File Information

Filename: l3d007.tex

Archived at: ctan:/tex-archive/info/ltx3pub/

Author: Justin Ziegler

Document group: Project core team

Title: Technical report on Math Font Encoding

Version: 2.00

Date/Time: June 1, 1994/18:52:36 GMT

Keywords: Math fonts encoding

Abstract: This is a report of the LaTeX3 Project work on math font encoding.

General Information

The LaTeX3 Project:
c/o Dr Chris Rowley
The Open University
Parsifal College
Finchley Road
London NW3 7BG, UK

Tel: +44 171 794 0575

FAX: +44 171 433 6196

E-mail: `LTX3-Mgr@SHSU.edu`

To subscribe to the LaTeX3 discussion list:
Send mail to

`listserv@vm.urz.uni-heidelberg.de`

with the following line as the body of the message (substituting your own name):

`subscribe LaTeX-L First-name Surname`

To find out about volunteer work:
look at the document in the file `vol-task.tex`, which can be obtained electronically (see below).

To get project publications electronically:
Project publications are available for anonymous ftp retrieval from ctan hosts (`ftp.shsu.edu`, `ftp.dante.de`, `ftp.tex.ac.uk`) in the directory `/tex-archive/info/ltx3pub`.

The file `ltx3pub.bib` in that directory gives full bibliographical information including abstracts in BibTeX format.
A brief history of the project and a description of its aims is contained in `l3d001.tex`.

You may use the ftpmail service to access these files by mail rather than ftp. Send a message just containg the word
`help`
in a mail message to:
`ftpmail@ftp.shsu.edu`
for more information about this service.

- For offers of financial contributions or contributions of computing equipment or software, contact the project at one of the above addresses, or the TeX Users Group.
- For offers of technical assistance, contact the project at one of the above addresses.
- For technical enquiries and suggestions, send e-mail to the `LaTeX-L` list (see above) or contact the project by letter or FAX at the address above.

Technical Report on Math Font Encoding

Justin Ziegler

Started on June 13, 1993
Last change: June 1, 1994
Printed: April 13, 1995
Filename: `l3d007.tex`

Foreword

I'm pleased to present the final report on "Math Font Encoding" produced by Justin Ziegler for the LaTeX3 project to the public.

Justin has worked for three months at the Johannes Gutenberg University Mainz. His work was generously sponsored by GUTenberg (The French TeX Users Group) and by the ZDV of the University of Mainz (Data Processing Center), the latter providing Justin with office space and taking care of the administrative details.

In the past years a lot of work went into integrating new fonts into the TeX system. Only five years ago, typesetting with TeX basically meant typesetting in Computer Modern. Nowadays many users can choose (at least theoretically) from several thousands of fonts. Today, NFSS is the standard font selection in LaTeX and due to this mechanism and the fontinst-package by Alan Jeffrey virtually every PostScript font, in fact, every font for which a `tfm`-file can be obtained, can be used, out of the box, with LaTeX.

But for these thousand text fonts there are only five font families for use in math formulas to go with them. Even worse, every of these math font sets are encoded in a different way making it nearly impossible even for an expert TeX user to use different fonts for math in different jobs.

The work undertaken by Justin is the first of several steps to solve the problems at hand, the final goal being the development of a system that allows the user to change math fonts as painlessly as it is now possible with text fonts.

Based on Justin's analysis and his proposal, the LaTeX3 Project is now undertaking to provide a prototype implementation for math fonts, starting with the Computer Modern fonts as well as the Euler Math fonts. We expect this implementation to be available for public usage during 1995.

Mainz, December 6, 1994

Frank Mittelbach
Technical Director LaTeX3 Project

Acknowledgement

I wish to thank the many people without whom my stay in Germany would not have been possible, and the work I did would not have been done. This includes:

GUTenberg who financed my stay;

Ehoud Ahronovitz for helping me with the administrative side of things, for giving me the opportunity of coming here, and spending extra time with me to make sure that everything went all right;

Frank Mittelbach for his friendly welcome, for the organization, time and guidance;

Bernard Gaulle the past president of GUTenberg, for the organization and logistics;

Klaus Merle for lending all the material that I used;

Chris Rowley for the organization, and help;

Stefan Steffens for answering patiently all my stupid questions, and helping me integrate Mainz and the university;

The LaTeX3 project which partially financed my stay in England for the Aston conference;

Barbara Beeton and Alan Jeffrey who commented my papers, and answered more stupid questions;

Jörg Knappen who gave me advice on the project, and with whom I discovered the Mainz night life;

All the computer center employees for making my stay more pleasant;

All the people who took the trouble to answer my mail, for their time and effort;

Donald E. Knuth who created TeX.

Contents

Chapter 1

Introduction / Overview

1.1 The technical environment

I worked in the ZDV of the university of Mainz in Germany. In German ZDV stands for "Zentrum für Daten Verarbeitung". Which means: Data Processing Center. This is where the main — soft and hardware — maintenance team works.

I worked on an X-terminal like a lot of other people in the university. For writing my documents I used GNU Emacs together with the AUCTEX package.

1.2 A few Definitions

1.2.1 TEX: a page description language

The best definition I can find for TEX is: "one third compiler, one third interpreter, and one third word processor". It was written in 1975 by D. E. Knuth and a group of students. One of its main features is its portability. A document written on one machine can be used on another machine. Knuth also insisted that TEX would not change. So a document written in 1980 is still usable in 1990.

The language defined by TEX is very specific, in so far as it is designed to describe a page layout. TEX processes the page like a rectangle, or more exactly like a box, that can be filled with smaller boxes. These smaller boxes can similarly be filled with smaller boxes, and etc... The smallest box one can get is a vertical / horizontal line, or a character (a glyph), or just some space. TEX has variables in which one can put boxes, or different types of numbers. One can define functions — usually called macros — in a way similar to lisp. The if-then-else statement is there, and combined with recursion it can be used to make loops.

In spite of its limitations due to its specificity, TEX defines a Turing machine. The syntax is very disagreeable, but one can get used to it: somebody wrote a basic interpreter in TEX. The only difference between TEX and a usual compiler, is that TEX stops the compilation when it gets to the pcode, and just puts it into a file. This file, called the device independant file, can then be sent to a printer, a screen, or any other printing device.

Today many people use TEX. All TEX users have got together and created TUG: TEX Users Group.

1.2.2 Plain TeX

Plain TeX is the standard set of macros and definitions that comes with TeX. It is written in TeX.

1.2.3 LaTeX: a document description language

Just as TeX is a language to describe pages, LaTeX is a language designed for describing whole documents, and their logical structure. The idea is that it lets the user concentrate on the contents of the document rather than the formating commands necessary for the document to look good. Thus it uses the logical mark-up concept. It was written by Leslie Lamport in 1985. Technicaly, LaTeX is "only" a cluster of macros written in TeX. This means that a LaTeX user has still got access to most of the TeX language. LaTeX includes the following facilities:

- Cross referencing.
- Automatic construction of a table of contents.
- Automatic construction of an index.
- Bibliography referencing.
- Basically the same math mode as TeX.

1.2.4 The LaTeX3 project

During the 1989 TUG conference at Stanford, the decision was taken to produce an improved and expanded version of LaTeX, that was going to be called LaTeX3.

The major difference in the new version will be the addition of a good interface through which designers can specify how classes of documents should be formated.

Frank Mittelbach is the technical director of the project; he and Chris Rowley are responsible for the management.

1.2.5 Fonts, glyphs, and slots

TeX would not be able to produce any nice documents if it did not have any fonts. One cannot get a nice looking 'A' or '$\mathcal{A}$', or any other letter if nobody has previously designed it.

All TeX really does, is produce a file that contains a set of instructions. Each instruction looks like the following: "place here the picture that is in such and such a file, in position number x." *The files that contain all the pictures (the letters and other symbols), are called "fonts". All the pictures that are in a font are called "glyphs".* Every glyph in a given font has a specific and known position. *I shall use the word "slot" to refer to a given position in a font.* Some slots can be empty, but most of them contain a glyph.

1.2.6 Font encodings

When TeX refers to the glyph number x, it must know which glyph is in position number x. This knowledge is contained in the encoding. In some cases one could say that the letters are in the ASCII order. But this is not sufficient, because the ASCII code does not include all the glyphs that people wish to put in their documents. Therefore, one must link every single font with a given encoding, and make the encoding known by TeX. Many different encodings exist, sometimes even for the same group of glyphs. But there are also many fonts that use the same encoding.

A mathematical definition of an encoding could be the following: *An encoding is a set of glyph names in a given order.*

1.2.7 The “Computer Modern Fonts”

When D. E. Knuth created TeX, he also created a set of fonts called the *Computer Modern Fonts*. Most of them were based on an encoding that is called the Computer Modern Encoding throughout this document.

All file names of Computer Modern Fonts start with the two letters ‘`cm`’.

1.2.8 Metafont: a font description language

Metafont is a language / program especially designed to describe glyph shapes, and more generally whole fonts. It was used to generate all the “Computer modern” fonts. The Metafont user must describe or “program” the curves for each glyph. Then Metafont produces an array of black and white dots for each glyph. The dots can be made as small as necessary to fit the precision of the printing device.

1.2.9 TeX version 3

In the beginning of 1990, under a lot of pressure (from the TeX User Community), D. E. Knuth produced a new version of TeX. TeX version 3 was born. The main improvements were the following:

- Up to 256 glyphs per font. The previous versions of TeX could only use the first 128 glyphs of a font.
- Virtual fonts. A normal font has all its glyphs in a file, and this file is in actual fact the font. Virtual fonts enable people to group 256 glyphs taken from many different fonts, and make TeX think it is using one normal font. For instance, one could make a virtual font with lowercase letters in bold, and uppercase letters in italic. The user would work as if he was using one font, but the results would in actual fact be a combination of two fonts. A very good example implementation of virtual fonts is the creation of “Small Caps” fonts: the uppercase letters could come from a roman upright font at 12 points, whereas the lowercase ones could come from a roman upright at 10 points.

  Virtual fonts enable still more ingenious things, like replacing glyphs with a set of TeX macros. One can then consider, for example, automatic raising or lowering of some letters.
- Better hyphenation. TeX version 3 can have up to 256 different hyphenation tables, and can produce good automatic hyphenation even when a word contains accents. The latter was not possible in previous versions. More generally the hyphenation mecanisms have been improved.
- The new ligature mecanism is more powerful. The result of a ligature is no longer only one glyph, but can be a set of glyphs...
- Special ligatures can be done at the beginning and at the end of words. Thus when a given letter is at the end of a word, its shape can be different from the shape it would have in the middle of a word.

- Better automatic adjusting of interword space.
- More little details that make everybody happy...

1.2.10 The "DC Fonts"

Although D. E. Knuth included a lot of "European glyphs" in his Computer Modern fonts, more were needed. In 1989 TeX users got together in Cork, and designed some new fonts called the *DC Fonts.* Thanks to the new features of TeX version 3 (256 glyphs per font encoding), DC fonts included for example more special letters for Catalan and Scandinavian languages.

The DC Fonts used what is now called the *Cork encoding.* All DC fonts file names start with '`dc`'.

1.3 My work

One of TeX's nicest features is its ability to typeset mathematical formulae. There has now been over ten years of experience typesetting mathematical material with TeX. During this time, TeX's math mode has been used to set a wide variety of material, including traditional mathematics, categorical diagrams, chemical reactions, computer programs and textual material such as '$5\frac{1}{2}$% or $\mathrm{M}^{\mathrm{lle}}$.

In recent years, with the arrival of the Cork standard for typesetting European text, and the Virtual Font standard, the fonts available for use in TeX have radically changed. The current situation is that there are over 14,000 text fonts available for use in TeX, but only five math fonts:

- Computer Modern
- Computer Concrete with Euler
- Lucida Math
- Lucida New Math
- Math Time

Each of these fonts use different encodings, and each comes with its own selection of TeX macros.

Although the Cork encoding is rapidly being established as the standard encoding for European Latin text, there is no similar encoding for mathematics. The result is:

- complex macro packages for using each math font.
- it is difficult to set mathematics with Cork text, since the Cork encoding does not include the uppercase Greek.
- installing PostScript math fonts such as Mathematical Pi is very difficult.

Furthermore, the present math encoding includes glyphs like old-style digits, and game card suits ($\spadesuit$) that just do not belong in a math encoding. On the other hand, many new glyphs have been designed and should be included in the math encoding.

To solve these problems, a new math encoding, using all the power of TeX version 3, is needed. For this reason I have been trying to re-organize all the glyphs that are needed to typeset mathematical formulae with TeX, according to various technical constraints.

The new math encoding that I am helping to produce is hopefully going to be part of the LaTeX3 package, and comes as one of the general improvements of LaTeX.

First I learnt to use TeX. In a second stage, I had to study and understand the technical constraints that apply on the grouping of mathematical glyphs in a font. Only then could I actually start thinking about which glyphs should go where. I intensively used LaTeX — so that I permanently had an up to date record of what had been done — and email, to communicate with the people I was working with.

Chapter 2

The TeXnicalities of math typesetting

2.1 A brief description of TeX's math facility

Logical markup like LaTeX. For the design of TeX's user interface, one of Knuth's concerns was that in the source code of a mathematical document the formulae should be readable in a linear manner. Thus when a mathematician thinks, he says to himself: "n over $n-1$", and when a TeXnician works, in order to produce the result $\frac{n}{n-1}$ he just has to type: `n \over {n-1}`.

The user is no longer bothered by trying to get this bit of text higher than this other bit of text. He just gives TeX the logical meaning of what should be typeset, and it is correctly placed.

The two math modes. There are two ways to enter TeX's math mode, which produce slightly different results with the same input. One mode is called the *display mode,* and produces *display style,* while the other is called *text mode,* and produces *text style.* The following input:

```
$ \int_0^1 \frac{1}{x}\;dx $
```

produces *text style:* $\int_0^1 \frac{1}{x}\;dx$, which can be mixed with text, whereas

```
$$ \int_0^1 \frac {1}{x} \;dx $$
```

produces *display style:*

$$\int_0^1 \frac{1}{x}\;dx$$

which is automatically centered and surrounded by space.

Automatic size change according to meaning. When the user says to TeX: "this letter is a superscript", or "this number is a subscript", TeX automatically typesets the letter (or the number) in a smaller font size. TeX does that same size adjustment for setting limits on glyphs like $\sum$, or $\int$.

Automatic placing for sub/superscript and for limits. At the same time as TeX changes size automatically when the user specifies a sub- or superscript, TeX also raises and lowers the resulting text. When placing limits over a $\sum$, for example, TeX automatically centers them over the sum:

$$\sum_{i=0}^{i=n} i = \frac{n(n+1)}{2}$$

Size change for big operators. One can see in the previous example that the two $\sum$ signs (one in the text and one in the example), are not set in the same size. TeX changes the size of some big operators when they are set in a centered environment like that example is. The integral also changes size.

Automatic spacing and math classes. As one can see in the previous example, TeX also spaces various glyphs in a special way. For instance the space around the $+$ sign is quite large, whereas the space between the n and the open parentheses is comparatively reduced. Turning off the automatic mathematical spacing for the $+$ sign would produce the following: $(n{+}1)$ versus $(n+1)$.

On a TeXnical point of view, the math spacing is done by dividing all mathematical glyphs into classes. For each class TeX has different spacing rules. Thus a class 1 glyph followed by a class two glyph would not induce the same spacing as a class 1 followed by a class 3. There is no point in giving all the spacing rules here. The different classes are listed below[1]:

1. *Ordinary:* lowercase Greek characters, and those symbols that are just 'symbols';
2. *Large operators:* integral and sum signs, and 'big' objects such as `\bigcap`, or `\bigotimes`. Large operators are centered vertically, and they may behave differently in text style, and in display style[2].
3. *Binary operators:* plus, minus, and look alikes;
4. *Binary relations:* equal, less than, subset, and friends;
5. *Opening symbol:* opening brace, bracket, parentheses, etc...
6. *Closing symbol:* closing brace, etc...
7. *Punctuation:* most punctuation marks, with an exception or two;
8. *Variable family:* described further on in section 2.3.3.

More symbols/glyphs. Last but not least, TeX's math facility gives the user easy access to special symbols: Greek letters, $\aleph$[3] $\cap$, $\subset$, and many others that are often used in mathematical formulae.

2.2 Math styles

When Knuth wrote "The TeXbook", he extended the 'display', and 'text style' terminology. If TeX is typesetting sub- or superscript material, one says that it is in *script style.* Furthermore, if TeX is typesetting sub- or superscript when it is already in script style, one says that it is in *scriptscript style.* The style terminology must not be confused with the size terminology that is described further on: text size, script size, and scriptscript size.

2.3 Font families

2.3.1 What are font families? / a definition

In math mode, TeX does not load fonts in the same way as it does in text mode. For maths, Knuth thought best to organize the fonts in families, and give each family a number. One font family can contain three fonts.

[1] Thanks to Victor Eijkout for the comments.
[2] See below for explanations.
[3] $\aleph$ is a Hebrew letter, not a Greek one.

The normal use is to load in a single family the same font in three different sizes. One size for the main text, one size for superscript and subscript, and one size for the exceptional super-superscript, or super-subscript. A good example ought to make things clear:

$$\int_0^\infty e^{\alpha.x^\alpha}\, dx = \ ?$$

It is clear that the x is smaller than the e, and that the second α is smaller than the first, which is the same size as the x.

2.3.2 The organization of mathematical glyphs

In the present version of TeX the mathematically used glyphs are organized in 4 families:

Family 0: Computer Modern Roman (`cmr`) This is a normal upright roman text font. It is loaded in a math family in order to typeset things like log or sin. The other reason for which it is loaded into a math font family is that it contains the uppercase Greek alphabet, so that the user can typeset Ψ and Γ, or even Υ. A few other symbols are also taken from `cmr`: ';' '=' '( )' '[ ]' ':' '+' ... See figure in appendix D.

Family 1: Computer Modern Math Italic (`cmmi`) The `cmmi` font is one of the special math fonts. For a non-expert user, its letters look just like normal *italic* letters. But in actual fact they are slightly different in their shapes, especially the lowercase. The reason for the letters being different is so that the variable a can be easily differenciated from the article 'a' used in "a horse" for example.

Whereas `cmit`[4] contains ligatures, `cmmi` does not, and includes instead the Greek lowercase and uppercase alphabets in italic.

A strange feature of `cmmi` is that it contains some old style digits. Thus one can write $\oldstyle{1789}$ or $\oldstyle{1942}$ which are quite different from 1789 and 1942. But these digits are never used in maths, so they do not belong in a font that is designed for use in maths.

The `cmmi` font also includes some other useful[5] symbols / glyphs that one can see on the corresponding figure in appendix D.

