Unicode mathematics in LaTeX: Advantages and challenges

Will Robertson

Abstract

Over the last few years I've been tinkering with Unicode mathematics in XeTeX. In this paper, I discuss Unicode mathematics in the context of LaTeX with the unicode-math package.

1 Introduction

XeTeX was the first widely-used Unicode extension to TeX. Several years ago Jonathan Kew added OpenType maths support to XeTeX [12] following Microsoft's addition of mathematics to the OpenType specification as they were preparing Microsoft Word 2007. Around that time I built a prototype implementation of a Unicode maths layer for LaTeX, called unicode-math, but with very few OpenType maths fonts available, and other projects consuming my time, the project lost momentum and I never managed to finish the package and upload it to CTAN.

That has now changed. In the leadup to the TUG 2010 conference I thoroughly revisited the code, re-writing most of it in the LaTeX3 programming environment 'expl3'. (A brief introduction to expl3 is given by Joseph Wright in [24].) Long-standing issues were resolved and support for LuaTeX was begun. It is now ready for greater distribution with TeX Live 2010.

In a happy twist of fate, the STIX fonts have recently been released and can be used by this package. Details to follow.

1.1 Outline

In Sections 2 and 3, I cover the origins and nature of Unicode mathematics, and what fonts are currently available which use it. In Sections 4 and 5, I address specific details of how to use Unicode maths in LaTeX, and comment on some challenges faced when doing so; in Section 6, I present my thoughts for the possible future of this work. Finally, in Sections 7 and 8 I discuss some technical aspects of the package and its development process.

2 What is Unicode maths?

Before we talk about Unicode maths, it is necessary to discuss the computer typesetting of mathematics from the very beginning, or at least since TeX was first created.

2.1 Origins

TeX was designed alongside a set of text and maths fonts, the 'Computer Modern' family. The original Computer Modern maths fonts were limited by restrictions of the time, consisting of three separate fonts with 128 glyphs each (for each design size).

Later, the well-known amsmath package provided a complement of glyphs designed to match Computer Modern; these extra maths fonts extended the repertoire of standard symbols that could be expected to be used by most mathematicians.

As well as the amsmath fonts, a (small) number of other maths fonts were also created for TeX systems, including Lucida[1] and MathTime Pro.[2] Each maths font developed generally contained a different set of glyphs, and as a consequence of this the developers who had to write the TeX support layer for each font generally had to start from scratch to implement the font encoding that bound symbols to glyph slot numbers. This tedious process is one factor in explaining the general dearth of maths fonts for TeX-based systems.

2.2 The newmath encoding

In the 1990s, the Math Font Group[3] was created to design an 8-bit math font encoding [7] to alleviate this problem of having to invent *ad hoc* encodings for each new maths font. This 'newmath' encoding was carefully designed to include as many maths symbols as possible, and each symbol was assigned a standard glyph slot. New fonts could just follow this system, and switching maths fonts would be as easy as switching text fonts since the newmath font encoding would automatically know where all the symbols were located.

The project produced a LaTeX implementation to support the 'newmath' encoding, but it was never completed for a variety of reasons. While XeLaTeX and LuaLaTeX are now available to access OpenType fonts that use Unicode maths, there may be still some interest in retaining (and finally releasing) newmath for future large-scale maths font encoding support — perhaps in order to support the STIX fonts and/or the proposed Latin Modern Math font in eight-bit LaTeX [13].

2.3 Unicode maths and the STIX fonts

After newmath the attention of the Math Font Group turned to Unicode, namely to answer the question: 'What maths symbols have actually been used and invented in published technical writing?' The particulars of this phase of history have been covered by Barbara Beeton's report of the project at the time [2]. To sum it up very briefly, members of this project,

[1] `http://tug.org/lucida`
[2] `http://www.pctex.com/mtpro2.html`
[3] `http://www.tug.org/twg/mfg/`

$\cos^2 \phi + \sin^2 \phi = -e^{i\pi}$

```
\setmathfont{Cambria Math}
$\cos^2 \varphi + \sin^2 \varphi = -e^{i\pi}$
```

Example 1: A minimal example of the unicode-math package.

now known as the STIX Project, gathered together a comprehensive list of symbols used in mathematics from as many sources as they could find and submitted these symbols to the Unicode consortium for addition to the Unicode specification. From their labours, we now have a formal description of thousands of glyphs that a maths font should contain and the particulars of how those glyphs should look and behave [4].

Having defined the symbols to appear in Unicode mathematics, a group of scientific publishers commissioned a new font family to be the reference implementation for the newly specified Unicode mathematics [3]. These STIX fonts were designed to blend with Times New Roman which was, I believe, (and perhaps still is) the most commonly used font in technical publishing.

2.4 OpenType maths and the modern era

But mathematics typesetting needs more than just glyphs. TeX itself uses a number of parameters built into the maths fonts it uses in order to place mathematics on the page in a form suitable for high-quality typesetting, such as where superscripts should be placed, whether delimiters should grow to encompass the material they surround, what alternative glyph to use for 'big operators' when in displaystyle rather than textstyle, and so on. The details have been elucidated and illustrated splendidly by Bogusław Jackowski [11]. A system to utilise Unicode maths must contain analogous information and use similar algorithms to produce acceptable results.

