fonts

An Extended Maths Font Set For Processing MathML

Taco Hoekwater
Bittext VOF
Singel 191
3311 PD Dordrecht
The Netherlands
bittext@cybercomm.nl

abstract
Last years autumn, work started on a new set of mathematical fonts that are intended to cover the full range of characters included in MathML as well as those included in the proposals for mathematical extensions in the next version of Unicode.
This paper presents the first result of that work: A new Times-compatible maths font set consisting of about 1500 symbols and a few alphabets; along with a collection of TEX macros to use them.
These fonts are donated to the public domain by Kluwer Academic Publishers and are available in both MetaFont source and Adobe Type 1 formats.

keywords
mathematical typesetting, fonts, PostScript

Introduction

Typesetting mathematical material is and always has been a complicated matter. Not only does it require a rather specialized typesetting engine, but formulas also need some quite peculiar fonts. Even if the typesetting engine takes care of most of the complications involved with horizontal spacing, resizing of fonts and bouncy baselines, you still need fonts that contain the rather strange glyphs that are needed to display formulas beautifully.

Mathematical fonts have been a bit of a problem child in the past. Although there are quite a few font solutions around that can take care of relatively simple 'high-school' equations, there are a number of scientific fields that have had to improvise to convey the meaning of their notations simply because the needed glyphs were not available.

For other fields of science, the situation is marginally better: there are fonts available that contain the 'correct' characters, but only in a design that is visually incompatible with the normal text font of the article to be written.

The current work tries to cover all of the known fields that need special characters, so that there will be at least one *full* solution, compatible with Adobe's Times-Roman font.

Sources of information

Over the past few years, two groups have been working very hard to improve the situation, and their work gives valuable information regarding the needed glyphs.

The first group is the MathML working group, the group that is responsible for the coding of mathematics for the World Wide Web. This group has created a DTD fragment that allows both layout-based and content-based markup to be used for on-line mathematics [1]. The entity set used by this DTD fragment is essentially the same as an older ISO technical report, ISO 9573-13 [2].

The second group (STIX) is essentially a collaboration between scientists and publishers. This group [3] has tried to compile a comprehensive table of actually used/wanted mathematical glyphs, to be submitted to the Unicode organization [4, 5]. Their current table of glyphs lists about 2000 separate characters (including cyrillic, phonetics and chemistry).

The combined efforts of these two groups have resulted in about 1500 different glyphs for mathematical typesetting as well as the specification of about 20 alphabets that need at least all lowercase and uppercase latin characters.

A third interesting source of information are the specialized mathematical fonts that are already out there: fonts such as the StMary's Road Symbols and Waldi's Symbols, and all of the proprietary fonts that come with scientific software such as Mathematica and Scientific Workplace.

Getting started

In the autumn of 1998, Kluwer Academic Publishers in Dordrecht (The Netherlands) decided to adopt the MathML coding method for their new SGML-based production environment. Since the publisher currently still relies on paper for most of its income, a solution had to be found for the lacking characters & fonts. The design and implementation of a Times-compatible version of these fonts in Adobe Type 1 format has been commissioned to Bittext VOF, on the condition that the resulting fonts will be donated to the Public Domain.

The information sources that were to be used as a reference for the fonts to be designed were rather disorganized. The first job was trying to combine all possible information sources into one large collection of glyphs + descriptions that could then be used as a lookup–table, eliminating the need to ask questions to the two groups mentioned above for every second glyph.

Because of the extensive work already done by the STIX group, the part of merging the descriptions was fairly simple. But unfortunately the demonstration glyphs that were used by the STIX HTML table(s) were not really usable for the creation of high-quality fonts. The typical example glyph looked like the ones in Figure 1.

Figure 1 Original glyph description images

So, the next thing to be done was to find as many reasonably good renderings in already existing fonts as possible. This was done by generating bitmaps of all fonts that were known to me (using a few TEX macros and GhostScript). These bitmaps also show the boundingbox of the character as well as an 'em-square'. A few typical results of this method can be seen in Figure 2.

