Experimental unicode mathematical typesetting: The unimath package

Will Robertson

2006/02/20 v0.01

Contents

1 Introduction

This document describes the unimath package, which is an *experimental* implementation of a macro to unicode glyph encoding for mathematical characters. Its intended use is for XeTeX, although it is conjectured that small effect needs to be spent to create a cross-format package that would also work with □.

As of XeTeX v. 0.995, maths characters can be accessed in unicode ranges. Now, a proper method must be invented for real unicode maths support. Before any code is written, I'm writing a specification in order to work out what is required. Fairly significant pieces of the NFSS may have to be re-written, and I'm a little unsure where to start.

2 Current NFSS methods

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

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

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

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

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

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

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

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

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

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

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

Delimiters, accents, and radicals are not dealt with yet.

3 Specification

In the ideal case, a single unicode font will contain all maths glyphs we need. Barbara Beeton's STIX table provides the mapping between unicode maths glyphs and macro names (all 3298 of them!). A single command

`\setmathsfont[`⟨*font features*⟩`]{`⟨*font name*⟩`}`

would implement this for every every symbol and alphabetic variant. That means `\alpha` to α, `\leq` to $\leq$, etc., `\mathcal{H}` to $\mathcal{H}$ and so on, all for unicode glyphs within a single font.

Furthermore, this package should deal well with unicode characters for maths input, as well. This includes using literal Greek letters in formulae, resolving to upright or italic depending on preference. This, and alphabetic variants via such commands as `\mathcal`, will be dealt with via XeTeX's 'last minute' font mapping features. (Or maybe not!)

Finally, maths versions must also be provided for. While I guess version selection in LaTeX will remain the same, the specification for choosing the version fonts will probably be an optional argument:

`\setmathsfont[`version=bold,⟨*font features*⟩`]{`⟨*font name*⟩`}`

All instances of 'maths' in command names will be aliased to 'math' for our American (or abbreviatory-minded) friends. Instances above of

`[`⟨*font features*⟩`]{`⟨*font name*⟩`}`

follow from my fontspec package, and therefore any additional ⟨*font features*⟩ specific to maths fonts will hook into fontspec's methods.

3.1 Dealing with real life

Let's face it; there will probably be few cases where a single unicode maths font suffices. The upcoming STIX font comes to mind as a notable exception. It will therefore be necessary to delegate specific unicode ranges of glyphs to separate fonts.

At the lowest level, it will probably be necessary on occasion to simply use just one or two glyphs from another font; either because they look better or they're simply unavailable in the default font in use. This doesn't really require anything that won't already exist; a command analogous to `\DeclareMathSymbol` that accepts unicode ⟨*slot*⟩ ranges.

More generally, it would be nice to be able to say

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

where ⟨*unicode range*⟩ is a comma-separated list of unicode slots and ranges such as `{27D0-27EB, 27FF, 295B-297F}`. Furthermore, preset names ranges could be used, such as `MiscMathSymbolsA`, with such ranges based on unicode chunks. The amount of optimisation required here to achieve acceptable performance has yet to be determined. Techniques such as saving out unicode subsets based on ⟨*unicode range*⟩ data to be `\input` in the next LaTeX run are certainly a possibility.

File I
The **unimath** package

This is the package.

```
\ProvidesPackage{unimath}
  [2006/02/20 v0.01 Unicode maths definitions]
```

Things we need:

```
\newcounter{um@fam}

%% Kees can der Laan's simplification of Van der Groot's loop:
\def\um@Loop#1\um@Pool{#1\um@Loop#1\um@Pool}
\def\um@Break#1\um@Pool{}

\long\def\um@FOR #1 = [#2:#3] #4{%
  {\csname#1\endcsname =#2\relax
   \um@Loop #4%
     \expandafter\advance\csname#1\endcsname\@ne
     \expandafter\ifnum\csname#1\endcsname>#3\relax
       \expandafter\um@Break
     \fi
   \um@Pool}}

\RequirePackage{fontspec}
```

Test: ∝ ∞∟∠∡∢ ∤∦ ∧ ∨ ∩ ∪ ∫ ∬ ∭

4 Trying to understand LaTeX

Here's LaTeX's definition of `\DeclareMathSymbol`. Let's try an make sense of it.

`\DeclareMathSymbol`

`#1`: Symbol, e.g., `\alpha` or 'a'
`#2`: Type, e.g., `\mathalpha`
`#3`: Math font name, e.g., `operators`
`#4`: Slot, e.g., `F1`

<*neveroutput>

```
\def\DeclareMathSymbol#1#2#3#4{%
```

First ensure the math font (e.g., `operators`) exists:

```
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
```

Convert the slot number to two hex digits stored in `\count\z@` and `\count\tw@`, respectively:

```
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
```

The symbol to be defined can be either a command (`\alpha`) or a character (`a`). Branch for the former:

```
      \if\relax\noexpand#1% is command?
     \edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
        \reserved@a
```

If the symbol command definition contains `\mathchar`, then we can provide the info that a previous symbol definition is being overwritten:

```
        \ifin@
          \expandafter\set@mathsymbol
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \@font@info{Redeclaring math symbol \string#1}%
```