For this purpose, Microsoft extended the OpenType specification to include tables of structured information for mathematics typesetting, generalising and extending the original algorithms within TeX.

OpenType maths has been described in more detail by Ulrik Vieth both in the context of its historical development [21] and with a particular emphasis on how the OpenType parameters correspond to TeX's own [23]. He has also discussed some of the deficiencies of TeX's mathematics engine [20], most of which are now addressed with OpenType maths.

3 The unicode-math package

With Unicode mathematics able to encode the maths glyphs we need, and the OpenType font format able to store the required parameters to use the new maths fonts, the only thing missing is the typesetting engine to put the pieces together. Microsoft Word 2007 and 2010 contains one, and so does XeTeX and LuaTeX. It is important to recognise that a Unicode maths font is suitable for both Word and TeX-based systems, which I believe will aid the adoption of the Unicode maths approach.

The unicode-math package is an initial attempt to write a high-level interface to Unicode maths for LaTeX documents. After loading the package, users can write

```
\setmathfont{Cambria Math}
```

as shown in Example 1 to select Cambria Math or any other Unicode maths font.

Readers may be familiar with the fontspec package, which is a high-level interface for loading fonts (usually OpenType fonts) in XeLaTeX and now also LuaLaTeX [18]. Where fontspec is designed for loading fonts to change the text font of the document, unicode-math allows a similar interface to select the maths font.

Previous work in this area has been performed by Andrew Moschou with his mathspec package for XeLaTeX. With mathspec, a text font can be loaded to substitute the alphabetic symbols of the mathematics setup — say to use Minion Pro Italic for the Latin symbols and Porson for the Greek symbols — but all other maths symbols are left untouched. A similar process has been shown previously for maths fonts in eight-bit LaTeX by Thierry Bouche [5]. The unicode-math package, by contrast, is designed to use OpenType maths fonts that contain all glyphs and associated information necessary to replace the existing LaTeX maths setup.

The two packages are therefore designed for different purposes; use mathspec if most of your maths needs are fulfilled by a pre-existing maths package (such as mathpazo) but you would like your maths alphabets to be taken from the text font; alternatively, use unicode-math if you have an OpenType maths font that you would like to use for typesetting all aspects of the mathematics.

The unicode-math package almost completely replaces LaTeX's maths setup. Control sequences are provided to access every Unicode maths symbol, and literal input of all such characters in the source is also supported. Maths can be copied from another source (such as a web page or PDF document) and pasted directly into the LaTeX document and the

Cambria: $$\int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t$$

Asana: $$\int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t$$

STIX: $$\int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t$$

Euler: $$\int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t$$

```
\def\laplace{\hfill$\displaystyle
  \int_0^\infty \mathup e^{-st}f(t)\,\mathup dt
              $\\[1ex]}
Cambria: \setmathfont{Cambria Math} \laplace
Asana:   \setmathfont{Asana Math}   \laplace
STIX:    \setmathfont{XITS Math}    \laplace
Euler:   \setmathfont[math-style=upright]
                    {Neo Euler}    \laplace
```

Example 2: Available OpenType maths fonts at the time of writing.

content will be retained, albeit with some loss of its presentational aspects (most notably subscripts and superscripts).

With some minor exceptions, no changes to the mathematical document source should be necessary to be able to switch fonts using Unicode maths. ConTEXt has an analogous system [14], and we have discussed future plans for coordinating our efforts to be consistent where possible and reduce duplication of work between ConTEXt and LaTEX.

3.1 What fonts are available?

This is all well and good, but the system doesn't do much good if there are no fonts to take advantage of it. *Cambria Math*, by Tiro Typeworks,[4] was the first OpenType maths font released (through Ascender Corp.), commissioned originally for Microsoft Office 2007.

There are three open source OpenType maths fonts currently available, developed using the free font editor FontForge[5] to add the OpenType maths parameters. These fonts are:

- Apostolos Syropoulos's *Asana Math*,[6] which has its origins in the 'Pazo' fonts, which are a clone of Palatino with additional maths support;
- Khaled Hosny's *XITS Math*,[7] which is a fork of the STIX fonts to include preliminary OpenType maths layout information (XITS will eventually be deprecated by an official release of the STIX fonts with the same functionality); and,
- Khaled Hosny's *Neo Euler*,[8] which is a Unicode re-working [10] of Hermann Zapf and Donald Knuth's Euler font.

These four OpenType maths fonts are shown in Example 2, in which note the fact that the maths font can now change part-way through a document.

Of these, XITS Math and Asana Math will both be included in TEX Live 2010, and they can be loaded with (respectively)

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

without any font installation necessary.

Readers may be interested in Daniel Rhatigan's dissertation [17] on the history of and design comparisons between the Times-, Euler-, and Cambria-based maths fonts (recall that STIX is modelled after Times).

4 Advantages

The main advantage of using Unicode maths is that it becomes easy to switch between maths fonts. There are some more benefits than simply standardising the way maths fonts are loaded, however.

