Experimental unicode mathematical typesetting: The xmaths package

Will Robertson

2006/02/20 v0.01

Contents

1 Introduction 1

2 Methods 1

3 Current NFSS methods 2

4 Specification 2
4.1 Dealing with real life 3

I The xmaths package 3

5 Testing 4

6 Trying to understand LaTeX 4

7 Symbol definitions 7

II STIX table data extraction 7

1 Introduction

This document describes the xmaths package, which is a proof-of-concept *experimental* implementation of a macro to unicode glyph encoding for mathematical glyphs. 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 □.

2 Methods

As of XeTeX almost v. 0.995, maths characters can be accessed in unicode ranges. Previously, to use unicode characters as mathematical glyphs, Bruno Voisin[1] has used the method of inserting text boxes into the maths mode with the following code:

```
\DeclareFontFamily{U}{appsym}{}
\DeclareFontShape{U}{appsym}{m}{n}{<-> "Apple Symbols"}{}
```

[1] Thread 'Use of Apple Symbols font in XeLaTeX' on the XeTeX mailing list, September 17, 2004.

```
\DeclareTextFontCommand{\applesym}{%
  \normalfont\fontencoding{U}\fontfamily{appsym}\selectfont}
\DeclareRobustCommand{\textapplehighplus}    {\applesym{\char"253C}}
\DeclareRobustCommand{\textappleblacksquare} {\applesym{\char"25A0}}
...
\DeclareRobustCommand{\applehighplus}    {\text{\textapplehighplus}}
\DeclareRobustCommand{\appleblacksquare} {\text{\textappleblacksquare}}
...
```

A similar approach has been taken here for simple glyph access in the past. 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.

3 Current NFSS methods

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

Maths symbol fonts `\DeclareSymbolFont{`‹ *name* ›`}`‹ *NFSS decl.* ›

Declares a named maths font such as `operators` from which symbols are defined with `\DeclareMathSymbol`.

Maths alphabet fonts `\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.

`\DeclareSymbolFontAlphabet{`‹ *cmd* ›`}{`‹ *name* ›`}`

Alternative (and optimisation) for `\DeclareMathAlphabet` if a single font is being used for both alphabetic characters (in their expected ASCII slots) 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 `\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 at this point in time.

4 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.

Finally, maths versions must also be provided for. To avoid having to re-write and re-design yet more LaTeX macros, let's leave it for now with a command such as `\setmathsversionfont{`‹ *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.

4.1 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.

File I
The **xmaths** package

This is the package.

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

We want amsmath's `\text` macro; not much else at this stage.

```
\RequirePackage{amstext}
```

Things we need:

```
\unless\ifdefined\@tempcntc
  \newcount\@tempcntc
\fi
```

Through fontspec, select a font to use for *all* subsequent maths glyphs. Remember: we're using unicode now!

```
\newcommand*\setmathfont[2][]{%
  \newfontfamily\mathfont[#1]{#2}}
\setmathfont{Cambria Math}
```

`\DeclareUnicodeMathSymbol`

#1: Unicode scalar value (hex.)
#2: math group
#3: command name
#4: description

This command is unlike `\DeclareMathSymbol` in that it doesn't take a mandatory font family to be rendered in. Instead, the maths font is decided by the user (this being unicode, all glyphs hopefully come from the same font); at present we assume each glyph comes from the same font, but this assumption will surely be relaxed in the future.

The description line is a bit of self-documenting that will surely come in handy.

```
\newcommand*\DeclareUnicodeMathSymbol[4]{%
  \DeclareRobustCommand#2{%
    \ensuremath{#3{\text{\mathfont\char"#1}}}}}
```

A question arises as to how multiple fonts should be incorporated. It is desired that one maths font supply all maths glyph required. For this reason I hesitate to provide a hook directly in `\DeclareUnicodeMathSymbol`.

</package>

5 Testing

After the macros this package uses are defined, a brief test verifies that they are working:

<*testing>

```
\DeclareUnicodeMathSymbol
  {1D6FD}
  {\test@italic@beta}
  {\mathalpha}
  {example italic beta}
```

</testing>

`\test@italic@beta`: β

6 Trying to understand LaTeX

<*neveroutput> 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`

```
\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@}}}
```

We need to both redefine `\DeclareMathSymbol` to deal with unicode slots, as well as `\DeclareSymbolFont` to deal with `yy` families.

To start with, we've got as many `\fam`s as before. So we want to redefine

```
\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
```

to

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

\mtu: math fam 24 of 255. <*testing>

```
\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

\let \unicode@math@symbol \DeclareUnicodeMathSymbol
\unicode@math@symbol{0221D}{\propto               }{\mathrel}{is proportional to}
\unicode@math@symbol{0221E}{\infty              }{\mathord}{infinity}
\unicode@math@symbol{0221F}{\rightangle                    }{\mathord}{right (90 degree) angle}
\unicode@math@symbol{02220}{\angle                }{\mathord}{angle}
\unicode@math@symbol{02221}{\measuredangle        }{\mathord}{angle-measured}
\unicode@math@symbol{02222}{\sphericalangle       }{\mathord}{angle-spherical}
\unicode@math@symbol{02223}{\mid                }{\mathrel}{/mid r:}
\unicode@math@symbol{02224}{\nmid                          }{\mathrel}{negated mid}
\unicode@math@symbol{02225}{\parallel           }{\mathrel}{parallel}
\unicode@math@symbol{02226}{\nparallel           }{\mathrel}{not parallel}
\unicode@math@symbol{02227}{\wedge                         }{\mathbin}{/wedge /land b: logical and}
\unicode@math@symbol{02228}{\vee                           }{\mathbin}{/vee /lor b: logical or}
\unicode@math@symbol{02229}{\cap                           }{\mathbin}{intersection}
\unicode@math@symbol{0222A}{\cup                           }{\mathbin}{union or logical sum}
\unicode@math@symbol{0222B}{\int              }{\mathop}{integral operator}
\unicode@math@symbol{0222C}{\iint               }{\mathop}{double integral operator}
\unicode@math@symbol{0222D}{\iiint              }{\mathop}{triple integral operator}
```

Test: ∝ ∞∟∠∡∢ ∤∦∦ ∧ ∨ ∩ ∪ ∫ ∬ ∭

</testing>

7 Symbol definitions

<*package>

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). The source file for the current version is:

http://www.ams.org/STIX/bnb/stix-tbl.ascii-2005-09-24

The (mostly awk, and fairly crude at that) script to generate the appropriate input files is presented in Part II.

```
%\input stix-tex-plane0.tex
%\input stix-tex-plane1.tex
```

File II

STIX table data extraction

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) )
   {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
```