Family 2: Computer Modern Symbols (`cmsy`) One can find in this font the calligraphic alphabet that some scientists use: $\mathcal{ABCDEFGH\ IJKLMNO\ PQRSTUVXYZ}$; plus lots of other symbols that only mathematicians could want to use: $\cap \cup \ominus \otimes \bigtriangleup \exists \forall \subset \leq \succ \leftarrow$ ... See figure in appendix D.

Family 3: Computer Modern Extensibles (`cmex`) All three sizes in this family are the same. `cmex` mainly contains symbols that change size, automatically. One can produce:

$$\left\{ \begin{array}{ll} u(x,y,z,t) & = u_0(x,y,t) + U(x,y,z,t) \\ v(x,y,z,t) & = v_0(x,y,t) + V(x,y,z,t) \\ w(x,y,z,t) & = w_0(x,y,t) + W(x,y,z,t) \\ w'(x,y,z,t) & = w'_0(x,y,t) + W'(x,y,z,t) \end{array} \right.$$

with four or ten lines, and the '{' will get bigger and bigger of its own accord, without the user specifying anything more. `cmex` also contains wide accents, so

[4] The normal italic Computer Modern font.
[5] Only for scientists though.

one can produce: $\widehat{a}$ $\widehat{ar}$ $\widehat{arg}$. I have previously spoken about the automatic size change of some operators, whether in text, or in display, style. These double sized 'big operators' are in `cmex`: $\bigcap$ *and* $\coprod$ in text style, and in display style:

$$\bigcup \textit{ and } \coprod \textit{ and} \ldots$$

The total contents of `cmex` is shown in a figure appendix D.

Most of the glyhs in `cmex` have a stange metric particularity, that makes them TeX specific. Thus no other typesetting system can use those glyphs. Vice versa TeX could not use those glyphs if they were made for another typesetting system. I spent a certain amount of time trying to understand all the triks hidden in `cmex`, and wrote a document on the topic (see appendix C). The math font group was then able to take decisions concerning the replacement of `cmex`.

The AMS symbol fonts: `msam` and `msbm`. Many more mathematical glyphs, and an extra blackboard bold alphabet. They are not part of the standard TeX, and are not loaded automatically in a family, but they are used on many sites. They were designed for the AMS: American Math Society, for use with TeX, and are now very widely spread. Their contents is shown in figures, appendix D.

2.3.3 How does TeX identify glyphs?

Glyph names. In Plain (see section 1.2.2) many glyph names are defined. They refer to some of the numerous glyphs TeX can typeset.

The user can also define his own names for glyphs. To a glyph name must be associated a family number, and a position in the given family. On top of that TeX likes to know which class the glyph belongs to. As well as the classes that have already been defined, there is an extra one:

The *'variable family'* class and the `\fam` variable. This class has nothing to do with spacing, and, to my mind, treating it as a class is one of Knuth's mistakes. It is used in particular for letters, but it could have other uses. If the calligraphic, upright, and italic letters all have the same position in their respective fonts, one does not want to define a different name for each letter in each shape. Instead, TeX has a `\fam` variable, that contains the number of the current family where glyphs should be taken from. So when a glyph is of class *'variable family'*, it is taken from the family number `\fam`. But that is not enough. Some times the `\fam` variable can be equal to -1, and there is no family number -1. In such a case a default family number is used. So together with the class and the position, one can assign the default family number for each glyph name. When a glyph is not defined as being variable family, it always comes from the same family, and its family number is linked to its name in the same way as the class number.

Example: when the user enters math mode, `\fam` is equal to -1, the letters come from the default family. By typing: `$abda$` which produces '*abda*', one can see that the default family for letters is family number 1 (See family descriptions). If the user assigns the family variable to 0 then the letters will come from family 0. Thus `$\fam0 abda$` produces 'abda'. (See family descriptions).

2.4 Font metric files: The "`.tfm`" files

2.4.1 A theoretical overview

When TeX is typesetting a page, and making all the calculations that are necessary for this, it does not need the actual picture of the glyphs. All TeX needs at this stage is the dimensions of the glyphs, and other numerical data. That information is in the "`.tfm`"[6] files, and every font has one. Without it, the font is unusable as far as TeX[7] is concerned.

For mathematical typesetting TeX uses all the information that a "`.tfm`" file can give. One of the first things I had to do was to study and understand the machinery hidden in the math fonts "`.tfm`" files. From a general point of view a font metrics file can contain the following data: [8]

Font dimensions. These are global parameters for the whole font. In a normal text font one would find the slant (positive on an italic or slanted font), the size of the interword space, other interword spacing parameters, more general spacing parameters, and the x-height. The latter is the height of the 'x' glyph, and is used for correct accent positioning.

The fonts in family 2 and 3 are a little special as far as font dimensions are concerned. TeX looks in family 2 and 3 for more font dimensions than usual. This extra information is used for special math spacing.

Glyph dimensions. Each glyph has a height, a width and a depth specified in the "`.tfm`" file. The height of the box that surrounds a glyph is equal to the height of the glyph plus the depth of the glyph, whereas the width is that of the glyph. I think it is important to say that *these values are theoretical, and can be quite different from the real size of the glyph.* Thus some glyphs are bigger than their box. A good example of this is the italic 'f': *f*. The top right end, and the bottom left end stick out of the box. The right hand side of a given glyph box is also the left hand side of the next[9] glyph box.

Kerns. They are necessary for the *automatic adjustment* of the spacing between two glyphs. Many non professional electronic typesetting systems have for a long time ignored this refinement of traditional typesetting. The problem is the following: for visual comfort all the letters of the alphabet cannot be spaced in the same manner. For instance when an 'A' is followed by a 'V', the two letters must be brought closer together to produce 'AV' versus 'AV'. In other cases letters must be separated a little to produce 'aj' versus 'aj', or 'f!' versus 'f!'. Otherwise the spacing does not look correct compared to the spacing of surrounding letters. In the "`.tfm`" file, for each glyph one can specify kerns with every other glyph *of the font. When two glyphs that are kerned in the "`.tfm`" file are found side by side in the right order, TeX automatically brings them closer together, or farther away.*

Ligatures. Here again, the idea is to improve visual comfort, and reading. Some letters when followed by other particular letters do not look right. In this case

[6] `tfm` stands for "TeX font metric file."

[7] There are some slight exceptions to this rule: in some cases a given font can use another font's "`.tfm`" file. But the visual results are not very good.

[8] This is not restricted to TeX. Although the file formats maybe slightly different, Postscript type fonts and others use similar metric files. One can find programs to convert the files from one format to another.

[9] The box on the right of the first one.

the two glyphs side by side must be replaced by another glyph that will look much better. This is called a ligature. The best and very well known example occurs when an 'f' glyph is followed by an 'i' glyph. The non-ligatured glyphs look like 'fi', and *TeX automatically replaces* them with the ligature that looks like 'fi'. In the "Collection La Pleiade", one can see many other ligatures if one looks hard enough.

In TeX version 3 the concept of ligatures is more general. It can use more than two letters, and has other interesting new features.

Italic corrections. For this I can only quote Frank Mittelbach:

> "At the points where one switches from slanted or italic to upright, the glyphs usually come too close together, especially if the last slanted/italic glyph has an ascender[10]. The proper amount of extra white space that should be added at this boundary is called the 'italic correction'. Its value depends on individual glyph shape, and is therefore stored in the "`.tfm`" file for each glyph. [...] For an upright font the italic corrections are usually null. [...] In slanted and italic fonts, the italic corrections are usually positif..."

Example: in the word *dif*ferent, the first f runs into the second one. Whereas in the word *dif* ferent, a little space is left between the two f's. That space is the f's italic correction.

'Skewchar' kerning. The skewchar is a specific character that is used for placing mathematical accents. In math mode, when an accent is placed on a glyph, the accent is first centered on top of the glyph's box, and then shifted rightwards by the amount of the kern between the glyph and the skewchar.

Each font should have its own skewchar. For most characters, the "`.tfm`" file specifies the kerning of each letter with its skewchar. This is true for the computer modern fonts, but other font designers may have chosen not to use this feature.

Why choose one skewchar rather than another? This is because the character ⁀ chosen by Knuth does not have any other kerning that could have been disturbed by the skewchar kerning. This choice may not always be good for all fonts, because it depends on what the character in position '127 is. Thus a font designer might choose another skewchar and put the necessary kernings in the "`.tfm`" file. Accent glyphs can be used as skewchars, because they are not usually subject to kerning from other glyphs."

Charlists. Charlists enable several characters in a font to be linked together. The `cmex` font uses charlists a lot: by just typing `charlist oct "000": oct "020": oct "022": oct "040": oct "060"` in the metafont source code, one links in order of increasing size all the left parenthesis that are in the font. Thus with this information contained in the "`.tfm`" file, TeX can find the parentheses that has the correct size for what is currently being typeset.

Charlists are used for:

- Linking variable-size delimiters,
- Linking variable-width accents,

[10] Here is something that has not been defined. The following letters have ascenders: l,k,h,f,t,b,d, in lowercase. One can guess what descenders are.

- Pairing the “big operators” that are typeset in different sizes in display style, and text style.

Extensibles. Extensible glyphs can change size vertically (not horizontally), according to the context. A good example is given in section 2.3.2 where the { grows automatically.

An extensible glyph is identified with one of its pieces. One simply has to decide which piece is going to be used for this identification. In the following example: `extensible oct"060": oct"060", 0, oct"100", oct"102";` — which appears in the metafont code of `cmex`, the first oct”060” is the identifier of the whole extensible glyph. The next three characters are the top, middle, and bottom pieces of the glyph whose identifier is oct”060”. The last character code is that of the piece to be repeated as many times as necessary between the top and middle, and between the bottom and middle pieces. All pieces are optional except the repeatable piece.

This mecanism is also used for the construction of the radical sign. But it only works for glyphs that grow vertically. Therefore the horizontal braces and the horizontal extendable arrows cannot use this facility.

2.4.2 Example: analysis of ‘`cmmi`’ metrics

I shall use here the usual TeX notation for writing octal numbers. Thus all numbers preceeded by a little quote sign like ’77 are in octal.

- Most characters in ‘`cmmi`’ are kerned with the skewchar.
- Many Greek uppercase and lowercase letters are kerned to: ‘.’ ‘,’ and ‘/’ respectivly ’72, ’73, and ’75. This takes us right up to position ’50.
- Characters from ’50 to ’73 are not kerned at all. This includes: funny horizontal half arrows, two hooks for the arrow construction set, two triangles, the old style digits, the ‘.’, the ‘,’ and the ‘$<$’.
- The ‘/’ sign is kerned with 1[11], A, M, N, Y, Z. Nothing to say about ‘$<$’ and $*$ and ∂.
- Then come the uppercase Latin letters. They are not kerned among each other. They are not kerned either with the lowercase letters. Just like the Greek letters, some of them are kerned with ‘.’ ‘,’ ‘/’.
- In my `.pl` file, it looks as though N and X have got two different kerns with ’75. (Not yet any explanation for this.) The 3 musical signs are not kerned with anything. The horizontal parentheses are not either.
- The lowercase Latin letters are not kerned with each other, except ‘d’ that is kerned with Y, Z, j, f. Some of them are kerned with ‘,’ ‘.’ ‘/’ in a way similar to that of uppercase letters.
- The last characters are not kerned at all.

For compatibility reasons, all these kerns will have to be in the new encoding.
See appendix C for a complete description and analyses of `cmex10.tfm`.

[11] The digit.

Chapter 3

Dividing all the glyphs into groups

3.1 More vocabulary

An "encoding table". This conveys the traditional meaning of an encoding (see section 1.2.6). That is to say a set of 256 glyphs in a given order. The expression "encoding table" is usually abbreviated: "encoding".

A "slot": the usual word used for referring to a position in an encoding. A slot can contain a glyph, or be empty. It is represented by an integer between 0 and 255. A slot is *not* a family in spite of the usage some people make of this word.

The "math kernel". This terminology is used to specify the minimal group of fonts that is necessary for the math facility to work, as described in the TEX documentation[1]. In D. E. K.'s package (Plain) the math kernel consists of the families numbered from 0 to 3. Together with the kernel, many other fonts can optionally be loaded and used.

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

The "default alphabet": the alphabet that is used when a user types `$abc$`. With Plain TEX's math encoding that produces abc.

"Glyph compatibility": two encodings (or M-encodings) are glyph compatible, if they contain the same glyphs. The latter do not systematically have to be in the same positions. However identical glyphs must have the same metrics. The kerning and ligaturing information must also be identical in both M-encodings.

3.2 General approach

Taking all the glyphs one by one, and putting them in a font encoding would have been to easy, and above all not satisfactory. Instead one must divide all the necessary glyphs

[1]And LATEX, AMSTEX, etc, documentation.

into groups and subgroups, and then tried to match groups in individual encoding tables according to all the constraints.

For instance, a typical group is the Latin alphabet: it includes the uppercase letters A-Z, and the lowercase letters a-z. Mathematicians often use accents on letters. For this they need a dotless 'i' (looks like: 'ı') and a dotless 'j' (looks like: 'ȷ') with every Latin alphabet. Thus the Latin alphabet group contains the uppercase and lowercase letters, the dotless 'i' and the dotless 'j'.

The grouping is based on different types of constraints: some technical, and some based on glyph usage. These are detailed in the next section.

3.3 Grouping constraints

Before being able to group the glyphs, grouping rules had to be established. [2]

At first I did not realise the importance of the design similarity constraint for the person designing the font. Thus the first two proposals did not really take it into account at all.

Kerning. See section 2.4.1 for a definition of kerning. The kerning information for a given font can be found in it's `tfm` file. Thus two glyphs from the same encoding table can be kerned together, but the letter 'f' belonging in a given encoding table, cannot be kerned with a glyph (the open parenthesis for instance) belonging in another encoding table. A group resulting from this constraint is: the group of glyphs that must be kerned with the default alphabet. This group and the alphabet will have to live in the same encoding table. This constraint is considered to be one of the most important.

In fact this type of grouping is not so much grouping together all the glyphs that must be kerned, but putting together in one group the glyphs that need to be kerned with another group. In order to facilitate the counting.

Ligaturing. See section 2.4.1 for a definition of ligaturing. In a similar manner to kerning, ligatures request that various glyphs live in the same encoding. If the letter 'f' is to be ligatured with the letter 'i' and produce the 'fi' ligature, then those three glyphs 'f','i', and 'fi' must live in the same encoding. In actual fact ligatures are not really used in math fonts. But they may be necessary one day. So empty slots should be left for ligatures where possible.

Design similarity: another reason for which the letter A must live in the same font encoding as the letter B, and all the other letters. All the glyphs in a normal text encoding are designed to be visually compatible with each other. This should also be the case in a math encoding. But all the compatible glyphs cannot live in the same font. There are simply too many of them. So one has to make a choice. Which glyphs must be alike? A lot of groups result from this constraint, which even comes into play when putting the groups together into encodings. A good example is the sim group. 'Sim' is the name given to the glyph: $\sim$. Many mathematical symbols contain such a sim. $\approx$ cannot be separated from $\sim$ because they must look alike, and for that they must be designed by the same person. Even more, the $\sim$ and the $\approx$ should be produced in metafont using the same sub-routine, with the same parameters. This also explains why it is important that the letters of a given style all live together.

[2] I've put in appendix A, C, and B, three of the documents that I wrote for this purpose.

Charlists. The reader is advised to re-read section 2.4.1 if he no longer remembers what charlists are. The information that such and such a glyph is part of a charlist is in the `tfm` file. Therefore charlists are also restricted to one font. Because of this all glyphs that are intended to be linked in a charlist must be put in the same font. Concerned by this restriction are:

- Wide accents, which are linked with a charlist in order of increasing size,
- Big delimiters: same as accents,
- The two sizes of big operators which are linked,
- All the different sized radicals.

Extensibles. In case of memory deficiency the reader is advised to take another look at the relevant passage in section 2.4.1 again. As for charlists, the extensible information is part of the `tfm` file. The different pieces of an extensible glyph must therefore live in the same font. Concerned by this restriction are:

- Extensible delimiters (not all delimiters are extensible). This constraint is doubled by the fact that an extensible delimiter is often the last element of a charlist. Thus many glyphs must live together.
- Radicals: the last element of the radical charlist is an extensible: it grows as high as necessary. In the same way as delimiters, the glyphs used to build the extensible radical are a subgroup of the radical charlist group, and therefore must live with the other members of the charlist.
- Vertical arrows or bars[3].

Constructed symbols. Some glyphs in a font are especially designed to be put next to each other. Good examples are the horizontal arrows, and the horizontal curly braces. Because of their horizontal characteristic, the extensible mechanism cannot be used. So the

$$\underbrace{\text{horizontal curly brace}}$$

is built up with abbuting glyphs. These glyphs must be of the same weight, and very well adjusted in order to fit together properly. They must therefore live in the same font.

3.4 Constraint importance

The design constraint is less important than the kerning constraint. Whereas keeping empty slots for ligaturing has very little importance compared to the two former constraints.

Charlists' and extensible lists' members **must** stay together, without exception. One could establish the following order of importance:

1. Extensibles,
2. Charlists,
3. Constructed symbols,
4. Kerning,

[3]But not horizontal arrows.

5. Ligatures,
6. Design similarities,
7. Empty slots for ligatures.

3.5 A few groups

- The Greek letter sets,
- The Greek-like glyphs,
- The Latin letter set,
- The Latin-like material
- The digits,
- The vertical arrows,
- The horizontal arrows,
- The accents, wide, double, underaccents,
- The core symbols: must live with the default alphabet,
- The subset group,
- The greater than group,
- etc ...

A lot of the above groups were still divided into smaller groups in order to make things fit in the encoding tables. Compromises had to be made, in order to respect the constraints set by compatibility.

Chapter 4

Making encoding tables

Similarly to the constraints governing the grouping, the constraints governing the division into encoding tables listed below were not at all obvious, and had to be thought of, and fully understood.

The construction of encoding tables largely depends on the main goals of a new math encoding.

4.1 The constraints of group grouping

Glyphs access. (alphabets, variable family) This is another technical constraint due to the way TeX accesses glyphs. It is also a user interface constraint, because the idea is to make alphabets easily accessible to the user.

Due to the variable family mechanism (explained in section 2.3.3), It is very practical for the user that font encodings contain only one alphabet. Thus when the fonts are loaded into the families, different letters can be accessed by changing the `\fam` variable, and typing the usual letters on the keyboard. For instance, when `\fam` is equal to -1, the default family is used. When `\fam` equals 2 the user can get the script alphabet. For this the user need only type `$\fam=2 A,B,C$` and the letters $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are produced.

The alternative would be to have many alphabets in one encoding. In that case, to access script letters A,B,C for example, the user would have to type `$\scriptA,\scriptB$`. That would be much more difficult to read, and less practical.

This constraint – due to glyph access – sets the shape of the whole M-encoding and has a very high priority.

