Euromath system: alphabets and fonts

Richard M. Timoney
School of Mathematics
Trinity College
Dublin 2
Ireland
richardt@maths.tcd.ie

April 1997

Abstract

We describe some of the principles behind the design of the Euromath system, concentrating on some aspects that are needed to explain how the system deals with alphabets and fonts and are not immediately apparent from a demonstration of the system. In particular we outline what an SGML DTD is and what SGML documents look like.

Our aim is to describe the considerations that are relevant to fonts, the SGML-encoding of non-ascii symbols and characters and the way the Type 1 fonts needed for the Euromath system (version 2) were generated using Type 1 versions of the TeX fonts.

1 Introduction to Euromath

As there are opportunities during the workshop[1] to see the Euromath system working, and there are published descriptions of the capabilities of the system (see [6], [7], [8], [4] and [4]), we concentrate on some of the aspects that are a little behind the scenes and are slightly technical.

It may help to have a brief overview of the purposes that the Euromath system was designed for. In summary, it was designed as a homogeneous computer working environment for mathematicians, which would provide access to the range of electronic services and facilities that mathematicians would use in their work. In particular, this includes

[1]Second EmNet/NIS Wokshop on Electronic Publications (El-Pub-97), organised by the Siberian Branch of the Russian Academy of Sciences at Novosibirsk, April 24-26, 1997.

1. Editing mathematical documents, which could be papers, letters, slides for talks, or lecture notes for students.

2. Being able to deal with the national languages of European countries. The system was envisaged as being designed for European mathematicians in particular and thus of course it should be able to deal with documents or data in the various European languages.

3. The system should provide an easy mechanism for retrieving data from mathematical databases. The idea of being easily able to consult the database of the Zentralblatt für Mathematik was a priority.

4. Electronic communication of information to other mathematicians (so that formulae could be electronically transmitted in a clean way within electronic messages) should be facilitated by the system.

One could elaborate on these goals but we refer to [6] for more information. The main point to explain is that it was decided to use a single editor as the common interface to all these tasks. The editor is a structured one which can be extended by programming in an API (applications programming interface) and the documents which the system deals with are SGML documents. SGML denotes "Standard Generalised Markup Language" and is an ISO standard for document storage (ISO standard 8879-1986). When this decision was made, SGML was an established standard and was recognized as a way of coping with large volumes of data (for example the voluminous data in an aircraft maintenance manual) in an efficient way. It is now used by many publishers as the basis for their electronic systems for managing the publication process and for keeping track of catalogues of their published material.

This structured aspect of the Euromath system is apparent to the user because every document that is edited in the Euromath system must belong to a document type (or document class). Thus, when a user starts creating a document with the system, the first step is to choose the type for the new system. Although SGML allows many different document types to exist (and there are many in existence), only a finite number are supplied with the system. These are article (which suffices for mathematical papers in general and chapters of books as well), slides (a simpler type suitable for preparing overhead projector slides for a lecture), letters (for writing letters, memos and FAX messages), html (for editing documents in the hypertext markup language used on the world-wide web) and a few others.

If the user chooses the article type, a template will appear with an almost empty article. See Figure 1 for a view of that template (it is a screen dump). A closer look will show that the top canvas is highlighted (or selected ready for input) and the message at the top of the window indicates that what is selected is intended for the Title component of the (so far empty) article. Selecting the next two canvasses would reveal that they are for the authors and the date. The fourth canvas is for the abstract and this is already indicated by the heading 'Abstract' above the empty canvas. This heading is not part of the document but part of the style sheet used by default for articles. A number of style sheets are provided, but there could be more. Following the abstract there is space for sections (each with a heading and automatic numbering) and references or a bibliography. If the user filled in these canvasses, the style sheet would dictate the font size and style used for the different components.