I suspect the most directly useful aspect of Unicode maths will be relieving (most of) the headache around finding *and using* a particular math font glyph. The STIX fonts are available as a fallback font for all symbols that are part of Unicode maths. After all, most maths symbols are geometrically abstract enough that they do not need to be directly matched with the text font.

4.1 Readable source

Unicode maths provides the ability for maths symbols and characters to be input in Unicode directly in the source file, as shown in Example 3. For example, you may input a literal 'α' directly into a source document rather than typing '`\alpha`'. A convenient way to achieve this input style is to use the auto-completion of text editors such as TeXShop and TEXworks, in which typing a unicode-math control sequence and then hitting the 'escape' key will produce the literal input character. Since the original control sequence still must be typed letter-by-letter, this technique doesn't improve input speed, but makes source documents far more readable and amenable

[4] http://www.tiro.com/projects.html
[5] http://fontforge.sourceforge.net/
[6] http://ctan.org/pkg/asana-math
[7] http://github.com/khaledhosny/xits-math
[8] http://github.com/khaledhosny/euler-otf

```
\[ 𝐄 = − ∇ 𝜙 − \frac{∂𝐀}{∂t} \]
\[ 𝐁 = ∇ × 𝐀 \]
\[ ∇ · 𝐃 = 𝜌 \]
\[ ∇ × 𝐇 − \frac{∂𝐃}{∂t} = 𝐉 \]
```

Example 3: Example of LaTeX source using Unicode math input with literal maths characters. Such input can be pasted from another source or typed with the aid of 'smart completion' in a text editor.

to casual editing. (Completion files for unicode-math will be distributed with the package.)

With direct Unicode input for symbols in a LaTeX document, only small changes to the regular syntax are required to approach the simplicity of Murray Sargent's 'nearly plain-text encoding of mathematics' [19], which can be used in Microsoft Office to achieve a TeX-like efficiency at writing maths while obtaining a WYSIWYG view of the document. (I personally still prefer the TeX way, however, since you can use macros and so on to retain consistency and give your symbols meaning.)

4.2 Mathematical alphabets

Unicode maths contains glyph slots to contain all styled alphabetic symbols used in mathematics, including bold, blackboard, script, etc., styles. The complete listing is shown in Example 4. Each style contains variations on some or all of the lowercase and uppercase Latin and Greek characters and Arabic numerals. The commands shown for switching alphabets force each particular shape, hence their explicit names such as `bfit` for 'bold italic'; general `\mathbf` and `\mathsf` commands are also provided to switch to the correct upright or italic shape depending on the context (see Section 4.3 and Example 6). Note that `\mathbf` is used to access bold symbols in both Latin *and* Greek; this is a great useability improvement over traditional LaTeX that requires either `\boldsymbol` or the bm package (or a specific maths font package) to access bold Greek letters.

As an aside, note that the command `\mathrm` from LaTeX is renamed in unicode-math to `\mathup` to emphasise the fact that it can be used for upright *Greek* symbols as well. The old name is still provided for backwards compatibility, of course.

As authors wish to use fonts with alphabet styles that are not currently present in Unicode, the system must be able to cope with the addition of new alphabets and new alphabet styles. The most relevant example here is the existence in the STIX fonts of a variety of these non-Unicode ranges, most notably the 'calligraphic' style in contradistinction to the

`\mathit`	𝑎𝑏𝑐	𝑋𝑌𝑍	𝛼𝜉𝜃	𝛹𝛯𝛺	
`\mathbfit`	𝒂𝒃𝒄	𝑿𝒀𝒁	𝜶𝝃𝜽	𝜳𝜩𝜴	
`\mathup`	abc	XYZ	αξθ	ΨΞΩ	123
`\mathbfup`	𝐚𝐛𝐜	𝐗𝐘𝐙	𝛂𝛏𝛉	𝚿𝚵𝛀	𝟏𝟐𝟑
`\mathbb`	𝕒𝕓𝕔	𝕏𝕐ℤ			𝟙𝟚𝟛
`\mathtt`	𝚊𝚋𝚌	𝚇𝚈𝚉			𝟷𝟸𝟹
`\mathsfit`	𝘢𝘣𝘤	𝘟𝘠𝘡			
`\mathbfsfit`	𝙖𝙗𝙘	𝙓𝙔𝙕	𝞪𝞽𝞱	𝞧𝞝𝞨	
`\mathsf`	𝖺𝖻𝖼	𝖷𝖸𝖹			𝟣𝟤𝟥
`\mathbfsfup`	𝗮𝗯𝗰	𝗫𝗬𝗭	𝝰𝞃𝝷	𝝭𝝣𝝮	𝟭𝟮𝟯
`\mathscr`	𝒶𝒷𝒸	𝒳𝒴𝒵			
`\mathbfscr`	𝓪𝓫𝓬	𝓧𝓨𝓩			
`\mathfrak`	𝔞𝔟𝔠	𝔛𝔜ℨ			
`\mathbffrak`	𝖆𝖇𝖈	𝖃𝖄𝖅			

Example 4: Mathematical alphabets in Unicode from the STIX fonts.