Font access. This only concerns the font that will replace `cmex`. For compatibility reasons, the math font group decided that it would be reasonable to try and replace `cmex` by a font that can be loaded in one size, *and* in three sizes. Therefore the `cmex` replacement can only take:

- Wide accents,
- Big delimiters,
- Big Operators,
- Radicals (with a small change),
- Vertical extensible arrows.

It would be too long to justify the decision here, but the relevant document is in appendix C. One of the consequences of this is that one cannot put an alphabet in `cmex`'s replacement encoding. An aphabet must be available in all three sizes. Other glyphs are also victims of this limitation.

Kerning. Obviously, if glyphs in two separate groups must be kerned, then those two groups must live together.

Design similarity. This is a designor's constraint and therefore has low priority. Because of this low priority, it often happens that big design similarity groups are subdivided into smaller ones. In such case one must try in sofar as is possible to put the smaller groups back together.

Bold face. Mathematicians and physicists often use boldface glyphs. These can either be directly available in some of the encodings, whereby the encoding will specify: here should go a bold uppercase 'A' — and that could be next to a non-bold glyph; or none of the encoding tables specify whether or not the glyphs are bold, and a bold version of the whole M-encoding or of each encoding table can be made — as with text fonts.

To reduce the total number of glyphs in the M-encoding, the second possibility has been chosen. But this induces another constraint on the global M-encoding: the individual encoding tables must be designed in such a way that the most commonly used bold glyphs are put together.

Compatibility with other font-using programs. Since the invention of ASCII code, the first 32 slots of fonts were often not used for glyphs, but reserved for control codes. Today many programs are still not designed to use the first 32 slots of a font. Thus fonts should not contain any glyphs in those slots. But this would be a big waste for TEX, because it can use glyphs in slots below 32.

However, if the glyphs in the critical slots do not have any kerning relation-ship with other glyphs in the font, then the former can be put in another font, and be used with little difficulty even in problematic software. This seemed a fairly good compromise, so it was decided to fill slots below 32 with glyphs that do not have any kerning with the others, and could thus be separated from them.

On the same lines: some programs are unable to use fonts that do not have a space in position 32. To solve this problem, only one slot in concerned, so it was decided to include a space in every font. This should not be a problem.

Grouping TEX specific glyphs: another compatibility issue. The present `cmex` font/encoding contains glyphs that cannot be used by other typesetting systems, because they are set in a strange way. Similarly `cmsy` contains one glyph that is set in a strange way: the radical sign. Therefore the whole of `cmsy` is unusable for other programs. Such a mistake must not be reproduced.

It is hoped that the new TEX math encoding will set a standard, that will not only be used by TEX, but by all systems that typeset mathematical formulae. If everything 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 M-encoding. Thus it will be very easy to use the same fonts on different systems. One day a TEX user will be able to take a mathematical font from 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 next to different math fonts. 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 changes.

From a commercial point of view, if a font designer creates a math font for Adobe, the work necessary for adapting it to TeX must be reduced to the minimum. Otherwise nobody will provide any new math fonts for TeX.

TeX specific glyphs are the following:

- The delimiters,
- The large and small 'bigops',
- The radicals.

Compatibility with Plain and LaTeX. Let us consider a user that has typed a document with the present math encoding, and in so doing has saturated the available families. If the new math encoding does not garantee Plain and LaTeX glyph compatibility with a *maximum of 4 fonts,* then the document will not be able to run with the new math encoding: not enough families. Thus one should make the first four encoding tables of the global M-encoding glyphs compatible with the Computer Modern cluster: cmr, cmmi, cmex, and cmsy.

Compatibility with AMSTeX, AMSLaTeX, and LAMSTeX. Let us consider this time a user that has typed a document with the existant AMSTeX or AMSLaTeX package, and in so doing has saturated the available families. If the new math encoding does not give AMSLaTeX and AMSTeX glyph compatibility with less than 6 encoding tables, then that document will not run with the new math encoding, for lack of family reasons.

The first 6 encoding tables must be one way glyph compatible with the fonts provided in the AMS packages.

Trying to give the Plain TeX user a logical cluster of new glyphs. No comment.

4.2 The Aston-LC math encoding

This is one of the proposals first thought of, but it is not the one finally chosen, because it had many problems.

LC stands for latin core. The main characteristic is the separation of the Greek letter sets from the Latin ones. In keeping these two sets separate, we give the greek letters an identity of their own, thus making them quite independent of the rest. The idea goes in the direction of orthogonal grouping. All the encodings that contain letters would have them in the Cork encoding positions, thus making access very simple. In fact this positioning concept will be taken farther: Cork encoded glyphs that are in the new encoding, will keep their Cork position.

4.2.1 The encoding tables

The text symbols: the TS encoding. Here would be included the old style numerals, and most of what is to be taken out of the present math encoding, because it does not belong with the rest of the math glyphs. Other symbols could be added in this encoding.

This encoding is not part of the M-encoding, but it will contain symbols that previously were accessed via the math fonts. In normal usage, this font will not be loaded in a family. It will simply be loaded as a normal text font.

The base: a Cork encoded latin text font. The main use of this font would be to typeset function names like `\log`. The idea being that the user can actually choose this font among the existing Cork encoded fonts. Thus 'sin' can actually be typeset in the same style as the text, or in another special style to match the rest of the math glyphs.

The core: the MC encoding. It would not contain any Greek glyphs (unlike `cmmi`). The basic accents (only one size) would be here, next to the default numerals. It would also include all the upper and lowercase default latin alphabet, all of the symbols that are most commonly used, and glyphs that must be kerned with the default alphabet.

The Greek alphabets: the MG encoding. This encoding table would contain all the upper and lowercase Greek letters in upright and italic, plus some variable shape Greek letters, also in upright and italic, and some numeric Greek letters. Any other Greek related glyphs would also live in MG. If place is still available, one could include some symbols. An advantage of putting the italic Greek and upright Greek together, is that both are often requested in medium and in bold weight.

The extensibles: the MX encoding. This encoding would look very much like the present `cmex` encoding: the usual extensible characters, together with some new ones. It could include any characters that have strange TeX features like big descenders. Thus glyphs that are not compatible with the outer world would be kept together.

The math symbols: the MS1, MS2, MS3... encodings. Each of these encodings would contain a set of Latin letters, like for instance script or blackboard bold, in upper or lowercase, or both, together with a set of matching accents if needed. In some cases a place should also be reserved for a set of matching numbers. The rest would be filled up with symbols. There could be an MS_i encoding for:

- Calligraphic,
- Script,
- Open,
- Old german, (Fraktur)

4.2.2 Other requested typefaces

- A "text-like" italic or slanted font for computer science identifier-names and the like. This would be Cork encoded.
- 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.
- Bold italic for use as variables: an MC or Cork encoding.
- Bold Old german (occasional).
- Bold script (occasional).

- Sans serif lightface (occasional): Cork encoded font.
- Sans serif boldface (occasional): Cork encoded font.
- Bold symbols: the same encodings loaded in bold.
- Ultra bold symbols: the same encodings loaded in bold.

4.2.3 Summarizing the family occupation

The following encodings are needed in the kernel:

1. A Cork encoded upright text font.
2. An MC encoded font containing the default alphabet, digits, accents, and symbols.
3. An MS_1 encoded symbol font for calligraphic/script.
4. An MX encoded extensible font.
5. An MG encoded font for Greek italic and upright.
6. An MS_2 encoded symbol font for Open and symbols.
7. An MS_3 encoded symbol font for Old german and symbols.

This occupies 7 families, and leaves 9 free for anything else, (like bold or sans...) and makes many symbols available.

4.2.4 Pros and cons

This proposal did not respect the limit of 4 and 6 families (compatibility with PlainTeX and AMSTeX), nor did it enable the Latin and Greek to be kerned together, nor could the Greek be kerned with the same symbols as the Latin alphabet, unless these were repeated. Generally, to get the equivalent of Plain TeX, one would have had to load 5 families, and to get the functionalities of AMSTeX, one would have needed to load 7 families.

One of the advantages was the orthogonality of the individual encoding tables, i.e. there were no strange mixes like Latin and Greek, or anything of the sort.

The main reasons for rejecting this proposal are:

- it is a big family consumer. In particular bold Latin and Greek would occupy two extra families, and they are frequently requested.
- it does not enable kerning between the Greek and punctuation which is needed for compatibility — the punctuation is in a separate encoding table from the Greek.

The next proposal is more attractive...

4.3 The Aston LGC math encoding

LGC stands for Latin Greek core. One of the main features of this proposal is that the Greek and Latin alphabets have been put together. In one font they could be upright, and in the other they could be italic. A good reason for doing things this way is that the font dimension called slant may give a few unexpected problems if italic and non italic glyphs are mixed in the way that they would have been in the Aston LC math encoding.

4.3.1 The encoding tables

The text symbols: the TS encoding. This would be the same as in the previous proposal, and would be used in a similar manner.

The base: a Cork encoded latin text font. This would be the same as in the previous proposal, and they would be used in a similar manner.

The core: the LG encoding. Instead of the MC encoding (in the previous proposal), the core could be duplicated. Once in upright, and once in uppercase. The LG encoding would contain one instance of both Latin and Greek letter sets. So two LG encoded fonts would be used (upright, and italic).

As far as the other slots are concerned, they could be filled in with the most used math symbols (similarly to the MC encoding), these would then appear once in upright, and once in bold. An alternative to such a duplication would be to make an LG1 encoding that would contain different symbols from an LG2 encoding, and these would always be in upright, whereas the letters would be specified as italic in LG1, and upright in LG2.

Note. The user could choose whether he wants to load both LG1 and LG2, or only one of the two.

The extensibles: the MX encoding. It would be the same as in the previous proposal, and it would be used in a similar manner.

The math symbols: the MS1, MS2, MS3... encodings. These would be the same as in the previous proposal, and they would be used in a similar manner.

4.3.2 And the rest?

Similarly to the previous proposal, many other fonts could be loaded in all the free families.

4.3.3 Summarising the Family occupation

1. An LG encoded font containing Latin and Greek italic. (This could be LG1 if necessary. See explanations above.)
2. An MS_1 encoded symbol font for calligraphic/script.
3. An MX encoded extensibles font.
4. An LG encoded font containing latin and Greek upright. (This could be LG2 if necessary. See explanations above.)
5. An MS_2 encoded font for Open and symbols.
6. An MS_3 encoded font for Old German and symbols.

Only 6 families are occupied. This leaves 10 families free for anything else, (like bold or sans...) and makes many symbols available.

4.3.4 Pros and cons

One of the advantages of this proposal is that kerning can be done between Latin and Greek (as long as they are in the same shape), and between Greek and other symbols present in the encoding such as punctuation. Also when bold is requested, one gets the bold Latin and the bold Greek in the same font table, which again consumes less families than having the two separate.

This proposal occupies less families than the previous one.

Reasons for abandoning the Aston LGC math encoding:

- The user must be able to choose the look of his log, sin, and friends. He may want them to be either text compatible, or compatible with the other math alphabets and the rest of the math glyphs in general. The choice must be left open, and the math font designer must not impose his decision on the user.
- A solution to the previous problem is to include another font for this purpose, as in the previous proposal. But then the family occupation rises up to 7, and three Latin alphabets are loaded, of which one (the LG upright) is probably not going to be used much. Thus a lot of precious space is wasted.
- The 'Yaasp' proposal is much more attractive.

Chapter 5

The proposed YAASP encoding

This chapter is the final proposal that was made. It is also the body of the official document that was produced. The reader may find some similarities with the previous sections, for instance some of the definitions can be found in section 3.1. Also a lot of the points discussed in the global policy section have already been discussed.

5.1 Introduction

This document aims to put on paper what could be the backbone or the skeleton of a new math encoding for TeX. 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 many glyph groups defined in another document called "Towards a list of math glyphs". Same author.

5.2 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 to "encoding".

A "slot". It is the usual word used for referring to a position in an encoding, that can contain a glyph. It is usually an integer between 0 and 255. A slot is certainly not a family, nor 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 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". It is considered here as a whole; not just one 256-glyph encoding table, but a set of encoding tables. This concept will be referd to as "M-encoding".

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

The "core symbols". They 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.

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

"Glyph compatibility". Two encodings are glyph compatible when they contain the same glyphs but not in the same positions. As well as containing the same glyphs, it must also be possible for the corresponding fonts to contain the same sidebearings and the same kerning and ligaturing information.

5.3 Global policy

5.3.1 Text in math mode

It is generally agreed that for best quality documents, if one wants to put text in a math formulae, one should switch back into text mode, using something like the `\text` macro in the AMS package. None of the encodings can be expected to support bad usage. In a math encoded font, the letters are not kerned in the same way as in a text encoded font, and there are no letter ligatures, because they are not needed.

5.3.2 A global rule for boldface

It has been decided not to mix ligth face and bold face symbols in the same encoding, but to generate a separate boldface version of all lightface math encoded fonts when necessary. This does not make it impossible to mix the two: either one can use the `\boldsymbol` approach, or one can load an extra bold face font in a given family, and have it directly and permanently accessible.

5.3.3 Sans serif and typewriter fonts

Extra fonts could be designed in sans serif, or in typewriter using some of the proposed new encodings. Another solution is to load the Cork encoded sans serif fonts (or typewriter fonts) in free families. In either case, the new math encoding will not have any slots containing specifically sans serif or typewriter glyphs.

5.3.4 Concerning the Euler shapes

Euler shapes could be a good example implementation of the new math encoding. Thus no Euler glyphs will be included in the new encoding. However, it may be useful to use the Euler Fraktur for a first implementation example, if the new encoding includes a Fraktur or old German alphabet.

5.4 Concerning Cyrillic letters

These would be available, but not as part of the math encoding. They would be loaded as an extra family, with whatever encoding exists, together with suitable `\mathchardef`s.

5.4.1 Compatibility with other typesetting systems

Grouping all TeX specific glyphs in one font encoding

The present `cmex` font contains glyphs that cannot 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. It would be a good idea to make sure that this does not happen again.

The LaTeX3 project is hoping 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 about.

Concerning this problem, the real question is: is the LaTeX3 project setting a real standard for the next few years, or just making another TeX math font encoding? If the answer is: "the LaTeX3 project is setting a real standard for the next few years", then TeX specific glyphs must be grouped in one font. If that is really not possible, then one can maybe consider putting them in two fonts.

If a font designer designs a math font for adobe, the work necessary for adapting his font to the TeX world should be reduced as much as possible. Otherwise nobody will provide any fonts for TeX.

The TeX specific glyphs that are concerned here could be visually compatible with many math fonts.

Which are the TeX specific glyphs? So far:

- The delimiters.
- The large and small 'bigops'.
- The radicals.

The space issue

To enable easier font exchange between the TeX world and the rest of the world, the new math encoding will have a space in position 32 (decimal) of every encoding table, if it is possible.

5.4.2 General document compatibility

It is not worth beeing totaly compatible. A lot of glyph positions will change, thus direct `\mathchardef`s will not always work. Documented names from AMSLATeX (this includes names from LATeX, TeX, and AMSTeX, and LAMSTeX) ought to be kept.

5.4.3 Grouping all Plain and LaTeX glyphs in 4 fonts

The main reason for this is compatibility. Let us 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 guarantee Plain and LaTeX glyph compatibility with a minimum of 4 fonts, then that document cannot run with the new math encoding: not enough families. Thus one should make the first four font encodings of the new math encoding glyph compatible with the group made by: cmr, cmmi, cmex, cmsy.

5.4.4 Grouping all AMSTeX and AMSLaTeX glyphs in less than 6 fonts

The main reason for this is compatibility. Let us 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 guarantee AMSLaTeX and AMSTeX glyph compatibility with less than 6 fonts, then that document cannot run with the new math encoding: not enough families. The first 6 font encodings must be one way glyph compatible with the fonts provided in the AMS package.

5.4.5 Replacing `cmex`

The math font group has decided that the encoding due to replace the `cmex` encoding will be designed in such a way that the corresponding font can be loaded in three sizes or in one size.

Loading such a font in three sizes produces better typesetting. But the resulting page and line-breaks will not be the same as when the font was loaded in one size only. Some people will not like that change, in their old documents. But for new documents loading the extensibles font in three sizes will be better.

5.4.6 Accents in maths

There seems to be an agreement that math accents should not change with the font or style of the letter. But it is not a problem to keep the accents that are already in plain TeX (and the `cm` fonts) in the same position as in the T1 encoding. That will allow them to be variable family; thus any T1 encoding could be loaded, and its accents used. Since there will be millions of documents using bold hats this possibility must be preserved even if by default all math accents are non-variable family.

5.5 The base: a Cork encoded text font

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

If it is a Latin upright font, it would probably also be used by physicists (and chemists) for operators, and more generally whenever upright letters are needed.

Separating this set from the rest enables the user to decide how 'log' and 'sin', etc. should be typeset. Thus the multiletter operators can be compatible with the text font, or with the rest of the math glyphs, or even set in yet another font.

Math mode should not be used for setting text phrases in mathematical material. For example:

```
$$x=y \quad {\rm is\ a\ direct\ consequence\ of} \quad y=x$$
```

would be better input as:

```
$$x=y \quad\hbox{\rm is a direct consequence of}\quad y=x$$
```

or, better still:

```
$$x=y \quad\text{is a direct consequence of}\quad y=x$$
```

where `\text` is a macro which sets its argument in horizontal mode. This avoids unpleasant surprises such as:

```
$$X \quad{\rm is\ a\ sub-object\ of}\quad Y$$
```

However, for upward compatibility with existing plain TeX documents, it will still be possible to set text phrases in math mode, as long as they only contain ⟨character⟩s of type Variable.

5.6 The "text symbol" encoding: the TS encoding

Here one would put the old style numerals, and most of what is comming out of the present math encoding. Other symbols could be added. The Text Symbol encoding is definetly *not part of the math kernel*. But since it will contain symbols that previously were accessed via the math fonts, its encoding must be supplied. This font will not be loaded in a family. It will just be loaded as a normal text font[2].

5.7 The core: the MC encoding (263)

Counting: 1,10,1 , 54,5 , 124,14 , 12,24,9 ,9= 263 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 elsewhere. 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: 5
- All the Greek material: 124
- The Greek friends, next to the Greek: 14
- The core symbols for kerning reasons (punctuation and delimiters): 12
- The core symbols for design reasons: 24

  More for kerning reasons:
- The basic geometric delimiters: 9

  Some new glyphs:
- New basic delimiters: 9

Sacrifices can be made in the greek material, and in the core symbols for design reasons.

[2] If some users really feel the need to load it in a math family, they can.

5.8 The MX encoding: 243

Count up: 1 1 78 8 10 24 6 16 7 7 26 47 = 230

The usual extensible characters, together with some new ones, would live here. Here any characters that have strange TeX features, like big descenders are included, thus grouping glyphs that are not compatible with the outer world.

