An encoding proposal: YAASP
Yet Another Aston Proposal
This is not a finished document

Justin Ziegler
ziegler@goofy.zdv.uni-mainz.de
ziegler@educ.emse.fr

Started: 3 July
Printed: May 5, 1997
Last change: 18 August.
Version 2.00
Filename: yaasp.tex with NSFF2

Abstract

The idea here is to put on paper what could be the backbone or the skeleton of the new math encoding. This is not the complete description of an encoding, but a sort of grid, or global picture of what things could look like. This document refers to glyph groups that are defined in "Towards a list of math glyphs". Same author.

1 Lets start with a few definitions

An "encoding table": this conveys the traditional meaning of an encoding. That is to say a set of 256 glyphs in a given order. The expression "encoding table" is usually abbreviated with encoding.

A "slot": is the usual word we use for referring to a position in an encoding, that can contain a glyph. It usually is an integer between 0 and 255. A slot is certainly not a family, or anything to do with it.

A "math kernel": this terminology is used to specify the fonts that are necessary for the math facility to work as it is described in most TEX documentation[1]. In DEK's implementation of math the math kernel consists of the families from 0 to 3. On top of the kernel, many other fonts with whatever encoding is available, could be optionally loaded and used.

A "math encoding": the way it is considered here, is as a whole. Not just one 256-glyph encoding table, but a set of x encoding tables, where x is the number of fonts in the math kernel. We will refer to this concept with the abbreviation "M-encoding".

The "core symbols": core symbols are made of two groups. The group of symbols that must live with the default alphabet for kerning reasons, and the group of symbols that must live with the default alphabet for design reasons.

[1]documentation on LATEX, AMSTEX, etc, also fits in here.

The "default alphabet": it is the alphabet that is used when a user types `$abc$`. In the present encoding that produces *abc*.

2 Global policy

2.1 Grouping all TEX specific glyphs in one font encoding

The present `cmex` font contains glyphs that can not be used by other typesetting systems, because they are set in a strange way.

The present `cmsy` font contains one glyph that is set in a strange way — the radical sign, and thus makes that whole font unusable for the outer world. I think it is a good idea to make sure that this does not happen again.

We are hopping to set a new standard, that will not only be used by TEX, but by all systems that typeset mathematical formulae. If every thing goes according to plan, in the next few years many math fonts will exist, for many different systems, and they will all use the same encoding. The fact that they all use the same encoding, means that it will be very easy to exchange fonts from one system to another. So one day a TEX user will be able to take a math font used by Microsoft Word, and convert it easily in order to use it with TEX.

If TEX specific glyphs are grouped in one font, there will only be one problematic font. As it happens, all TEX specific glyphs are more or less geometric, so they could be used with more than one math font.

On the other hand, if TEX specific glyphs are spread around in many fonts, then many "imported" fonts will not be usable by TEX without major messing around.

Concerning this problem, I think that the real question is: are we setting a real standard for the next few years, or are we just making another TEX encoding?

If the answer to that question is: we are setting a real standard for the next few years! Then TEX specific glyphs must be grouped in one font. If that is not possible, and I mean really not possible, then we can consider putting them in two fonts.

2.2 Grouping all Plain and LATEX glyphs in 4 fonts

The main reason for this, is compatibility. Lets consider a user that has typed a document with the present math encoding, and has saturated the families for this document. If the new math encoding does not garanty Plain and LATEX glyph compatibility with a minimum of 4 fonts, then that document can't run with the new math encoding: not enough families.

2.3 Grouping all AMSTEX and AMSLATEX glyphs in less than 6 fonts

The main reason for this, is compatibility. Lets consider a user that has typed a document with the existant AMSTEX or LATEX package, and has saturated the families for this document. If the new math encoding does not garanty AMSLATEX and AMSTEX glyph compatibility with less than 6 fonts, then that document can't run with the new math encoding: not enough families.

3 The base: a Cork encoded text font

Main use: things like `\log`. This would generably be a Latin font.

