Baskerville

The Annals of the UK TEX Users' Group **Editor: Editor: Sebastian Rahtz** **Vol. 5 No. 4**

ISSN 1354–5930 **August 1995**

Articles may be submitted via electronic mail to `baskerville@tex.ac.uk`, or on MSDOS-compatible discs, to Sebastian Rahtz, Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, to whom any correspondence concerning *Baskerville* should also be addressed.

This reprint of *Baskerville* is set in Times Roman, with Computer Modern Typewriter for literal text; the source is archived on CTAN in `usergrps/uktug`.

Back issues from the previous 12 months may be ordered from UKTUG for £2 each; earlier issues are archived on CTAN in `usergrps/uktug`.

Please send UKTUG subscriptions, and book or software orders, to Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB. Fax/telephone: 0121 476 2159. Email enquiries about UKTUG to `uktug-enquiries@tex.ac.uk`.

Contents

I Editorial

1 Cathy Booth memorial

As explained in the last issue of *Baskerville*, the committee of UKTUG has been working to honour the memory of Cathy Booth in its various activities and donations to good causes. Continuing an occasional tradition started by Malcolm Clark, we decided to award a prize in Cathy's name at this year's TEX User Group conference. At the meeting a poll of delegates was arranged, and Donald Knuth was also asked to nominate his favourite papers. The first choice was overwhelmingly for the work of T. V. Raman, who presented a paper, and demonstrated his audio 'views' of TEX documents. For anyone who has met Raman, or experienced his work, this will come as no surprise — his system for speaking mathematical formulae (and any other TEX construct!) is an incredible achievement, recognized by the national award for his doctoral thesis last year. Raman (and his guide-dog Aster) received a remarkable standing ovation when the Cathy Booth prize was awarded at TUG95, for what one voter aptly described as his 'contribution to humanity'. We are very pleased to be allowed to reprint his paper from the conference preprints in this issue of *Baskerville*.

2 Looking for a new captain of *Baskerville*

At the 1995 meeting of the TEX User's Group, I was elected as Secretary of the group by the Board of Directors. UKTUG members who are also members of TUG will be aware that the group has been going through rough times recently, with its publication program adrift, and its relationship with other groups being questioned. Since I want to give my best effort to getting TUG back on its feet, and working for the good of all TEX users, I have very reluctantly decided that I will have to give up work for the UK TEX Users' Group, including the editorship of *Baskerville*. A replacement is being sought — if you think you can do it, let's hear from you. The most important criterion is that you be able to garner 24 pages of interesting material every other month — all sorts of people can help with the typesetting, proofing, production etc, but it is the firm overall control that we want.

3 Corrigendum

In *Baskerville* 5.3 Rosemary Bailey gave a list of the members of the committee of UKTUG in 1991–1992. Unfortunately, she omitted the names of two members: Angus Duggan and David Osborne. She has asked the *Baskerville* editor to pass on her apologies.

4 Words from the Treasurer

Peter Abbott asks me to remind/inform members of the following points:

4.1 Book discounts

Any book from the Addison-Wesley Complete Computer Science catalogue may be ordered from UKTUG. The published price should be discounted by 10% rounded to the nearest 5p. If you are unsure please let me know the ISBN and I will quote a price. A-W books are delivered direct so I would appreciate notification of delivery.

Books from O'Reilly are as listed on the sheet included in *Baskerville* from time to time. O'Reilly books are forwarded by me so again I would appreciate notification of receipt.

Cheques should be made payable to 'UKTUG' and sent to Peter Abbott (see banner for address, phone etc).

4.2 4allTeX CD-ROM 3rd edition

This newly released CD is available for £25. It now has *two* CDs, the latest emTEX, and more goodies than you can imagine. A few copies of the 2nd edition are on sale for £5.

4.3 EmTEX new release

By the time you read this you should have received the update mentioned in the last edition of *Baskerville*. Please contact me if you paid the £30 in 1995 or the £5 for the update service, and have not received your disks.

II An Audio View of (LA)TEX Documents

T. V. Raman
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square, Building 650
Cambridge, MA 02139
Email: `raman@crl.dec.com`

Summary
ASTER —Audio System For Technical Readings—is a computing system that produces audio renderings from the *same* (LA)TEX source used to produce the printed document. [Raman 1992] described our preliminary work on this project. At the time, correct handling of user-defined (LA)TEX macros was described as one of the key issues in building a fully extensible audio rendering system. ASTER [Raman 1994] has now been fully implemented. This paper reports on the approach used in ASTER to handle user-defined macros.

ASTER treats macro definitions as introducing new object types into the document logical structure. The (LA)TEX macro consists of two parts; a declaration, and a series of TEX commands that the macro expands into. The macro expansion is nothing but a visual rendering rule that specifies how TEX should display instances of the object represented by the macro.

ASTER provides an equivalent mechanism for extending the class of logical structures that are recognized. Once ASTER has been told about a user-defined macro, audio rendering rules for the new object type introduced by this macro can be defined in AFL (Audio Formatting Language).

The approach used not only makes ASTER fully extensible; it points out a unique advantage of (LA)TEX—the ability of the author to encode semantic meaning into the markup by extending the document model in ways appropriate to the specific document instance that is being encoded.

1 Introduction

ASTER

ASTER—Audio System For Technical Readings—is a computing system that aurally renders electronic documents marked up in the (LA)TEX family of markup languages (see [Raman 1994] for details). ASTER uses the structural markup present in the electronic source to advantage in producing high-quality, interactive audio renderings. This paper focuses on a specific aspect of the problem; namely that of flexibly rendering the extended document logical structure encapsulated in a (LA)TEX document.

One primary advantage of (LA)TEX is the flexibility it provides the author in defining logical structures that are specific to a particular document instance. In this sense, the class of logical structures that can be encapsulated in a (LA)TEX document is extensible. (LA)TEX macros allow an author to abstract away the layout details. At the same time, they provide a powerful mechanism for defining new constructs that are not already present in the document style (DTD in SGML parlance) in use. As a consequence, when introducing a new piece of mathematical notation, an author can first define a new (LA)TEX macro that produces a desired layout, and then use this newly defined construct throughout the document.

The flexibility of the (LA)TEX macro facility initially proved a major stumbling block in building a fully extensible audio rendering system. A system that attempts to produce aural renderings by *mapping* the built-in (LA)TEX commands to an equivalent aural representation faces the severe shortcoming of not being able to render documents that contain user-defined macros. At the same time, it is impossible to translate such user-defined (LA)TEX macros into a suitable aural representation. This is because TEX in its full glory is a Turing-complete programming language, and saying “we can translate a general TEX macro to audio” is equivalent to saying that “Given a TEX program, we can predict the result”. Being able to achieve the above without actually running TEX on the program (document fragment) would amount to being able to solve the Halting Problem!

In the rest of this paper, we describe the solution used in ASTER to circumvent this difficulty. The solution we used in fact turns the presence of user-definable (La)TeX macros into an advantage. Such user-defined constructs allow AsTeR to glean even more information about the document logical structure than would be possible if the document were encoded using only the built-in (La)TeX operators; as a consequence, the audio renderings produced are also significantly better.

2 Document Models in AsTeR

AsTeR produces audio renderings by first extracting the document logical structure. In this model, all forms of rendering, *i.e.* visual, aural, etc. are regarded as a projection of the structure present in the information being conveyed onto the medium being used to communicate the information. Thus, typesetting a document requires visual formatting—projecting the information structure onto a two-dimensional visual tablet; aural rendering requires presenting the structure using various features of the auditory display.

The recognizer used in AsTeR extracts logical structure present in documents encoded in the (La)TeX family of languages. An important feature of this recognizer is that it works on the entire gamut of encodings, ranging from plain ASCII documents, *i.e.* no explicit markup, up to documents containing completely unambiguous encodings of the logical structure.

The basic document model used in AsTeR is the attributed tree. Each hierarchical level of the document is modeled as a node in this tree. Each node can have content, children and attributes. Using object-oriented terminology, each different kind of node of the tree is called an *object* and represents a document element. Thus, “chapter”, “section”, “paragraph”, and “sentence” are all objects. If a document contained five sections, its representation in AsTeR would have five instances of object “section”. This object-oriented terminology is used because AsTeR actually uses CLOS objects in this fashion. The use of an object-oriented language was instrumental in allowing us to develop and implement the ideas in AsTeR incrementally and effectively.

This attributed tree structure is augmented to represent mathematical content; we call this augmented representation the *quasi-prefix form*, (see figure 1 below). Expressions that are completely unambiguous, *e.g.* $x+y$, are captured in their prefix form. In addition to linearizing the underlying tree structure, mathematical notation uses *visual attributes* such as superscripts and subscripts, whose interpretation is context-dependent. We extend the prefix form to capture such visual attributes—hence the name *quasi*-prefix.

$$\begin{tikzcd}
\text{left-superscript} & \text{accent} & \text{superscript} \\
 & \textbf{math object} \arrow[ul] \arrow[u] \arrow[ur] \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{left-subscript} & \text{underbar} & \text{subscript}
\end{tikzcd}$$

Figure 1. A math object with attributes. Each of the attributes themselves contain math objects.

The next section describes how this model is extended to encapsulate the use of user-defined constructs in (La)TeX.

3 Extended Logical Structure

The (La)TeX facility can be used to extend the document logical structure by defining new constructs. Thus, an author preparing a manuscript on inference logic might define

```
\newcommand{\inference}[2]{{#1\over#2}}
```

and write

```
\inference{x}{y}
```

and use this construct throughout the document.

Notice that defining the `\inference` as shown above and using it to encode inference statements is distinct from and more powerful than just using the TeX built-in operator `\over` throughout the document. A commonly mentioned advantage in this context is that using the newly defined construct `\inference` will permit the author to easily change the notation used to denote *inference*. Notice, that this is in fact the same as saying that

> If distinct elements in a document instance are marked up using distinct constructs, then it is possible to recognize and process these elements in a multiplicity of ways.

In AsTeR, the (La)TeX facility of defining a second `\inference` macro that produces a different layout for *inference* can be generalized to the notion of different *audio renderings* for *inference*.

As explained above ("Document models"), AsTeR achieves its aural renderings by building a rich internal representation of the document content. In this representation, each document element[1] E is represented by an instance of object O_E. AsTeR provides a predefined type O_E for each of the built-in constructs in (LA)TEX. Thus, we could represent the use of `\inference` defined above in terms of object O_{over}. However, notice that this would mean losing valuable information. When building up the internal representation, the additional semantic information provided by the author's use of the `\inference` construct is very useful. In addition, expanding all (LA)TEX macros results in a pure layout representation, which is not appropriate for producing aural renderings (see [Raman 1992]). If we were to represent instances of `\inference` in terms of O_{over}, AsTeR would be forced to render `\inference` the same as the `\over` construct. Though the author in this particular example may have chosen to use the same visual rendering for inferences that is normally used for fractions, the same may not carry over well to the aural domain.