For various reasons discussed in "Replacing `cmex`?" (Same author), the math font group has decided that the MX encoding will be designed in a way such that when it is loaded in one size (like in the present TeX), every thing works OK, and the user can still have access to the new symbols. However, the MX encoding will produce better quality typesetting when loaded in three sizes.

Detailed contents of MX:

- Maybe a skewchar: 1

  The space is questionable here, because MX will not be usable by other typesetting systems, see comment:
- Maybe a space: 1
- Big and extensible TeX delimiters from cmex: 78

  Any characters that have strange TeX features like big descenders:
- The radicals: 8

  Technically the following can come out, but then they must go in MSP. To make this possible one could take the bar accent out of MSP.
- Horizontal curly braces: 10
- All existant big and small "bigops" except the integrals: 24
- The existant Plain vertical extensible arrows: 6

  One has to limit the number of wide accents, otherwise there is not enough place.
- The wide tildes, and the wide hats: 16

  New glyphs:

  It is a little bothering that the following will be separated from their small versions, but there is nothing much that can be done 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: 47

Note: all integral glyphs must be kerned with themselves, so that two integrals following each other can be kerned easily with a little care. Same for various other glyphs.

5.9 The math symbol 'privilege' font "MSP": 250

Count up: 1,1, 54 , 18,7,3 , 23 , 8,4,2,4,4 , 20,7, 10 , 8,8,12 , 14 , 6,4,5,4,5 , 8,2 , 16 = 250

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

- The script/calligraphic Latin letter set:54

  According to AMS statistics, the script/cal are used more often than the Blackboard bold.

  Having the script/calligraphic here also achieves backward compatibility with the existing TeX, without loading MS1 and MS2.
- The basic accents: 18
- The double accents: 7
- The underaccents: 3

  The following 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 These should probably 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": 14 or 10
- "Plain vertical arrows" : 6
  Does not include the extensible arrows. The latter are in MX as before.
- "Plain oblique arrows": 4
  Also called "Plain other arrows".
- "Latex arrows": 5
- Plain miscellaneous geometric symbols: 4

  Extras — these are new glyphs — to improve a little what TeX can already do:
- Extra arrows for use with plain: 5

  The "lasy" triangles are included in the AMS fonts and thus are included in the following group:
- AMS left-right open triangles: 8

  Should live with the "Plain oblique arrows":
- "AMS obliques":2

  Some new glyphs: Some of this could come out.
- Wide accents bar: 8

5.10 The MS1, MS2, Math-Symbol encodings

Each of these encodings will contain a set of Latin letters, like for instance fraktur or blackboard 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. An MS_i encoding is needed 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 writes "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 there are two possibilities:

1. designing one encoding table where the positions A-Z (and probably a-z and 0-9 even if they are not 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 family 4. A designer would then choose one such font with this encoding for family 4 (thereby effectively deciding what `\cal` and a lot of other symbols look like (the ones whose `\mathchardef` points into family 4)). 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. Having 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 is preferable depends a bit on the number of symbols that ought to go in the standard."

Also Jörg Knappen writes: "I strongly support having two different encodings, one for cal and one for script. If users have the choice between cal and script, they prefer script (at least in Mainz[3]). However, the old calligraphic alphabet still needs to be supported for compatibility reasons."

5.11 The MS1 encoding: 232

Count up: 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:

[3]Maybe Americans prefer it the other way round.

6. The “AMS horizontal arrows” group: 22
7. The AMS other arrows group: 12
8. The “Greater than AMS” group: 30
9. The “Subset AMS” group: 12
10. “AMS Equals friends”: 10
11. “AMS Miscellaneous geometric symbols”: 21
12. “AMS Vdash group”: 10
13. “AMS boxes and friends: 15

    For fun if there is place to spare:

14. Alan’s arrow construction set: ?

5.12 Other requested typefaces

- In general, users may want MC fonts in arbitrary styles (bold sans serif 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 which case no family would be required.
- Bold italic for use as variables: either optionally loaded as a second font with MC or cork encoding (using only variable family 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).
- A cyrillic alphabet. Loaded as an extra family, or in text.

5.13 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.

Comments:

- This leaves 10 families free for anything else, (like bold or sans...) and makes many symbols available.
- The first four encoding tables put together give total TeX, LaTeX compatibility.
- The first six give total TeX, LaTeX, AMSTeX, AMSLaTeX, LAMSTeX compatibility.
- The six put together: do wonders, using no more font families than the present AMSTeX.

5.14 Discussion

5.14.1 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. This last point is the most important, and gives new and better automatic math spacing. (For compatibility reasons, the Greek italic must be kerned 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 consumption 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.

One can get more than compatibility with plain TeX only 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 than the open, so putting it with the accents is a step towards grouping most used glyphs together.

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

With the MSP encoding concept, the MSi encodings can really be considered as (optional) extensions. Thus somebody who knows he does not 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 family 0 for things like `\log` and `\sin`, means that the Greek users can replace it by a Greek font. (Apparently Greek mathematicians set these function names using a Greek alphabet).

5.14.2 Disadvantages

If there is not enough space for all the required symbols, one can make an MS3 encoding that would contain the other version of script/cal, together with extra symbols.

5.14.3 Comments

In this proposal the core is really made of two fonts, and the kernel is made of four.

Comments from Alan about the space slot in MX:

> "MX will be used by TeX, and the dvi drivers may be outputting to a device that does not 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 `\fam 0` (naughty them!) and in order to have filled up slots 0 to 3 rather than leaving a gap in family 0."

Chapter 6

The glyph groups

6.1 Introduction

The goal of this document is to try and list the future contents of the new math encoding. The glyphs have been divided into groups and subgroups.

This is a draft and a workbench document. Some of it is out of date, and usually the comments are not for the public. There are lots of spelling mistakes, I know! The reader should be indulgent.

6.2 Extra font dimensions

- The design size,
- The default script size,
- The default scriptscript size,
- Suggested value for mathsurround (in MC)
- Math_axis (in every font)
- Thin mu skip,
- Med mu skip,
- Thick mu skip,
- Recommended rule weight
- Baselineskip: leadingheight,
- Baselineskip: leadingdepth
- Suggested by JMR: the big and bigg params.

6.3 Kerning

Better kerning should be made possible in the Latin math italic, if it is possible. Normal kerning information is put in the `.tfm` file. But in math mode, for things to get kerned as specified in the `.tfm` file the left atom must be of ordinary type. If the user interface redefines everything that must be kerned as being ordinary, old documents will start looking different, and this is not wanted. To avoid this, the user interface could define a macro `\mathkerning{...}` that would use the kerning specified in the `.tfm` file, without globally making all characters ordinary.

The following glyphs should be kerned:

- The spacing of [and (and) and] followed by letters should be adjusted.

- The spacing of letters followed by [ ] ( ) should be adjusted. This is an important one.
- Keep the kerning with . , / for most letters !
- At least keep the kerning between d and Y,Z,j,f. Maybe add some others: dx, dy, dα, dθ, dϕ ...
- New kerning of the integral with itself. This would only be available via a `\mathkerning` macro (see previous comment).
- Kerning of the period with itself,
- Kerning of the centered period with itself.
- Jan M.R. is sure that kerning is needed between Latin and Greek. More precise information is needed.

6.4 The following should be taken out of the present math encoding

- The old digits: 10.
- The 2 paragraph signs: ¶, §.
- The Yen sign: Y.
- The double dagger sign ‡.
- The four card families: $\clubsuit, \heartsuit, \diamondsuit, \spadesuit$.
- The musical signs: $\flat, \natural, \sharp$?????????
- The maltese cross. (AMS)
- The ⌢ seems not to be needed in maths.
- The circled R must come out of the math symbols. (AMS)
- The raised asterisk,
- The two triangles in cmmi: cmmi'56 '57 ???????

All these will be put in to the “Text symbols” encoding, that would come in many faces, and be text dependent. Other glyphs could be put in the “Text symbols” font:

- More numerals,
- The perthousand sign.
- Maybe this is a good place for the ‘fraction’ characters from adobe.
- <florin>, <ellipsis> etc.
- The superior and inferior digits, and put in kerning so that <onesuperior> <fraction> <twoinferior> produces a 1/2.
- The single dagger finds a place here although it is in maths as well. This makes them two different symbols, and enables both to have more specific shapes.
- A real copyright symbol, TM (trademark) and SM (service mark).
- An interrabang (a combination of ? and !) new. (bb)

Alan Jeffrey has worked on the ‘text symbol font’. Actually it is now called the companion text font. He has written more on this topic. “alanje@cogs.susx.ac.uk”

6.5 The Greek glyphs: 124

The following shapes must be included:

- All the Upper-case in upright. 24
- All the Upper-case in italic. 24
- All the Lower-case in upright. 24
- All the Lower-case in italic face. 24. So far: $24 \times 4 = 96$

- All the variable shapes in upright. 10
- All the variable shapes in italic. 10
- All the special numeric letters in upright. 3
- All the special numeric letters in italic. If lack of space prefer the italic shapes to the upright ones. 3.
- Some control glyphs: 2 (probably useless)

6.5.1 Variable shapes: 10

This list comes basically from: Jörg Knappen. They are all listed here including the ones that are already in the `cm` fonts:

1. Lower-case Phi,
2. Lower-case Pi,
3. Lower-case Kappa, (AMS)
4. Lower-case beta (new),
5. Lower-case Rho,
6. Lower-case Epsilon,
7. Lower-case Sigma,
8. Lower-case Theta.
9. Upper case chi (new),
10. Upper case for upsilon.

6.5.2 Extra letters for numerals: 3

Source: Jörg Knappen. They are all listed here including the ones that are already in the `cm` and `ams` fonts. Must be given in lower-case. Upper-case Greek numerals exist, although extremely rare. For the sake of completeness one could fill them in. But they are surely not the hottest characters needed. (Jörg)

1. Qoppa (new),
2. Sampi (Sanpi?) (new) (Jörg Knappen),
3. Digamma (AMS).

6.5.3 Control glyphs: 2

1. An italic control glyph, i.e. the following Greek letter is not taken from the upright, but from the italic Greek,
2. A variable shape control glyph, i.e. the following Greek letter is not taken from the normal set of letters, but form the variant shape set. This will not work for all letters. Thus may not be a good idea.

Note: From Alan about the control slots for Greek, "Er, I'm not very sure about those, since they'll affect kerning. I'd prefer to have the choice between italic / upright made by the document designer. And I'm not sure why anyone would want to get at an upper case ξ by a macro `\uppercasegreek{\xi}`!" — "True they will affect the kerning. But one could use them differently from what you suggested. Although I'm not sure it is interesting, the ligature mechanism does not have to be visible for the user, i.e. he can still type `\Gamma`, which is expanded to `\up\gamma`."

6.6 Extra Greek-like material: 14

This group of glyphs should not really be separated from the rest of the Greek material.

1. An upright partial sign,
2. An italic partial sign,
3. An upright partial sign with a slanted bar, AMS `\eth`
4. A `\thorn` WASY'151 but this one is not very good. There is a better one in dcmr'136. There is one is wslipa10'102 (Jörg)
5. A barred upright lambda, ?????? (probably not Jörg)
6. A barred italic lambda, this is preferred. (Jörg)
7. An upright mho sign (upside down Omega),
8. The back to front epsilon: AMS "7F `\backepsilon`,
9. Arabic letter dal: looks something like a back to front c.
10. Hebrew letter msbm'151,
11. Hebrew letter msbm'152,
12. Hebrew letter msbm'153,
13. The $\aleph$ or `\aleph` in position CMSY'100,
14. The Nabla, ∇ in CMSY'162

The barred signs may be obtained by ligatures, or could be constructed with kerning. In any case some slots for ligatures must be left free if possible.

6.7 The Latin letters: One set= 54 glyphs

We shall assume here that all lower case alphabets contain a dotless 'i' and a dotless 'j', so that they can take accents other than a dot.

- The usual `cmmi` italic shapes. Upper-case and lower-case.
- The calligraphic shapes. Upper-case and lower-case. The lower-case shapes are presently maybe not available.
- The script shapes. Upper-case and lower-case. The lower-case shapes are presently maybe not available.
- The black board bold shapes. Upper-case and lower-case.
- The Fraktur style. Upper-case, and lower-case.

6.7.1 The calligraphic and/or script styles

BB: "How are "calligraphic" and "script" different here? I've never seen what Knuth calls calligraphic and what most mathematicians call script (the "curly" style) used in the same context, so they are presumably not distinct from one another in actual usage."

The two should be included if there is enough space. Otherwise one is enought.

6.7.2 A hyphen char ?

These Latin letters are not meant for typesetting words. It is assumed that all multi-letter words should be typeset using the text fonts, not the math fonts. Thus the hyphen character is not needed in the math encoding.

6.7.3 Computer science and identifiers

It looks as though the new math encoding will not contain anything specially designed for computer science. Computer scientists will have to use `cmti*` in an extra family for long identifiers.

6.7.4 Chemists and chemical formulae

Considering the fact that chemists do use a lot of mathematical expressions, they need the total math mode as it is. On top of that they need a special mode for writing chemical equations. One of the particularities of this chemical mode would be the different placing of sub- and superscript. A possible implementation is something like `\EnterChemicalMode` and `\ExitChemicalMode`, which would in actual fact load a new set of fonts (or only the font in family 2), in order to have a different font dimensions in family 2.

6.8 Latin-like material: 5

This group should live next to the Latin letter set.

1. An upright d. This is needed for standard mathematical typesetting.
2. A horizontally barred italic h, for physicists.
3. A slanted barred italic h, for physicists.
4. An italic upper-case Vee with a bar, the bar is meant to be horizontal. `jvpurcel@vela.acs.oakland.edu`
5. An upright upper-case Vee with a bar, the bar is meant to be horizontal, and extends through both sides of the Vee almost like a strikeout. `jvpurcel@vela.acs.oakland.edu`

6.9 The different ways needed to write numbers

- The normal set of numbers in `cmmi`: upright lining.
- The blackboard bold numbers. (Used in physics and a field of maths. See Alan J. for more details.) [Note: presently no satisfactory bbb numbers seem to exist.]

6.10 Empty slots?

Some free slots could be included, so that people can put their ligatures in when they are trying to convert fonts coming from other worlds.

Alan J. can give good explanations for this.

6.11 Arrows

Arrow construction should be possible. But to make sure it does not fail when used in different sizes, every single glyph used for this purpose, will be *specifically* designed for this use. *All of them will be in the same font table.* This does not mean that a given construction block can't be used for different types of arrows. This sort of thing has to be thought of, and forecasted. These construction blocks must not be used for any other purpose — like for instance the equal or minus sign.

All arrows from `cm`, and from `msam/msbm`, should be taken if necessary. Maybe some others too.

6.11.1 The "Plain horizontal arrows" group: 14 (Alan:10)

The first 6 are in cmmi'050 to '055:

1. leftharpoonup

2. leftharpoondown
3. rightharpoondown
4. rightharpoonup
5. lhook
6. rhook

   From `cmsy`:

7. leftarrow '40
8. Leftarrow '50
9. leftrightarrow '44
10. Leftrightarrow '54
11. rightarrow '41
12. Rightarrow '51
13. CMSY'67 this is the `\mapstochar`
14. CMSY'66 the negation sign/slash: 1

6.11.2 Extra arrows for use with plain arrows: 5 (Alan 5)

1. It would be reasonable to add a `\mapsfromchar` in order to build things like: <–|: 1
2. It would be reasonable to add a `\Mapstochar` that could go with the double arrows to build things like |=> : 1
3. It would be reasonable to add a `\Mapsfromchar` that could go with the double arrows to build things like <=| : 1
4. A - for extending arrows: 1
5. A = for extending arrows: 1

6.11.3 The "Plain vertical arrows" group: 6 (Alan 6)

- updownarrow cmsy'154
- Updownarrow cmsy'155
- uparrow cmsy'042
- downarrow cmsy'043
- Uparrow cmsy'052
- Downarrow cmsy'053

6.11.4 Plain vertical extensible arrows: 6 (Alan 6)

1. Top sing arrow: cmex'170
2. Bottom single arrow cmex'171
3. Top double arrow cmex'176
4. Bottom double arrow cmex'177
5. Middle double arrow cmex '167
6. Middle single arrow cmex '077

6.11.5 Plain extra vertical arrows: 0

Nothing added here.

6.11.6 The plain other arrows: 8 (Alan)

First the oblique arrows:

1. CMSY'45
2. CMSY'46
3. CMSY'55
4. CMSY'56

   What else: ?

6.11.7 The "Ams obliques" group: 2

1. msbm'36
2. msbm'37

6.11.8 The "Latex arrows" group: 5

The four characters in position LASY'50 to '53 from the lasy font (These appear in the wasy font as well) must be put with the arrows. They are arrow heads. The squig \arrow in position '73 of lasy should also be included: 5

6.11.9 The "Ams other arrows" group: 4

- Circle arrows MSAM: '10 to '11 :2
- Horizontal arrows MSAM:'113 '114 :2

6.11.10 AMS horizontal arrows: 22 (Alan 23)

This includes all the horizontal arrows and the negated ones, that are listed page 280 of "The joy of tex".

1. leftarrowtail
2. leftleftarrows
3. leftrightarrows
4. leftrightsquigarrow
5. lefttrightharpoons
6. Lleftarrow
7. looparrowleft
8. looparrowright
9. nleftarrow
10. nLeftarrow
11. nLeftrightarrow
12. nleftrightarrow
13. nrightarrow
14. nRightarrow
15. rightarrowtail
16. rightleftarrows
17. rightleftharpoons
18. rightrightarrows
19. rightsquigarrow
20. Rrightarrow
21. twoheadleftarrow
22. twoheadrightarrow

6.11.11 Ams vertical arrows (Alan: 6) (here: 8)

1. MSAM:'24 upuparrows
2. MSAM:'25 downdownarrows
3. MSAM:'26 upharpoonright
4. MSAM:'27 downharpoonright
5. MSAM:'30 upharpoonleft
6. MSAM:'31 downharpoonleft
7. MSAM:'36 Lsh
8. MSAM:'37 Rsh

6.11.12 Some control glyphs for access to arrows

These do not appear in the `.dvi` file, they simply enable the construction of some arrows and slahsed arrows using the ligature mechanism.

6.12 All sorts of accents

6.12.1 Basic size accents: 18

- All those that are created by macros in the Ams package: the 3 dotted accent, and the 4 dotted accent. 2
- The ones in TEX: e` e´ eˇ e˘e¯ e˚ eˆ e˙ e¨ e˜e⃗. They all come from `cmr` except for the last two from `cmmi`. 11
- Extra: a back-to-front vector arrow, 1
- Extra: a double sided type vector arrow, 1
- Extra: a square bracket used as an accent, 1
- Extra: The previous one turned upside down, 1
- Extra: an arc is requested by AMS, 1

Note: The e˝ in `cmr` is not needed in maths, it is just a Hungarian accent.