Otherwise, throw an error if the command name is already taken by a non-symbol definition:

```
        \else
            \expandafter\ifx
            \csname\expandafter\@gobble\string#1\endcsname
            \relax
            \expandafter\set@mathsymbol
               \csname sym#3\endcsname#1#2%
               {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \else
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
```

And if the symbol input is a character:

```
      \else
        \expandafter\set@mathchar
          \csname sym#3\endcsname#1#2
          {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      \fi
    \endgroup
```

Everything previous was skipped if the maths font doesn't exist in the first place:

```
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi}
```

The final macros that actually define the maths symbol with TEX primitives. If the symbol definition is for a macro:

```
\def\set@mathsymbol#1#2#3#4{%
  \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
```

Or if it's for a character:

```
\def\set@mathchar#1#2#3#4{%
  \global\mathcode`#2="\mathchar@type#3\hexnumber@#1#4\relax}
```

</neveroutput>

Summary For symbols, something like:

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

For characters, something like:

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

5 This package

We need to both redefine `\DeclareMathSymbol` to deal with unicode slots, as well as `\DeclareSymbolFont` to deal with 8-bit family numbers.

5.1 Enlarging the number of maths families

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

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

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

```
⟨*□□□□□□□⟩
\newfam\mta\newfam\mtb\newfam\mtc\newfam\mtd
\newfam\mte\newfam\mtf\newfam\mtg\newfam\mth
\newfam\mti\newfam\mtj\newfam\mtk\newfam\mtl
\newfam\mtm\newfam\mtn\newfam\mto\newfam\mtp
\newfam\mtq\newfam\mtr\newfam\mts\newfam\mtt
\newfam\mtu
⟨/□□□□□□□⟩
```

\mtu: math fam 24 of 255.

5.2 \DeclareMathSymbol for unicode ranges

This is mostly an adaptation from LaTeX's definition.

```
\def\DeclareUnicodeMathSymbol#1#2#3#4{%
```

First ensure the math font (e.g., operators) exists:

```
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
```

No longer need to perform the obfuscated hex conversion, since \XeTeXextmathchar (and friends) has a more simplified input than TeX's \mathchar.

```
    \begingroup
```

The symbol to be defined can be either a command (\alpha) or a character (a). Branch for the former:

```
      \if\relax\noexpand#1% is command?
     \edef\reserved@a{\noexpand\in@{\string\XeTeXextmathchar}{\meaning#1}}%
        \reserved@a
```

If the symbol command definition contains \XeTeXextmathchar, then we can provide the info that a previous symbol definition is being overwritten:

```
        \ifin@
          \expandafter\set@xmathsymbol
             \csname sym#3\endcsname#1#2{#4}%
          \@font@info{Redeclaring math symbol \string#1}%
```

Otherwise, overwrite it if the symbol command definition contains plain old \mathchar:

```
        \else
      %\edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
          %\reserved@a
          %\ifin@
          %  \expandafter\set@xmathsymbol
          %     \csname sym#3\endcsname#1#2{#4}%
```

Otherwise, throw an error if the command name is already taken by a non-symbol definition:

```
          %\else
            %\expandafter\ifx
```

```
            %\csname\expandafter\@gobble\string#1\endcsname
            %\relax
              \expandafter\set@xmathsymbol
                 \csname sym#3\endcsname#1#2{#4}%
            %\else
          %  \@latex@error{Command `\string#1' already defined}\@eha
            %\fi
          %\fi
        \fi
```

And if the symbol input is a character:

```
      \else
        \expandafter\set@xmathchar
          \csname sym#3\endcsname#1#2{#4}%
      \fi
    \endgroup
```

Everything previous was skipped if the maths font doesn't exist in the first place:

```
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi}
```

The final macros that actually define the maths symbol with XeTeX primitives. If the symbol definition is for a macro:

```
\def\set@xmathsymbol#1#2#3#4{%
  \global\XeTeXextmathchardef#2"\mathchar@type#3 #1 #4\relax}
```

Or if it's for a character:

```
\def\set@xmathchar#1#2#3#4{%
  \global\XeTeXextmathcode`#2="\mathchar@type#3 #1 #4\relax}
```

[For later] or if it's for a character code:

```
\def\DeclareUnicodeMathCode#1#2#3#4{%
  \expandafter\set@xmathcode
    \csname sym#3\endcsname{#1}{#2}{#4}}
\def\set@xmathcode#1#2#3#4{%
  \global\XeTeXextmathcode#2="\mathchar@type#3 #1 #4\relax}
```

```
⟨*□□□□□□□⟩
\zf@fontspec{}{Cambria Math}
\DeclareSymbolFont{test}{EU1}{CambriaMath(0)}{m}{n}
\DeclareUnicodeMathSymbol{\infinity}{\mathord}{test}{"221E}
\DeclareUnicodeMathCode{65}{\mathalpha}{test}{119860}
⟨/□□□□□□□⟩
```

Test infinity: $A\infty$

5.3 User interface to `\DeclareSymbolFont`

`\setmathfont [#5]`: font features

#6: font name

Use fontspec to select a font to use.

```
\newcommand\setmathfont[2][]{%
  \zf@fontspec{#1}{#2}%
```