Representing Extended Logical Structure

AsTeR solves the problem of representing and rendering the extended logical structure arising from user-definable macros by considering each macro definition as introducing a new object type. Instances of a macro M, are represented by instances of object O_M. Thus, in the example shown above, the definition of the construct `\inference` introduces a new object type $O_{\mathrm{inference}}$. The (LA)TEX macro consists of two parts; a declaration, and a series of TEX commands that the macro expands into. The macro expansion is nothing but a visual rendering rule that specifies how TEX should display instances of the object represented by the macro.

AsTeR provides an equivalent mechanism for extending the class of logical structures that are recognized. Once AsTeR has been told about a user-defined macro, audio rendering rules for the new object type introduced by this macro can be defined in AFL (Audio Formatting Language). Notice that such audio rendering rules have to be defined by the user, just as the (LA)TEX macro is defined by hand. It is not possible in general to translate the TEX macro into a set of audio rendering rules. This is because the TEX macro is capable of performing any arbitrary computation permitted by the operators present in the TEX language [Knuth 1984]—a Turing-complete programming language.

4 Rendering Information

AsTeR renders information by applying *rendering rules* to the internal representation described above ("Document models"). The system of rendering rules used in AsTeR and the language in which they are written (AFL—Audio Formatting Language) are described in detail in [Raman 1994]. In a sense, AFL is to audio formatting as Postscript is to visual formatting, although AFL is a much smaller language.

Here, we show a small example of such a rendering rule for a user-defined macro. In the following, we use CLOS generic function read-aloud. For the present, let us assume that function read-aloud executes the necessary actions to render its argument.

After extending AsTeR to process the (LA)TEX macro `\inference` shown above ("Logical structure"), we can define

```
(defmethod read-aloud((inference inference))
  "Sample rendering for object inference."
  (read-aloud (argument 1 inference))
  (read-aloud "implies")
  (read-aloud (argument 2 inference)))
```

Given $\frac{A}{B}$, this produces "A implies B".

If we wished to produce a rendering that inverts the order in which the arguments to macro `\inference` are rendered, we would define:

```
(defmethod read-aloud((inference inference))
  "Renders inference with arguments reversed."
  (read-aloud "We know")
  (read-aloud (argument 2 inference))
  (read-aloud "because")
  (read-aloud (argument 1 inference)))
```

which produces "We know B because A".

Switching between these two rendering rules has the effect of inverting a proof-tree! Notice that writing a new rendering rule for an object O_E has the same effect as redefining the (LA)TEX macro that corresponds to E.

[1]We use the term *element* loosely to mean a logical unit of the document.

ASTER makes it easy to write several rendering rules for the same object and also allows rendering rules to be partitioned into rendering *styles*. Such *styles* can be thought of as being analogous to LATEX styles, but with one important difference. Due to the non-interactive nature of traditional paper documents, a paper is typically typeset in a given style. It is not possible for the reader to change the style in which the document is typeset. Typically, we do not feel the shortcoming of not being able to change the way a mathematical expression is rendered when reading a printed paper because the eye is capable of reading the various parts of an expression in any order that is convenient. However, when listening to an aural presentation, the listener does not have this flexibility. In other words, an active reader peruses a printed paper, a passive display, whereas in the case of audio, these roles are reversed—the aural display scrolls *actively* past a passive listener.

ASTER overcomes these difficulties by being a fully interactive system. It is possible for the listener to interrupt the rendering, change the rendering style in use, and listen to the document. In an interactive session with ASTER, switching between rendering styles (a collection of rendering rules for different objects) and invoking individual rendering rules can be done with a few keystrokes, making it easy for a listener to obtain many different views of a document. This facility enables *active* listening.

ASTER derives its power from representing document content as objects and by allowing multiple user-defined rendering rules for individual object types. These rules can cause any number of audio events (ranging from speaking a simple phrase, to playing a digitized sound). The pitch of the voice, the physical head-size of the virtual speaker, the volume, and many other parameters can be changed by rendering rules, making it easy to create sound cues to help display structure. In fact, the design of ASTER does not restrict the system to producing purely aural renderings; there is nothing to preclude us from defining renderings that produce truly multimodal output; *i.e.* renderings where the traditional visual rendering is augmented with aural feedback. We conjecture that such multimodal renderings may prove very useful for persons with learning impairments.

To give an example of a multimodal rendering, the logo for ASTER is

ASTER

and is produced by (LA)TEX macro `\asterlogo`. After appropriately extending ASTER to recognize this macro, we can define an audio rendering rule for object *asterlogo* that produces a bark when rendering instances of this macro. Thus, the same piece of markup `\asterlogo` produces the picture of Aster[2] when rendered visually, and an appropriate sound[3] when rendered aurally.

This feature was exploited to advantage when producing the audio formatted version of the author's thesis. The dedication page of the thesis contains a large picture of Aster, and the audio formatted version[4] contains a verbal description of the picture, accompanied by the sound of Aster panting in the background. You can listen to this example on the WWW—visit the ASTER home page by following the link to the ASTER demonstration from my home page[5] and clicking on the picture of Aster.

Several ideas come together to make all this possible. First, logical structure is of paramount importance—not its display on any one particular medium. The more a document makes structure explicit, the better the document can be displayed on (projected onto) several different media.

Next, the use of (LA)TEX macros to encode structure makes it possible to have a system like ASTER, in which the internal structure can be extended to fit a document. This allows the encoding of the structure in a flexible, uniform, and consistent representation such as an attributed tree, with the addition of the quasi-prefix form for dealing with mathematics.

Finally, providing different rendering rules and styles and a flexible way to switch among them makes it possible to obtain multiple views of a document in an interactive fashion.

5 Conclusion

The approach used in ASTER to exploit the additional semantic information present in the electronic encoding in the form of user-defined constructs points to an important feature of markup systems like (LA)TEX that is currently missing to a certain extent in systems like SGML. When ASTER was at its inception, I firmly believed that one should use a semantic-oriented DTD to encode a document in order to be able to produce high-quality audio renderings. I still

[2]Aster is my guide-dog.

[3]The bark is that of a generic dog, Aster is too well trained to bark, and could not therefore be recorded.

[4]An audio formatted version of the thesis produced by ASTER (about 6 hours) is being distributed by RFB—Recordings For The Blind—as the first fully computer-generated talking book.

[5]|http://www.research.digital.com/CRL/personal/raman/raman.html|

believe this; however the work on ASTER does point out one shortcoming with the fixed document DTD model. Given that mathematical and technical notation is being invented all the time, a fixed DTD forces the author to encode new constructs using *only* primitives that are provided by the DTD. As a consequence, authors end up using a presentation-oriented encoding even though the DTD in use is one that is semantically oriented.

To make this concrete, consider the case of the *inference* construct described above ("Logical structure"). If the document were being encoded using a fixed non-extensible DTD that only provides a *fraction* element, the author would be forced to encode *inference* using this element.

Since in general it is not possible to define an all-encompassing DTD that covers every possible kind of math notation (those currently known and those yet to be discovered) extensibility of the DTD as provided by (LA)TEX is of vital importance.

Another good example of this facility in (LA)TEX being put to good use is the HyperTEX system —an extension to TEX that allows the user to view his legacy (LA)TEX documents as online hypertext. Conceptually, we can think of `\ref` and `\label` as being object types; traditionally, these cause specific marks to appear on paper when rendered visually by TEX; to a system like HyperTEX these turn into *active* links that a user can follow interactively.

The ability to produce multiple renderings of the same object provided by ASTER was introduced in the context of aural presentations. However, such multiple presentations become equally relevant when interactively perusing online documents visually. For instance, when reading a document that presents a complex proof, a user may wish to have the same proof displayed as an outline in one window, and as a proof-tree in another (see [Lamport 1993]). In the case of paper documents, the user has to use her imagination to achieve such multiple views —though she is aided in this by the visual notation. In the interactive scenario presented by electronic documents, the previewer can provide some additional functionality to aid in this process.

References

[Knuth 1984] Knuth, D. E. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, Massachusetts, 1984.

[Lamport 1993] Lamport, L. "How to write a proof". Technical Report 94, DEC Systems Research Center, Palo Alto, CA, 1993. To appear in *American Mathematical Monthly*.

[Raman 1992] Raman, T. V. "An audio view of TEX documents". *TUGBoat* **13**(3), 372–377, 1992.

[Raman 1994] Raman, T. V. *Audio System for Technical Readings*. Ph.D. thesis, Cornell University, 1994.

III HH Gets Carried Away: the hhmuf, hhflxbox and hhcount packages

Herman Haverkort
Email: `herman@fgbbs.iaf.nl`

Summary
In this article I present some of the features of three packages, hhmuf, and hhcount, that I have written recently. However, this presentation is far from complete. A more detailed manual and the packages themselves can be obtained from the author at FGBBS (tel. +31 85 21 70 41) or CTAN (`macros/latex/contrib/supported/hh`. hhmuf offers 'multinotes' —special cheery footnotes to be used in special situations, including so-called 'forbidden environments'. hhflxbox's provides self-scaling frames: encircling macros are provided but you can define whatever features you like by means of the macros provided. Finally hhcount is presented: macros to handle simple and composite counters in a fancy way.

1 Some Features of the hhmuf Package

1.1 Reusable Footnotes and Recycled Markers

Suppose you have to typeset some tables containing many entries which are amplified by footnotes outside the table. Several entries, possibly but not necessarily in the same table, refer to the same footnote. If references to the same footnote appear in different tables on different pages, the footnote text should be set on all pages involved, while the footnote marker should be the same each time the footnote is set. The first to avoid unnecessary turning over, the second to avoid confusion.