Note: The ⌢ seems not to be needed in maths. It could be put in the text companion font.

6.12.2 Double accents: 7

1. A bar and a dot on top,
2. A dot and a bar on top,
3. 2 dots with a bar on top,
4. A bar with 2 dots on top,
5. A hat and a tilde on top,
6. A hat and bar on top,
7. A double bar,

Note: For the double accents, Spivak and Ralf Rey could do some archive research at the AMS. Similar research could be done at the APS, and the CUP.

6.12.3 Variable size accents: $7 * 8 = 56$

Variable size has meant 5 different sizes until today. That number could be raised to 8.

Note: If the accents are in a font loaded in three different sizes, the choice mechanism of `\mathaccent` will only look in the current style size (unlike the delimiter choice mechanism). Thus although one could hope to multiply the number of available sizes by three, in actual fact in a given style the number of automatically available sizes would not be multiplied. All the same this *would* give better results in each style, but it would also create *compatibility problems* i.e. formulae heights and widths may change. Even if not done in an automatic way, the user would still have a larger range of accents to choose from. Compatibility problems could be avoided by redefining `\mathaccent` to a `\mathchoice`. Thus the accents could always come from text style, and the accented material could come from the current style. But this does not work either. In doing so one would no longer be able to take the base accents from the current style. Although one could make two macros. See paper “Repacing `cmex`?”, same author.

1. e⃗ the vector. 8
2. e˜ the tilde. 8
3. eˆ the hat. 8
4. e¯ the bar. 8
5. Some people request a variable size arc. 8
6. The back to front vector arrow, 8
7. The double sided vector arrow, 8

6.12.4 Under accents: 3 so far

Requests exist for the following:

1. A tilde,
2. A breve (˘)
3. A bar

Like for the double accents, research could be done at the AMS... ???????

6.13 Core symbols

The symbols that have some reason to live with the default math material. There are mainly two reasons for them to be there: one is kerning, and the other is design similarity.

6.13.1 For kerning reasons: 12

1. The period . CMMI
2. The coma , CMMI
3. The semi colon ; CMR
4. The colon : CMR
5. The exclamation mark ! CMR
6. The (
7. and the) respectively opening-class and closing-class, CMR
8. The [
9. and the] respectively opening-class and closing-class, CMR

10. The {,
11. And the } (design similarity reasons also) in positions ’146 and ’147 of CMSY,
12. The ‘/’ as a delimiter[1], and as a fraction sign, CMR

6.13.2 Basic geometric delimiters: 9

Should go in the core, for kerning reasons, like the other ( ) and [ ]. If they don’t fit in the core, they must go with the basics. The ones listed here are all in CMSY, around ’142, and ’150.

1. $\rangle$
2. $\langle$
3. $|$
4. $\|$
5. $\rceil$
6. $\lceil$
7. $\rfloor$
8. $\lfloor$
9. The $\backslash$ in position ’156

Test: $|f|$, $\|f\|$, $\lfloor f\rfloor$,$\lceil f\rceil$, $\langle f\rangle$ $f\lceil$, $f\lfloor$,$f\langle$, $f\backslash$ strange that no kerning seems to be needed here, where as it is necessary for the bracket.

6.13.3 New basic size delimiters: 9

Basic size means the same size as the parentheses and brackets in `cmr`. The following is a preference order list of desired new delimiters:

1. A $|||$ for use as $|||f|||$ a norme,

   Semantic brackets: $[\![$ and $]\!]$
2. The opening semantic bracket,
3. The closing semantic bracket,
4. Opening multi set brackets $\{|$
5. Closing multi set bracket $|\}$

   Unicode contains another style of brackets, they call them tortoise shell brackets. They look like:

```
      /
     /
    /
    |
    |
    |
    \
     \
      \
```

   These are like parentheses, but with straight lines. No curves.
6. The opening tortoise shell bracket,

[1] This is not accessible via a single key. The key ‘/’ produces the sign / taken from `cmmi`.

7. The closing turtoise shell bracket,

   Triangle brackets, something like:

```
     /|
    / |
   /  |
   \  |
    \ |
     \|
```

8. The opening triangle bracket,
9. The closing triangle bracket.

6.13.4 For design similarity reasons: 24

All this group must live with the default alphabet for design similarity reasons.

1. The question mark ? must live with the ! CMR
2. The percent sign % must live with the ! and ? CMR
3. The at sign @ must live with the % CMR
4. The $ sign must live with the @ % ? ! CMR
5. The & must live with $, % .. CMR
6. The # in CMR
7. The inverted &: must be found.
8. The ℓ as a rounded 'l'. CMMI
9. The centered dot $\cdot$ for use as a multiplication sign, must live with the period. CMSY
10. The asterisk $*$ for use as a multiplication sign, in position '003 in CMSY.
11. The $\propto$ sign must live with @, %, ℓ. In position '057 of CMSY.
12. The $'$ or prime in position '060 of CMSY, one cannot separate the prime from the the prime ligature slots. (2 ligatures) Kerning of letters with the prime is not possible, because the latter is set in superscript. ????????
13. The backprime from MSAM'070 should live with the prime. ??????
14. The ∞ sign in position '061,
15. The $\emptyset$ in position '073 of CMSY,
16. The `\check` mark in MSAM'130, ?????
17. The `\between` double parentheses in MSAM'107 should go with the normal parentheses.
18. The $\Re$, could come out if necessary,
19. The $\Im$ in position '074, and '075 of CMSY, could also come out if necessary.
20. The $\dagger$ or dagger in CMSY'171, ?????
21. The smile, ???????
22. The frown form CMMI ????????
23. Could maybe include the circled S from MSAM'163. ?????
24. The Weierstrass symbol: $\wp$ only in one style, (could come out)
25. The powerset symbol from `Martin.Ward@durham.ac.uk`. (could come out if necessary) It looks something like this:

```
       XXXXXXXXXXX
   XXXXXXXX    XXXX
 XXXXX           XXXX
 XXXXX           XXXX
```

```
XXXXXXXX         XXXX
XXX  XXX         XX
XX   XXXXXXX  XXXXXX
XX    XXXXXXXXXXXX
XX    XXXX
XX    XXXX
XX      XXXX
XX      XXXX
XXXX    XXXX
XXXX    XXXX
XXXX  XXXX
  XXXXXX
```

6.14 Symbols from lasy that must be kept:

The first four lasy symbols are in msam.

The ones in positions '50 through '53 are arrow heads, and are counted as such in the corresponding group.

Character '60 is in the msam.

'61 is not in the msam, and should be kept.

Char'62: same as msam'03 ?

Char'63: same as msam'06?

Lasy'72: same as msbm'163 or msam'166 ?

Lasy'73: same as msam'40 ?

A list of what should be kept from lasy:

- Character '61: 1
- ???

This makes a total of 1.

6.15 The "Subset" groups

Note: None of these have anything to do with the \sim glyph.

6.15.1 The "subset plain" group: 4

- The $\subseteq$ in position CMSY'022
- The $\supseteq$ in position CMSY'023
- The $\subset$ in position CMSY'032,
- The $\supset$ in position CMSY'033,

6.15.2 The "subset Ams" group: 12

- From MSBM'040 to MSBM'43 : 12

6.15.3 The "In/ni plain" group: 2

1. The $\in$ sign in position CMSY'062,
2. The $\ni$ sign in position CMSY'063,

6.15.4 The “sqsubset plain & Ams” group: 4

These do not have a place any where else:

1. The $\sqsubseteq$, cmsy’166,
2. The $\sqsupseteq$, cmsy’167,
3. The sqsubset from MSAM’100,
4. The sqsupset from MSAM’101,

6.16 The “Greater than” group

6.16.1 The “Greater than Plain” group: 8

- The $\leq$ in position CMSY’024,
- The $\geq$ in position CMSY’025,
- The $\ll$ in position CMSY’34,
- The $\gg$ in position CMSY’35
- $<$ less than CMMI’074,
- $>$ Greater than: CMMI’076,
- The alternative leq: msam’66
- The alternative geq: msam’76

6.16.2 The “Greater than AMS” group: 30

- From MSBM’000 to ’005: 6
- From MSBM’010 to ’015: 6
- From MSBM’024 to ’025: 4
- From MSBM’154 to ’155: 2
- From MSAM’060 to ’061: 2
- From MSAM’065 and ’067: 2
- From MSAM’075 and ’077: 2
- From MSAM’121 to ’124: 4
- From MSAM’156 to ’157: 2

6.16.3 The “greater than with sim” group: 8

1. MSBM’022,
2. MSBM’023,
3. MSBM’032,
4. MSBM’033.
5. MSAM’046,
6. MSAM’047,
7. MSAM’056,
8. MSAM’057

The ‘shapee’ `\sim`, and the geometric `\sim` are considered to be the same glyph, i.e. the difference that is sometimes visible is considered to be a bug.

6.17 The “Succ” groups

6.17.1 The “Succ without sim plain” group: 4

1. CMSY’026,
2. CMSY’027,
3. CMSY’036,
4. CMSY’037

6.17.2 The "Succ without sim AMS" group: 10

- MSBM'006, '007: 2
- MSBM'016, '017: 2
- MSBM'026, '027: 2
- MSAM'062 - '064: 3
- MSAM'074: 1

6.17.3 The "Succ with sim Ams" group: 8

- MSBM'020, '021: 2
- MSBM'030, '031: 2
- MSBM'166, '167: 2
- MSAM'45,
- MSAM'55,

The 'shapee' `\sim`, and the geometric `\sim` are considered to be the same glyph, i.e. the difference that is sometimes visible is considered to be a bug.

6.18 The "Sim" group: 12

1. sim CMSY'030
2. approx CMSY '31
3. simeq CMSY'047
4. wr CMSY'157
5. The bold MSBM'034
6. MSBM'035
7. MSBM'150
8. The bold MSBM'163
9. The bold MSBM'164
10. MSBM'165
11. MSAM'166, backsim
12. MSAM'167, backsimeq

6.19 Binops

6.19.1 The "Small binops plain" group: 20

1. cap CMSY
2. cup CMSY
3. uplus CMSY
4. sqcap CMSY
5. sqcup CMSY
6. big circle CMSY
7. big triangle up CMSY
8. big triangle down CMSY
9. vee CMSY
10. wedge CMSY
11. oplus CMSY
12. ominus CMSY
13. otimes CMSY
14. oslash CMSY
15. odot CMSY

16. amalg CMSY
17. bullet CMSY
18. circ CMSY
19. diamond CMSY
20. star (5 branches): CMMI'77

6.19.2 Small plain left right triangles: 2

These should be replaced by the ones in the AMS fonts for math usage.

They are also used as bullets, and should go in the text symbol encoding for such a usage.

1. triangle left: CMMI'56
2. triangle right CMMI'57

6.19.3 AMS left right open triangles: 8

These are also in LASY.

- vartriangle left
- vartriangle right
- triangle left eq
- triangle right eq

  The previous four are in `msam`
- Same 4 negated in `msbm`: 4

6.20 Basic Symbols: 24

A group of symbols used for typesetting basic mathematics. These are mainly geometrics. Some have been added for similarity reasons:

1. = The equals sign, CMR'075
2. - The minus sign, CMSY'00
3. + The plus sign, CMR'053
4. The $\times$ multiplication sign CMSY'002,
5. The `\divide` sign $\div$ CMSY'004
6. The `\divideontimes` from msbm'076 should live with divide and times.
7. The rtimes from msbm'157 should live with the times.
8. The ltimes from msbm'156 should live with the times.
9. The $\pm$ sign in position CMSY'006,
10. The $\mp$ sign in position CMSY'007,
11. The $\equiv$ in position CMSY'021, Difficult to separate from other similar relations.
12. The $\forall$ sign in position '070,
13. The $\exists$ sign in position '071,
14. The `\nexists` sign from msbm'100
15. The $\neg$ sign CMSY'072,
16. The `\varpropto` from AMS "5F. ????? or should this be left as a geometric?????
17. The varemptyset from MSBM'77, ????
18. Could go here: the upside down F: Finv from msbm'140 ???????
19. And the back to front G: Game from Msbm'141 ???????
20. Unary minus like en dash, could be CMR'173 but I personally think it should be shorter.
21. The `\varnothing` from MSBM'77,
22. smallsetminus from msbm'162
23. The $\perp$ perp or bot sign in position '077,
24. top sign CMSY'076

6.21 Radical

6.21.1 The TeX radicals: 10

Currently available in `cmex` are:

- Five radical signs: 5
- The vertical bit needed to construct the big radical '165 : 1
- The top bit of the constructed radical. '166 : 1

`cmsy` includes the basic size of the radical. It has always been loaded in three sizes, and must remain so. If it is taken out of `cmsy`, and put in a `cmex` replacement, then this point must be taken into consideration.

If the new `cmex` is loaded in one size, it must contain three different sizes of the radical in order to stay compatible with plain: 3

6.21.2 New radicals: 2

Request made by: `HITT% USOUTHAL.BITNET@SHSU.edu` .

One can overload the little vertical extensible module of the radical, in order to produce a left quantum operator. For the right quantum operator, the glyphs could be available, but the radical macro can't be used. A specific macro could be designed and it would need two glyphs: the top right hand corner, and the repeatable vertical bit: 2

6.22 The integrals family: 18

6.22.1 Big 'bigops' size: 7

1. The single integral.
2. The double integral. Could be done with kerning if there is not enough space.
3. The triple integral. Could be done with kerning if there is not enough space.
4. The single O integral.
5. The double O integral.
6. The sigma integral. For physics: Jörg.
7. The slash integral. For physics: Jörg.

6.22.2 Small 'bigops' size: 7

The same as in big 'bigops' size.

6.22.3 Small size: 7

This refers to the size of the `\smallint` in CMSY.

1. The single normal integral.
2. The single O integral.
3. Double O integral. ???? (Jörg thinks yes)
4. Double normal integral ???? (Jörg thinks yes) Could be done with kerning if there is not enough space.
5. Triple normal integral ???? (Jörg thinks yes) Could be done with kerning if there is not enough space.
6. The sigma integral.
7. The slash integral.

Mail from HSS:

> \doubleoint is used by Becker in ”electromagnetic fields and interactions” (Dover). I also saw \tripleoint used in electromagnetic theory books although both are somewhat archaic.
>
> Concerning the small version (in cmsy), I suggested this solely for reasons of completeness. The need for it is less now that cmex will be loaded in three sizes. But the small version of \int & \oint look a bit large when used in inline formulas. The \smallint & \smalloint etc. may be a choice for some authors in the latter case.
>
> YH also pointed out, that the upright versions of integrals are very common in textbooks. since the integral sign is one of the most common symbols used in maths, it may not be a bad idea to include upright versions of *all* integral signs in cmex (with corresponding small versions in cmsy) again for reasons of completeness.

6.23 AMS Vdash group: 10

1. MSBM'054 nparallel
2. MSBM'055 nmid
3. MSBM'056 nshortmid
4. MSBM'057 nshortparallel
5. MSBM'061 nVdash
6. MSBM'062 nvDash
7. MSBM'063 nVDash
8. MSBM'160 shortmid
9. MSBM'161 shortparallel
10. MSAM'015 Vdash
11. MSAM'016 Vvdash
12. MSAM'017 vDash

6.24 Plain and lasy miscellaneous symbols: 6

1. CMSY'20
2. CMSY'140
3. CMSY'141

Should live with the two previous:

4. MSBM'060 What about the back to front version of this ?????
5. LASY'061 the bow tie,
6. LASY'62 the small box. It is smaller than the one in the AMS fonts. ????

6.25 AMS equals friends: 10

1. msam'155
2. msam'154
3. msam'120
4. msam'73
5. msam'72
6. msam'54
7. msam'53
8. msam'52
9. msam'51
10. msam'44

6.26 AMS miscellaneous geometric symbols: 21

1. msam'174
2. msam'173
3. msam'171
4. msam'170
5. msam'165
6. msam'164
7. msam'161
8. msam'160
9. msam'151
10. msam'150
11. msam'147
12. msam'146
13. msam'141
14. msam'140
15. msam'136
16. msam'135
17. msam'134
18. msam'133
19. msam'132
20. msam'131
21. msam'050
22. msam'005

6.27 AMS boxes and friends: 15

1. MSAM'000
2. MSAM'001
3. MSAM'002
4. MSAM'003
5. MSAM'004
6. MSAM'006
7. MSAM'007
8. MSAM'014
9. MSAM'106
10. MSAM'110
11. MSAM'111
12. MSAM'112
13. MSAM'115
14. MSAM'116
15. MSAM'117

6.28 The horizontal curly braces: 10

Their design should be the same as the vertical braces. Add two horizontal extension modules for them, since if they are drawn with rules, digitization errors may cause them not to line up with the horizontal brace glyphs. What's more, this would enable the designer to choose there boldness.

Plus two extra middle bits. So that the designer is not restricted by the number of slots. Knuths design could use that little amount of glyphs, but maybe other will need more.

Count: for the downwards brace: 2 end bits, 1 middle bit, 2 extensible modules. That makes a total of 5 per curly brace. One up, and one down: makes 10.

6.29 Big and extensible TeX delimiters from cmex: 78

This group includes delimiters that are in `cmex`. And an extra little extensible module for the { and }.

- Four different sizes of (and): 8
- Extensible versions of (and): 6
  Left and right extensible modules are '102 and '103. Top and bottom are '060, '061, '100, '101.
- Four different sizes of [ ]: 8
- Extensible version of [and]: 6
  The extensible modules, one for the right bracket, and one for the left bracket are: '066, '067. The top and bottom pieces are: '062 to '065.
- Four different sizes of $\{$ and $\}$: 8
- Extensible module for $\{$ and $\}$: 7
  The extensible module ('76) used for the curly braces is very small, because it is added twice: once above the middle piece, and once below the middle piece. Its height is half that of the parentheses extensible module. Other pieces are: '070 - '075.
- An extra extensible module for the $\{$ and $\}$: 1
  There is only one extensible module for both the left and the right curly brace in `cm`. This is because the left-right spread of a curly brace is symmetrical in `cm`, unlike the parentheses for example. This may not be the case for other designs.
- Four different sizes of $\langle$ and *rangle*: 8
- Four different sizes of $\backslash$ and $/$: 8
- Four different sizes of $\lfloor$ and $\rfloor$: 8
  The extensible version is build with the same pieces as the extensible brackets.
- Four different sizes of $\lceil$ and $\rceil$: 8
  The extensible version is build with the same pieces as the extensible brackets.
- Extensible vert and parallel: 2
  Extensible versions of the vertical bar and the double vertical bar. They are their own extensible modules: '014 and '015.

6.30 Bigops

6.30.1 Old bigops from TeX: 28

There are two glyphs for each bigop.