Script style: $\mathscr{ABCXYZ}$
Calligraphic: $\mathcal{ABCXYZ}$

```
\setmathfont
  [range=\mathscr]{XITS Math}
\setmathfont
  [range=\mathcal,StylisticSet=1]{XITS Math}
Script style: $\mathscr{ABCXYZ}$\\
Calligraphic: $\mathcal{ABCXYZ}$
```

Example 5: Accessing the non-Unicode calligraphic style in the STIX fonts.

'script' style that *is* included in Unicode. Example 5 shows the differences between these two styles; some mathematicians are used to using these two alphabet styles separately (with script letters accessed through the mathrsfs package, for example). Here, the XITS fonts have encoded the calligraphic shapes in the position of the script glyphs under the OpenType font feature `ss01`, which is accessed through fontspec font features as `StylisticSet=1`.

In time, I believe that the calligraphic alphabet will be incorporated into the Unicode standard, but until then the unicode-math package must be able to use it explicitly as an exceptional case. The system in unicode-math for creating new alphabet styles in this way is not completely generalised yet, but work in this area is planned for the future (including the addition of alphabets neither Latin nor Greek that might be also used in a mathematical context, such as Russian).

`math-style=`

`ISO`	a	z	B	X	α	β	$\mathit{\Gamma}$	$\mathit{\Xi}$
	$\boldsymbol{a}$	$\boldsymbol{z}$	$\boldsymbol{B}$	$\boldsymbol{X}$	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\mathit{\Gamma}}$	$\boldsymbol{\mathit{\Xi}}$
`TeX`	a	z	B	X	α	β	Γ	Ξ
	$\mathbf{a}$	$\mathbf{z}$	$\mathbf{B}$	$\mathbf{X}$	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\mathbf{\Gamma}$	$\mathbf{\Xi}$
`upright`	a	z	B	X	α	β	Γ	Ξ
	$\mathbf{a}$	$\mathbf{z}$	$\mathbf{B}$	$\mathbf{X}$	**α**	**β**	$\mathbf{\Gamma}$	$\mathbf{\Xi}$
`french`	a	z	B	X	α	β	Γ	Ξ
	$\mathbf{a}$	$\mathbf{z}$	$\mathbf{B}$	$\mathbf{X}$	**α**	**β**	$\mathbf{\Gamma}$	$\mathbf{\Xi}$

Example 6: Different output styles without changing the input source according to the `math-style` option.

4.3 Flexible output

The unicode-math package does not assume a one-to-one mapping between the Unicode characters in the source and the Unicode glyphs in the output. In fact, the design of the maths setup, by default, is such that there is no semantic difference between upright and italic letters in the input source; consistent output is achieved regardless of the style of the input source.

Claudio Beccari [1] has detailed the requirements of typesetting mathematics according to the ISO standard (ISO31/XI), which requirements differ in important ways from the typical output of LaTeX mathematics. More recently, Ulrik Vieth [22] discussed many of the details of mathematical typesetting in the context of mathematical physics; the features offered by the unicode-math package help to provide the flexibility required to achieve these ideas for any maths font available. (Packages to perform this in classical LaTeX, such as the isomath package, require maths fonts set up with a particular encoding.) As an example of the different approaches to mathematical typesetting, Example 6 shows how documents are able to be typeset per ISO standards or in a more classical TeX-like format without changing the source text of the mathematics. Similarly, the output style of bold characters can also be adjusted.

As the package can load fonts for maths glyphs dynamically, multiple fonts and multiple styles can be used between various characters or families or alphabets of characters. Example 7 shows an example in which the maths was typed 'as usual', but different glyphs and glyph ranges were assigned fonts with different colours (grayscaled for *TUGboat*). This particular example may not be very practical, but it illustrates that the system is flexible enough to accommodate a wide range of effects. Even single characters within an alphabet may be chosen, such as in Example 8 where the 'π' symbol alone is chosen to be typeset upright.

$$F(s)=\mathscr{L}\biggl\{f(t)\biggr\}=\int_0^\infty \mathrm{e}^{-st}f(t)\,\mathrm{d}t$$

```
\setmathfont{Cambria Math}
\def\SET#1{\setmathfont[#1]{Cambria Math}}
\SET{range={\mathop,\mathscr}, Colour=red}
\SET{range={\equal}, Colour=00BB22}
\SET{range={\mathopen,\mathclose}, Colour=blue}

\[ F(s)=\mathscr{L}\,\biggl\{f(t)\biggr\}
   = \int_0^\infty \mathup e^{-st}f(t)
   \, \mathup d t \]
```

Example 7: Hooks make it possible to use a variety of fonts or styles — in this case, colours — for different maths characters or families/alphabets of maths characters.

1: $\{\alpha,\dots,\pi,\dots,\omega\}$

2: $\{\alpha,\dots,\mathrm{\pi},\dots,\omega\}$

```
\setmathfont{Cambria Math}
1: $\{\alpha,\dots,\pi,\dots,\omega\}$

\setmathfont
  [range={"1D70B},math-style=upright]
  {Cambria Math}
2: $\{\alpha,\dots,\pi,\dots,\omega\}$
```

Example 8: An example of selecting a different font for a single alphabetic glyph. The glyph slot `"1D70B` corresponds to the pi symbol in the mathematical Greek Unicode range.