Typesetting such tables is not very easy in basic LaTeX. A relatively easy way to do the job is probably as follows:

1. first get markers for all the footnotes and define macros to set the footnote markers;
2. then define a macro `\tablenotes` which typesets the footnotes;
3. then typeset the tables, using the macros defined in the steps mentioned above;

Here is some example input:

```
% step 1:
\newcommand\getfootnotemarker[1]{%
  \stepcounter{footnote}%
  \newcounter{#1}%
  \setcounter{#1}{\value{footnote}}%
  \expandafter\newcommand\csname #1\endcsname
    {\footnotemark[\value{#1}]}}
\getfootnotemarker{notea}%
\getfootnotemarker{noteb}%
\getfootnotemarker{notec}%

% step 2:
\newcommand\tablenotes{%
  \footnotetext[\value{notea}]{%
                     First example footnote}%
  \footnotetext[\value{noteb}]{%
                     Second example footnote}%
  \footnotetext[\value{notec}]{%
                     Third example footnote}}
```

```
% step 3:
\tablenotes
\begin{center}
\begin{tabular}{|l|l|}%
\hline
  name       & amount in \$ \\
\hline
  Achterberg & 100          \\
  Bosman     & 150\notea    \\
  Evers      & 125\noteb    \\
  Gerritsen  & 145          \\
  Hooier     & 170\notec    \\
  Jansen     & 165\notea    \\
\hline
\end{tabular}
\end{center}
```

And the resulting output:

name	amount in $
Achterberg	100
Bosman	150[6]
Evers	125[7]
Gerritsen	145
Hooier	170[8]
Jansen	165[6]

This article is too short to demonstrate it, but when typesetting multiple tables this way problems are likely to arise. If you typeset another table on the same page, calling `\tablenotes` again will cause the footnote texts to be typeset twice on the same page. If you typeset another table on another page, *all* table footnote texts will be typeset on that page, even if the table which is on that page does not refer to all of them.

The hhmuf package solves these problems, although the solution of the twice-on-the-same-page problem may be buggy in rare contexts. If you use the hhmuf package you can replace the previous listing by the following:

```
\mufhire note1:{First example footnote}%
\mufhire note2:{Second example footnote}%
\mufhire note3:{Third example footnote}%
%
\begin{center}
\begin{tabular}{|l|l|}%
\hline
  name       & amount in \$      \\
\hline
  Achterberg & 100               \\
  Bosman     & 150\muf note1:{}  \\
  Evers      & 125\muf note2:{}  \\
  Gerritsen  & 145\muf:{Example
          of an incidental note} \\
  Hooier     & 170\muf note3:{}  \\
  Jansen     & 165\muf note1:{}  \\
\hline
\end{tabular}
```

[6] First example footnote
[7] Second example footnote
[8] Third example footnote

```
\end{center}
```

And get this result:

name	amount in $
Achterberg	100
Bosman	150♠
Evers	125△
Gerritsen	145♣
Hooier	170⊖
Jansen	165♠

You will note that `\mufhire` *label*`:{`*footnote text*`}` is used to define footnotes. Its opponent is, of course, `\muffire` *label*`:`, which undefines footnotes, while `\muf` *label*`:{}` is used to set previously defined footnotes. As shown in the example, `\muf:{`*footnote text*`}` can be used to set incidental footnotes. `\muf:{`*footnote text*`}` actually acts as an abbreviation for hiring, typesetting and firing an incidental note. Thus it is well-suited for setting normal in-text footnotes.

hhmuf does not use common footnote markers. Most common sets of symbols have a well-defined order which makes them ill-suited for hhmuf since the hhmuf macros do not respect that order. While defining footnotes, markers are assigned in turn. Thus there is no need to restart footnote numbering every chapter or every page, because you never run out of markers, unless you hire a lot of them without ever firing any. Restarting footnote numbering does not make sense anyway because there is no such thing as a *first* marker.

The set of markers used by hhmuf can be fully specified by the user, either by selecting one of the predefined sets or by compiling a new one. The predefined sets are the following:

set name	markers included
black	● ♦ ▼ ♣ ■ ▲ ◀ ♠
circlox	⊘ ⊙ ⊠ ⊕ □ ⊛ ⊟ ⊗ ⊚ ⊡ ⊖ ⊞
fuss	∗ ♢ ⊛ ⋇ ♯ ⋆ ♣ ℵ ∞ ↻
geometry	♦ □ ▼ △ ■ ◁ ▲ ♢ ◀ ▽
misc	♠ △ ⊖ ♣ × ♢ ⊗ ≀ ⊙ ∨ ⊕ ∞ ⋆ ⊘ + ◁ ⊤ ● ∇
music	♯ ♭ ♮
strokes	⊤ × ⋎ + ⊻ ⋕ ⊼ ≀

For details see the documentation in the package file.

1.2 The Forbidden Environment Problem

Suppose I typed several pieces of text that have to be typeset all in the same special way. For that purpose I (re)defined an environment `specialtext`:

```
\renewenvironment{specialtext}%
    {\par$\bigtriangledown$\par}%
    {\par$\bigtriangleup$\par}
```

▽

If I have a piece of text like this paragraph, which refers to a footnote,[9] this should cause me no problems.

△

Then I defined a package option in the package I used to typeset my document. If I specified that option when loading the package, then `specialtext` would be defined as follows:

```
\renewenvironment{specialtext}%
    {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
    {\egroup
```

♠ First example footnote
△ Second example footnote
♣ Example of an incidental note
⊖ Third example footnote
[9]which occurs frequently in my ever changing text

```
     \begin{center}\fbox{\box0}\end{center}}
```

Now problems arise. I typeset the same text again:

If I have a piece of text like this paragraph, which refers to a footnote,[10] this should cause me no problems.

The paragraph is shown framed all right, but something went wrong with the footnote. With the new definition of `specialtext`, the footnote suddenly appears in a 'forbidden' environment and therefore it actually disappears. Although the in-text marker is typeset, there is no note at the foot of the page.

While writing this article I discovered Kresten Krab Thorup's style file `ftn.sty`,[11] which attempts to solve this problem but does so quite buggily. Footnotes disappear when forbidden environments are nested. When multiple footnotes are type-set in forbidden environments, footnotes are repeated and their numbering is wrong. A modified version of `ftn.sty` exists (by Zdenek Wagner), which solves the repetition problem correctly, and suppresses incorrect numbers by omitting them: at least that is what I got. I tried to contact Krab Thorup about making `ftn.sty` more robust, but have not so far succeeded. Nevertheless Krab Thorup's style file contained some very useful ideas, which I combined with my own ideas to construct a set of macros which seem to be quite robust. I will now show you the result: how easy it is to use hhmuf's footnotes in forbidden environments.

The kind of unpleasant surprises presented above is easy to prevent when typesetting footnotes with `\muf` instead of `\footnote`, as in:

```
\begin{specialtext}
If I have a piece of text like this paragraph,
which refers to a footnote\muf:{which occurs
frequently in my ever changing text}, this
should cause me no problems.
\end{specialtext}
```

When using the first definition of `specialtext`, this yields:

▽

If I have a piece of text like this paragraph, which refers to a footnote,[×] this should cause me no problems.

△

In the following example the second definition of `specialtext` is used, but I added one line of code to 'protect' the environment:

```
\renewenvironment{specialtext}%
    {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
    {\egroup
     \begin{center}\fbox{\box0}\end{center}}
\mufoff[specialtext]
```

And here is the result:

If I have a piece of text like this paragraph, which refers to a footnote,[◇] this should cause me no problems.

Without changing the text in my document, I could redefine `specialtext` to use forbidden environments in such a way that my footnotes did not disappear. `\mufoff` did the job.

1.3 Shortcomings: `minipage` *Fans Beware!*

hhmuf does not support `minipage`s yet. If `\muf` is used in a `minipage` environment, the footnote will be placed at the foot of the 'master page' instead of under the `minipage`!

[11] Available at CTAN as `macros/latex209/contrib/misc/ftn.sty`

[×] which occurs frequently in my ever changing text

[◇] which occurs frequently in my ever changing text

2 The hhflxbox Package

hhflxbox contains a number of boxing macros. The kernel consists of `\iframe`, which boxes things and sets self-scaling frames around, and `\sframe`, which sets more complex self-scaling and -stretching frames. Besides hhflxbox provides the encircling macros `\ringbox`, `\bellybox` and `\outringbox` (which use `\iframe`), the macros `\sepbox` and `\separbox`, which set empty space around boxes, and `\broadbox`, which boxes its argument in a `\vbox` of which the width is the line width minus some specified value.

2.1 `\sepbox` *and* `\separbox`

For the introduction of `\sepbox` and `\separbox` it is convenient to look at `\bellybox` first. `\bellybox` is one of the hhflxbox macros which can be used to encircle things, for example ③, which is set with: `\bellybox :{3}`.

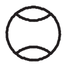
You probably notice that the circle around the digit is somewhat tight. This problem can be solved by putting a `\separbox` around the digit, as in `\bellybox:{\separbox{1pt}{3}}`, which yields: ③. Actually `\separbox{`*dimension*`}{`*stuff*`}` puts *dimension* wide empty space around *stuff* on all sides.

A more general form is:
`\sepbox(`*leftspace*`,`*topspace*`,`*rightspace*`,`*bottomspace*`){`*stuff*`}` which adds empty spaces of the specified widths to the sides of the box containing *stuff*.

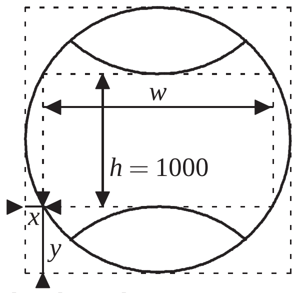
2.2 `\iframe`*: Isomorphous Frames*

`\iframe` is only a frame drawing *tool*: it does not draw frames itself but it can take care of the proper positioning and scaling of frames drawn by other macros. To explain the functioning of `\iframe` it is probably best to give an example of the development of a framing macro using `\iframe`.

Suppose we want to set self-scaling frames which have the following shape:



then we could imagine a box-shaped area in the frame which will contain the frame's contents (the inner dashed box in the figure below). Also we could imagine a box surrounding the frame (the outer dashed box in the figure).



Since `\iframe` expects the inner box height to be 1000 times the `\unitlength`, all dimensions have to be chosen so that the inner box height equals 1000 indeed. Then `\iframe` can scale the frame by setting the `\unitlength`. Furthermore `\iframe` expects the lower left corner of the outer box to have coordinates $(0,0)$. Taking these expectations in account we can design a macro which draws the frame:

```
\newcommand\sillyshape{%
  \begin{picture}(2000,2000)
    \thicklines
    \put(1000,1000){\arc(0,1000){360}}
    \put(1000,-498){\arc(662,749){83}}
    \put(1000,2498){\arc(-662,-749){83}}
  \end{picture}}
```

(`\arc` is defined in the curves package by I. L. Maclaine-Cross.) Now we can define a silly shape framing macro by defining `\sillyframe` as: `\iframe\sillyshape(`x`,`y`){`w`}{0pt}\ifrch\ifrcv:{#1}`. Actually we will set #1 in a `\sepbox(0pt,2pt,0pt,2pt){#1}` to prevent the frame from touching its contents. In the above example we have $x = 134$, $y = 500$ and $w = 1732$, so we write:

```
\newcommand\sillyframe[1]{%
 \iframe\sillyshape(134,500){1732}{0pt}%
  \ifrch\ifrcv:{\sepbox(0pt,2pt,0pt,2pt){#1}}}
```

```
Now we can put \sillyframe{all}
\sillyframe{sorts} \sillyframe{of}
\sillyframe{things} in silly frames.
```

Now we can put all sorts of things in silly frames.