Figure 2 New glyph description images

Of course this method did not work for all possible glyphs, but at least it helped in deciding how to design the unknown glyphs (of which most were, of course, related to an already existing character).

After the bitmap-generation was done, the HTML lookup table was reorganized into blocks of glyphs that all had the same mathematical logical interpretation: one HTML file for arrow relations, one for binary relations, one for large operators, etc. This reorganization was necessary because one wants to design all similar glyphs at the same time to improve consistency. At that time, the original STIX table was organized according to the (tentative) Unicode positions, without any logical structure.

A piece of the new tables can be seen Figure 3. With all this work in place, it was possible to start the actual implementation.

	xme/179				
E23E		R	cudarrl		left, curved, down arrow
E23F		R	cudarrr		right, curved, down arrow
E240		R	harrcir		left and right arrow with a circle
E241	stmary/95	R	roarr	S\rightarrowtriangle	right open arrow
E242	stmary/94	R	loarr	S\leftarrowtriangle	left open arrow
E243	stmary/93	R	hoarr	S\leftrightarrowtria*	horizontal open arrow
E244	msam/32	R	zigrarr		right zig–zag arrow
E245		R	nvhArr *		not, vert, left and right double arrow
E246		R	nvrArr		not, vert, right double arrow
E247		R	nvlArr		not, vert, left double arrow
E248			angzarr		right angle with down zig–zag arrow
E249	qua6/126	R	curarrm		curved right arrow with minus
E24A	qua6/96	R	cularrp		curved left arrow with plus
E24B		R	ufisht		up fish tail
E24C		R	dfisht		down fish tail
E24D		R	rarrsim		right arrow, similar

Figure 3

Creating the fonts

A choice had to be made. The requested output format for the fonts was Adobe Type 1 [6], so two different workflows were possible:

1. Use an interactive editor such as Fontographer or FontLab to create the requested PFB files directly.
2. Use MetaFont and convert the result using Richard Kinch's MetaFog (see [7] and [8] for details of this conversion process and the programs involved).

Using an interactive editor has three big advantages:

- It generally works faster than programming in MetaFont;
- it is very simple to 'steal' shapes from existing fonts;
- and it is possible to edit all glyphs in one font file at the same time (MetaFont is limited to 256 characters per font).

The drawback of using an editor is the output, which is essentially write-only. It's not possible to re-use the created glyphs to create for example a sans–serif version.

MetaFont's advantages are

- its programmability: using dedicated macros in MetaFont it is easier to remain consistent.
- The MetaFont code is re-usable simply by changing the underlying macros and parameters.

MetaFont turned out to be the winner, not because this particular job really needs extensive programming facilities, but because it seems very likely that we will want to redo the maths font set for different font families in due time. This is a job which will be much easier using existing MetaFont code than it would scratch in an interactive program. Because in the latter case we would be obliged to restart from scratch again.

Planned for the future are at least a 'bold math' version of the Times compatible fonts, a sans–serif version to be used with e.g. Helvetica and Frutiger, and a Computer-Modern compatible implementation.

The actual implementation

In the actual implementation, there are conceptually only a few fonts: one very large font containing all of the 1500 special glyphs and a few separate fonts for the different alphabets.

The symbolic font

As said earlier, MetaFont cannot really handle fonts that have more than 256 characters in them. For this reason, the source has been split into sub-font chunks that are somewhere between 200 and 250 glyphs each (in implementation order):

- Arrow relations and other arrow symbols
- Ordinary symbols such as harpoons and angles
- Binary relations
- Negated binary relations
- Binary operators
- Delimiters and other extensibles
- Dingbats and various ordinary symbols
- Large operators
- Accents

These groups are now in various states of completion, from almost completely finished (the arrows) to rough pre-implementations (the large operators). Figure 4 gives an idea of what the resulting fonts look like (this is page two of the font table for the binary relations, ranging from character 81 to 174)

All of the sub-fonts share a common MetaFont base file, and almost all character definitions are defined in a very indirect manner. For example, here is the code for character 84: 'geqslantdot':

```
beginchar(geqslantdot_slot,42hu,27vu,6vu);
geqslantdot;
endchar;
```