If it is a Latin upright font, it would probably also be used by physicists[2] for

[2]I guess I should also mention chemists.

operator, and more generally whenever upright letters are needed.

Separating this set from the rest, enables the user to have ‘log’ and ‘sin’, etc typeset in many different ways. Thus the multiletter operators can by compatible with the text font, or with the rest of the math glyphs, or even set in yet another font.

Note: this font is not for typesetting words. That must be done in text mode.

4 The “text symbol” encoding: the TS encoding

Here we would put the old style numerals, and most of what we have agreed to take out of the present math encoding. Other symbols could be added. The Text Symbol encoding is definetly *not part of the math kernel.* But seeing as it will contain symbols that previously were accessed via the math fonts, we must supply its encoding. This font will not be loaded in a family. It will just be loaded as a normal text font[3].

5 The core: the MC encoding (256)

Counting: 1+10+1 + 54+6 + 124+12 + 13+23+12 = 256 glyphs

The accents are no longer here. They had no real reason to be here. Most of them are geometrics anyway. But they do have reasons to be else where. One of the main consequences of taking these accents out is that the core can be made more coherent, and more complete. The MC encoding would contain:

- The skewchar in position 0: 1
- The core digits: 10
- The space character in position 32: 1
- The core Latin alphabet, which is the default alphabet, in uppercase, and lowercase, together with the dotless i and j: 54
- The Latin friends: 6
- All the Greek material: 124
- The Greek friends, next to the Greek: 12
- The core symbols for kerning reasons (punctuation and delimiters): 13
- The core symbols for design reasons: 23

  More for kerning reasons:
- The basic geometric delimiters: 12

  Some new stuff:
- New basic delimiters:

[3]If some users realy feel the need to load it in a math family they can.

6 The extensibles: the MX encoding

Count up: 1+1+ 77+8 + 8 + 24+6 + 7+7 + 16+26 + 60 = 241

The usual extensible characters, together with some new ones would live here. We include here any characters that have strange TEX features like big descenders. Thus grouping glyphs that are not compatible with the outer world.

The MX encoding, will be designed in such a way, that if it is loaded in one size (for compatibility with the present TEX), every thing works OK, and the user can still have access to the new symbols. On the other hand, the MX encoding is best loaded in three sizes, and produces better quality typesetting.

We are left with the following:

- Maybe a skewchar: 1

  I guess the space is questionnable here, because MX will not be usable by other typesetting systems, see comment:

- Maybe a space: 1
- All existant delimiters including the pieces needed for constructing the extensible versions, this includes the single and double vertical bar. There are 4 sizes per delimiter: 77
- Any characters that have strange TEX features like big descenders. Radicals including the one from CMSY'160: 8.
- Horizontal curly braces: 8
- All existant big and small "bigops" except the integrals: 24
- The single and double vertical arrows: 6

  Have to limit the number of wide accents, otherwise not enough place.

- The wide tildes, and the wide hats: 16

  New stuff:

  I am a little concerned about the fact that these will be separated from the small versions, but there is nothing much to do about it:

- The big "big integral" family: 7
- The small "big integral" family: 7
- The new big and small 'bigops': 26
- New multisized and extensible delimiters: 60

7 The math symbol 'privilege' font "MSP": 256

1+1+ 54 + 19+7+3 + 23 + 8+4+2+4+4 + 20+7+ 10 + 8+8+12 + 24 + 6+4+4+4 + 8+2 + 16= 256

- A skewchar in position 0: 1
- A spacechar in position 32: 1

- The script/calligraphic Latin letter set:54

  Because according to AMS statistics, the script/cal are used more often than the Bbb.

  Another reason is so that we can achieve backward compatibility with the existing TEX without loading MS1 and MS2.