1. The sqcup
2. The circled integral
3. The circled dot
4. The circled plus
5. The circled times
6. The sums
7. The prods
8. The normal integrals
9. The bigcups
10. The bigcaps
11. The U plus
12. The wedges
13. The vees
14. The coprods

6.30.2 New double sized 'bigops': 26

All these should come in two sizes, in the same font, like the present `\bigcup`. One for display style, and one for text style. That makes two glyphs for each one.

1. A double sized sqcap $\sqcap$ `\bigsqcap` (can be found in `cspex`)
2. Two sized $\bigcirc$ with $\vee$ inside. $\ovee$ proposed name: `\ovee`, and `\bigovee`. Can be found in `cspex` and `stmary`.
3. Two sized $\bigcirc$ with $\wedge$ inside. $\owedge$ proposed name `\owedge`, and `\bigowedge`. Comment from Alan:

   > "As far as I'm aware nobody has *ever* used these glyphs in a paper. I put them in St Mary's Road because I needed them at the time, but I shortly abandoned writing the paper they were going to be used in. Please don't include them! (If we are going to, we need to include ¡ovee¿ and ¡owedge¿ as well as ¡bigovee¿ and ¡bigowedge¿ which are the ones you described.)"

4. Dijkstra choice: ⫾ CSPEX
5. A wide Dijkstra choice. CSPEX . Comment from Alan:

   > If this is the glyph I think it is, it's not quite a wide Dijkstra choice in shape (although mathematically it's the same thing as Dijkstra choice). The two glyphs are:
   >
   > <dijkstrachoice> looks remarkably like [and] glued together.
   >
   > <oblong> looks like <sqcap> but with the square completed.
   >
   > <oblong> is used in CSP in conjunction with <sqcap>, so it's quite important that they look the same. In particular, they need to be of the same width because if they're not, formulae sometimes don't line up properly...

6. Parallel `\bigparallel` just a double sized version of parallel.
7. Interleaving $\interleave$: `\biginterleaving`

   > 'Interleaving' and 'parallel' are used in (at least) three different ways:
   >
   > - As delimiters ||foo|| and |||foo|||. These should come in basic-sized and extensible versions.
   > - As binary operators p || q and p ||| q. These can be the same glyphs as for the basic-sized delimiters.
   > - As 'big' operators $||_i\ p_i$ and $|||_i\ p_i$ similar to `\bigcup`. These should come in text style and display style versions.
   >
   > The big operators are not the same glyphs as the extensible delimiters.

8. `\bigcupdot`: A 'U' with a dot in it. Something like: ⊍
9. `\bigcapdot`: an upside down 'U' with a dot in it. Something like: ⩀
10. An inverted & called `\dnasrepma`
11. Large operator symbol based on, an asterisk sign.
12. Large operator symbol based on a times sign.
13. Large operator symbol based on, a hash sign.
14. Large operator symbol based on, an ampersand sign.

6.31 Non classified existing symbols

Here is a list of some symbols that do not have a place elsewhere:

- The different shapes of # should find a place, although one is already in the core group.

6.32 A list of new glyphs

The following symbols should be added in the math fonts. Some have already been designed by various people, so it should be possible to find them...

6.32.1 Basic size operators: 2

Basic size means the same size as the operators in `cmsy`.

1. Something like `\cupdot` and
2. Something like `\capdot` Frank M. can justify these.

6.32.2 New multi-sized, and extensible delimiters: 47

Count: 8, 6, 1, 8, 8, 8, 8, makes 47 .

A multi-sized delimiter means: 4 sizes for each side : 8 glyphs. Plus and extensible version: top, bottom, extension module for both sides: 6 glyphs. Sometimes also a middle: 8 glyphs. Total: 16 or 14.

1. Four sizes of the semantic brackets $[\![$ and $]\!]$: 8
2. An extensible version of the semantic brackets: 6
   (Top - Middle - Bottom) * 2 makes 6.
3. An extensible version of $|||$ for use as $|||f|||$ (a norme). Just the extension module: 1
4. 4 sizes of multiset brackets $\{|$ and $|\}$: 8
5. An extensible version of multi-set brackets: 8
   (Top - Bottom - Middle - extensible module) *2 makes 8.

   Unicode contains another style of brackets, they are called tortoise shell brackets. They look like

```
  /
 /
/
|
|
|
|
|
\
 \
  \
```

6. Four sizes of tortoise shell brackets: 8
   No extensible version. could add them in.

7. Four sizes of triangle brackets: 8
   They look like:

```
        /|
       / |
      /  |
      \  |
       \ |
        \|
```

6.32.3 Geometrics: 21

1. The `ams` smaller or equal and greater or equal must not be forgotten.
2. Linear 'is implied by' if o– and o–o [bb: There are also versions of these with filled-in circles.]
3. From JMR: something like: $\lfloor$ maybe the same upside down.
4. From JMR: Something like: $\rfloor$ maybe the same upside down.
5. `Here are some other symbol I once needed: \opm, \omp. Why? There is \oplus and \ominus and there is +, -, \pm and \mp, but there's no \opm and \omp.`

   `I once needed them in a context, where \oplus and \ominus were used as subscripts to indicate symmetric and antisymmetric wave functions that were normalized. It is easy to specify formulae that include both cases using \pm and \mp, but suddenly there was no \opm and \omp. I constructed symbols using the circle from the copyright sign, but that was not actually the perfect size.`

6.32.4 New arrows

- Alan J. wrote: `\arrownot` and `\Arrownot`, so that for example `\arrownot\mapsto` is visually compatible with `\nrightarrow`. Describing the same thing he also wrote: Add the 'building blocks' for the AMS negated relations, for example a `\arrownot` to build `\nlongrightarrow` and `\nrightarrowfill`.
- The building blocks to make `\mapsfrom` <–| `\Mapsto` |=> and `\Mapsfrom` <=|
- Lfloor, Rfloor, Lceil, Rceil like $\lfloor\!\lfloor$
- Arrows with triangles on the end.

$$\lhd\!-\!-,\ -\!-\!\rhd,\ \lhd\!-\!-\!\rhd$$

- Equals like symbol: $<-->$ with $==$ underneath.

6.32.5 Non geometrics: 19

1. Possibly something like $\bar{(}$ and $\bar{)}$ if the bar was touching the parentheses.
2. Banana brackets: look (sort of) like $(\!|$ and $|\!)$. Or they look like bananas if you believe Jeremy... Alan: The St. Mary Road font includes samples of them, in a line-drawing style. Since I have not seen them in real use, I cannot comment, if this style or rather the look of & in `cmr` is appropriate.
3. lightning (`wasy`) — probably does not belongs in maths. Alan: It is actually used though! It means 'interrupt' in process theory, c.f. Communicating Sequential Processes, Hoare, Prentice Hall 1985. I don't know how widely used outside process theory it is though...

4. Katakana character that looks like a spiral. (bb)
5. A lower-case sigma with a long tail that goes a little bit below the baseline.
6. The two versions of the # hash sign must not be forgotten. It seems that one is geometric, and one is not: the slanted hash sign and the upright hash sign.
7. An `\inviota` is sometimes requested on the net. I'll send you a reference file for it. (Jörg)
8. More ?

Chapter 7

Final conclusions

The 'Yaasp' proposal, which is the final proposal made, is given in chapter 5.

Working on the LaTeX3 project in Mainz was very interesting for many reasons:

- People next to me were working on net management and system maintenance. I used email intensively for communicating with other people working on the project. I used tar files and other programs to send large amounts of information to other people.

  It was a very good introduction to the network oriented studies I will be doing in my last year. I don't think that I really knew before what the network was. Now I have a better idea.

- I installed a test version of NFSS2. It was a good introduction to software installing, and enabled me to discover a few more UNIX tools. I hope I will be able to use this knowledge for installing various packages for TeX and especially emacs in the Ecole des Mines de Saint Etienne.

- The work I did was to a large extent research work and thus involved many topics for which no previous experience was available. For this reason the work seemed to go slowly, and we often had to go back to the drawing board and re-think points that we thought were already finished and done with. All of this, of course, was made worse by the fact that a lot of the communication was done via email.

  However, the final result was very positive. By the end of the three months, a complete proposal for a new math font set-up was produced. For Frank Mittelbach technical director of the LaTeX3 project, this is a good achievement, and a big step forward. The next stage is to try and implement the proposal, and start testing it.

- Last but not least, I greatly improved my English and my German. I learnt about another country, about its educational system, and about its habits, which one can only grasp by working in the country. I learnt how to integrate in a foreign environment, and how to deal with a few distressing problems: it was not easy to keep calm when my car packed up the week-end before I had planned to go back to France. In one's own country garage mechanics aren't easy people to deal with; matters get even worse when it is in a foreign country.

The whole experience was very enriching in many domains. The difficulty I had in speaking German proved to me that the teaching of languages in the *Ecole des*

Mines de Saint Etienne is not good enough, and must be improved and given more importance.

Appendix A

Analysing TeX's positioning of \mathaccents

The placing of mathematical accents in TeX is done with the following control sequence: `\mathaccent"xyzz{box}`. `"xyzz` is the code that identifies the accent, and '`box`' is the entity to come under the accent. The box can be any type of box known by TeX: a single letter, a vbox, an hbox, etc... The accent code used is a usual math code (refer to any book on TeX for more details). The accent is basicaly centered over the given box, but there are ways of influencing the way the centering is done. The vertical placement is as we shall see a little more tricky.

A.1 The accent choosing

Let us call x the box that is due to be accentuated, and u its width. We shall first suppose that x is a single character.

$$x: \quad \rightarrow\boxed{\mathrm{X}}\leftarrow\mathrm{u}$$

If the accent is part of a list of successors[1], TeX chooses the one whose total width is *as close as possible* to u, but still *smaller or equal* to u.

Note. A list of successors can only take place in one and one font only, and Knuth reminds us of this fact in rule 12: "[...] has a successor in its font [...]" This makes me wonder... The hat and tilde come in their smallest size in cmr position '136 and '176. The other sizes are in cmex position '142 - '147. The following:

`$\mathaccent"005E{e}$, ... ,$\mathaccent"005E{eeee}$`

produces: $\hat{e}$ and $\hat{ee}$ and $\hat{eee}$ and $\hat{eeee}$. So as expected there is no automatic sizing, seeing as the hat in cmr is not part of a charlist. Same test with tilde:

`$\mathaccent "007E{e}$, ... ,$\mathaccent"007E{eeee}$`

produces: $\tilde{e}$ and $\tilde{ee}$ and $\tilde{eee}$ and $\tilde{eeee}$. In order to get a hat that changes size automatically one must call the smallest hat in `cmex` directly , like `\widehat` does. `\widehat` is defined as: `\mathaccent "0362`. Family three indicates cmex, and 62 is the hex position of the smallest hat in `cmex`. Thus `\widehat{abda}` produces: $\widehat{abda}$.

[1]See document on charlist for more details about successors.

A.2 The horizontal placing

The accent followed by its italic correction is put into a box y whose width we shall call w.

$$y: \quad \rightarrow\boxed{\text{acc }\backslash/}\leftarrow \quad w$$

Let us call s the kern between the character in box x and the *skewchar*[2] — taken in that order. The box y is first centered on top of the box x and then shifted rightwards by the amount s.

If the box x is not a single character, *the amount s is taken equal to zero, therefore the skewchar is ignored and the box y is normally centered*[3] *over the box x.*

A.2.1 The 'skewchar'

The *skewchar* is a specific character that D.E. Knuth decided to use — in the way described above — for placing accents. Each font should have its own skewchar.

For most characters, the `.tfm` file specifies a particular kerning of each letter with the skewchar. This is true for the computer modern fonts, but other font designers may not have used this feature.

Why choose one skewchar rather than another? This is because the character ⁀ chosen by Knuth does not have any other kerning that could have been disturbed by the skewchar kerning. This choice may not always be good for all fonts, because it depends on what the character in position '127 is. Thus a font designer might choose another skewchar and put the necessary kernings in the `.tfm` file. Accent glyphs can be used as skewchars, because they are usually not subject to kerns from other glyphs.

As long as one uses the `cm` fonts, it does not make much sense to change the skewchar, unless one wants to get strange effects, or unless one intends to change the `.tfm file`. One should remember that an assignment to skewchar is not undone at the end of the group, but is a part of the global font information. A local change therefore involves saving the original value, and then restoring it.

A.2.2 The italic correction

The presence of the italic correction here is a little mysterious. Its effect is to shift the accent to the left compared to the position it would have without it. One should note that it is still added when the x box is more than one character.

A.2.3 Conclusions

It is useless changing the skewchar unless one also modifies the `.tfm` file(s), which contain(s) the info for the skewchar kerning, and for the italic correction.

Accents from any font can be positioned on characters from any other font if there is a skewchar in the character font.

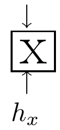
A.3 The vertical placing

This is a little more tricky. Here as well one can start by supposing that the character to be put under the accent is single. All previous notations are kept. χ is the x-height

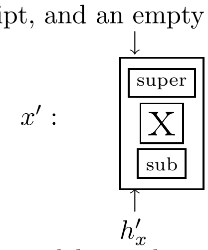
[2] See below for more details about the skewchar.

[3] But the box y contains the italic correction, which influences the centering.

of the accent's font i.e. `\fontdimen5`. Let us call h_x the height of the x box:



and h'_x the height of a box x' containing the character to be put under the accent, together with an empty superscript, and an empty subscript.



Now set δ with the minimum of χ and h_x, and increase it by $h'_x - h_x$.

The end result is a `\vbox` z containing box y (the accent correctly positioned lateraly), followed by a (vertical) kern of $-\delta$, and then box x' . A normal accent char has the folowing aspect: . This explains why the kern of $-\delta$ is needed. Without it the gap under the accent would be to big. One can now understand why an 'O' used as an accent over an 'e' produces the following: .

If the height h_z of z is smaller than h_x then a kern is added on top of z in order to make the end height that of x. Finally, the width of z is set to the width of x.

If the character to be accentuated is not single, δ is not increased by $h'_x - h_x$. In other words, one can forget about the subscript and superscript.

A.3.1 Conclusions

The x-height is used for the vertical placing of accents. This means that one cannot mix in the same font table glyphs designed with different x-heights.

Appendix B

A close look at extensible characters

The re-encoding of the math fonts cannot be thought of without a closer look at TEX's mechanisms for dealing with extensible characters. This includes all characters that come in different sizes, all characters that are constructed, and the operators that usually come in two sizes. The ultimate questions are: how to implement these characters? What sort of kerning can be done with them?

B.1 Let us start with the easiest: The operators

A large operator like $\sum$ will be vertically centered with respect to the axis when it is typeset. Thus, large operators can be used with different sizes of text. This vertical adjustment is not made for symbols of other classes. [The TEXbook p.155]

This is a sum in tex: $\sum_{i=0}^{i=n} i$ and this is a prod in text: $\prod_{j=0}^{i=m} j$. Same test in display maths:

$$\sum_{i=0}^{i=n} vi \quad \prod_{j=0}^{j=m} j$$

The sum is defined as follows:

```
\mathchardef\sum = "1350
```

If TEX is in display style, it looks to see if the character in position ”50 of family 3 (The extensible `cmex` family) has a successor[1]. If it does then the successor is taken. When not in display style, TEX just takes character ”50 from family 3.

B.1.1 Conclusions for operators

Both occurrencies of a double sized operator must be in the same font table, because they are linked by the successor mechanism.

[1] See below for explanations on successors.

B.2 How characters can be linked

B.2.1 `charlists`

They enable several characters of the same font to be linked together. By simply typing `charlist oct"000": oct"020": oct"022": oct"040": oct"060"` in the metafont source, one links all the occurrences (in the `cmex` font) of the left parentheses in increasing size-order.

They are used for:

- Linking variable-size delimiters,
- Linking variable-width accents,
- Making pairs of operators that come in textsize and display size.

B.2.2 The `extensible` lists p318 metafont book.

An extensible glyph is identified with one of its pieces. One has to decide which piece is going to be used for this identification. For instance, in `extensible oct"060": oct"060", 0, oct"100", oct"102";` the first `oct"060"` is the identifier of the whole glyph. The next three characters are the top, middle, and bottom pieces. The last character position is that of the piece to be repeated as many times as necessary between the top and middle, and between the bottom and middle pieces. All three pieces are optional. When they are not needed, they are replaced with '0'. But if a zero is put in the repeater position, then character '0' will effectively be used as a repeater.

B.2.3 Restrictions compiled from p318 of the Metafont book

- An `extensible` identifier can only appear at the end of a `charlist`.
- A kerning/ligtable label can only appear at the end of a charlist.
- One cannot use an extensible identifier as a kerning label.
- One cannot use an extensible identifier as a ligature label.

B.2.4 Conclusions

Kerning with parentheses is going to be very tricky...

A delimiter is made of two sets of characters that can be in two separate font tables. The glyphs in these two sets (the delimiters), can be kerned with the characters that accompany them in their font. So one must place them correctly.

Note: There are no parentheses in the `cmmi` fonts, so this possibility has not been used.

There is a list of all `cm` extensibles, and charlists in "Computer modern typefaces" p66.

B.3 The vertical constructables, or "those that come in pieces" — *Delimiters*

B.3.1 A few notes

First information about delimiters p.171 of the TeXbook. They are not all of the same type. Some are (bigl,bigr) Open/close atoms, and others are (bigm) rel and (big) ord atoms. On the other hand, a `\left \right` grouping is inner.

When a delimiter gets larger, its height and depth both grow by the same amount.

In the `cmex` font, most of the vertical constructable glyphs are below the baseline, in the `.tfm` file.

B.3.2 A first description of the choice mechanism

A delimiter is defined by a small "character" and a "large character". **These characters can be in different families, and therefore in different font tables.** We shall call the small character a in family f_a, and the large character b in family f_b.

The search first starts by considering the three different sizes of char a in its family[2]. When testing a character in a given font table, its successors are tested before going to the next bigger font. In other words, for each member of a family, starting with the smallest, and going to the biggest, TeX first looks at char a and then at its successors[3] in the same font table. If nothing suitable is actually found within the family f_a a similar search is done in the family f_b based on character b.

The search stops when the character being tested has a sufficient height plus depth, or when it is extensible[4].

If either of the couples (a, f_a), and (b, f_b) are set to $(0, 0)$ then the corresponding part of the search is bypassed.

If none of the characters actually found are suitable, the biggest is taken, i.e. the one with the greatest height plus depth.

B.3.3 A second description of the choice mechanism by Victor E.

TeX first tries the small variant, and if that is not satisfactory (or if the left part of the delimiter code is 000) it tries the large variant. If trying the large variant does not meet with succes TeXtakes the largest delimiter encountered in this search. If no delimiter at all is found, (which can happen if the right hand part is also 000), an empty box of width `\nulldelimiterspace` is taken.