Note that I do not claim this kind of silly frame to be good-looking: it is just an example.

The dimension `0pt` in the example above determines the minimal height of the silly frame's inner box. Sometimes it is necessary to define it because LaTeX's picture environment suppresses small line segments.

The macro `\ifrch` determines what should be done if the frame's contents width/height ratio is too small. By specifying `\ifrch` we instruct `\iframe` to center the contents. Instead of `\ifrch` we could have specified `\ifrl` or `\ifrr` to have the contents flush left or right.

The macro `\ifrcv` determines what should be done if the frame's contents height/width ratio is too small. `\ifrcv` yields vertical centering, while `\ifrt` and `\ifrb` yield top and bottom flushing.

If we put frames around for example page numbers, then the self-scaling properties of isomorphous frames may have an unpleasant result: numbers of the same type, like page number 21 and page number 25, might get differently sized frames because of their different natural sizes. This can be solved by redefining `\sillyframe` to specify a *unit name*, since all things typeset with the same unit name get equally sized frames. The unit name, for example `pagenr`, should be placed between the vertical alignment specification and the colon, like in:

```
\newcommand\pagenrframe[1]{%
  \iframe\sillyshape(134,500){1732}%
    {0pt}\ifrch\ifrcv pagenr:{%
      \sepbox(0pt,2pt,0pt,2pt){#1}}}

framed numbers like \pagenrframe{\oldstylenums
{21}} and \pagenrframe{\oldstylenums{25}}.
```

which yields (after compiling our document twice):

framed numbers like 21 and 25.

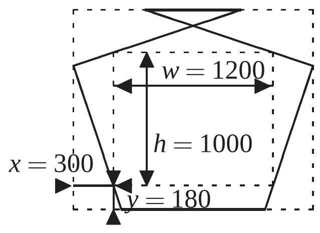
However, this is not fully satisfactory yet: now the frames are equally sized but the first frame is positioned higher than the second. This is no bug, it is a feature. No really, it is! It is, however, a sometimes unwanted feature. The solution is using `\lcenter` to center the frames on their line, like in:

```
pages \lcenter{\pagenrframe{\oldstylenums{21}}}
and \lcenter{\pagenrframe{\oldstylenums{25}}}
```

resulting in:

pages 21 and 25

As a final example of isomorphous frames, consider the following framing macro. Note that the inner box height is 1000 again, as expected by `\iframe`.



```
\newcommand\jarshape{%
  \begin{picture}(1800,1500)
    \thicklines
    \put(360,0){\line(-1,3){360}}
    \put(0,1080){\line(3,1){1260}}
    \put(540,1500){\line(1,0){720}}
    \put(1440,0){\line(1,3){360}}
    \put(1800,1080){\line(-3,1){1260}}
    \put(360,0){\line(1,0){1080}}
  \end{picture}}

\newcommand\jarframe[1]{%
 \iframe\jarshape(300,180){1200}{10pt}%
  \ifrch\ifrb:{\separbox{1pt}{#1}}}
```

2.3 `\ringbox`, `\bellybox` *and* `\outringbox`*: Encircling*

`\ringbox{`*optional unit name*`}:{`*stuff*`}` sets a circle around *stuff*. The specification of a unit name is optional; its use is explained above.

`\outringbox` is very much like `\ringbox`, but the following example demonstrates their difference:

```
1\ringbox:{2}3 and 1\outringbox:{2}3
```

yields:

1②3 and 1②3

If `\ringbox` is used, the circle contributes to the width, height and depth of the result. If `\outringbox` is used, the circle does not contribute any width, height or depth, so that the text is typeset as if the circle were not present and the circle were added after typesetting the text.

The result of `\bellybox` is a circle which contributes a bit to the dimensions of the encircled result but also sticks out a bit (by 10 percent of its radius to be sort of exact). So `\bellybox` is an intermediate form of `\ringbox` and `\outringbox`.

2.4 `\sframe`*: Stretchable Frames*

Putting `\sframe` to good use is a rather complex task. `\sframe` assembles user-defined frame components which actually are macros which set the values of a box and several dimension registers. Therefore I decided to give only an example of what can be achieved in this article; for explanation see the manual and demo files available at FGBBS.

If one has defined suitable macros `\fancycolumn` and `\fancytympan`, one can define:

```
\newcommand\templebox[1]{\sframe
  [1]\fancycolumn [2]\fancytympan
  [1]\fancycolumn [-]\-%
  {\separbox{3pt}{#1}}}
```

which should be read as: after 3pt wide empty space is set around `#1`, first add columns to the left and the right, second put a tympan on top of the result, and never put anything at the foot. Then typing this:

```
\templebox{hello there!} and \templebox{%
  \vbox{\hbox{b}\hbox{y}\hbox{e}}}
```

will yield:

hello there! and b y e

2.5 `\broadbox` *for Setting Line Wide Frames*

`\broadbox` can be useful to set frames that fill the line. Its use is best explained through an example. Suppose we want to set a paragraph of text in a line wide temple box. Then the lines will be filled by (from left to right): a column, empty space added by `\separbox`, text, empty space and a column. In other words: the whole line is available for setting text, except for the space needed by the columns and the empty space set by `\separbox`. The columns are 12pt each while `\separbox` adds 3pt wide empty space to the left and the right: that makes a total of 30pt. So we write: `\templebox{\broadbox{30pt}{\broadbox can be` … `\textit{dimension}.}}`, which yields a paragraph typeset like this. So `\broadbox {`*dimension*`}{`*stuff*`}` sets *stuff* in a `\vbox` which has width line width minus *dimension*.

2.6 Environment Versions

Some of the macros defined in hhflxbox are available as LaTeX environments. For example: instead of `\broadbox {30pt}{`*text to be boxed*`}` one could also use `\begin{broadboxed}{30pt}` *text to be boxed*`\end {broadboxed}`. Similarly one could use the environments `sepboxed`, `separboxed` and `sframed` instead of the macros `\sepbox`, `\separbox` and `\sframe`.

Actually I have to confess something: I lied to you about the typesetting of the section about `\broadbox`. I did it with:

```
\newenvironment{templeboxed}{%
  \begin{sframed}%
    [1]\fancycolumn [2]\fancytympan
    [1]\fancycolumn [-]\-
    \begin{separboxed}{3pt}
      \begin{broadboxed}{30pt}
}{%
      \end{broadboxed}%
    \end{separboxed}%
  \end{sframed}%
}

\begin{templeboxed}%
  \broadbox can be useful to set frames that
       :        :        :        :        :
  width line width minus dimension.
\end{templeboxed}
```

I hope you will forgive me my cheating. I mean, without using these environments, typesetting verbatim stuff is so troublesome …

3 Some Features of the hhcount Package

3.1 Simple Number Formatting

Let us start by summarizing the simple number formatting macros which are provided by hhcount:

example input	corresp. output	other example output
`\fctabdigit{2}`	2	29
`\fcolddigit{2}`	2	29
`\fcloweralpha{2}`	b	cc
`\fcbigalpha{2}`	B	CC
`\fcsmallalpha{2}`	B	CC
`\fclowerroman{2}`	ii	xxix
`\fcbigroman{2}`	II	XXIX
`\fcsmallroman{2}`	II	XXIX
`\fcbigromanlined{2}`	$\underline{\mathrm{II}}$	$\underline{\mathrm{XXIX}}$
`\fcsmallromanlined{2}`	$\underline{\mathrm{II}}$	$\underline{\mathrm{XXIX}}$
`\fcbigdice{2}`	⚁	⚅⚅⚅⚅⚄
`\fcsmalldice{2}`	⚁	⚅⚅⚅⚅⚄
`\fcbigscore{2}`	‖	𝍸𝍸𝍸𝍸𝍸||||
`\fcsmallscore{2}`	‖	𝍸𝍸𝍸𝍸𝍸||||
`\fcfnsymbol{2}`	†	

The next step in complexity are number formatting macros which give context-dependent output. This is implemented by using the *context switches* `\if@fcoldstyle` and `\if@fcsmall`, which are set by context switching macros like `\fcinheading` and `\fcintext`. We say that a context switching macro is active if it was the last one to affect the context switches.

example input	output when `\fcinheading` is active	output when `\fcintext` is active
`\fcdigit{14}`	14	14
`\fcalpha{14}`	N	N
`\fcroman{14}`	XIV	XIV
`\fcromanlined{14}`	XIV	XIV
`\fcdice{14}`	⚅⚅⚁	⚅⚅⚁
`\fcscore{14}`	𝍸𝍸||||	𝍸𝍸||||

By default `\fcinheading` is active; `\fcintext` is active when using `\ref` or `\pageref` (those two macros are redefined by hhcount).

3.2 How to Define Composite Counters

I will now try to give an impression of the way in which composite counters can be defined using hhcount. However, this is *not* a manual. After reading the following paragraphs you may be able to hack a composite counter together yourself, by imitating what is done below and experimenting with some small modifications of your own. If you want to be taught how to use hhcount efficiently and effectively, then you should read the manual.

Suppose we want to set up a three-level section numbering system for some sub-document in another document, for example the rules of a club embedded in some booklet about that club. The section numbers should be composed from the values of three (sub-)counters: `ruleschapter`, `rulessection` and `rulesparagraph`. Chapter numbers should be represented by capital alphabetic characters; elementary section and paragraph numbers by arabic digits. What should be done?

First we select a *series identifier* for our composite counter. Series identifiers are natural numbers which are assigned to composite counters. Each composite counter should be assigned a unique identifier. Because identifiers `1` to `8` and `12` are reserved for common purposes we select `9` for our rules section numbers.

