { } { \coloneq }
    \RenewDocumentCommand \equalscolon { } { \eqcolon }
    \RenewDocumentCommand \coloncolonequals { } { \Coloneq }
  }
```

```
⟨/package⟩
```

File XX
um-code-amsmath.dtx

22 *Compatibility with amsmath*

```
⟨*package⟩
```

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:

```
\AtEndOfPackageFile * {amsmath}
  {
    \tl_remove_once:Nn \@begindocumenthook
      {
        \mathchardef\std@minus\mathcode`\-\relax
        \mathchardef\std@equal\mathcode`\=\relax
      }
    \AtBeginDocument
      {
        \Umathcharnumdef\std@minus\Umathcodenum`-
        \Umathcharnumdef\std@equal\Umathcodenum`=
      }
    \cs_set:Npn \@cdots {\mathinner{\unicodecdots}}
    \cs_set_eq:NN \dotsb@ \cdots
```

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

```
⟨*XE⟩
    \def \resetMathstrut@
      {%
        \setbox\z@\hbox{$($}%)
        \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
      }
```

The subarray environment uses inappropriate font dimensions.

```
    \@@_check_and_fix:NNnnn \subarray \cs_set:Npn { #1 }
      {
        \vcenter
        \bgroup
        \Let@
        \restore@math@cr
        \default@tag
        \baselineskip \fontdimen 10~ \scriptfont \tw@
        \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
        \lineskip \thr@@@@ \fontdimen 8~ \scriptfont \thr@@@@
        \lineskiplimit \lineskip
        \ialign
        \bgroup
        \ifx c #1 \hfil \fi
        $ \m@th \scriptstyle ## $
        \hfil
        \crcr
      }
      {
        \vcenter
        \c_group_begin_token
        \Let@
        \restore@math@cr
        \default@tag
        \skip_set:Nn \baselineskip
          {
```
Here we use stack top shift + stack bottom shift, which sounds reasonable.
```
            \@@_stack_num_up:N \scriptstyle
            + \@@_stack_denom_down:N \scriptstyle
          }
```
Here we use the minimum stack gap.
```
        \lineskip \@@_stack_vgap:N \scriptstyle
        \lineskiplimit \lineskip
        \ialign
        \c_group_begin_token
        \token_if_eq_meaning:NNT c #1 { \hfil }
        \c_math_toggle_token
        \m@th
        \scriptstyle
        \c_parameter_token \c_parameter_token
        \c_math_toggle_token
        \hfil
        \crcr
      }
⟨/XE⟩
```
The roots need a complete rework.
```
⟨*LU⟩
  \@@_check_and_fix:NNnnn \plainroot@ \cs_set_nopar:Npn { #1 \of #2 }
    {
      \setbox \rootbox \hbox
        {
          $ \m@th \scriptscriptstyle { #1 } $
        }
      \mathchoice
        { \r@@@@t \displaystyle      { #2 } }
        { \r@@@@t \textstyle         { #2 } }~
        { \r@@@@t \scriptstyle       { #2 } }
        { \r@@@@t \scriptscriptstyle { #2 } }
      \egroup
    }
    {
      \bool_if:nTF
        {
          \int_compare_p:nNn { \uproot@ } = { \c_zero }
          && \int_compare_p:nNn { \leftroot@ } = { \c_zero }
        }
        {
          \Uroot \l_@@_radical_sqrt_tl { #1 } { #2 }
        }
        {
          \hbox_set:Nn \rootbox
            {
              \c_math_toggle_token \m@th
              \scriptscriptstyle { #1 }
              \c_math_toggle_token
            }
          \mathchoice
            { \r@@@@t \displaystyle      { #2 } }
            { \r@@@@t \textstyle         { #2 } }
            { \r@@@@t \scriptstyle       { #2 } }
            { \r@@@@t \scriptscriptstyle { #2 } }
        }
       \c_group_end_token
    }
⟨/LU⟩
  \@@_check_and_fix:NNnnn \r@@@@t \cs_set_nopar:Npn { #1 #2 }
    {
      \setboxz@h { $ \m@th #1 \sqrtsign { #2 } $ }
      \dimen@ \ht\z@
      \advance \dimen@ -\dp\z@
      \setbox\@ne \hbox { $ \m@th #1 \mskip \uproot@ mu $ }
      \advance \dimen@ by 1.667 \wd\@ne
      \mkern -\leftroot@ mu
      \mkern 5mu
      \raise .6\dimen@ \copy\rootbox
      \mkern -10mu
      \mkern \leftroot@ mu
      \boxz@
    }
⟨*LU⟩
    {
      \hbox_set:Nn \l_tmpa_box
        {
          \c_math_toggle_token \m@th
            #1 \mskip \uproot@ mu
          \c_math_toggle_token
        }
      \Uroot \l_@@_radical_sqrt_tl
        {
          \box_move_up:nn { \box_wd:N \l_tmpa_box }
            {
              \hbox:n
                {
                  \c_math_toggle_token \m@th
                    \mkern -\leftroot@ mu
                    \box_use:N \rootbox
                    \mkern \leftroot@ mu
                  \c_math_toggle_token
                }
            }
        }
        { #2 }
    }
⟨/LU⟩
⟨*XE⟩
    {
      \hbox_set:Nn \l_tmpa_box
        {
          \c_math_toggle_token \m@th
            #1 \sqrtsign { #2 }
          \c_math_toggle_token
        }
      \hbox_set:Nn \l_tmpb_box
        {
          \c_math_toggle_token \m@th
            #1 \mskip \uproot@ mu
          \c_math_toggle_token
        }
      \mkern -\leftroot@ mu
    \@@_mathstyle_scale:NnnN #1 { \kern } { \fontdimen 63 \g_@@_sqrt_font_cmd_tl } \g_@@_sqrt_font_cmd_
      \box_move_up:nn
        {
        \box_wd:N \l_tmpb_box + (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box)
            * \number \fontdimen 65 \g_@@_sqrt_font_cmd_tl / 100
        }
        { \box_use:N \rootbox }
    \@@_mathstyle_scale:NnnN #1 { \kern } { \fontdimen 64 \g_@@_sqrt_font_cmd_tl } \g_@@_sqrt_font_cmd_
      \mkern \leftroot@ mu
      \box_use_drop:N \l_tmpa_box
    }
⟨/XE⟩
  }
⟨/package⟩
```

File XXI

um-code-epilogue.dtx

23 *Epilogue*