We need to hook into fontspec here to check if a family is loaded twice. This might be important if loading lots of individual glyphs.

```
\stepcounter{um@fam}%
\DeclareSymbolFont{um@fam\theum@fam}{EU1}{\zf@family}{\mddefault}{\updefault}%
```

Now when the list of unicode symbols is input, we want a suitable definition of its internal macro.

```
\def\unicode@math@symbol##1##2##3##4{%
  \DeclareUnicodeMathSymbol{##2}{##3}{um@fam\theum@fam}{##1}}%
```

And now we input every single maths char. See File II for the source to `stix-tex.tex`.

```
\um@text@input{um@fam\theum@fam}%
\input stix-tex.tex}
```

```
⟨*□□□□□□□□⟩
\setmathfont{Cambria Math}
\DeclareUnicodeMathSymbol{\testalef}{\mathord}{um@fam\theum@fam}{"2135}
⟨/□□□□□□□□⟩
```

Test aleph: ℵ
$\int Ax \stackrel{\text{def}}{=} \nabla \times \mathcal{Z}$

5.4 Setting up the ascii ranges

We want it to be convenient for users to actually type in maths. The ASCII Latin characters should be used for italic maths, and the text Greek characters should be used for upright/italic (depending on preference) Greek, if desired.

`\um@mathcode@offset` This is a wrapper macro to save space:

```
\newcommand\um@mathcode@offset[3]{%
    \DeclareUnicodeMathCode
      {\numexpr\the\@tempcnta+#2\relax}
      {\mathalpha}{#1}
      {\numexpr\the\@tempcnta+#3\relax}%
}
```

`\um@text@input` And here're the text input to maths output mappings, wrapped up in a macro.

```
\newcommand\um@text@input[1]{%
```

Latin alphabet, uppercase and lowercase respectively:

```
  \um@FOR @tempcnta = [0:25] {%
    \um@mathcode@offset{#1}{65}{119860}%
    \um@mathcode@offset{#1}{97}{119886}%
  }%
```

Filling a hole for 'h', which maps to U+210E: PLANCK CONSTANT instead of the expected U+1D455: MATHEMATICAL ITALIC SMALL H (which is not assigned):

```
  \DeclareUnicodeMathCode
    {104}{\mathalpha}{#1}{8462}%
```

Greek alphabet, uppercase (note the hole after U+03A1: GREEK CAPITAL LETTER RHO):

```
\um@FOR @tempcnta = [0:23] {%
  \DeclareUnicodeMathCode
    {\ifnum\@tempcnta>16
       \numexpr\the\@tempcnta+913\relax
     \else
       \numexpr\the\@tempcnta+913+1\relax
     \fi}
    {\mathalpha}{#1}
    {\numexpr\the\@tempcnta+120546\relax}%
```

And Greek lowercase:

```
    \um@mathcode@offset{#1}{945}{120572}%
  }%
}
```

Uppercase Latin	$ABCDEFGHIJKLMNOPQRSTUVWXYZ$
Lowercase Latin	$abcdefghijklmnopqrstuvwxyz$
Uppercase Greek	$A\mathrm{B}\Gamma\Delta\mathrm{E}\mathrm{Z}\mathrm{H}\Theta\mathrm{I}\mathrm{K}\Lambda\mathrm{M}\mathrm{N}\Xi\mathrm{O}\Pi\Sigma\mathrm{T}\Upsilon\Phi\mathrm{X}\Psi$
Lowercase Greek	$\alpha\beta\gamma\delta\varepsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi o\pi\rho\sigma\tau\upsilon\varphi\chi\psi$

File II

STIX table data extraction

The source for the TEX names for the very large number of mathematical glyphs are provided via Barbara Beeton's table file for the STIX project (`ams.org/STIX`). A version is located at `://..//////-.` but it's not currently up to date.

A single file is produced containing all 3298 symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols..

```
#!/bin/sh

cat stix-tbl.asc |
awk '
 BEGIN {OFS="|"}
 {if (usv != substr($0,2,5) )
   {if (substr($0,2,1) != " ")
    {usv = substr($0,2,5);
     texname = substr($0,84,25);
     type = substr($0,57,1);
     description = tolower(substr($0,233,350));
     {if (texname ~ /[\\]/)
       print usv, texname, type, description;}}}}' - |
awk -F"|" '
  (($3 != " ") && ($3 != "F") && ($3 != "D")) {
   print "\\unicode@math@symbol{" "\"" $1 "}{" $2 "}{" $3 "}{" $4 "}";
```

```
}' - |
sed -e ' s/{N}/{\\mathord}/   ' \
    -e ' s/{A}/{\\mathalpha}/ ' \
    -e ' s/{P}/{\\mathpunct}/ ' \
    -e ' s/{B}/{\\mathbin}/   ' \
    -e ' s/{R}/{\\mathrel}/   ' \
    -e ' s/{L}/{\\mathop}/    ' \
    -e ' s/{O}/{\\mathopen}/  ' \
    -e ' s/{C}/{\\mathclose}/ ' > stix-tex.tex
```