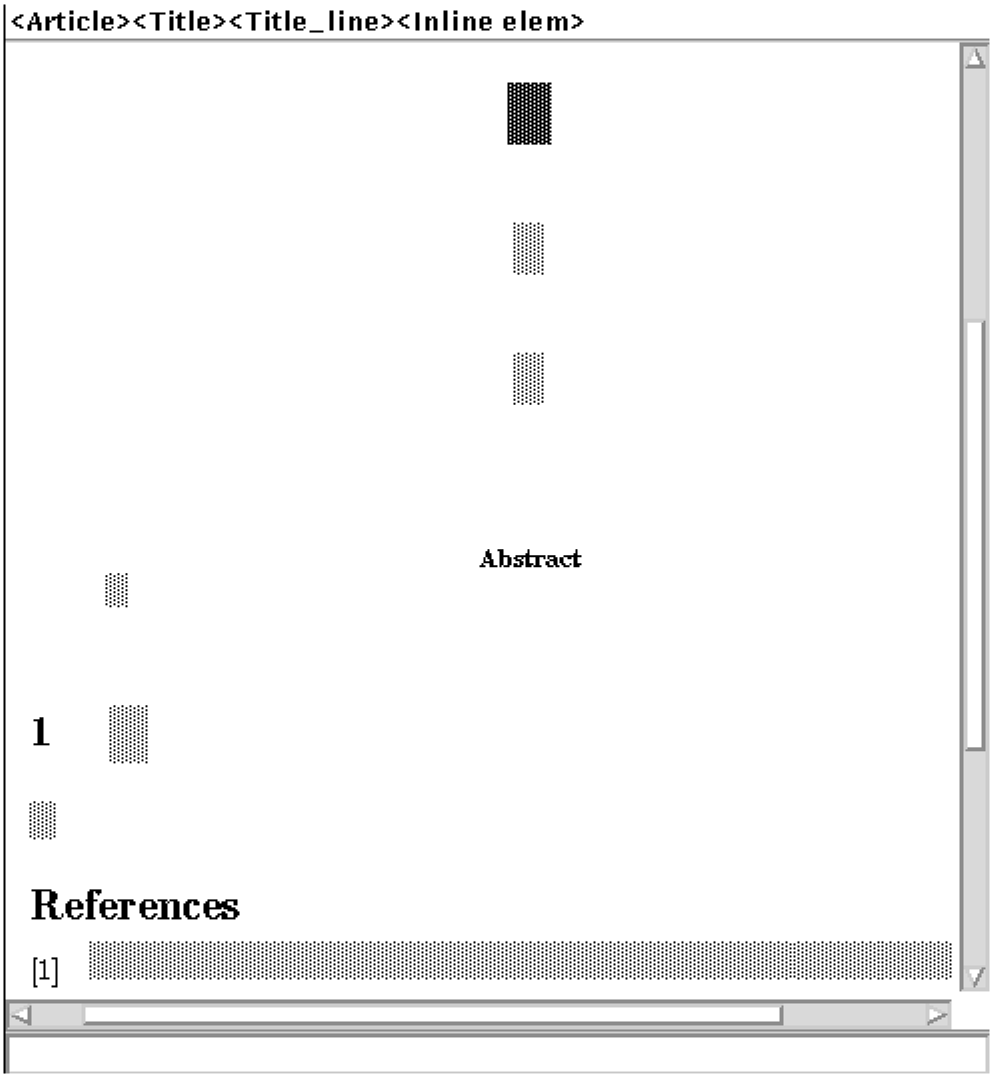

Figure 1: Template for article

2 Document types

The initial template presented to the user of the Euromath system (for example the article template in Figure 1) can be thought of as a visual representation of the underlying structure. Abstractly that structure is perhaps best thought of as an abstract tree (or directed graph) with the root = "Article" and child nodes for the Title, Author(s), Date, Abstract, Sections and Bibliography (or references). The template is just one way of visualizing the tree and it is a difficulty with explaining the system to new users that the abstract structure is the key idea, whereas the page layout is a secondary thing. The layout is driven by one particular style sheet, but could be transformed by switching to a new style where quite different fonts and layout could be used for the various headings, the text, the section numbering and the references. One style might suppress the date (not show it at all), for example.

The formal description of the structure is given by a DTD (= Document Type Definition) in the SGML syntax. As a lot of details are involved in the actual DTD, we give some fragments (some of which are simplified) to explain what is involved. The DTD for the Euromath article starts out like this.

```
<!DOCTYPE article [
<!ELEMENT article - - (Title?,Authors?,Date.a?,Abstract?,
Sections?,biblio?)>
. . .
```

The sequel (of the DTD) must explain the structure of the content of each of the elements that are mentioned (Title, Authors, and so on). To get a complete DTD, each element must have a defined structure, which can be defined in terms of other elements or one of a small number of predefined types including "EMPTY" and "#PCDATA" which is the SGML code for 'pure text'. Recursion may be involved in the DTD.

From the first line of the DTD, we can see that the elements directly below the top 'article' level come in a certain order (the commas separating the element names indicate that there is an order). Also all the elements below 'article' are optional (that is the significance of the ?). In the initial template all of these are included by default, but the system would allow them to be deleted (if for example we did not want an abstract).

To give an example how the rest of the structure is defined, here are some more lines from the DTD.

```
<!ELEMENT Authors - O (Author+)>
<!ELEMENT Author  - O (Name,Affil?)>
<!ELEMENT Name    - O #PCDATA>
```

We can see that the "Authors" element consists of one or more "Author" elements (that is the significance of the +) and each Author must have a "Name" and may have an "Affil" (which is intended for an address of some kind). The Name field is only allowed to contain pure text (and no substructure) in this simplified example.