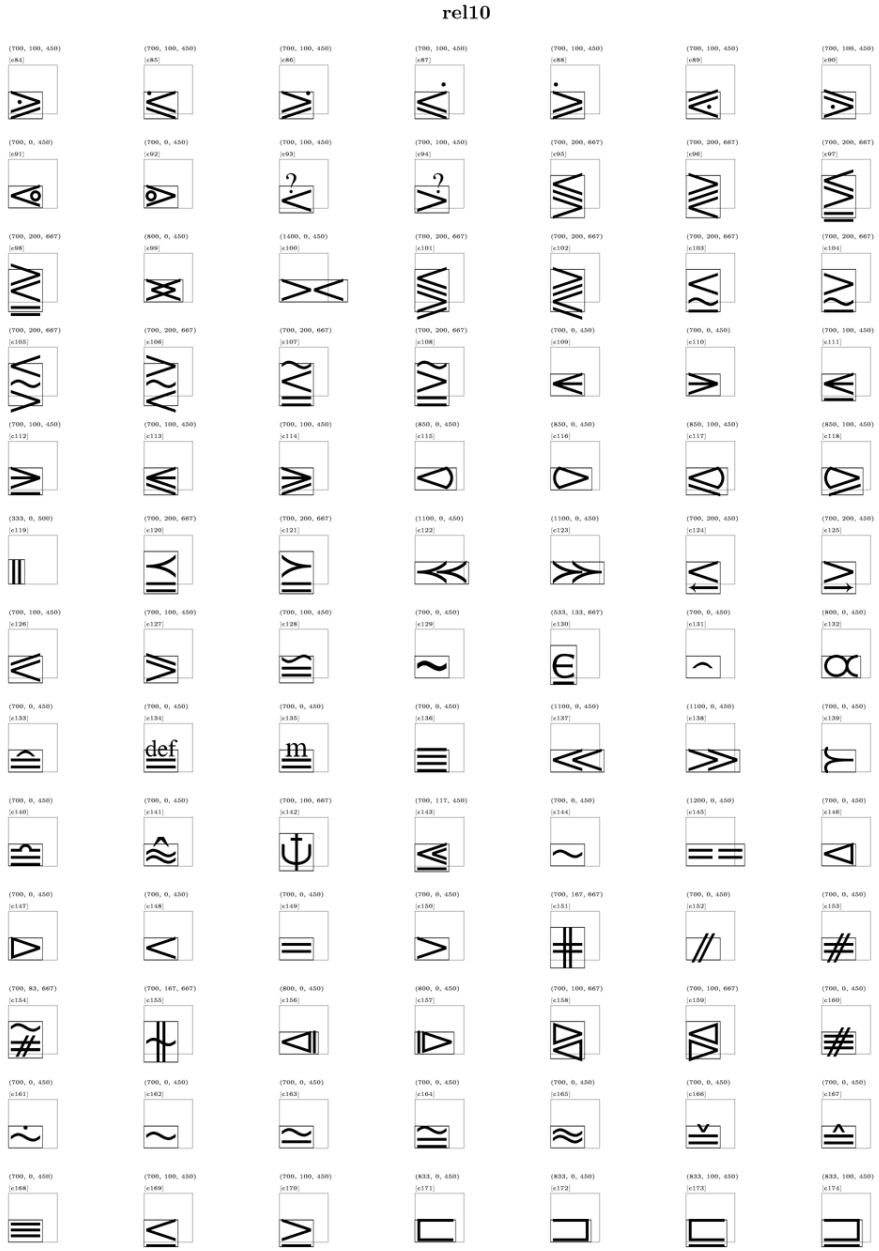

Figure 4

`geqslantdot_slot` is a number that gives the location of this character in the font. This list of numbers is another input file.

The macro `geqslantdot` is also defined in a separate macro file, and looks like this:

```
def geqslantdot =
leqslantdot;
```

```
hreflect;
enddef;
```

Of course, the referred–to `leqslantdot` represents the character that is the horizontal mirror of this one:

```
def leqslantdot =
leqslant;
dotat ((w-side-3thick,axis));
enddef;
```

And here, `leqslant` is defined in terms of `leq` and `eqslant`. All of the glyphs where this kind of indirection was feasible work this way: lots of glyphs are built up from (parts of) other glyphs, with added rotations or reflections around the axis.