```
⟨*package⟩
```

Lots of little things to tidy up.

23.1 *Resolving Greek symbol name control sequences*

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

```
\AtBeginDocument { \@@_resolve_greek: }
```

```
\cs_new:Npn \@@_resolve_greek:
  {
    \clist_map_inline:nn
      {
        Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa,Lambda,
        alpha,beta,gamma,delta,epsilon,zeta,eta,theta,iota,kappa,lambda,
        Mu,Nu,Xi,Omicron,Pi,Rho,Sigma,Tau,Upsilon,Phi,Chi,Psi,Omega,
        mu,nu,xi,omicron,pi,rho,sigma,tau,upsilon,phi,chi,psi,omega,
        varTheta,varsigma,vartheta,varkappa,varrho,varpi,varepsilon,varphi
      }
      {
        \tl_set:cx {##1} { \exp_not:c { mit ##1 } }
        \tl_set:cx {up ##1} { \exp_not:N \symup \exp_not:c { ##1 } }
        \tl_set:cx {it ##1} { \exp_not:N \symit \exp_not:c { ##1 } }
      }
  }
```

23.2 *Unicode radicals*

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

`\@@_redefine_radical:`

```
\AtBeginDocument{ \@ifpackageloaded { amsmath } { } { \@@_redefine_radical: } }
```

`\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`.

```
⟨*XE⟩
\cs_new:Nn \@@_redefine_radical:
  {
    \cs_set_nopar:Npn \r@@@@t ##1 ##2
      {
        \hbox_set:Nn \l_tmpa_box
          {
            \c_math_toggle_token \m@th
            ##1 \sqrtsign { ##2 }
            \c_math_toggle_token
          }
     \@@_mathstyle_scale:NnnN ##1 { \kern } { \fontdimen 63 \g_@@_sqrt_font_cmd_tl } \g_@@_sqrt_font_cm
        \box_move_up:nn
          {
            (\box_ht:N \l_tmpa_box - \box_dp:N \l_tmpa_box) * \number \fontdi-
men 65 \g_@@_sqrt_font_cmd_tl / 100
          }
          { \box_use:N \rootbox }
     \@@_mathstyle_scale:NnnN ##1 { \kern } { \fontdimen 64 \g_@@_sqrt_font_cmd_tl } \g_@@_sqrt_font_cm
        \box_use_drop:N \l_tmpa_box
      }
  }
⟨/XE⟩
```

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

```
⟨*LU⟩
\cs_new:Nn \@@_redefine_radical:
  {
    \cs_set:Npn \root ##1 \of ##2
      {
        \Uroot \l_@@_radical_sqrt_tl { ##1 } { ##2 }
      }
  }
⟨/LU⟩
```

23.2.1 *Active fractions*

Active fractions can be set up independently of any maths font definition; all it requires is a mapping from the Unicode input chars to the relevant LATEX fraction declaration.