- The basic accents: 19
- The double accents: 7
- The underaccents: 3

  This must stay here:

- The "Basic symbols" group: 23

  The next 5 are needed for compatibility with plain:

- The "Greater than plain" group: 8
- The "Subset plain" group: 4
- The "In / ni plain" group: 2
- The "Sqsubset plain & ams" group: 4
- The "Succ without sim plain" group: 4
- The "Small binops plain" group: 20
- The small ints: 7 I feel these should live with the other 'succ' members for design reasons:
- The "Succ without sim ams" group: 10

  The next three make a homogenous group, and must live with sim. Sim itself must live here because of compatibility with Plain:

- The "Greater than with sim" group: 8
- The "Succ with sim" group: 8
- The "Sim" group: 12

  The arrows, for compatibility, (improved a little though):

- "Plain horizontal arrows": 24
- "Plain vertical arrows" does not include the extensible ones: 6
- "Plain oblique arrows": 4
- "Latex arrow heads": 4
- Plain miscellaneous geometric symbols: 4

  The lasy triangles completed with the ones from AMS:

- AMS left right open triangles: 8

  Should live with the "Plain oblique arrows":

- "AMS obliques":2

  Some new stuff: Some of these could come out.

- Wide accents bar: 8

8 The MS1, MS2, Math-Symbol encodings

Each of these encodings will contain a set of Latin letters, like for instance fraktur or black board bold, in uppercase or lowercase or both. In some cases a place should be reserved for a set of matching numbers too (i.e. Open). The rest would be filled up with symbols. We need an MS$_i$ encoding for:

- An extra script/calligraphic, (see below comment on script and calligraphic) the default caligraphic is in the MSP encoding.
- Open + (Arrows or relations) + other geometrics.
- Old german,

Note: Barbara Beeton "regarding script vs. calligraphic, i do understand the difference; however, at AMS I believe we only very rarely get a request to use both styles in the same paper".

For that we have two possibilities:
1) We design one encoding table where the position A-Z (and probably a-z and 0-9 even if they aren't all filled) are supposed to contain a "calligraphy/script" set of characters. Then there would be instances of that encoding that would contain script chars and others that would contain calligraphic chars. Suppose our standard would say that this encoding is to be used as fam4. A designer would then choose one such font with this encoding for fam4 (thereby effectively deciding how `\cal` and a lot of other symbols look like (the ones whose mathchardef points into fam4)). For those who in addition would like to use another script/call math alphabet: they can then just allocate one of the free families. Access to this would then be trivial.
2) We have two different encodings one for cal and one for script. The remaining symbols in both encodings would be different too. Thus both encodings would need to be part of the standard suite of math encoding tables.

Which solution to prefer depends a bit on the the number of symbols that we want to put into the standard.

¿From Jörg: I strongly support to have two different encodings, one for cal and one for script. Why? If the user's have the choice between cal and script, they prefer script (at least here in Mainz[4]). On the other hand, the old calligraphic alphabet still needs to be supported for compatibility reasons.

9 The MS1 encoding: 232

Counting: 1 1 54 10 32 36 30 12 10 21 10 15,= 232

1. A skewchar in position 0: 1
2. A spacechar in position 32: 1
3. The BBB alphabet uppercase and lowercase: 54
4. The BBB digits: 10
5. The last WIDE ACCENTS: arc, back to front vector, and double sided vector, normal vector: 32

   For ams inclusion:
6. The "Ams arrows" group: 36

[4]Maybe American style/taste prefer it the other way round.

7. The “Greater than AMS” group: 30
8. The “subset ams” group: 12
9. AMS equals friends: 10
10. Ams miscellaneous geometric symbols: 21
11. Ams Vdash group: 10
12. AMS boxes and friends: 15
    For fantasie if we have place to spair:
13. Alan arrow construction set:

10 And the rest ?

- In general, users may want MC fonts in arbitrary styles (bold sans MC for instance) in order to get the Greek letters in their favourite styles.
- A “text-like” italic or slanted for computer science identifier names and the like. This would be Cork encoded and optionally loaded.
- A “bold upright” for use as variables – e.g. vectors in physics notation rather than the arrow over an italic letter. This would be Cork encoded, and optionally loaded or accessed via the `\boldsymbol` concept in that case no family is required.
- Bold italic for use as variables: either optionally loaded as a second font with MC or cork encoding (using only variable fam symbols) or accessed via something like `\boldsymbol`.
- Bold Old german (occasional) suggested `\boldsymbol` approach.
- Bold script (occasional) suggested `\boldsymbol` approach.
- Sans serif lightface (occasional): optionally loaded cork encoded font.
- Sans serif boldface (occasional): optionally loaded cork encoded font.
- Bold symbols: either `\boldsymbol` or optionally loaded in remaining slots.
- Ultra bold symbols: either `\boldsymbol` or optionally loaded in remaining slots.
- An MC-encoded bold font containing upright bold Latin glyphs, plus bold upright and bold slanted Greek. This would contain all of the most commonly requested bold glyphs in one font (rather than many more).

11 Summarising the families used by the proposed YAASP M-encoding

1. Family 0: A Cork encoded upright text font.
2. Family 2: An MC encoded font containing the default latin and Greek italic+upright, and core symbols...
3. Family 1: An MSP encoded font containing cal/script and accents...

4. Family 3: An MX encoded font including all extensible glyphs, and double sized operators...
5. Family y: An MS1 encoded symbol font for Open, and arrows or relations.
6. Family z: An MS2 encoded symbol font for Old german.

This leaves 10 families free for anything else, (like bold or sans...) and makes many symbols available.

What's more: The first four give total TEX, LaTEX compatibility.

Still more: The first six give total TEX, LATEX, AMSTEX, AMSLATEX, LAMSTEXcompatibility.

The six put together: make wonders, using no more font families than the present AMSTEX.

12 Discussion

12.1 Advantages

For MC: A big advantage here, is kerning. In this encoding kerning is possible between the Latin default alphabet, and both italic and upright Greek alphabets. This is necessary for compatibility, and for tidyness. On top of this both letter sets (in actual fact there are three) can be kerned with the core symbols that are in the MC encoding. That last point is the most important, and gives new and better automatic math spacing. (For compatibility reasons, we must kern the Greek italic with the period, the comma, and the slash.)

The bold version of the MC encoding gives the user access to a lot of bold letter sets in one go. The global family consumtion is therefore largely reduced: 1 bold font instead of 2 or 3.

Taking the accents away from the letters, means that the accents do not change when the text face changes. i.e. bold letters, and medium letters get the same accents.

Putting all the wide accents together, goes in the way of orthogonal grouping. Thus all the accents could be metafonted together, and have a few variables for adjustment.

One can get more than compatibility with plain TeX using 4 families (the same number as standard TEX currently uses).

One can get more than compatibility with AMSTEX using 6 families. This is less or equal than the number of families used by AMSTEX.

The calligraphic alphabet is more used that the open, so putting it with the accents is a step towards grouping most used stuff together.

This proposal gives a little room in the MC for free space, and good core material.

With the MSP encoding concept, the MSi encodings can really be considered as (optionnnal) extensions. Thus somebody who knows he doesn't need the arrow kit and the Blackboard bold letter set does not have to load them. Same for Fraktur.

All the TEX specific glyphs are grouped in MX. Thus all the other fonts could be used by other typesetting systems.

Using the Cork encoded font in fam 0 for things like `\log` and `\sin`, means that the Greek users can replace it by a Greek font. (I have been told that Greek mathematicians set those function names in Greek).

12.2 Disadvantages

If there is not enough space for all the symbols we want to get in, I suggest that we make an MS3 encoding, that would contain the other version of script/cal, and extra symbols.

12.3 Comments

Now the core is really made of two fonts, and the kernel is made of four.

Comment from Alan about the space is MX:

> MX will be used by TeX, and the dvi drivers may be outputting to a device that doesn't accept anything but a space in position 32. So if you don't include a space here, then the MX-encoded fonts have to be split into two device fonts by the drivers.

Comment from Alan about the Cork encoded font:

> I think it would be good to specify that this is family 0, for compatiblity with current TEX documents containing explicit `\fam0` (naughty them!) and in order that we've filled up slots 0–3 rather than leaving a gap in family 0.