Investigating a variant means in sequence:

- If the current style is scriptscript style, the scriptscript font of the family is tried.
- If the current style is script style or smaller the script font of the family is tried.
- Otherwise the text font of the family is tried.

Note: The plain format puts the `cmex10` font in all three styles of family three.

Looking for a delimiter at a certain position in a certain font means:

[2] i.e. the scriptscriptsize then the scriptsize and finaly the textsize. This is done in an optimized way: if the current size is bigger than scripscriptsize, no point looking in the scriptscriptsize font, same goes for the other two sizes.

[3] See later explanations.

[4] Obviously in this case the appropriate delim has been found, seeing as it can be made to any given size bigger than the non-extensible characters.

- If the character is large enough, accept it.
- If the character is extensible, accept it.
- Otherwise, if the character has a successor, (the same but bigger), try the successor.

B.3.4 The final width of the delimiter ?

If the character chosen ends up to be an **extensible** one, *the resulting width is that of the repeatable piece.* Otherwise (in the case of a normal character) the width is that of the chosen character *plus its italic correction.*

B.3.5 Conclusions for delims

- The different sizes of a delimiter can be spread in two font tables if it is necessary.
- A given delimiter does not absolutly have to have two sizes.
- One can adjust the width of the repeatable piece for extensibles.
- One can adjust the italic correction of the normal "single glyph" delimiters, in order to get it further away from things like f, j, g and p. In the present case of '(' (in text size) it comes from the text font `cmr*` so obviously has not got any italic correction. The vertical bar has not got any either (checked in the `.pl` files). This is quite a global solution and the italic correction will be added in all cases: if it is small it may not bother anybody and should have the right effect in most cases.

B.4 References

The TEXbook "Construction of math symbols": 151 mathchoice: no good; 152: about delimiters and size choosing; 178: using phantom and vphantom - no good; 358: how large operators are assigned in plain, and some horizontal constructables; 359: all the 24 delims that can change size and the big and bigg macros; 360-361: nothing.

The TEXbook "math symbols": 127-128: nothing, 289: nothing, 290: interesting things about delims.

The TEXbook "Mathcode": 134: tiny little bit at the bottom about mathcodes, 154: a list of the 8 classes and (3) about variable family and (bottom) about mathcode, 155: the mathcode "8000 + mathchardef + mathinner, 156: delcodes and delimiter, 157: radical, 214: nothing, 271: nothing, 289: nothing, 319,326: answer to exercises, 344: where all the mathcodes are set so that '1' comes from fam 0 and 'a' from fam 1, 345: where the delcodes for plain tex are set.

The TEXbook "mathop": 155 cf mathcode, 178 cf construction of math symbs, 291 a bit about mathaccent, radical 324-325 361

The TEXbook "dilimiters": 156: delcode; 157: radical; 214: nothing; 289: nothing; 271: nothing; 290: in the middle "A delim..."; 345: see at the top; 359: Plain tex definitions of some delimiters; 146: A list of plain tex delims; 147: the bigs; 148-149: details for use of left right; 150: still more extensible chars (bottom not on growing delims); 171: info on the type of atoms made by big bigr bigl bigm; 437: openings and closings; 442: The search of the appropriate delimiter: good.

About radicals rule 11 page 443 appendix G. TEXbook

TeXbook: About operators rule 13 page 443, successors. Interesting about italic corrections. The way limits are typeset rule 13a.

About parameter usage p447 TeXbook. About math spacing p170 TeXbook.

TeX for the impatient: p.126 nothing, p.194-196 a list of operators and a few explanations.

TeX by topic p.194: Large operators and their limits, the choosing method: good.

Appendix C

Replacing `cmex` ?

While working on the new math encoding, the writer realised that the fact the `cmex` font is only loaded in one size, and not in three like the other math fonts, was going to create a few problems. This paper deals with the following topics:

- What is in `cmex`?
- Which special mechanisms does TeX use to access glyphs from `cmex`?
- What could be added to `cmex`?
- What could be taken out of `cmex`?

The aim of this paper is to help the MFG[1] design the MX encoding as a replacement and improvement of the `cmex` encoding.

Note: Most of what is written in this paper is pure theory, and has not been applied or tested.

Acknowledgements: thanks to Alan Jeffrey and Barbara Beeton for their constructive comments, and help.

C.1 What is in `cmex`?

C.1.1 Delimiters

- Four different sizes of ( ) and extensible versions. Left and right extensible modules are '102 and '103.
- Four different sizes of [ ] and extensible versions. The extensible modules, one for the right bracket, and one for the left bracket are: '066, '067.
- Same for { and } ; the extensible module is: '076.

  Note: The extensible module here is very small, because it is added twice: once above the middle piece, and once below the middle piece. Its height is half that of the parentheses' extensible module. Interesting to see that there is only one extensible module for both the left and the right curly brace. This is because the left-right extension of a curly brace is symmetrical, unlike the parentheses for example.

[1] Math Font Group.

- Four different sizes of $\langle$ and $rangle$. No extensible version.
- Same for $\backslash$ and $/$. No extensible version.
- Four different sizes of $\lfloor$ and $\rfloor$ and an extensible version. Extensible modules: '066, and '067.
- Same for $\lceil$ and $\rceil$. Same extensible modules as the previous one.
- Glyphs in positions '014 and '015 are the extensible versions of the vertical bar and the double vertical bar. They are their own extensible modules.

C.1.2 Large operators

Large operators come in pairs:

- The sqcup
- The circled integral
- The circled dot
- The circled plus
- The circled times
- The sums
- The prods
- The normal integrals
- The bigcups
- The bigcaps
- The U plus
- The wedges
- The vees
- The coprods

C.1.3 Wide accents

- 3 sizes for the hat
- 3 sizes for the tilde

C.1.4 Radicals

- Five radical signs
- The vertical bit needed to construct the big radical: '165
- The top bit of the constructed radical: '166

C.1.5 Arrows

- The three pieces for the construction of the vertical double arrow: '167 '176 '177
- The three pieces of the vertical single arrow: '077, '170, '171

C.1.6 Horizontal curly braces

- The four pieces for the construction of horizontal curly braces: '172 – '175

C.2 TeX's behavior with `cmex` glyphs

C.2.1 Large operators

- A large operator is vertically centered with respect to the math axis. This means that, whatever the surrounding glyph size, things will not look too bad.
- With the following definition of a large operator: `\mathchardef \sum = "1xyy`, if TeX is in *display style*, it looks to see if the character in position `"yy` of family `x` has a successor. If it does then the successor is taken. When not in *display style*, TeX simply takes character `"yy` from family `x`. Thus in text style, in script style and in scriptscript style the same glyph is used.

C.2.2 Vertical delimiters, and friends

Radicals are delimiters, and vertical arrows also, so let us only speak about delimiters. Here is a quote from Victor Eijkhout's book:

> A delimiter has two codes: a small variant, and a large variant. TeX first tries the small variant, and if that is not satisfactory (or if the left part of the delimiter code is 000) it tries the large variant. If trying the large variant does not meet with success TeX takes the largest delimiter encountered in this search. If no delimiter at all is found, (which can happen if the right hand part is also 000), an empty box of width `\nulldelimiterspace` is taken.
>
> Investigating a variant means in sequence:
>
> - If the current style is scriptscript style, the scriptscript font of the family is tried.
> - If the current style is script style or smaller the script font of the family is tried.
> - Otherwise the text font of the family is tried.
>
> Looking for a delimiter at a certain position in a certain font means:
>
> - If the character is large enough, accept it.
> - If the character is extensible accept it.
> - Otherwise, if the character has a successor (the same but bigger), try the successor.

Using the three size mechanism probably did not seem necessary to Knuth. Generally large delimiters are used in display style, and not in script or scriptscript style. However, they can also be used in the small styles.

C.2.3 Wide accents

For the choice of accents, TeX only considers one font, but looks to see if the current accent has a successor. Unlike the delimiter choice mechanism, the accent choice mechanism does not go through all three sizes. TeX chooses the accent in such a way that the accent width is as close as possible but smaller than the width of the box to cover.

C.2.4 First conclusion

For operators, a distinction in size is made between text style and display style, whereas with the usual automatic size choosing mechanism, glyphs in text style and in display style are taken from the text size font, and are therefore the same size.

C.3 Consequences of loading `cmex` in 3 different sizes

C.3.1 Consequences for operators

For operators, let us consider two completely separate policies. In the one case `cmex` is unchanged, and loaded in three sizes. In the other case, an imaginary font derived from `cmex` called `cmex`′ is loaded in three sizes. In `cmex`′ the operators no longer have two sizes, thus glyphs like `\bigcup` do not have a successor in their font.

All the following supposes that no new macros have been written. What would TeX's automatic behavior would be?

If `cmex`′ is loaded in three sizes. In such a situation TeX has a large version of `\bigcup` (and other operators) in text size, a small version in script size, and yet a smaller version in scriptscript size.

The operators can be centered: no problem.

In script and scriptscript style the resulting 'big operators' would be smaller than if they were produced with today's standard TeX, and today's standard `cmex`. What is more, one would be smaller than the other, which is also not the case with today's standard TeX, and today's standard `cmex`.

In display style one would get big operators from the text size font: this is acceptable. *But one would also get a big operator in text style,* and that does not conform with today's standard TeX behavior.

If the existing `cmex` encoding is loaded in three sizes. The operators can be centered: no problem.

In script style, one would get the smallest version of a large operator. But coming from a small size font, that will produce something very small. In scriptscript style, same behavior as in script style, but the result would be even smaller. Thus in script, and in scriptscript style, the large 'big operators' would never automatically be used. Hence the `cmex`′ encoding.

In text style, TeX would produce the small version taken from the text size font. In display style TeX would produce the big version of operators taken from the text size font.

So in text style and in display style, there would be no change compared to what today's standard TeX produces. But script and scriptscript style would produce different results.

In both cases, things could be improved if macros were written to override the present behavior of `\bigsqcup`. One could think of things like `\mathchoice`, but ...

C.3.2 For vertical delimiters, radicals, vertical arrows

Let us start by supposing TeX is in scriptscript style, and it has to typeset a large delimiter. One should consider two cases:

The delimiter has an extensible variant. In this case the search will start in scriptscript size, and continue until TeX finds the extensible variant of scriptscript size. Then the search will stop, and the extensible will be used. This extensible will come from scriptscript size, and therefore probably not look the same as it would in today's setup, where all extensibles come from text size.

The delimiter does not have an extensible variant. As previously, the search starts in scriptscript size. If nothing big enough is found in scriptscript size, the search continues in script size. If still nothing is found, the search then continues in text size. If necessary the biggest delimiter from text size will be used. If the search stops in text size, there is no difference with what TeX produces today. But if the search stops before reaching text size, the chosen delimiter will be different from the one TeX would use in the present configuration. Its strokes would be finer, and better adapted for use in script style.

If one supposes that TeX is in script style, the previous two cases also apply, except that every occurrence of 'scriptscript' must be replaced with 'script'. If one supposes that TeX is in text style, the result of loading three different sizes of `cmex` would be the same as it is in TeX's current configuration.

C.3.3 For horizontal curly braces

If they are automatically taken from script size, or from script script size when necessary, the spacing changes a little, because the dimensions in the `.tfm` files would be different. A consequence of this could be different line and page breaks.

However, it would be nice if curly braces did come out of the correctly sized fonts. Then their boldness would match the surrounding text. But apparently from a macro programming point of view things could be difficult, even if the glyphs are available and loaded.

C.3.4 For wide accents

See first paragraph of previous section.

If accents were taken from the current size, things could only look better. The accent width would be closer to that of the material under the accent, and the accent's boldness would be better adjusted.

Note: Unlike the delimiter choice mechanism, the accent choice mechanism is restricted to one font, and one size. It will thus not look in text size when it is in script size for instance. So in script style, accents will always come out of the script size font, and in scriptscript style, accents will always come out of the scriptscript size font, etc....

C.3.5 Conclusion

Nearly everything in `cmex` could have lived in a normal three sized math font, and maybe that would have been better. The only problems would have come from the specific “big operator” behavior required by Knuth.

Also one must not forget that Knuth did not want to leave any empty slots.

The reduced amount of memory that was available on the machines with which TeX was first used could have been another reason for loading `cmex` in one size only.

C.4 What could be added to `cmex`?

Let us now consider possible evolutions of `cmex`. In spite of the terminology “adding to `cmex`”, the font resulting from these evolutions would have a different name.

C.4.1 If the `cmex` encoded font is loaded in three sizes

In this case big operators would not produce the usual results, and the rest would be slightly different, as stated above.

- One could add wide accents, but one would get slightly different (better) results. Thus wide accents would match the script and scriptscript styles. Macros could be made available as a style option to keep the old behavior, if necessary.
- One could increase the number of different sizes for accents.
- One could add big delimiters and their extensible versions, without any problem! Things will be slightly better adjusted in script and scriptscript style. Macros could be made (available as a style option) to keep the old behavior, if necessary.
- One could increase the number of different sizes for delimiters, and one could probably reduce the height of the extensible module in order to make the growing of delimiters more gradual.
- One could add some vertical extensible arrows! Things will be slightly better adjusted in script and scriptscript style.
- One can add small and large ‘big operators’ without any problem!
- Big improvement: one could add loads of other glyphs (symbols, etc.. . .) that would come in all three sizes.

C.4.2 If the `cmex` encoded font is only loaded in one size

- One could add big delimiters and their extensible versions without any problem!
- One could increase the number of different sizes for delimiters, and one could reduce the height of the extensible module in order to make the growing of delimiters more gradual.
- One could add some vertical extensible arrows!
- One could add large operators without any problem!
- One could add wide accents without any problem, and the present behavior of wide accents would not change. But if wide accents are meant to match the script and scriptscript styles, then wide accents must go in another font that would be loaded in different sizes.

- One could increase the number of different sizes for accents.
- One could add other stuff, but it would only come in one size.

C.4.3 If a cmex′ encoded font is loaded in three sizes

The imaginary cmex′ encoded font, previously described in this document is considered here.

One could add the same things as when cmex is loaded in three sizes. The only difference is: if no macro programming were done, the text style and display style will produce the same 'big operators'. In script and scriptscript style the 'big operators' would be in different sizes from one another and smaller than those in text style.

C.5 Conclusions

If one loads cmex in three different sizes, many things are improved, and with a \mathchoice the initial behaviour of large operators could be kept, or available as a style option.

If cmex is kept in a single size, one must decide whether to put wide accents in or not.

C.6 The beginning of my cmex10.pl file

```
(FAMILY CMEX)
(FACE O 352)
(CODINGSCHEME TEX MATH EXTENSION)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM O 37254272422)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.0)
   (STRETCH R 0.0)
   (SHRINK R 0.0)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.0)
   (DEFAULTRULETHICKNESS R 0.039999)
   (BIGOPSPACING1 R 0.111112)
   (BIGOPSPACING2 R 0.166667)
   (BIGOPSPACING3 R 0.2)
   (BIGOPSPACING4 R 0.6)
   (BIGOPSPACING5 R 0.1)
   )
(CHARACTER O 0 ...
```

C.6.1 Comments about the cmex10.pl file

- The xheight is not equal to zero.
- The space is equal to zero.

- With the following:

```
(CHARACTER O 100
   (CHARWD R 0.875003)
   (CHARHT R 0.039999)
   (CHARDP R 1.760019)
   (VARCHAR
      (TOP O 70)
      (BOT O 73)
      (REP O 76)
      )
   )
```

that is in the `.pl` file, one can produce something that looks like a growing integral:

$$\left\int \frac{3.q}{\frac{3\pi.r^2}{3.q.b.c}} \right.$$

- The pieces used to construct the horizontal curly braces are not linked in any way.
- The bottom pieces of the extensible parentheses are overloaded for `\rmoustache` and `\lmoustache`. One of these could be linked (charlisted) with the integrals, so that `\left\bigint` could produce a growing integral like the delimiters.
- The bottom pieces of the curly braces ('072 and '073) are also overloaded for `\lgroup` and `\rgroup`.
- The middle pieces of the curly braces are overloaded for `\arrowvert` and `\Arrowvert`. Other single and double extensible bars with different spacing.
- The extensible module of the curly braces is overloaded for `\bracevert`. A fat vertical bar.
- The extensible modules of the parenthese are overloaded to produce more fat vertical bars.
- More overloading: the construction pieces of the extensible brackets are also used for the $\lceil\rceil\lfloor\rfloor$. Thus the top left bracket piece ('062) identifies the left bracket; the top right bracket piece identifies the left bracket; the bottom left bracket piece identifies the $\lfloor$ extensible version; the bottom right piece identifies the $\rfloor$; the left extensible module identifies the $\lceil$; and the right extensible module identifies the $\rceil$. This over-loading may not be desired.
- For the wide accents and the curly braces the depth is nil.
- All the delimiter glyphs in `cmex` are set with a very small height and a big depth. This is because the radical primitive is also used for delimiters. For radicals, the `.tfm` height of the glyph is used to determine the size of the `\hrule`.
- The extension modules do not have any height at all. Same for the arrow heads.
- The four integrals have italic corrections.
- Small versions of operators have a nil height, whereas big versions have a small height and a big depth:

```
(CHARACTER O 116
 (COMMENT This is the small \bigotimes)
 (CHARWD R 1.1111145)
 (CHARDP R 1.000013)
 (NEXTLARGER O 117)
 )
(CHARACTER O 117
 (COMMENT This is the big \bigotimes)
 (CHARWD R 1.511116)
 (CHARHT R 0.100001)
 (CHARDP R 1.500012)
 )
```

What is more, in the metafont code, both big and small versions of bigops are under the baseline.

- There are no kerns or ligatures in `cmex`.

C.7 Characters under the baseline

C.7.1 Which?

It is understood and agreed that the radical glyphs need to be virtually completely under the base line. TeX uses their small height to measure the thickness of the radical's over line. But delimiters and 'bigops' are also placed under the baseline for no obvious reasons: both types of glyphs are just centered on the maths axis.

C.7.2 'Bigops' and metafont code

The metafont padded operator. Although both big and small versions of bigops are placed completely under the baseline (height=0), the big version ends up — in the `tfm` file, with a non-nil height. Many people agree that the reason for this is that the metafont code for the large version of 'bigops' contains the `\padded` macro. This last places some extra space around the glyphs. This extra space would serve for separating the 'bigops' from the limits they may take.

Large integrals do not have any padded macro, and thus in the `tfm` file, they are placed completely under the base line. The reason for the difference between integrals, and other 'bigops' could be that the limits of the former are usually placed next to the glyph, and not on top. Even when limits are placed on top of the integral, the results do not look too bad, because the integral is very narrow.

The reason for 'bigops' being set under the baseline is still unknown. Whether or not they would be correctly centered on the math axis, if they were placed over the base line is not sure either.