The obvious advantage of this approach is that most of the sub-fonts consist of only a few dozen 'basic macros', who will be the only things that need to be changed for different versions to be implemented in the future. Of course, this scheme also helps out in the attempt to remain consistent in design throughout the sub-fonts.

Alphabets

From the discussions on the `math-font-discuss` mailing list, we came to a rather large list of alphabets that we seem to be needing:

- Math Italics
- Serif Text: Upright, Italics, Bold, Bold Italics
- Sans–serif Text: Upright, Slanted, Bold, Bold Slanted
- Greek Symbols: Upright, Italics, Bold, Bold Italics
- Blackboard: Upright, Slanted
- Fraktur: Upright
- Calligraphic: Upright, Bold
- Formal Script: Upright, Bold

Luckily, a lot of those alphabets can be borrowed or nearly borrowed from already existing (free) fonts. For instance, the Serif Text and Sans–serif Text can be taken from Times-Roman and Helvetica. Greek was borrowed from the Omega Unicode font, and Fraktur from the Euler fonts.

But others require quite some work. For instance, most current math texts use a Computer-Modern derivative for Blackboard Bold (`msbm` from the AMS fonts), which does not intermix well with Times. It is a bit crude and some of the capitals are somewhat different from Times; also all of the lower case letters are missing. A completely new version has been created, which results in:

This is 𝔑𝔈𝔚 𝔅𝔩𝔞𝔠𝔨𝔟𝔬𝔞𝔯𝔡 and this is 𝕆𝕃𝔻 𝔹<∋¬×∋⧹.

Likewise, new versions have to be done for the two required Script fonts as well as for the Calligraphic Bold version. These last three fonts are the final part of the work, and where not finished at the time of writing.

Current Status

There is still a lot of work that remains to be done in the TEX macro area: macros and virtual fonts have to be created before the fonts can actually be used in a sensible way from within TEX. Now that the encoding tables for the base fonts are reasonably fixed, this is rapidly becoming a first priority.

In MetaFont coding, about 90% of the work needed for the Times–compatible version is now complete. All of the symbols are available and the alphabets are well under way. The new maths fonts have therefore reached the point where user feedback is needed to finish the shapes and debug the metric information.

All current alpha and beta versions of the fonts (in both MF and PFB formats) and the related macros and metrics are available for public scrutiny from my web page:

`http://www.cybercomm.nl/~bittext/fonts.html`

In the current planning, there should be a user-level beta of the fonts as well as TEX macros available just before EuroTEX'99 in Heidelberg.

Literature

1. W3C REC-MathML-19980407, *Mathematical Markup Language (1.0) specification, W3C Recommendation 07-April-1998*.
   `http://www.w3.org/TR/REC-MathML/`
2. ISO/IEC TR 9573-13:1991(E), *Information technology – SGML support facilities – Techniques for using SGML – Part 13: Public entity sets for mathematics and science*.
3. `http://www.ams.org/STIX`
4. The Unicode consortium: *The Unicode Standard, Version 2.0* Addison Wesley, 1996 ISBN 0-201-48345-9.
   `http://www.unicode.org`
5. The Unicode consortium: *Unicode Technical Report # 8, The Unicode Standard, Version 2.1* Author Lisa Moore, September 4, 1998,
   `http://www.unicode.org/unicode/reports/tr8.html`. On-line publication.
6. Adobe Systems Inc: *Adobe Type 1 Font Format* Addison-Wesley, June 1995.
7. Richard J. Kinch: *Converting MetaFont Shapes to Outlines*. Paper presented at the 1995 TUG Conference in St. Petersburg, Florida, USA. Appeared in print in *Tugboat 16.3*.
8. Taco Hoekwater: *Generating Type 1 Fonts from MetaFont Sources*, 1998. *MAPS* #20, pp. 264–275.