Then we define a macro which expands to the series identifier:

```
\def\rulesseries{9}
```

Next we specify how the three sub-counters are to be combined:

```
\combinecounters\rulesseries{%
  \\{ruleschapter}%
  \\{rulessection}%
  \\{rulesparagraph}}
```

And finally we define how the counter is to be formatted:

```
\setcounterformat\rulesseries{#1-#2-#3}{%
  \fcorfinally
  % capitals for chapter numbers:
  \fcformat{#1}{\fcalpha}%
  % digits for section numbers:
  \fcformat{#2}{\fcdigit}%
  % a period to separate section and paragraph
  % numbers:
  \fcformat{#3}[.]%
  % digits for paragraph numbers:
               {\fcdigit}%
  \fcordespair}
```

If you want to understand the definition above, read the manual.

3.3 The Result

Now the composite counter can be accessed by the macros `\theruleschapter`, `\therulessection` and `\therulesparagraph`, which give results like: “A”, “A2” and “A2.3”. The macros `\stepcounter {ruleschapter}`, `\stepcounter{rulessection}` and `\stepcounter{rulesparagraph}` can be used to step the counter.

When `\fcinheading` is active, rules paragraph numbers will be set like “A2.3”, but when `\fcintext` is active, the same number will be set like “A2.3”.

More complex distinctions in representation of counters are possible. hhcount provides a set of macros which can be used in the last argument of `\setcounterformat`. Those macros enable definition of counters which are set like “A2.3” in headings and like “section A2, par. 3” in text etc. For details see the manual.

3.4 hhcount and `makeindex`

Composite section numbers like “A2.3” cannot be handled by the `makeindex` program. Besides, `makeindex` has problems with sorting alphabetic numbers since it cannot determine whether or not they are roman numbers. hhcount provides a way to get around these problems.

All composite numbers defined by hhcount constructs are internally represented by a sequence of natural numbers, separated by hyphens and embedded in a macro call. A typical example is `\fancycounter 9-1-2-3-!`. The first number represents the series identifier (`9` in the example), while the following numbers represent the values of the relevant sub-counters.

hhcount provides macros `\initfancycounters`, `\indextolabels` and `\indextopages`. The first redefines the section and page numbering systems to use hhcount’s composite counters. `\indextolabels` sort of redefines `\index` to use the redefined section numbers and strip the `\fancycounter` and the `-!` off the composite counter representation. `\indextopages` does the similar thing for page numbers. In both cases the result is a sequence of natural numbers, separated by hyphens, which can be handled perfectly well by `makeindex`. By embodying the appropriate definitions in your index style (`.ist`) file `makeindex` will undo the stripping after sorting the page or section numbers, so that your index entries will still be typeset as defined by means of hhcount macros. Thus section numbers like “A2.3” can be used for references in the index. Inserting equation, table and figure numbers in the index is just as easy. It is even possible to have different kind of composite numbers in the same index, for example page as well as section numbers, because the series identifiers are not stripped off so that it remains possible to determine the proper series and formatting of each composite number. For details see the manual.

3.5 Bugs and Deficiencies

Class files tend to make the TEX compiler show on your terminal which chapter of your book or report is being processed. Error messages often contain the page number. When using hhcount there is a chance that the chapter and page numbers shown on your terminal look weird: you will be shown the internal representation of your counter (`\fancycounter 9-1-0-0-!` for example). This is caused by an incorrect timing of macro expansion: in this case `\fancycounter` is expanded too late (that is to say, not at all).

Late expansion with hhcount is typically a problem with error and other messages: I would be highly surprised if someone discovers something like `\fancycounter 9-1-0-0-!` outside verbatim environments in a typeset document. However, when compiling your document you might run into early expansion, which causes severe errors. With the latest version of hhcount this problem does not seem to emerge in ‘usual’ contexts; however I am not sure.

Front matter, appendix and back matter peculiarities (with respect to page and section numbering) are not automatically supported by hhcount. Class files are too different in that respect. If hhcount is to be used to handle the section and page numbering in documents containing front matter and appendices, it would probably be best to incorporate hhcount in the class file, instead of loading it as an additional package.

3.6 Gamesters Page Numbers

The following redefines the `page` counter so that page numbers will be set as dice (I designed this for a gamesters society):

```
\def\fcpageseries{12}
\combinecounters\fcpageseries{\\{page}}
\setcounterformat\fcpageseries{#1}{%
  \fcorfinally
  \fcformat{#1}\fcdice
  \fcordespair}
```

I could not resist giving you this as an final example.

IV Maths in LaTeX: Part 6, Harder arrays

R. A. Bailey
Queen Mary and Westfield College,
University of London

1 Recall

This is the sixth and final part of a sequence of tutorials on typesetting Mathematics in LaTeX. The first five appeared in issues 4.4, 4.5, 5.1, 5.2 and 5.3 of *Baskerville*. The series includes some things which can be found in [3], but I am working in more things which, while straightforward and necessary for Mathematical work, are not in [3] or [4]. In this final tutorial I cover the harder parts of arrays, including aligned equations.

In case you missed the previous tutorials, I remind you that I expect you, the reader, to do some work. Every so often comes a group of exercises, which you are supposed to do. Use LaTeX to typeset everything in the exercise except sentences in italics, which are instructions. If you are not satisfied that you can do the exercise, then tell me. Either write to me at

School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS

with hard copy of your input and output, or email me at `r.a.bailey@qmw.ac.uk` with a copy of the smallest possible piece of LaTeX input file that contains your attempt at the answer. In either case I will include a solution in the following issue of *Baskerville*: you will remain anonymous if you wish.

This tutorial covers things that LaTeX is not really very good at. You may ask why I have not simply referred you to the packages `amsmath`, `array` and `delarray`. One reason is that every package has its price: it may give you the functionality that you want at the expense of changing something that you are quite happy with. The other is that you often are not allowed to include style packages when you submit an article to a journal or conference proceedings. It is your choice whether to use the inelegant solutions presented here or to cut out the relevant pieces of code from various style packages.

2 Answers

2.1 Boxed subarrays

In the panel session at the end of the UKTUG meeting on 'The New Maths for the New LaTeX' on 7 June, one of the participants asked how to create an array in which there is a box around a subarray, as in

$$\begin{array}{c|cccc|}
\multicolumn{1}{c}{1} & 2 & 3 & 4 & \multicolumn{1}{c}{5}\\
\cline{2-5}
2 & A & B & C & D\\
3 & B & A & D & C\\
4 & C & D & A & B\\
5 & D & C & B & A\\
\cline{2-5}
\end{array}$$

The answer is to use `\cline` for the horizontal sides of the box and to put `|` in the columns specifier to obtain the vertical sides of the box, overriding this with `\multicolumn{1}` where necessary. Thus the input for the preceding array begins

```
\begin{array}{c|cccc|}
\multicolumn{1}{c}{1} & 2 & 3 & 4 &
\multicolumn{1}{c}{5}\\
\cline{2-5}
2 & A & B & C & D\\
...
```

2.2 Angle brackets

Several people have asked me why I insist that `\langle` and `\rangle` should be used for angle brackets when they prefer the shape of < and >. At a group theory conference in July I saw a good, if unconscious, demonstration of why < and > should not be used. A line of displayed Maths on an overhead projector transparency was

$$G =< a,b,c > \times < a,c,e >^x$$

Look at the spacing. TEX knows that $=$ and $<$ are both relations, so it puts no space between them, but it does put some space between the relation $>$ and the binary operator $\times$. If you put this equation in the running text, you will find that the line may break between the $<$ and the a. If you really prefer the shapes of $<$ and $>$ to $\langle$ and $\rangle$ then you should make yourself macros such as

```
\newcommand{\llangle}{\mathopen{<}}
\newcommand{\rrangle}{\mathclose{>}}
```

Then the display becomes

$$G = \mathopen{<}a,b,c\mathclose{>} \times \mathopen{<}a,c,e\mathclose{>}^x$$

However, you cannot make these new angle brackets expand by preceding them with `\left` and `\right`.

12 Arrays of equations

12.1 Don't do it

Many of us write our lecture notes on the board as a series of equations, more or less aligned, and are tempted to write in print in the same fashion. Don't. For one thing, printed material needs the connecting words that you normally say at the board, such as 'and' or 'it follows that' or 'substituting for … '. For another, alignment suggests to the reader that the equations are somehow related, so it should not be used merely because two displayed equations come one after another with no intervening text: use two separate lines of displayed Maths instead, using `\[` and `\]`.

12.2 Parallel definitions

For two or more parallel or analogous definitions or results, use the `eqnarray*` environment. If a typical line is $A = B$ then type that line as `A & = & B` and put `\\` at the end of each line except the last. Extra space can be added after any `\\` just as with ordinary arrays. For example, the parallel definitions of $\cap$ and $+$

$$\begin{aligned} W \cap X &= \left\{v\in V:v \in W \mbox{ and } v\in X\right\}\\ W + X &= \left\{w+x:w\in W \mbox{ and } x\in X\right\}\end{aligned}$$

have input

```
\begin{eqnarray*}
W \cap X & =& \left\{v\in V:v \in W
\mbox{ and } v\in X\right\}\\
W + X   & = & \left\{w+x:w\in W
\mbox{ and } x\in X\right\}
\end{eqnarray*}
```

12.3 Chains of equalities

The `eqnarray*` environment is also useful for a chain of equalities or inequalities, such as

$$\begin{aligned}\sum_{i=1}^{q} x_{ij}(M-x_{ij}) &= M^2 - \sum_{i=1}^{q} x_{ij}^2\\ &\leq M^2 - \frac{M^2}{q}\\ &= \theta M^2.\end{aligned}$$

Here each line after the first begins with & followed by = or some other relation, followed by another &. Any line may conclude with `\qquad\mbox{...}` to give a short explanation, just as in a single line of displayed Maths.

$$\begin{aligned} n(Q_3 - Q_1)^4 \,\mathrm{Var}\, W \quad &= \quad \left[(M-Q_1)\left(\frac{1}{f_M} - \frac{1}{2f_{Q_3}}\right) + (Q_3 - M)\left(\frac{1}{f_M} - \frac{1}{2f_{Q_1}}\right)\right]^2 \\ & \qquad + \frac{1}{2}\left[\left(\frac{M-Q_1}{f_{Q_3}}\right)^2 + \left(\frac{Q_3-M}{f_{Q_1}}\right)^2\right] \end{aligned}$$

Figure 1. An overlong equation

12.4 Overlong displays

Sometimes what is conceptually a single line of displayed Maths, whether it is an equation or not, is simply too long to fit on one line. Then you can use `eqnarray*`, choosing where to split the line. If you split it at a binary operator, it is usual to put the binary operator after the split. In this case you must precede it with `\mbox{}` so that TeX knows that it is a binary operator. The two lines in Figure 1 are given by

```
...\right]^2\\ & & \mbox{} + \frac{1}{2} ...
```