5 Challenges

The biggest problem I can see with the advent of Unicode maths, besides more fonts — I believe they'll slowly start to appear now that there are tools and programs to support them — is educating people into using them well.

5.1 Using the correct characters

Example 9 shows five different maths glyphs that are all triangular, while Example 10 shows the eight different slash-like glyphs; four in each direction. Consider whether it's clear, only from the description in the tables, which ones to use in different contexts.

Without careful documentation and good education, it may be hard for users to know which is the 'correct' glyph to use in many occasions. The markup in TeX and LaTeX has generally steered towards presentational aspects. But, as an example, with five different choices for which triangle to choose,

Slot	Command	Glyph	Class
U+25B5	\vartriangle	$x \vartriangle y$	relation
U+25B3	\bigtriangleup	$x \bigtriangleup y$	binary
U+25B3	\triangle	$x\triangle y$	ordinary
U+2206	\increment	$x\Delta y$	ordinary
U+0394	\mathup\Delta	$x\mathrm{\Delta} y$	ordinary

Example 9: Four triangular glyphs (from the STIX fonts) with five different uses but all with similar shapes.

Slot	Name	Glyph	Command
U+002F	Solidus	x/y	\slash
U+2044	Fraction slash	$x⁄y$	\fracslash
U+2215	Division slash	$x ∕ y$	\divslash
U+29F8	Big solidus	$x ⧸ y$	\xsol
U+005C	Reverse solidus	$x\backslash y$	\backslash
U+2216	Set minus	$x \smallsetminus y$	\smallsetminus
U+29F5	Reverse solidus operator	$x \setminus y$	\setminus
U+29F9	Big reverse solidus	$x ⧹ y$	\xbsol

Example 10: A multitude of symbols for different purposes. Glyphs taken from the STIX fonts.

different authors may inadvertently choose different (but visually similar) glyphs for the same purpose in their mathematics. Furthermore, font designers are going to need to carefully design these glyphs to be consistent with the STIX fonts, which have been designed as 'reference material' against which all aspects of Unicode maths can be compared.

My feelings are that new tools will be needed to write LaTeX mathematics more semantically (which I will talk about later in Section 6.2). But such tools will need to be specific for each scientific field that uses different notation. This is an open problem.

5.2 LaTeX vs MathML

Mathematics represented in TeX and MathML are really quite separate beasts, although TeX can (perhaps obviously) be used as an engine to typeset MathML [9, 16]. While LaTeX input is designed to be hand-written and has visual output as the primary goal, MathML is a machine-friendly (human-unfriendly!) language to represent mathematics far more unambiguously and verbosely. There is not much overlap between how LaTeX looks at Unicode maths and how MathML is used, although packages such as stex ('semantic TeX') wed the ideas of 'Content MathML' to LaTeX (I briefly discuss semantic input of maths later in Section 6.2).

MathML and LaTeX often use different names for the symbols in Unicode maths. For example, the infinity symbol ∞ (U+221E) is `\infty` in TeX and `∞` in MathML. There are very few naming conflicts, but do bear in mind that the W3C names for maths symbols can occasionally be incompatible with the names used in unicode-math. As an example, consider the two 'set minus' characters in Example 10, which inherit their names from Plain TeX and the amssymb package, respectively. U+2216 is `\smallsetminus` and U+29F5 is `\setminus`. However, MathML does it differently due to a historical accident: U+2216 is referred to by either `∖` or `∖` or a number of other synonyms; U+29F5 is as-yet unnamed [6]. The general mismatch between these two Unicode maths glyph naming schemes might make it difficult to move between MathML and LaTeX if one is used to writing symbol names in MathML and starts writing LaTeX mathematics, or vice versa.

Despite the semantic advantages of Content MathML, however, it is still not supposed to be used as an input language for mathematics; MathML and the language of LaTeX maths are simply designed for different things. Therefore, in practise I don't believe there will be any problems resulting from the differences in glyph naming between the two.

6 Thoughts for the future

Unicode is clearly here to stay, and we are entering a time where, for the first time, fonts for mathematics can be built with standard OpenType font tools, and they can be used in a variety of cross-platform environments — from XeTeX and LuaTeX to Microsoft Office to MathML on the web. I hope and believe that this will herald the more profuse production of maths fonts than we've seen in the past.

The unicode-math package is only the first step for modernising the maths support in LaTeX. I consider the future of maths in LaTeX to be supported by three main pillars of functionality: font support; structural improvements to the input language supported by advanced layout algorithms; and 'semantic'-style input. Font support is broadly covered by the unicode-math package, which leaves two topics to discuss below.

6.1 Layout of mathematics

For 'structural improvements to the input language', I really mean improvements for writing the kinds of things that the amsmath package has typically been used for; namely, it provides high(er)-level tools to describe the layout of mathematical expressions. While the amsmath package has been extremely popular for many years, it is not perfect. The best candidate to extend it is the breqn package [8], which is now maintained by Morten Høgholm. (breqn is completely compatible with amsmath, thus transitioning from one to the other is very easy.)