SGML is a dual-purpose language. As well as providing a means for defining DTD's, it also provides a way to describe documents that obey a DTD. For instance, here is the beginning of a Euromath article which has no title as yet but is authored by Richard M. Timoney:

```
<!DOCTYPE ARTICLE SYSTEM "article.dcl">
<ARTICLE>
<TITLE>
</TITLE>
<AUTHORS>
<AUTHOR>
<NAME>
Richard M. Timoney
</NAME>
</AUTHOR>
</AUTHORS>
. . .
```

It is not hard to see that the explicit way the content of the document is tagged facilitates automatic processing of the document. If one wanted to count the number of authors in various documents of type article, it would be easy to do design a program for that purpose. If one wanted extract certain fields to make a summary or report about the various documents in a collection, that is also not a big problem provided the different components of the DTD are separated out sufficiently explicitly.

When designing a DTD, it is necessary to try and envisage the likely purposes for the documents that will be created following the DTD. There is a temptation to be very explicit in the design and to think of tags for every conceivable purpose, but this is not a very practical thing to do and imposes a heavy overhead on anybody creating a document according to the DTD. Thus compromises are necessary to arrive at a DTD which is sufficiently versatile for the likely uses of the documents, while not being too complicated to use.

3 Text in SGML

For the present discussion, the most important aspect is how SGML copes with pure "text" (know by the SGML keyword #PCDATA, which stands for Parsed Character Data). To be precise, one should mention that the way PCDATA is represented in SGML is something that can be defined in a DTD, but we concentrate on the usual way it is done. Normally the whole of an SGML document is represented in plain readable ASCII. Some of the ASCII letters are used for special purposes. In particular the characters $<$ $>$ & " have special purposes and cannot be used to represent themselves. The rest of the printable ASCII symbols are used just as they are. Thus the ordinary alphabet lower case and upper case (a-z and A-Z) are represented just as themselves and many of the other letters on an American keyboard (such as 0-9 and various punctuation marks and parentheses) can be used as they are. There are ISO standard SGML names (known as entity names) for just about any symbol. Hence `é`is the entity for é and `É` is the entity name for É, for example.

Apart from a comprehensive set of names for a huge range of accented letters and symbols (technically know as entity names), there are no such things as "code pages" in SGML. There is no specified order for the different entities and no numerical code associated with them in the ISO standard. Thus "code pages" in the sense of personal computers do not enter into the theory of SGML and, in practice, there is no need to know what code page may or may not have been used by the sender in order to read an SGML document created elsewhere.

These are substantial advantages of SGML.

4 System design

Although no code pages are involved in SGML itself, SGML editors, such as the Euromath system do need to concern themselves with numerical coding for various internal purposes. These concerns arise as follows.

1. The key codes of the user's keyboard on the local system have to be mapped so that the Euromath system understands them in the same way as the local system.
2. The fonts used in the local system (for display while editing - the Euromath system shows a screen version of the printed document and the users manipulates this directly) have an inherent ordering or coding.
3. There are menus of symbols in the system, which are used to access symbols or characters which may not be available from the keyboard. These menus need to be designed in some logical way so that the user can easily find the appropriate menu and, secondly, the order of the characters within the menu should make it convenient for the user to find the appropriate entries in the menus.

In the Euromath system, we have adherent to the numerical codes defined in the X-window system for the Latin 1, Latin 2 and Cyrillic alphabets. When it comes to Mathematical symbols, the X-window codes are not at all comprehensive and so they have not been used. By contrast, there are standard SGML entity names defined for almost all the mathematical symbols and only a few new ones were needed.

The number of mathematical symbols is very large and it would not be convenient to search for them in one huge menu. (In fact the number of symbols is also greater than the maximum number allowed by the system for a single menu.) It was necessary to divide the symbols up into groups for the different menus in such a way that the average mathematical user would be able to guess quickly which menu to look in for a given symbol. Mathematical Greek symbols are in one menu by themselves and there should be no great difficulty explaining that to any user. Mathematical relations are in another, thought this is harder to explain. Symbols like $\in<\subset$ seem to be clearly relations but this may not be apparent in all cases. A criterion which was used in case of doubt was that any symbol that had a negated version must be a relation. Another menu has arrows, but there are certainly arrows like $\rightarrow$ and double arrows like $\Rightarrow$ which can also operate as relations, and are therefore in the relations menu and not in the arrows menu. There is a menu for most of the remaining special symbols such as quantifiers, signs like tensor product signs and other miscellaneous things. Blackboard bold, Euler fraktur and calligraphic letters are in separate menus (and there are actually no standard SGML entity names for them - one might argue that they are just letters in a different font, but usually $\mathbb{R}$ is really a different symbol from R).