To split an even longer line, you may want the second and succeeding lines to come partly underneath the first line. You can do this by enclosing the whole of the first line in `\lefteqn{ }`, thus fooling TeX into thinking that it has no width. Starting subsequent lines with `& &` gives that necessary bit of indentation. In this example

$$\begin{aligned} &\sum\left\{\sum\{f(B) : B\cap A = \emptyset\} : A \supseteq J\right\} = \\ &\qquad \sum\left\{\sum\{f(B) : A \supseteq J,\ A\cap B = \emptyset\} : B\cap J = \emptyset\right\} \end{aligned}$$

the lines begin

```
\lefteqn{\sum ...
 & & \sum ...
```

12.5 Numbered aligned equations

The environment `eqnarray` works just like `eqnarray*` except that each line is numbered, in the same sequence as `equation`s. If you want any line to be not numbered, just put `\nonumber` before the end of the line. If you want to refer somewhere else to the number, put a `\label` on the line in the usual way. Thus

$$\begin{aligned} \bar{F}(x_1,x_2) \quad &= \quad \int_0^\infty \exp(-\theta a_1 x_1^c - \theta a_2 x_2^c) \frac{\theta^{b-1}\lambda^b e^{-\theta\lambda}}{\Gamma(b)} d\theta \\ &= \quad \frac{\lambda^b}{(\lambda + a_1 x_1^c + a_2 x_2^c)^b} \qquad (1) \end{aligned}$$

is created with

```
\begin{eqnarray}
\bar{F}(x_1,x_2) & = & \int_0^\infty ...
{\rm d}\theta \nonumber \\
 & = & \frac{ ...
\end{eqnarray}
```

12.6 What is `eqnarray`*?*

The two environments `eqnarray` and `eqnarray*` differ only in the numbering of lines. Each creates a piece of displayed Maths containing a special sort of array. The array has only three columns. The first column is in `\displaystyle` and is right-aligned. The second is in `\textstyle` and is centred. The third is in `\displaystyle` and is left-aligned. The space between columns is controlled by `\arraycolsep` just as for ordinary arrays. The space between rows is (unless you put something after the `\\`) what you would get in an ordinary array by putting `\\[\jot]`.

12.7 Simultaneous equations

Simultaneous equations are often written with a vertical alignment for each variable and for the binary operators in between them, as well as for the equals sign, as the following example shows.

$$\begin{array}{rcrcrcrcrl} x_1 &-& x_2 &+& x_3 &-& x_4 &+& x_5 &= 1\\ 2x_1 &-& x_2 &+& 3x_3 && &+& 4x_5 &= 2\\ 3x_1 &-& 2x_2 &+& 2x_3 &+& x_4 &+& x_5 &= 1\\ x_1 && &+& x_3 &+& 2x_4 &+& x_5 &= 0 \end{array}$$

This is too many alignments for an `eqnarray*`, so an `array` has been used in displayed Maths, with every line ending with `\\[\jot]`. With a column for each variable and one for each binary operator, almost all pairs of adjacent columns should have the separation that TEX normally gives between an ordinary Maths symbol and a binary operator, which is `\medmuskip`. Unfortunately, you cannot set `\arraycolsep` to be equal to `0.5\medmuskip`: TEX will complain. So I have set `\arraycolsep` to zero: the command

```
    \setlength{\arraycolsep}{0pt}
```

has been placed before the array but within the displayed Maths, to limit its scope. Then the columns specifier `{*{4}{rc}r@{{}={}}r}` does the trick for the equals sign, which comes in every column: for the binary operators I have put `{}+{}` or `{}-{}` as least once in each column.

There are two other possibilities that could be used here. The `array` package allows you to put items in the columns specifier that will be incorporated in array entries before the boxes are made. So you could put the `{}` either side of each binary operator by putting it in the columns specifier once and for all. See [2, Section 5.3]. That would be useful if the binary operators in the array had differing widths. The second is to effectively set `\arraycolsep` equal to `0.5\medmuskip`. Now, `\medmuskip` is 4 mu plus some stretchability, and 1 mu is 1/18 of an em in the current font. So you can do

```
    \setlength{\arraycolsep}{0.11em}
```

and omit all the `{}`, so long as the current font does not change (by too much) between the issuing of that command and the setting of the entries in the array.

12.8 Which to use: `eqnarray` *or* `array`*?*

Regular readers will know that I am a big fan of LATEX. All the same, I think that the design of `eqnarray` is fundamentally flawed. It is not simply a method of aligning lines of displayed Maths, chiefly because it uses `\arraycolsep` to insert larger spaces than normal, but also because it changes between `\displaystyle` and `\textstyle` and because it is limited to three columns. For these last two reasons, it is also not a method of achieving a displayed array all of whose entries are in `\displaystyle` and whose rows are spread out, which would have been a useful environment.

So which should you use, `eqnarray` or `eqnarray*` or `array`? Each of them needs some work to give good results.

If you need a set of aligned equations carrying a single number then I recommend using `array` inside an `equation`. You will have to put in `\displaystyle` and `\jot` where necessary. If one or more lines must be individually numbered then there is nothing for it but to use `eqnarray`.

If an unnumbered set of aligned equations has only two alignment points you may be able to use `eqnarray*` if you are careful about the inter-column spacing. Thus if you put an `&` on only one side of an equals sign you must put a quad space on the other side. In the following display each line has the form

```
    ... \quad = & ... & \qquad ...
```

$$\begin{array}{rcll} g(x) &=& e^x & \text{for } x \in \mathbb{R},\\ h(y) &=& \ln y & \text{for } y > 0,\\ h'(y) &=& \frac{1}{y} & \text{for } y > 0. \end{array}$$

For a set with more alignment points, such as

$$\begin{array}{llll} f(1) = 1 & f(2) = 0 & f(3) = -2 & f(4) = 3\\ g(1) = 5 & g(2) = 7.5 & g(3) = 6 & g(4) = -4 \end{array}$$

or simultaneous equations, use `array` and be cunning with the columns specifier.

For parallel results, or for chains of (in)equalities, it would be good to have a form of `eqnarray` and `eqnarray*` in which the space on either side of the equals sign is what TeX normally puts between a relation and an ordinary Maths symbol, which is `\thickmuskip`. Now, `\thickmuskip` is 5 mu plus some stretchability, so we can use the same fudge that we used for simultaneous equations. It is no good changing `\arraycolsep` globally, because that would affect other `array`s. So you could make an environment to use in place of `eqnarray` such as the following.

```
\newenvironment{bettereqnarray}%
{\setlength{\arraycolsep}{0.14em}%
 \eqnarray}%
{\endeqnarray}
```

Compare the following display, made with `bettereqnarray` and `\nonumber`, with the previous form made with `eqnarray*`. Now that the spaces around the aligned = are correct, a second = can be placed on the same line.

$$\begin{array}{rcl}\sum_{i=1}^{q} x_{ij}(M-x_{ij}) & = & M^2 - \sum_{i=1}^{q} x_{ij}^2 \\ & \leq & M^2 - \frac{M^2}{q} = \theta M^2\end{array}$$

If you are uneasy about that fudge, set `\arraycolsep` to zero. Then put `& {}={} &` instead of `& = &` in the centre of the array.

There is a disadvantage common to both of these `bettereqnarray` environments: if you have any ordinary `array` within them then the value of `\arraycolsep` will almost certainly be wrong and you will have to reset it locally.

There are several better environments for aligned equations in the `amsmath` package, which is described in [1]. However, it does not seem to be possible to obtain these environments without the rest of the package, which you may not want: for example, it disables `\over`.

13 Exercises

Exercise 72 Möbius inversion gives:

$$B_\gamma = \sum_{\alpha\in\Gamma} z(\gamma,\alpha)S_\alpha, \tag{2}$$

$$S_\alpha = \sum_{\gamma\in\Gamma} m(\alpha,\gamma)B_\gamma. \tag{3}$$

Exercise 73 *Get the number cited here from the question above, by cross-reference.*

Now

$$\begin{array}{rcll} L_\alpha & = & X'S_\alpha X & \\ & = & \sum_{\gamma\in\Gamma} m(\alpha,\gamma)X'B_\gamma X & \text{by Equation (3)} \\ & = & \sum_{\gamma\in\Gamma} \frac{m(\alpha,\gamma)}{k_\gamma}C_\gamma. & \end{array}$$

Exercise 74

$$\begin{array}{rcl} \sum a_j b_k & = & a_1b_1 + a_1b_2 + a_1b_3 \\ & & + a_2b_1 + a_2b_2 + a_2b_3 \\ & & + a_3b_1 + a_2b_2 + a_3b_3. \end{array} \tag{4}$$

Exercise 75

$$\sum_{1\le j,k\le 3} a_j b_k \quad = \quad \begin{aligned} & a_1b_1+a_1b_2+a_1b_3 \\ & +a_2b_1+a_2b_2+a_2b_3 \\ & +a_3b_1+a_2b_2+a_3b_3. \end{aligned} \tag{5}$$

Exercise 76 Using the hint, we get

$$3(1-z)^2\sum_k \binom{1/2}{k}\left(\frac{8}{9}z\right)^k(1-z)^{2-k} =$$
$$3(1-z)^2\sum_k \binom{1/2}{k}\left(\frac{8}{9}\right)^k\sum_j \binom{k+j-3}{j} z^{j+k}$$

and now look at the coefficient of z^{3+l}.

Exercise 77 Solve the system of equations

$$\begin{aligned} 2x+\ \ y+5z&=4 \\ 3x-2y+2z&=2 \\ 5x-8y-4z&=1. \end{aligned}$$

Exercise 78 The dyad appears as

$$\mathbf{AB} \quad = \quad \begin{aligned} & A_xB_x\mathbf{ii}+A_xB_y\mathbf{ij}+A_xB_z\mathbf{ik} \\ & +A_yB_x\mathbf{ji}+A_yB_y\mathbf{jj}+A_yB_z\mathbf{jk} \\ & +A_zB_x\mathbf{ki}+A_zB_y\mathbf{kj}+A_zB_z\mathbf{kk} \end{aligned} \tag{6}$$

Exercise 79

$$\begin{array}{ll} f(x)=1 & f=(1,\ 1,\ 1,\ \ 1,\ \ 1) \\ g(x)=x & g=(1,\ 2,\ 3,\ \ 4,\ \ 5) \\ h(x)=x^2 & h=(1,\ 4,\ 9,\ 16,\ 25). \end{array}$$

14 Hand-crafting alignments

In this section I show a few tricks for difficult alignments. I show them because I know that I am not the only Mathematician who needs to produce these effects. I am not particularly proud of the methods I have used, because in each case I have had to either measure a length explicitly or use phantoms: I couldn't find a way of getting the right sizes automatically. So if any reader can write in with a better way of doing these things, I shall be delighted.