The breqn package's primary features are to simplify the input necessary over what is required for more complex structures in amsmath; the way that it does this is by incorporating complex algorithms to perform automatic breaking of mathematics over lines. This has long been regarded as impossible to perform correctly all of the time — and while no-one is arguing that breqn is always correct, it *usually* is. When it is not, the task is done manually as is presently the case anyway.

6.2 Semantic input of mathematics

If you look over the list of 'TeX names' used by unicode-math for the Unicode maths symbols, it is clear that the names chosen have often been chosen to be descriptive rather than semantic. For example, `\doteq`, `\bigwedge`, `\smwhtsquare` ('small white square'), and so on. This is not unique to unicode-math; this follows the general naming scheme for LaTeX math font symbols where the name of a symbol shouldn't be too specific for one general use.

However, when there are clear semantics for symbols it is generally more useful to use a semantic input style for that piece of mathematics. For example, with '$a \to b$' (and this is from regular LaTeX), it is clearly more sensible to write `$a \to b$` rather than `$a \rightarrow b$` when we're writing what would be said aloud as 'for/from a to b'. Similarly, (and more hypothetically), writing `\intersection` and `\union` is probably better than `\cap` and `\cup`, respectively, in that their meaning in the former is immediately obvious from the source document.

I am aware of two macro packages that attempt to provide a general semantic input style for mathematics in LaTeX: the cool ('content-oriented LaTeX') package and the aforementioned stex package. As an argument for using them, and by way of comparison between them, consider writing an integral

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x.$$

In pure LaTeX, we must write this in a purely presentational manner, explicitly writing subscripts and superscripts on the integral symbol, and inserting a manual space and upright font switch to write the '$\mathrm{d}x$'. The LaTeX source is:

```
\int_{x_0}^{x_1} f(x) \,\mathrm{d}x
```

By contrast, consider what this mathematical statement actually *means*: a direct integral of a function $f(x)$ from x_0 to x_1. There is more detail in the typesetting of the statement than in the mathematics of it! In the cool package, this is written

```
\Integral{f(x)}{x,x_0,x_1}
```

In stex (the package name is cmathml for just the mathematics component of stex), it is

```
\CintLimits{x}{x_0}{x_1}{f(x)}
```

Another pertinent example is for representing derivatives. To write $\frac{\mathrm{d}f}{\mathrm{d}x}$ in LaTeX requires using an explicit fraction with more markup for the upright 'd': `\frac{\mathrm d f}{\mathrm d x}`. The packages cool and cmathml respectively use `\D{f}{x}` and `\Cddiff{x}{f}`. For multiple derivatives the benefits are even more obvious; `\D{f(x,y)}{x,y,z}` or `\Cpartialdiff{3}{x,y,z}{f(x,y)}` instead of

```
\frac{\mathrm d^3}
  {\mathrm d x\,\mathrm d y\,\mathrm d z}
    f(x,y)
```

to obtain

$$\frac{\mathrm{d}^3}{\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z} f(x,y).$$

I haven't spent much time with the more recent cmathml package, but my experiences with writing mathematics using the cool package have been very positive. The additional semantics using this notation isn't helpful from an academic sense of adding more meaning to the document source (although that's also a good thing). The real benefit is that it makes these maths constructions easier to type.

Based on the work of cool and cmathml, I believe that standardising some of the ideas for semantic markup of mathematics will benefit document authors (many of whom, after all, use similar macros in their own texts, albeit in an *ad hoc* way) and help in the automatic translation of mathematics written in LaTeX to other markup systems like MathML, and vice versa.

The unicode-math package will not address such ideas directly; it is purely a system to use mathematics with OpenType fonts. But as this package becomes more mature and can be used as a solid foundation for Unicode mathematics, then it will be time to start thinking seriously about formalising ideas behind 'semantic mathematics'.

7 A technical note on alphabet remapping

In TeX and LaTeX, using different fonts for alphabets such as `\mathbf` and `\mathscr` involved setting the 'math code' of the ASCII Latin letters to 'variable' and simply switching the math font. This meant that, internally, \mathbf and friends simply resulted in a font switch, which is efficient and straightforward (although sometimes tricky to juggle with only sixteen maths fonts in eight-bit TeX).

Unfortunately, things are not so simple with unicode-math. Within Unicode, each alphabet style (for bold and script and so on) is encoded in a distinct Unicode range. For example, the italic mathematical 'w' accessed with `$w$` is symbol U+1D464. The bold upright mathematical '**w**' (`$\mathbf{w}$`) is U+1D430. In order to switch from one to the other using a command like \mathbf requires that the mathcodes for all affected letters must change locally inside its argument. This isn't *too* inefficient, since assigning \mathcodes is pretty fast, but it's not particularly elegant. It would be easier not to support the `\mathbf{...}`-style syntax at all and instead refer to such symbols with macros, such as \mbfw for the bold '**w**'. But we must support the switching-style commands for backwards compatibility.

There are some alternatives to doing things this way, but they all have trade-offs. The simplest solution would be to use XeTeX's input mapping feature that allows letters in the source to be transformed into other letters before typesetting. Thus, the 'variable math code' approach as used in LaTeX could be used for Unicode maths alphabets. However, this system is less flexible (features such as Example 8 would be more difficult) and an alternative approach would be required for LuaTeX.