```
\cs_new:Nn \@@_which_frac:nn
  {
    \bool_if:NTF \l_@@_smallfrac_bool {\tfrac} {\frac} {#1} {#2}
  }
```

```
\cs_new:Npn \@@_setup_active_frac:
  {
    \@@_mathactive_remap:nn {"2189}  { \@@_which_frac:nn {0} {3}  }
    \@@_mathactive_remap:nn {"2152}  { \@@_which_frac:nn {1} {10} }
    \@@_mathactive_remap:nn {"2151}  { \@@_which_frac:nn {1} {9}  }
    \@@_mathactive_remap:nn {"215B}  { \@@_which_frac:nn {1} {8}  }
    \@@_mathactive_remap:nn {"2150}  { \@@_which_frac:nn {1} {7}  }
    \@@_mathactive_remap:nn {"2159}  { \@@_which_frac:nn {1} {6}  }
    \@@_mathactive_remap:nn {"2155}  { \@@_which_frac:nn {1} {5}  }
    \@@_mathactive_remap:nn {"00BC}  { \@@_which_frac:nn {1} {4}  }
    \@@_mathactive_remap:nn {"2153}  { \@@_which_frac:nn {1} {3}  }
    \@@_mathactive_remap:nn {"215C}  { \@@_which_frac:nn {3} {8}  }
    \@@_mathactive_remap:nn {"2156}  { \@@_which_frac:nn {2} {5}  }
    \@@_mathactive_remap:nn {"00BD}  { \@@_which_frac:nn {1} {2}  }
    \@@_mathactive_remap:nn {"2157}  { \@@_which_frac:nn {3} {5}  }
    \@@_mathactive_remap:nn {"215D}  { \@@_which_frac:nn {5} {8}  }
    \@@_mathactive_remap:nn {"2154}  { \@@_which_frac:nn {2} {3}  }
    \@@_mathactive_remap:nn {"00BE}  { \@@_which_frac:nn {3} {4}  }
    \@@_mathactive_remap:nn {"2158}  { \@@_which_frac:nn {4} {5}  }
    \@@_mathactive_remap:nn {"215A}  { \@@_which_frac:nn {5} {6}  }
    \@@_mathactive_remap:nn {"215E}  { \@@_which_frac:nn {7} {8}  }
  }
```

```
\AtBeginDocument { \@@_setup_active_frac: }
```

23.3 *Synonyms and all the rest*

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

```
\protected\def\to{\rightarrow}
\protected\def\le{\leq}
\protected\def\ge{\geq}
\protected\def\neq{\ne}
\protected\def\triangle{\mathord{\bigtriangleup}}
\protected\def\bigcirc{\mdlgwhtcircle}
\protected\def\circ{\vysmwhtcircle}
\protected\def\bullet{\smblkcircle}
\protected\def\mathyen{\yen}
\protected\def\mathsterling{\sterling}
\protected\def\diamond{\smwhtdiamond}
\protected\def\emptyset{\varnothing}
\protected\def\hbar{\hslash}
\protected\def\land{\wedge}
\protected\def\lor{\vee}
\protected\def\owns{\ni}
\protected\def\gets{\leftarrow}
\protected\def\mathring{\ocirc}
\protected\def\lnot{\neg}
\protected\def\longdivision{\longdivisionsign}
```

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

```
\protected\def\backepsilon{\upbackepsilon}
\protected\def\eth{\matheth}
```

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

```
\protected\def\dbkarow {\dbkarrow}
\protected\def\drbkarow{\drbkarrow}
\protected\def\hksearow{\hksearrow}
\protected\def\hkswarow{\hkswarrow}
```

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

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

`\underbar`

```
\cs_set_eq:NN \latexe_underbar:n \underbar
\renewcommand\underbar
  {
    \mode_if_math:TF \mathunderbar \latexe_underbar:n
  }
```

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

```
\@ifpackageloaded{amsmath}
  {
   % define their own colon, perhaps I should just steal it. (It does look much bet-
ter.)
  }
  {
    \cs_set_protected:Npn \colon
      {
        \bool_if:NTF \g_@@_literal_colon_bool {:} { \mathpunct{:} }
      }
  }
```

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

```
\protected\def\digamma{\updigamma}
\protected\def\Digamma{\upDigamma}
```

Symbols

```
\cs_set_protected:Npn \| {\Vert}
```

`\mathinner` items:

```
\cs_set_protected:Npn \mathellipsis {\mathinner{\unicodeellipsis}}
\cs_set_protected:Npn \cdots {\mathinner{\unicodecdots}}
```

```
\cs_set_eq:NN \@@_text_slash: \slash
\cs_set_protected:Npn \slash
  {
    \mode_if_math:TF {\mathslash} {\@@_text_slash:}
  }
```

23.3.1 \not

The situation of \not symbol is currently messy, in Unicode it is defined as a combining mark so naturally it should be treated as a math accent, however XeTeX does not correctly place it as it needs special treatment compared to other accents. Furthermore a math accent changes the spacing of its nucleus, so \not= will be spaced as an ordinary not relational symbol, which is undesired.

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

This code is based on an answer to a TeX – Stack Exchange question by Enrico Gregorio[3].

\not

```
\DeclareDocumentCommand \not {m}
  {
    \tl_set:Nx \l_@@_not_token_name_tl { \cs_to_str:N #1 }
    \tl_if_empty:NT \l_@@_not_token_name_tl
      {
        \tl_set:Nx \l_@@_not_token_name_tl { \token_to_str:N #1 }
      }
    \cs_if_exist:cTF { not \l_@@_not_token_name_tl }
      {
        \use:c { not \l_@@_not_token_name_tl }
      }
      {
        \cs_if_exist:cTF { n \l_@@_not_token_name_tl }
          {
            \use:c { n \l_@@_not_token_name_tl }
          }
          {
            \tl_if_eq:nnTF {#1} {$} { \notaccent{} } { \notaccent } #1
          }
      }
  }
```

\NewNegationCommand
\RenewNegationCommand

```
\DeclareDocumentCommand \NewNegationCommand {mm}
  {
    \@@_set_negation_command:Nnn \cs_new_protected:cpn {#1} {#2}
  }
\DeclareDocumentCommand \RenewNegationCommand {mm}
  {
    \@@_set_negation_command:Nnn \cs_set_protected:cpn {#1} {#2}
  }
\cs_set:Nn \@@_set_negation_command:Nnn
```

[3] http://tex.stackexchange.com/a/47260/729

```
  {
    \tl_set:Nx \l_@@_not_token_name_tl { \cs_to_str:N #2 }
    \tl_if_empty:NT \l_@@_not_token_name_tl
      {
        \tl_set:Nx \l_@@_not_token_name_tl { \token_to_str:N #2 }
      }
    #1 { not \l_@@_not_token_name_tl } { #3 }
  }
```

```
\NewNegationCommand { = }    { \neq   }
\NewNegationCommand { < }    { \nless }
\NewNegationCommand { > }    { \ngtr  }
\NewNegationCommand { \gets     } { \nleftarrow }
\NewNegationCommand { \simeq    } { \nsime      }
\NewNegationCommand { \equal    } { \ne         }
\NewNegationCommand { \le       } { \nleq       }
\NewNegationCommand { \ge       } { \ngeq       }
\NewNegationCommand { \greater  } { \ngtr       }
\NewNegationCommand { \forksnot } { \forks      }
```

23.3.2 *Full-width remapping*

While this could be done with the full mathcode remapping machinery used for the other purposes, it would be fairly redundant with plain ASCII. Worse, this would slow down what is already an inefficient part of `unicode-math`.

Instead we use mathactive to do a plain old mapping from full-width to ASCII directly.

Until I get requests for it, I've not included symbols or punctuation here.

Numbers

```
\int_step_inline:nnnn {0} {1} {9}
  {
    \@@_mathactive_remap:nn {"FF10+#1} {\char\int_eval:n{`\0+#1}}
  }
```

Letters

```
\int_step_inline:nnnn {0} {1} {26}
  {
    \@@_mathactive_remap:nn {"FF21+#1} {\char\int_eval:n{`\A+#1}}
    \@@_mathactive_remap:nn {"FF41+#1} {\char\int_eval:n{`\a+#1}}
  }
```

23.4 *Legacy characters*

`\@@_undeclare_symbol:N`

```
\cs_new:Nn \@@_undeclare_symbol:N
  {
    \cs_set_protected:Npn #1
```

```
      { \@@_error:nx {legacy-char-not-supported} { \token_to_str:N #1 } }
  }
```

If you have better ideas about what to do here, please mention.

```
\@@_undeclare_symbol:N \arrowvert
\@@_undeclare_symbol:N \Arrowvert
\@@_undeclare_symbol:N \bracevert
```

```
⟨/package⟩
```

Fin

The official end of the package:

```
⟨package⟩\endinput
```

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.