14.1 Horizontal braces in arrays

When I use an `array` to show a chain of equalities, I often use an under- or overbrace to indicate how terms are grouped. Here is an example.

$$\begin{array}{ccccccc} y & = & \multicolumn{3}{c}{\overbrace{\hphantom{\bar{y}(1,\ldots,1) + \frac{\mathrm{CS}(x,y)}{\mathrm{CS}(x,x)}x'}}^{f}} & + & r \\ & = & \bar{y}(1,\ldots,1) & + & \frac{\mathrm{CS}(x,y)}{\mathrm{CS}(x,x)}x' & + & r \\ & & \uparrow & & \uparrow & & \uparrow \\ & & \text{fit for null} & & \text{extra fit} & & \text{residual} \\ & & \text{model} & & \text{for straight} & & \\ & & & & \text{line} & & \\ & & & & \text{model} & & \end{array}$$

$$3(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu) = (x-1, y+2)\left[\begin{array}{rr} 4 & 1 \\ 1 & 1 \end{array}\right]\begin{pmatrix} x-1 \\ y+2 \end{pmatrix}$$

Figure 2. Matrices aligned on their top rows

The problem is that the brace needs to span several columns. So the brace needs to be put in with a `\multicolumn` command, so it cannot automatically be set to the correct width. I solve this by using `\hphantom` to obtain the width of the spanned columns. Here there is an entry

```
\multicolumn{3}{c}{f}
```

in the first row; and the corresponding entry in the second row is

```
\multicolumn{3}{c}{
\overbrace{\hphantom{\bar{y} (1,\ldots, 1)
\quad+\quad ... x’}}}
```

I have used the fact that the intercolumn space is one quad.

An alternative solution is to use the command `\downbracefill` as the final argument of the `\multicolumn` in the second row. However, you have to be outside Maths mode to use `\downbracefill`, and putting it inside an `mbox` is no good because that destroys the expandability. So you have to set the whole thing in a `tabular` environment rather than `array`, and then enclose every other entry in `$ $`, which is a pain. There is also an analogous command `\upbracefill`.

14.2 Paragraph boxes in arrays

In the preceding display I have also set some explanatory text in paragraph boxes whose width is determined by the width of a Mathematical expression in the same column. For each such column I make a new length and use `\settowidth` to make it as wide as the desired Mathematical expression: see [3, page 95] or [4, page 101]. Then I make a `p` column element of that width.

For example, in the first column I get the correct width with

```
\newlength{\gnat}
\settowidth{\gnat}{$\bar{y} (1,\ldots, 1)$}
```

and then use it as follows.

```
\multicolumn{1}{p{\gnat}}{\centering
fit for null model}
```

14.3 Top-aligned matrices

In the last article I showed Figure 2 as an example of something that is not easy to do in LaTeX. To achieve this I have made a macro `\topthing` which takes as its single argument any item that must be aligned with the top rows of the arrays. The biggest arrays here have two rows, so `\topthing` produces a two-rowed array whose second row is empty. If there were bigger arrays here I would have to have a macro for each smaller size of array. The empty second row of the array contains a phantom zero: this works because zero is the standard height, as are all the entries that occur in the second rows of arrays. I don’t know how to fudge this for arbitrary heights of entries. Finally, I remove the extra space that is usually put on each side of an array.

The `\topthing` macro is defined by

```
\newcommand{\topthing}[1]%
  {\begin{array}{@{}c@{}}
   #1\\\phantom{0}
   \end{array}}
```

Then the equation in Figure 2 is done with

```
\topthing{3(...   (x-1, y+2)}
\left[\begin{array}{rr}
4&1\\1&1
\end{array}
...
```

Top-aligned matrices can be done very simply if you have access to the package `delarray`. Get it. See [2, Section 5.3.6]. However, be warned that `delarray` inputs the package `array`, which makes a small difference to the way that `|` works in all `array`s.

14.4 Bordered matrices

Using `\left[` and so on to get the right size of fences around a matrix works fine for unbordered matrices. But how do you get a bordered matrix such as the following?

$$\begin{array}{cc} & \begin{array}{ccccc} \emptyset & \{1\} & \{2\} & \{1,2\} & \{1,2,3\} \end{array} \\ \begin{array}{c} \emptyset \\ \{1\} \\ \{2\} \\ \{1,2\} \\ \{1,2,3\} \end{array} & \begin{bmatrix} 1 & -1 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{7}$$

The command `\bordermatrix` given by plain TEX will not do, because it uses parentheses for the fences and does not let you choose how the columns are aligned.

The problem is to get the fences of the right size while keeping the alignment of the two borders with the body of the matrix. My solution is to set the whole thing as a 2×2 array with no space between the columns. The top left entry is empty. The top right entry is a one-rowed array containing the column labels. The bottom left entry is a one-columned array containing the row labels. The bottom right entry is the body of the matrix, including the fences.

If all entries have the same width and height, that's enough. However, if any column in the body of the matrix has a different width from the corresponding column in the top border, then the narrower of the two has to be expanded. Set a new length equal to the wider item and then put the smaller item in a `makebox` of that width.

In the matrix (7) it is clear that, in every column, it is the label that is the widest element. So I made a macro to set the first row of the body of the matrix to the width of the labels, as follows.

```
\newlength{\perch}
\newcommand{\fish}[2]%
{\settowidth{\perch}{$#1$}
 \makebox[\perch]{$#2$}}
```

For example, the first entry in the main body of the matrix is

```
\fish{\emptyset}{1}
```

Note that it suffices to make a single entry in each column of the body as wide as the column label.

(By the way, I can never decide whether the minus signs should be taken into account or not when centering the columns of such a matrix. Here I decided not to, and so I defined

```
\newcommand{\minone}{\llap{$-$}1}
```

and then used `\minone` for each -1.)

If the row heights in the body of the matrix do not match those in the array of row labels, as in the following matrix, they can be adjusted with `\vphantom`. If `X` is the tallest item in the row, simply put `\vphantom{X}` in a single entry of the corresponding row of the other array: it will not affect the horizontal spacing.

$$\begin{array}{cc} & \begin{array}{cccccc} 1 & \dots & r & r+1 & \dots & n \end{array} \\ \begin{array}{c} 1 \\ \vdots \\ r \\ r+1 \\ \vdots \\ n \end{array} & \left[\begin{array}{c|c} \Sigma_1 & 0 \\ \hline 0 & \Sigma_2 \end{array}\right] \end{array} \tag{8}$$

15 Exercises

Exercise 80

$$\begin{array}{ccccccc} \|f\|^2 & = & e_1^2u_1\cdot u_1 & + & e_2^2u_2\cdot u_2 & + & e_3^2u_3\cdot u_3\\ & & 18\bar{y}^2 & + & \texttt{ss1} & + & \texttt{ss2}\\ & & & & \multicolumn{3}{c}{\underbrace{104474 \quad + \quad 2284}_{106758}} \end{array}$$

Exercise 81

$$\begin{aligned} \operatorname{Cov}(U,V) &= (2,1)\begin{bmatrix} 1 & -1\\ -1 & 4 \end{bmatrix}\begin{pmatrix} -2\\ 1 \end{pmatrix}\\ &= (2,1)\begin{pmatrix} -3\\ 6 \end{pmatrix} = 0 \end{aligned}$$

Exercise 82 *Typeset the matrix (8).*

16 Acknowledgements

While writing these tutorials I have had to expand my own knowledge of how to typeset Mathematics in LaTeX to cover topics that I had never really bothered with properly. I should like to thank David Carlisle, Frank Mittelbach and Chris Rowley for patiently answering my questions, even when they did not wholly approve of what I was writing. Of course, any remaining mistakes are my own, as are the personal opinions expressed.

Thanks also to many of my colleagues, both at Goldsmiths' College and at Queen Mary and Westfield College, for badgering me to explain how to do these things. And thanks to those readers who have sent kind messages of appreciation while the tutorials have been appearing.

References

[1] American Mathematical Society: *AMS-LaTeX Version 1.2 User's Guide*, American Mathematical Society, Providence, Rhode Island, (1995).

[2] Goossens, M., Mittelbach, F. & Samarin, A.: *The LaTeX Companion*, Addison-Wesley, Reading, Mass., (1994).

[3] Lamport, L.: *LaTeX: A Document Preparation System*, first edition, Addison-Wesley, Reading, Mass., (1986).

[4] Lamport, L.: *LaTeX: A Document Preparation System*, second edition, Addison-Wesley, Reading, Mass., (1994).

V Somethin' there is about you, That I can't quite put my finger on

Chris Rowley

I have been stimulated (goaded?) into penning these thoughts by Peter Cameron's expression of concerns about the future utility of TEX to mathematicians. Many of these I share and, indeed, feel all the more pressing since I have set myself up as being responsible for them (and even, in some cases, I do directly bear that burden).

I shall first consider some particulars of speaking mathematics and, in particular, voicing division; then conclude with some more general thoughts about communicating with and via computers.

1 Talking divisively

Peter, in company with Don Knuth, wants to write his TEX "as close as possible to the way [he] pronounces" it (I added the "he" deliberately, see below).

Well, he may say 'x over y' for $\frac{x}{y}$ or even 'n-one-factorial n-two-factorial n-three-factorial, over, n-one + n-two + n-three' for $(n_1!\,n_2!\,n_3!)\,/\,(n_1+n_2+n_3)$ but I suspect that he would not say 'x overwithdelims . . . ' for $\binom{x}{y}$ (`$x \overwithdelims() y$`)—and how would he cope with this (selected "at random" from an AMS paper)?

$$\left| \frac{\hat{v}(s)-\hat{v}(t)}{\left|\widetilde{D}u\right|([t,s[)} - \frac{f\left(\hat{u}(t)+\frac{\widetilde{D}u}{\left|\widetilde{D}u\right|}(t)\left|\widetilde{D}u\right|([t,s[)\right)-f(\hat{u}(t))}{\left|\widetilde{D}u\right|([t,s[)} \right|$$

Of course, Peter may put only very simple fractions into his letters but an average physicist is not so fortunate (or so communicatively challenged?).

