New Math Fonts in MetaFont

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

Abstract

In the autumn of last year, work started on a new set of mathematical fonts that are extended to cover the full range of characters included in MathML and the STIX proposals for math extensions in the next version of Unicode.
This article presents the intermediate results.

Introduction

Typesetting mathematical material is and has always been a complicated matter. Not only does it require a rather specialized typesetting engine, but formulas also needs 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 lot of 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 there were no fonts available containing the correct glyphs for that particular field of science.

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.

Sources of information

Over the past few years, two groups have been working very hard to improve the situation. 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. Needless to say, there were no fonts available that covered the entire range.

A third interesting set of information are the specialized mathematical fonts that are already out there: fonts like the StMary's Road Symbols and Waldi's Symbols, and the fonts that come with scientific software like 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 publishers currently still rely on paper for most of their 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, with 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–in–design were rather disorganized. So the first job was trying to combine all possible information sources into one large collection of glyphs + descriptions that consecutively could be used as a lookup-table without being forced to ask questions 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.

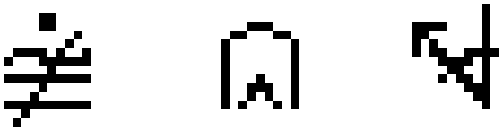

Figure 1

So the next thing to be done was to find as much 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.

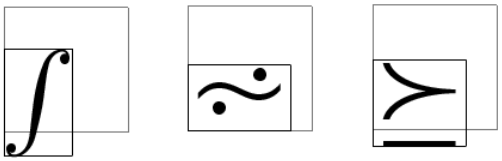

Figure 2

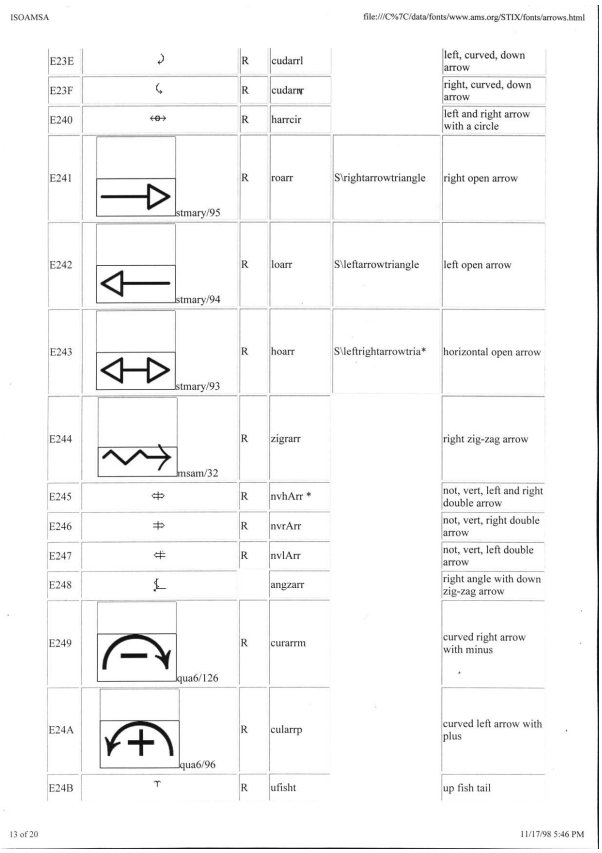

Figure 3

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 connected to an already existing character).

After that 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 the original STIX table was organized on the (tentative) unicode positions. 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.

Creating the fonts

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

1. Using an interactive editor like 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 already 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 big advantage is its programmability: using dedicated macros in MetaFont it is easier to remain consistent. And, of course, the MetaFont code is re-usable simply by changing the underlying macros and parameters.

MetaFont turned out to be the winner, not because of this particular job, but because it seems very likely that we will want to redo the font for different font families as well, a job which is much easier using already existing MetaFont code than

it would be restarting from scratch in an interactive program. 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 alike implementation.

The actual implementation

In the actual implementation, there are conceptually 21 fonts: one very large font containing all of the 1500 special glyphs and twenty 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
- Binary relations
- Negated binary relations
- Binary operators
- Delimiters and other extensibles
- Dingbats
- Ordinary symbols like harpoons and angles
- Large operators
- Accents
- Greek

The first eight of those are now in various states of completion, from almost completely finished (the arrows) to rough pre-implementions (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 all character definitions are as indirect as possible. For example, here is the code for character 84: 'geqslantdot':

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

`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:

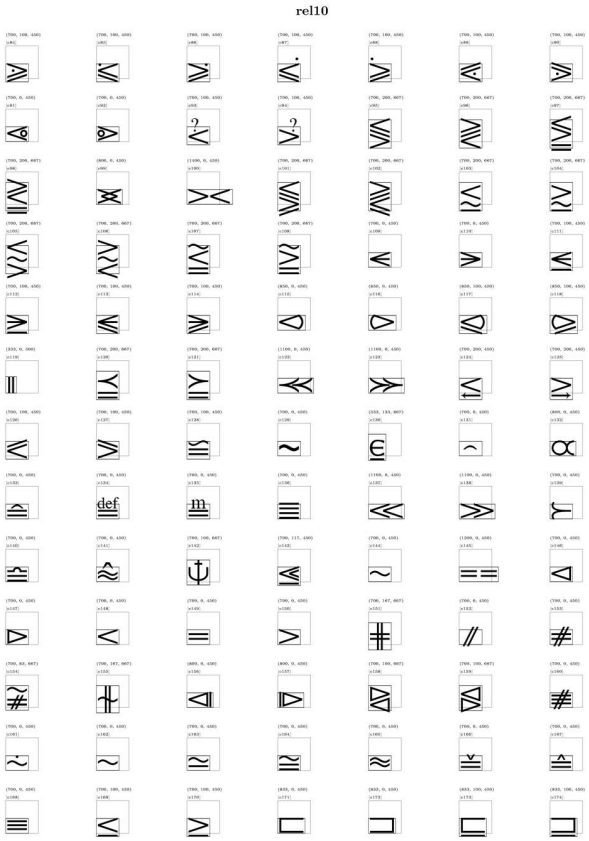

Figure 4

```
def geqslantdot =
leqslantdot;
hreflect;
enddef;
```

Of course, `leqslantdot` is 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 like this: 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', that hopefully will be the only things that will need to be changed for different versions to be implemented in the future.

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
- Blackboard: Upright, Slanted
- Fraktur: Upright
- Calligraphic: Upright, Bold
- Script: Upright, Bold

Luckily, a lot of those alphabets can be borrowed or nearly borrowed from already existing (free) fonts. But others require quite some work. For instance, most current math texts use a Computer-Modern derivate for Blackboard Bold (from the AMS fonts), which does not intermix well with Times (it is a bit skinny and some of the capitals are quite different from Times, besides missing the lower case letters). So a completely new version had to be created, which results in:

This is times NEW BLACKBOARD
and the OLD BLACKBOARD.

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 75% of the work needed for the Times–compatible version is now complete. Almost all of the symbols are available and the alphabets are well under way. The new math 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) as well as 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 and 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.