Another approach would be to use (math-)active characters for all maths symbols. In this approach, ASCII letters such as 'a' would be *active* in maths mode and expand (as if it were a macro) to a construction such as

```
\csname mathchar_\mathstyle_a \endcsname
```

where \mathstyle would resolve to 'up' or 'bf' (etc.) depending on the context and \mathchar_up_a and \mathchar_bf_a (etc.) would be defined accordingly with the appropriate Unicode maths glyph. This is more efficient for font switching but less efficient (and perhaps more fragile) when symbol remapping is not taking place. Using active characters is the technique used by the breqn package to do its automatic line breaking of mathematics, and extending that system for unicode-math would be quite logical. One way or the other, breqn compatibility is planned for unicode-math in the future.

Using LuaTeX for alphabet remapping (which is how ConTeXt's implementation works) is probably the best way to tackle this problem, but while unicode-math is written for XeLaTeX as well (and it will continue to be for the immediate future) we must stick with TeX-based programming solutions.

8 Experiences writing the package

Some aspects of writing the unicode-math package have been more organised than other LaTeX code I've written. As more and more LaTeX code is being developed publicly in source code repositories such as GitHub, BitBucket, and others, I would like to discuss quickly some of the infrastructure of this package's development.

8.1 Cross-platform development

The fontspec and unicode-math packages are both now targeted towards running on both XeLaTeX and LuaLaTeX. Despite small differences in how certain things are done, this generally works well for both.

Most of the code in unicode-math and fontspec has been written (or re-written) with the expl3 programming interface. This has proven to be a very useable interface to numerous high-level programming constructs; expl3 allows more complex ideas to be easily realisable within the limitations of TeX macro programming.

In this shared XeLaTeX/LuaLaTeX environment, Lua code is restricted to a minimum in order to minimise separate code branches for each engine, as much as possible. Functions in Lua are 'hidden' inside TeX macros, so all of the main programming in unicode-math resembles plain old TeX programming. I personally find this much easier to read than mixing Lua code and TeX macro code together.

8.2 Version control

I use the Git version control system, and for some time I've been using GitHub repositories[9] for most of my public code (LaTeX and otherwise). GitHub provides free accounts for developers of open source software, and their site includes a very functional bug tracker/issue reporter per project. Tracking bugs over several years is certainly no fun with email.

I've had many people contribute code and provide feedback through the GitHub project page, and I highly recommend such a public development environment for all package developers.

The main advantages to these systems, for me, are the ease with which others can collaborate on code or documentation writing and with how issues can be resolved. Having a public code repository also allows users to access historical versions of the code, which can be important for those on legacy systems who cannot upgrade their distributions but

[9] `http://github.com/wspr`

need old versions of some packages that are no longer available on CTAN in their original form.

8.3 Test suite

Inspired by the test suite available for the LaTeX 2ε and LaTeX3 codebase, I implemented a test suite for unicode-math based on a 'visual diff' between the output of each test file compared to a known 'reference' output that had been compiled some time beforehand. At the time of writing there are 128 tests in total, the output of which are included all together as a separate documentation file in the unicode-math distribution as a rather complete set of minimal examples showing various aspects of the package and its features.

In hindsight, using an image-based test was perhaps not the best way to approach regression testing with LaTeX. The test scripts use ImageMagick's `compare` tool, which first discretises the PDF output to a bitmap and compares the pixels between the output and the reference. Unfortunately, due to rounding errors this technique is prone to the occasional 'false negative' in which the bitmap output of a test might change by a few (very small) pixels but there's nothing wrong with the output of the test itself.

An unintended benefit of this technique, on the other hand, is that any changes in the fonts I am using are immediately detected. This makes the unicode-math test suite a useful way for me to see what's actually changing when the fonts that I use in the test suite are updated.

However, the visual diff is slow and, as mentioned above, not always accurate (although it is repeatable, at least). A more reliable and efficient approach might use `\showbox` and `\tracingoutput` to create a detailed (textual) log of the TeX boxes generated in the output. This 'box log' can be checked for differences against a normalised result produced by a prior test run, and this is the technique used successfully by the LaTeX 2ε and LaTeX3 test suites [15].

Regardless of whether it's the most efficient or the most reliable technique to use, the test suite is still essential for catching bugs before I release new versions of the package to the public. I can change code without fear of unexpected problems in the behaviour of the package.

9 Conclusion

It's early days for Unicode mathematics. The work here shows the first steps for using OpenType fonts in LaTeX for mathematics; while we have done little to change the style of the input, there are still clear advantages in more consistent commands and Unicode input in the source. Being able to support new fonts without any extra LaTeX support files will hopefully spur new efforts in building new maths fonts. I am looking forward to seeing what happens in the future.

I would like to thank the TeX Users Group for supporting my attendance of the TUG2010 conference, and extend further thanks towards some people without whom the unicode-math package couldn't exist: Barbara Beeton for all her work with the STIX project and for her thoughtful correspondence; members of the LaTeX3 project for, well, everything; Khaled Hosny and others for their work with luaotfload and LuaLaTeX in general; all of those who have collaborated with, enthusiastically commented on, and especially tested the code; Jonathan Kew for XeTeX; and Taco Hoekwater et al. for LuaTeX.