4.1 Fonts

The Euromath system can use arbitrary Type 1 fonts (see [1]). These are used both for screen display via X-windows and for printing, so that the problem of showing a faithful representation of the printed page on screen is simplified.

Type 1 fonts are outline fonts and this means that they contain a mathematical description of the shape of each character or symbol in the font. The display or (PostScript) printer calculates the boundary of the black region according to the size and position of the character and then blackens appropriate pixels according to its own resolution.

There are some constraints associated with using Type 1 fonts in the Euromath system. These include the following.

1. All the fonts to be used for a particular alphabet or set of symbols must have the same coding. Thus, for example, all the fonts to be used for the Latin 1 alphabet in different fonts (Times, Helvetica or courier, for example) and different styles of the same font (roman, italic and bold for instance) must have the same coding.

2. The fonts which are going to be available for direct keyboard input (as opposed to via menus) must be coded in the same way as the keyboard.

3. The alphabet or set of symbols in any menu of symbols must all belong to one font.

4. Accented characters must be included in the font under separate codes and may not be generated by overprinting an accent on top of a letter.

Our solution to these problems involved using Type 1 versions of the TeX fonts. One such set was generated by Basil K. Malyshev. They are freely available from the CTAN (Comprehensive TeX Archive Network) ftp sites and are known as bakoma fonts. This collection includes versions of the standard computer modern fonts for TeX (converted from the traditional MetaFont source invented by Knuth [3] to the Type 1 format) and also includes versions of the extended sets of mathematical symbols (which originated from the American Mathematical Society) and the new dc fonts which include accented letters fro Latin 1 and Latin 2. Unfortunately there is no Cyrillic font in this collection.

The problem with using these fonts was that they do not satisfy the various coding restrictions outlined above. Our solution was to re-encode the fonts to suit our purposes. This re-encoding was more extensive than re-encoding within a single font as the TeX encoding (even for the dc fonts) always involves mixing fonts in a way that is efficient in terms of saving on the total number of fonts needed, but is not compatible with many other systems.

The solution was to write a program (distributed with the Euromath system for anybody interested) to generate new Type 1 fonts by taking specified characters from different fonts and to write an encoding vector for the new font. This work was based on programs called t1utils for manipulating Type 1 fonts which are available by anonymous ftp and were written by I. Lee Hetherington. Type 1 fonts come in two different formats and the routines for drawing the characters are encrypted in both formats. However, the t1utils provide a way to read this encrypted part. One needs to be careful with combining parts of different fonts as there are various ways that a Type 1 font can call subroutines as part of the rules for drawing characters and these subroutines are numbered. However, these problems were overcome (at least for the bakoma fonts we used). Our amalgamating and re-encoding produced reasonably satisfactory results because the original fonts were all of the same "family". There were a few missing symbols in the collection because some common symbols are produced in TeX by overprinting (such as $\neq\not\subset\not\supset\notin\equiv$) and these were created by hand.

The Cyrillic fonts we used were not of the same TeX "family" and so the results are not aesthetically good. Moreover, we only have Cyrillic fonts in a limited number of styles (we have no Courier for example). Thus there is room for further work in this area.

Another area where it should be possible to make improvements concerns fine tuning of the fonts to work well at screen resolutions. The TeX-derived fonts we used are good at the resolutions of printers (say 300 dots per inch), but there are some undesirable effects on screen. There is a mechanism in Type 1 fonts for "hinting" which ensures that a thin line or descender on a character does not vanish altogether when drawn at small sizes on a low resolution device. There should be room for improvement in this area also.

References

[1] Adobe Systems Incorporated, Adobe Type 1 Font Format (version 1.1), Addison-Wesley, (1990).

[2] M. Gorecka, R. M. Timoney, T. M. Wolniewicz, *The Euromath interface to X.500 directory services*, Euromath Bulletin, **2** 1, 27-30, (1996).

[3] D. E. Knuth, The MetaFont Book, Addison-Wesley, (1986).

[4] H. Lenzing, B. von Sydow, R. M. Timoney, *Preparing for the future, the new Euromath system*, Euromath Bulletin, **2** 1, 19-26, (1996).

[5] B. K. Malyshev, *Problems of the conversion of METAFONT fonts to PostScript Type 1*, 1994 TeX Users group Annual meeting, Preprint, November (1994).

[6] B. von Sydow, *The design of the Euromath system*, Euromath Bulletin, **1** 1, 39-48, (1992).

[7] B. von Sydow, *Editing mathematics in the Euromath system*, Euromath Bulletin, **1** 2, 17-23, (1994).

[8] R. Timoney, *The construction of an interactive LaTeX translator for mathematical formulae*, Euromath Bulletin, **1** 2, 103-110, (1994).

Contents

1 Introduction to Euromath **1**

2 Document types **3**

3 Text in SGML **5**

4 System design **6**

4.1 Fonts 7