Changing the metrics in the metafont code. It is interesting to see how TeX would manage if the 'bigops' were placed over the base line. The best way to find out, is to change the metafont code of `cmex`, and see... I first did the test on `\bigoplus` because it was a good candidate — simple metafont code. I have never used metafont before. I copied all the necessary files into my directory, and changed their names to '`my*`'. I then did the following modifications in `mybigop.mf` (look for %%%%):

```
cmchar "\textstyle circle-plus operator";
beginchar(oct"114",20u#,10/6dh#,0); %%%% was ,0,10/6dh#)
adjust_fit(0,0); pickup pencircle scaled stem;
lft x6=hround u; x2=w-x6; top y8=h; bot y4=-d; %%%% was top y8=0
...

cmchar "\displaystyle circle-plus operator";
beginchar(oct"115",27.2u#,14/6dh#,0); padded 1/6dh#;
     %%%% was ,0,10/6dh#)
adjust_fit(0,0); pickup pencircle scaled curve;
lft x6=hround u; x2=w-x6; top y8=h; bot y4=-d; %%%% was top y8=0
...
```

When I ran Metafont on it there were no problems. After having put all the relevent files where they were meant to go, I gave the following to TEX:

```
Old cmex in text style: $\bigoplus i \coprod$
Old cmex in display style: $$-\bigoplus i\coprod \mathchar"034D$$

% change font:
\font\myfont=myex10
\textfont3=\myfont
\scriptfont3=\myfont
\scriptscriptfont3=\myfont

New myex in text style: $\bigoplus i \coprod$
New myex in display style: $$-\bigoplus i\coprod \mathchar"034D$$
The minus sign gives the height of the math axis, while the bottom
            of the word ‘base’ gives that of the baseline.

\bye
```

Here is the output:

Old cmex in text style: $-\bigoplus i \coprod$
Old cmex in display style:

$$-\bigoplus i \coprod \bigoplus^{base}$$

New myex in text style: $-\bigoplus i \coprod$
New myex in display style:

$$-\bigoplus i \coprod \bigoplus^{base}$$

The minus sign gives the height of the math axis, while the bottom of the word ‘base’ gives the baseline.

The results are quite obvious: in both case the `\bigoplus` is correctly centered. On the 300 dpi printer I have here there is no visible difference. What is more the `\mathchar"034D` proves that the metrics of the `\bigoplus` have changed: in the first case the glyph is placed under the baseline, and in the second case it is placed over the baseline. The difference is not visible if this document is printed with the wrong fonts.

Appendix D

Fonts and font encodings

The first 4 figures given on the next few pages are the standard fonts used in plain TeX for maths.

- **Computer Modern Roman:** loaded in family 0 shown on figure D.1.
- **Computer Modern Math Italic:** loaded in family 1 shown on figure D.2.
- **Computer Modern SYmbols:** loaded in family 2 shown on figure D.3.
- **Computer Modern EXtensibles:** loaded in family 3 shown on figure D.4.

Figure D.7 shows the DC-encoding with which the new math encoding is designed to live. Unlike the `cmr` encoding, the Cork encoding does not include any Greek glyphs, this prevents its use in family 0 for maths. But an upright text font is needed in family 0, for mixing sub- and super-script in text. This problem has until now prevented the wide spreading of the DC-fonts.

The next two fonts shown in figures D.5 and D.6 are the AMS fonts, designed especially for use in maths.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	˝0x
´01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	
´02x	ı	ȷ	`	´	ˇ	˘	¯	˚	˝1x
´03x	¸	ß	æ	œ	ø	Æ	Œ	Ø	
´04x	-	!	”	#	$	%	&	’	˝2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	˝3x
´07x	8	9	:	;	¡	=	¿	?	
´10x	@	A	B	C	D	E	F	G	˝4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	˝5x
´13x	X	Y	Z	[	“	]	ˆ	˙	
´14x	‘	a	b	c	d	e	f	g	˝6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	˝7x
´17x	x	y	z	–	—	˝	˜	¨	
	˝8	˝9	˝A	˝B	˝C	˝D	˝E	˝F	

Figure D.1: The `cmr` encoding: 128 glyphs.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	$\mathit{\Gamma}$	$\mathit{\Delta}$	$\mathit{\Theta}$	$\mathit{\Lambda}$	$\mathit{\Xi}$	$\mathit{\Pi}$	$\mathit{\Sigma}$	$\mathit{\Upsilon}$	˝0x
´01x	$\mathit{\Phi}$	$\mathit{\Psi}$	$\mathit{\Omega}$	α	β	γ	δ	ϵ	
´02x	ζ	η	θ	ι	κ	λ	μ	ν	˝1x
´03x	ξ	π	ρ	σ	τ	υ	ϕ	χ	
´04x	ψ	ω	ε	ϑ	ϖ	ϱ	ς	φ	˝2x
´05x	$\leftharpoonup$	$\leftharpoondown$	$\rightharpoonup$	$\rightharpoondown$	$\lhook$	$\rhook$	$\triangleright$	$\triangleleft$	
´06x	0	1	2	3	4	5	6	7	˝3x
´07x	8	9	.	,	$<$	$/$	$>$	$\star$	
´10x	∂	A	B	C	D	E	F	G	˝4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	˝5x
´13x	X	Y	Z	$\flat$	$\natural$	$\sharp$	$\smile$	$\frown$	
´14x	ℓ	a	b	c	d	e	f	g	˝6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	˝7x
´17x	x	y	z	$\imath$	$\jmath$	$\wp$	$\vec{}$	$\frown$	
	˝8	˝9	˝A	˝B	˝C	˝D	˝E	˝F	

Figure D.2: The `cmmi` encoding: 128 glyphs.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	$-$	$\cdot$	$\times$	$\ast$	$\div$	$\diamond$	$\pm$	$\mp$	″0x
´01x	$\oplus$	$\ominus$	$\otimes$	$\oslash$	$\odot$	$\bigcirc$	$\circ$	$\bullet$	
´02x	$\asymp$	$\equiv$	$\subseteq$	$\supseteq$	$\leq$	$\geq$	$\preceq$	$\succeq$	″1x
´03x	$\sim$	$\approx$	$\subset$	$\supset$	$\ll$	$\gg$	$\prec$	$\succ$	
´04x	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$	$\leftrightarrow$	$\nearrow$	$\searrow$	$\simeq$	″2x
´05x	$\Leftarrow$	$\Rightarrow$	$\Uparrow$	$\Downarrow$	$\Leftrightarrow$	$\nwarrow$	$\swarrow$	$\propto$	
´06x	$\prime$	∞	$\in$	$\ni$	$\bigtriangleup$	$\bigtriangledown$	$/$	$\mapstochar$	″3x
´07x	$\forall$	$\exists$	$\neg$	$\emptyset$	$\Re$	$\Im$	$\top$	$\bot$	
´10x	$\aleph$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	″4x
´11x	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$	
´12x	$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{S}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{W}$	″5x
´13x	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{Z}$	$\cup$	$\cap$	$\uplus$	$\wedge$	$\vee$	
´14x	$\vdash$	$\dashv$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	″6x
´15x	$\langle$	$\rangle$	$\vert$	$\Vert$	$\updownarrow$	$\Updownarrow$	$\backslash$	$\wr$	
´16x	$\surd$	$\amalg$	∇	$\smallint$	$\sqcup$	$\sqcap$	$\sqsubseteq$	$\sqsupseteq$	″7x
´17x	$\S$	$\dagger$	$\ddagger$	$\P$	$\clubsuit$	$\diamondsuit$	$\heartsuit$	$\spadesuit$	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure D.3: The `cmsy` encoding: 128 glyphs.

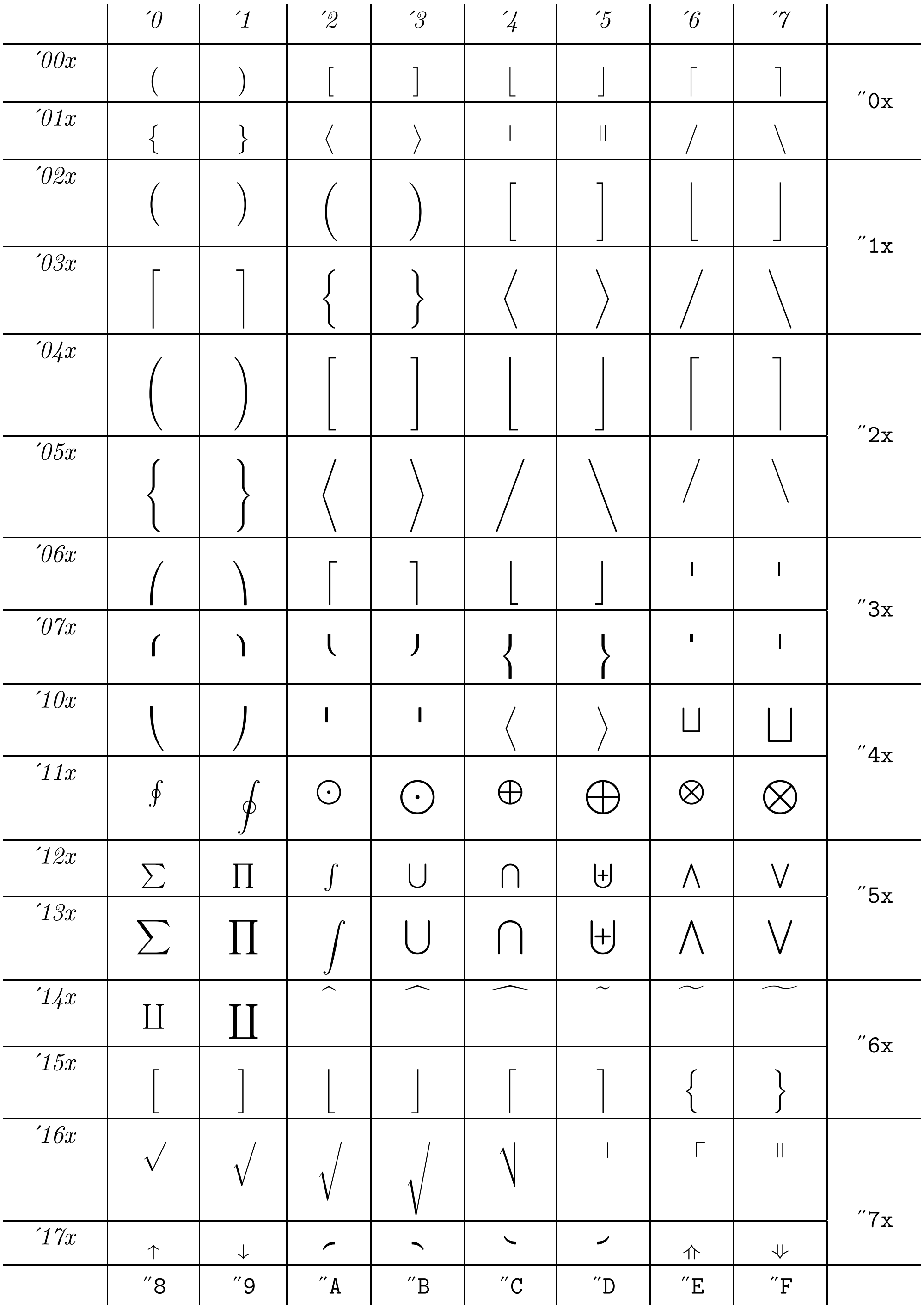


Figure D.4: The `cmex` encoding: 128 glyphs.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	$\boxdot$	$\boxplus$	$\boxtimes$	$\square$	$\blacksquare$	$\centerdot$	$\lozenge$	$\blacklozenge$	″0x
´01x	$\circlearrowright$	$\circlearrowleft$	$\rightleftharpoons$	$\leftrightharpoons$	$\boxminus$	$\Vdash$	$\Vvdash$	$\vDash$	
´02x	$\twoheadrightarrow$	$\twoheadleftarrow$	$\leftleftarrows$	$\rightrightarrows$	$\upuparrows$	$\downdownarrows$	$\upharpoonright$	$\downharpoonright$	″1x
´03x	$\upharpoonleft$	$\downharpoonleft$	$\rightarrowtail$	$\leftarrowtail$	$\leftrightarrows$	$\rightleftarrows$	$\Lsh$	$\Rsh$	
´04x	$\rightsquigarrow$	$\leftrightsquigarrow$	$\looparrowleft$	$\looparrowright$	$\circeq$	$\succsim$	$\gtrsim$	$\gtrapprox$	″2x
´05x	$\multimap$	$\therefore$	$\because$	$\doteqdot$	$\triangleq$	$\precsim$	$\lesssim$	$\lessapprox$	
´06x	$\eqslantless$	$\eqslantgtr$	$\curlyeqprec$	$\curlyeqsucc$	$\preccurlyeq$	$\leqq$	$\leqslant$	$\lessgtr$	″3x
´07x	$\backprime$	-	$\risingdotseq$	$\fallingdotseq$	$\succcurlyeq$	$\geqq$	$\geqslant$	$\gtrless$	
´10x	$\sqsubset$	$\sqsupset$	$\vartriangleright$	$\vartriangleleft$	$\trianglerighteq$	$\trianglelefteq$	$\bigstar$	$\between$	″4x
´11x	$\blacktriangledown$	$\blacktriangleright$	$\blacktriangleleft$	$\rightarrow$	$\leftarrow$	$\vartriangle$	$\blacktriangle$	$\triangledown$	
´12x	$\eqcirc$	$\lesseqgtr$	$\gtreqless$	$\lesseqqgtr$	$\gtreqqless$	$\yen$	$\Rrightarrow$	$\Lleftarrow$	″5x
´13x	$\checkmark$	$\veebar$	$\barwedge$	$\doublebarwedge$	$\angle$	$\measuredangle$	$\sphericalangle$	$\varpropto$	
´14x	$\smallsmile$	$\smallfrown$	$\Subset$	$\Supset$	$\Cup$	$\Cap$	$\curlywedge$	$\curlyvee$	″6x
´15x	$\leftthreetimes$	$\rightthreetimes$	$\subseteqq$	$\supseteqq$	$\bumpeq$	$\Bumpeq$	$\lll$	$\ggg$	
´16x	$\ulcorner$	$\urcorner$	$\circledR$	$\circledS$	$\pitchfork$	$\dotplus$	$\backsim$	$\backsimeq$	″7x
´17x	$\llcorner$	$\lrcorner$	$\maltese$	$\complement$	$\intercal$	$\circledcirc$	$\circledast$	$\circleddash$	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure D.5: The `msam` encoding: 128 glyphs.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	$\lvertneqq$	$\gvertneqq$	$\nleq$	$\ngeq$	$\nless$	$\ngtr$	$\nprec$	$\nsucc$	″0x
´01x	$\lneqq$	$\gneqq$	$\nleqslant$	$\ngeqslant$	$\lneq$	$\gneq$	$\npreceq$	$\nsucceq$	
´02x	$\precnsim$	$\succnsim$	$\lnsim$	$\gnsim$	$\nleqq$	$\ngeqq$	$\precneqq$	$\succneqq$	″1x
´03x	$\precnapprox$	$\succnapprox$	$\lnapprox$	$\gnapprox$	$\nsim$	$\ncong$	$\diagup$	$\diagdown$	
´04x	$\varsubsetneq$	$\varsupsetneq$	$\nsubseteqq$	$\nsupseteqq$	$\subsetneqq$	$\supsetneqq$	$\varsubsetneqq$	$\varsupsetneqq$	″2x
´05x	$\subsetneq$	$\supsetneq$	$\nsubseteq$	$\nsupseteq$	$\nparallel$	$\nmid$	$\nshortmid$	$\nshortparallel$	
´06x	$\nvdash$	$\nVdash$	$\nvDash$	$\nVDash$	$\ntrianglerighteq$	$\ntrianglelefteq$	$\ntriangleleft$	$\ntriangleright$	″3x
´07x	$\nleftarrow$	$\nrightarrow$	$\nLeftarrow$	$\nRightarrow$	$\nLeftrightarrow$	$\nleftrightarrow$	$\divideontimes$	$\varnothing$	
´10x	$\nexists$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$	$\mathbb{D}$	$\mathbb{E}$	$\mathbb{F}$	$\mathbb{G}$	″4x
´11x	$\mathbb{H}$	$\mathbb{I}$	$\mathbb{J}$	$\mathbb{K}$	$\mathbb{L}$	$\mathbb{M}$	$\mathbb{N}$	$\mathbb{O}$	
´12x	$\mathbb{P}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{S}$	$\mathbb{T}$	$\mathbb{U}$	$\mathbb{V}$	$\mathbb{W}$	″5x
´13x	$\mathbb{X}$	$\mathbb{Y}$	$\mathbb{Z}$	$\widehat{}$	$\widehat{}$	$\widetilde{}$	$\widetilde{}$		
´14x	$\Finv$	$\Game$					$\mho$	$\eth$	″6x
´15x	$\eqsim$	$\beth$	$\gimel$	$\daleth$	$\lessdot$	$\gtrdot$	$\ltimes$	$\rtimes$	
´16x	$\shortmid$	$\shortparallel$	$\smallsetminus$	$\thicksim$	$\thickapprox$	$\approxeq$	$\succapprox$	$\precapprox$	″7x
´17x	$\curvearrowleft$	$\curvearrowright$	$\digamma$	$\varkappa$	$\Bbbk$	$\hslash$	$\hbar$	$\backepsilon$	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure D.6: The msbm encoding: 128 glyphs.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	`	´	ˆ	˜	¨	˝	˚	ˇ	″0x
´01x	˘	¯	˙	¸	˛	‚	‹	›	
´02x	“	”	„	«	»	–	—		″1x
´03x	₀	ı	ȷ	ff	fi	fl	ffi	ffl	
´04x	␣	!	"	#	$	%	&	’	″2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	″3x
´07x	8	9	:	;	<	=	>	?	
´10x	@	A	B	C	D	E	F	G	″4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	″5x
´13x	X	Y	Z	[	\	]	^	_	
´14x	‘	a	b	c	d	e	f	g	″6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	″7x
´17x	x	y	z	{	|	}	~	-	
´20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	″8x
´21x	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ	
´22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	″9x
´23x	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§	
´24x	ă	ą	ć	č	ď	ě	ę	ğ	″Ax
´25x	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ	
´26x	ř	ś	š	ş	ť	ţ	ű	ů	″Bx
´27x	ÿ	ź	ž	ż	ĳ	¡	¿	£	
´30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	″Cx
´31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
´32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	″Dx
´33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
´34x	à	á	â	ã	ä	å	æ	ç	″Ex
´35x	è	é	ê	ë	ì	í	î	ï	
´36x	ð	ñ	ò	ó	ô	õ	ö	œ	″Fx
´37x	ø	ù	ú	û	ü	ý	þ	ß	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure D.7: The dcr encoding: 256 glyphs.