References

[1] Claudio Beccari. Typesetting mathematics for science and technology according to ISO31/XI. *TUGboat*, 18(1):39–48, 1997. `http://tug.org/TUGboat/tb18-1/tb54becc.pdf`.

[2] Barbara Beeton. Unicode and math, a combination whose time has come — Finally! *TUGboat*, 21(3):176–185, September 2000. `http://tug.org/TUGboat/tb21-3/tb68beet.pdf`.

[3] Barbara Beeton. The STIX project — From Unicode to fonts. *TUGboat*, 28(3), 2007. `http://tug.org/TUGboat/tb28-3/tb90beet.pdf`.

[4] Barbara Beeton, Asmus Freytag, and Murray Sargent III. Unicode support for mathematics. Unicode Technical Note 25 Version 9, Unicode, Inc., 2008. `http://www.unicode.org/reports/tr25`.

[5] Thierry Bouche. Diversity in math fonts. *TUGboat*, 19(2):120–134, 1998. `http://tug.org/TUGboat/tb19-2/tb59bouc.pdf`.

[6] David Carlisle and Patrick Ion. XML entity definitions for characters. Technical Report W3C Working Draft 21, W3C, 2008. `http://www.w3.org/TR/xml-entity-names/`.

[7] Matthias Clasen and Ulrik Vieth. Towards a new math font encoding for (LA)TeX. *Cahiers GUTenberg*, 28–29, 1998. `http://cahiers.gutenberg.eu.org/cg-bin/article/CG_1998___28-29_94_0.pdf`.

[8] Michael Downes. Breaking equations. *TUGboat*, 18(3):182–194, 1997. `http://tug.org/TUGboat/tb18-3/tb56down.pdf`.

[9] Hans Hagen. MathML [in ConTeXt]. *MAPS*, 27:66–119, 2002. `http://www.ntg.nl/maps/27/18.pdf`.

[10] Hans Hagen, Taco Hoekwater, and Volker R.W. Schaa. Reshaping Euler: A collaboration with Hermann Zapf. *TUGboat*, 29(2):283–287, 2008. http://tug.org/TUGboat/tb29-2/tb92hagen-euler.pdf.

[11] Bogusław Jackowski. Appendix G illuminated. *TUGboat*, 27(1):83–90, 2006. http://tug.org/TUGboat/tb27-1/tb86jackowski.pdf.

[12] Jonathan Kew. XeTeX Live. *TUGboat*, 29(1):146–150, 2008. http://tug.org/TUGboat/tb29-1/tb91kew.pdf.

[13] Johannes Küster. Newmath and Unicode. *Proceedings of EuroTeX 2005*, 2005. http://tug.org/TUGboat/tb27-0/kuster.pdf.

[14] Aditya Mahajan. Integrating Unicode and OpenType math in ConTeXt. *TUGboat*, 30(2), 2009. http://tug.org/TUGboat/tb30-2/tb95mahajan-cmath.pdf.

[15] Frank Mittelbach. A regression test suite for LaTeX 2_ε. *TUGboat*, 18(4):309–311, December 1997. http://tug.org/TUGboat/tb18-4/tb57mitt.pdf.

[16] Luca Padovani. MathML formatting with TeX rules, TeX fonts, and TeX quality. *TUGboat*, 24(1):53–61, 2003. http://tug.org/TUGboat/tb24-1/padovani.pdf.

[17] Daniel Rhatigan. Three typefaces for mathematics. Master's thesis, University of Reading, 2007. http://www.typeculture.com/academic_resource/articles_essays/pdfs/tc_article_47.pdf.

[18] Will Robertson. Advanced font features with XeTeX — the fontspec package. *TUGboat*, 26(3):215–223, 2005. http://tug.org/TUGboat/tb26-3/tb84robertson.pdf.

[19] Murray Sargent III. Unicode nearly plain-text encoding of mathematics. Unicode technical note 28, Unicode, Inc., 2006. http://www.unicode.org/notes/tn28/.

[20] Ulrik Vieth. Math typesetting in TeX: The good, the bad, the ugly. *MAPS*, 26:207–216, 2001. http://www.ntg.nl/maps/26/27.pdf.

[21] Ulrik Vieth. Do we need a 'Cork' math font encoding? *TUGboat*, 29(3):426–434, 2008. http://tug.org/TUGboat/tb29-3/tb93vieth.pdf.

[22] Ulrik Vieth. Experiences typesetting mathematical physics. In *Proceedings of EuroTeX*, 2009. http://tug.org/TUGboat/tb30-3/tb96vieth.pdf.

[23] Ulrik Vieth. OpenType math illuminated. *TUGboat*, 30(1):22–31, 2009. http://tug.org/TUGboat/tb30-1/tb94vieth.pdf.

[24] Joseph Wright. LaTeX3 programming: External perspectives. *TUGboat*, 30(1):107–109, 2009. http://tug.org/TUGboat/tb30-1/tb94wright-latex3.pdf.

◇ Will Robertson
School of Mechanical Engineering
University of Adelaide, SA, Australia
will dot robertson (at) latex-project dot org