The problem with 'over' (in the sense of 'divided by') derives from the development over time of the notation for division; the use of built-up fractions is one of the more bizarre of the many usages that historical accidents have bequeathed us. If the good old 'division sign' (whose TEX name I have, I suspect, never known) had won out then life would have been much easier for coders and typesetters of mathematical documents, and possibly also for mathematicians. I also have a feeling that our generation were perhaps the first to adopt such a sloppy mode of mathematical speech—the phrase 'quotient of x by y' seems only a little old-fashioned to me.

In practice, mathematicians can often speak to each other in many abbreviated forms just like 'over'. For example, in context I could say to Peter:

For t greater than 0, -1 0 1 t is non-singular.

and I would expect *him* to very easily (sic) understand that the '-1, 0, 1, t' should be formatted as: $\left(\begin{smallmatrix}-1 & 0\\ 1 & t\end{smallmatrix}\right)$

My 'day job' (but I do it evenings too) involves me in spending more time than your average mathematician communicating notation over the telephone, so I have become quite adept at inventing methods of speaking math notation to a fairly wide variety of people—this often involves private codes which I would not expect anyone else to understand. This has little to do with talking or writing to either computers or general mathematical audiences, nor should it have, but for me it has illustrated very clearly the fact that any particular convention, however well it works in a restricted context, is not a good paradigm for a general way of making mathematical documents portable. I would, of course, place the method that Don and Peter use to talk mathematics to each other firmly in this category of "private codes".

Knuth's idea of writing mathematics as he and Peter would say it is both impractical (as is well illustrated by most of plain TEX), and irrelevant to the real problem of communicating mathematics (not just the notation, but the structures), both between people and computers and inter-computer.

I shall pick over just one other point in Peter's article before moving to more general matters: he complains that some of us are "obsessed with the need for all operators to be prefix".

All operators? No, at least not in the mathematical sense. All commands, yes: but that is a consequence of yet another accident of mathematical history—if you need general functions with an arbitrary, and possibly not fixed, number of arguments then the functions should be prefix, otherwise neither the computer (without a lot of extra work) nor the user (often, remember, this is not a mathematician) will be able to understand (in the sense of "parse") them. As Leslie Lamport observed, this convention does also have the advantage that the syntax of prefix commands often also makes it necessary to delimit the arguments; this syntactical nicety is essential for human readability but not much appreciated by most mathematicians (see my remark about "What are we summing?" below). If you do not understand the importance of writing commands in this very inefficient (from the computer's viewpoint) prefix form, try learning more than about a dozen Postscript commands with one to three arguments and then try and read a file that uses them.

2 Talking to computers

I shall now make some more general observations concerning maths, communication and computers.

Knuth's bestiary of mathematical symbols and constructions is no better or worse than any other: from the perspective of anyone from outside mathematics they are all both mysterious and infuriating.

I have in other *fora* argued strongly against too much formalism in the definition of a language in which computers can communicate mathematical notation. I now realise that, for general formatting purposes, rather more structure needs to be expressed in the mark-up than that which Knuth (and, hence, Lamport) thinks necessary. I say 'thinks' since recent reports from Florida suggest that Don is unrepentant in thinking that he got it right—for example it is, apparently, OK (TM) if the computer never knows what is being summed by a summation sign, just like it does not need to know when it is starting a quotient construction.

Computers need a lot more information than is provided by most schemes in order to format notation properly; this is very eloquently and dramatically illustrated by the work of T. V. Raman, who is getting the computer to answer back (audibly!) so that he knows what it cannot understand. I can assure Peter that Raman does *not* want his computer saying 'over' at random places but rather needs it to be able to efficiently distinguish and locate the beginning, end and "type" of all substructures. If it is ever sensible to use the phrase "how mathematics *should* be spoken" (when this is the only available means of communication) then the only relevant answers must surely be Raman's?

It has been apparent to me throughout my mathematical life that the world would be a better place if mathematicians were more respectful of an audience's intelligence (rather than of her knowledge of the bizarre conventions of the subject itself) when writing about it; it would be nice to think that training them to 'talk to computers' would make them more polite, but I doubt it.

I agree that it would be nice to talk informally about mathematics to my computer, and I expect that to happen long before it can understand (in any format) all the implicit conventions contained in the way I write notation that is to be understood only by other mathematicians. However, I am sure that it will never be so good to talk about the subject, and others, with a computer as it is to do so with Peter.

3 Talking to each other

The article also touches on many other areas which contain genuine problems (and I cannot see them becoming less numerous in the near future). Amongst them are some that the new standard LaTeX attempts to tackle, and others that we know must be tackled by LaTeX3. One of the former is the ability to substitute for fonts that you do not have.

Peter's thoughts are of great value to those of us who are actively influencing the future of TeX, both as a typesetting system and as a mathematical lingua franca. Thus I hope we shall see many more articles like this one—and not just from mathematicians, please! I don't promise to argue with them all in print but I shall certainly read them carefully and, who knows, they may goad someone else into explaining a different viewpoint on a controversial issue.

VI TUG95 at St. Petersburg Beach, Florida

Michel Goossens
Email: `m.goossens@cern.ch`

[*Editor's note:* Michel has kindly allowed us to print portions of his longer report on TUG 95, which will be distributed on the Internet.]

1 Monday July 24th

Monday morning saw the official opening of the Conference, with TUG95 Organizing Committee Chairperson Mimi Burbank welcoming all participants before passing the floor on to Michel Goossens, who thanked everybody for coming, and emphasized the different structure of this conference as compared to previous years, since this time we had more workshop oriented sessions (in the afternoons) and were opening up the area of presentations to the world of electronic publishing at large, and SGML/HTML, Adobe Acrobat, CD-ROMs, hypertext, etc. in particular. At that point the representatives of NTG said a few words about their third edition of the 4AllTEX CD-ROM, and Wietse Dol took this occasion to offer the first pressed CD to Donald Knuth who was a guest of honour at this 16th (2^{2^2}) TEX Annual Meeting.

The first talk was by Jiří Zlatuška, who showed how METAFONT and TEX can work together to typeset combinations of text and graphics. His new approach based on TEX's extended ligature mechanism, reduces the number of METAFONT passes to one, and also simplifies the TEX-METAFONT interface. This permits easier typesetting of text along curves and in particular allows one to generate beautiful institutional seals and logos in various forms and combinations starting from the same base elements. He noted that, although POSTSCRIPT is often the first choice for including graphics information in TEX documents, METAFONT often offers improved legibility of the logos and the letters at smaller sizes.

The next speaker, Richard Kinch, discussed his work on building reliable POSTSCRIPT Type1 and Truetype outlines for the Computer Modern fonts. He emphasized that several METAFONT primitives (stroked pens, overlapping ink) have no equivalent in these formats, since they support only non-overlapping Bézier curves. His program, `MetaFog`, handles most of the difficult problems associated with these conversions. He went on to discuss how `MetaFog` works, pointing out some of its drawbacks and making a plea for including hints inside the METAFONT sources. He finally compared his results to other outline instances of the CM fonts.

After refreshments Alan Hoenig gave another one of his almost "perfect" pedagogical talks, this time showing how one can use Adobe's Poetica font set, comprising 21 fonts in two families, exploiting the possibilities of the virtual font mechanism. Alan showed us how his macro package together with the font metrics generated by Alan Jeffrey's `fontinst` package are able to typeset a sonnet by Shakespeare as though it were written in the most beautiful calligraphy of a scribe.

One can never exaggerate the importance of documentation, and to teach first-year students the benefits of that approach an experiment was started at Texas' A&M University by teaching Knuth's `WEB` system, that fully exploits combining code and documentation in the same source. It was found that students who had to "mix" program description and code acquired increased problem solving skills. They tended to analyse their problems not merely in function of the programming language used, but in terms of the more general literate programming paradigm. Thanks to this increased awareness the students who took the `WEB` course also were more successful in grasping data structures and program development in general.

The morning session was concluded by Włodek Bzyl who showed how by extending Nowman Ramsey's generic literate tool `noweb` with a few stand-alone front-end programs, it became relatively easy to create a `TEX-WEB` system that is easy to understand for the novice user. The system is extensible by allowing customized styles and additional features. As an example he showed the literate source of `plain.tex`, the TEX source of Knuth's `plain` format, and proudly handed a printed copy to Donald Knuth, who browsed through it with great interest.

After lunch Sebastian Rahtz discussed some advanced exotic features of Timothy van Zandt's `PSTricks` package, that provides an easy interface between the PostScript and TEX languages (`PSTricks`'s operating principles were described at TUG94—see *TUGboat* 15(3), pp. 239–246). The talk was a shortened version of a three-hour presentation by Denis Girou, a well-known `PSTricks` guru, at the GUTenberg meeting on June 1st in Montpellier (France). By using the electronic version of the slides, Sebastian could easily zoom in on fractions of text and drawings. Fractals, complicated curves, like cycloids (this is the first time that curves of this type published in books on calculus will actually be drawn correctly, Don Knuth remarked when seeing the beautiful and precise graphs), and three dimensional multi-colour calendars were only of few of the graphics gems possible by this approach.

Jerry Marsden introduced his FASTTEX system, a library of standardized system-independent shortcuts for TEX commands. At present versions for Mac and Unix exist. This approach speeds up the keying of input material and greatly increases the accuracy of the final text. Abbreviations exist for most of the well know formats and extensions, like the AMS packages. Commands can easily be edited or added. This approach is especially interesting in an environment where many non-specialist typists have to work together so that consistency and ease of input are important considerations.

After tea Denis Kletzing showed how he uses his `multienumerate` package to handle complicated list structures. This environment handles narrow numbered list entries by bundling them in multiple columns. The drawback of the approach is that the user must specify the actual layout by typing the explicit position of the entry via different `\mitem`*`ijk`* commands, where *`ijk`* are column identifiers. With a little more TEX programming a lot of the positioning can probably be made automatic, but the approach shows that it is relatively straightforward to extend LATEX to cope with moderately simple but useful structures.

Jon Stenerson described the experience gained by using the style files he developed for use with *Scientific Word* and which he described last year at TUG94 (*TUGboat* 15(3), pp. 247–254). He thought that the basic ideas of his original approach were still all right but that most of them will have to be rewritten for streamlining and to better reflect his present thoughts on the subject.

At the end of the day the editors of the journals edited by the various TEX User Groups gave an overview of the problems encountered and ways of improving communication and mutual re-publication of worthwhile material. A few weeks before the conference, an electronic discussion list coordinated by Christina Thiele (TUG and former editor of *TTN*) and Gerard van Nes (NTG, editor of *MAPS*) had been set up and allowed the editors to exchange valuable information. Presentations were made by Sebastian Rahtz (editor of the UKTUG's *Baskerville*), Michel Goossens (for Jacques André, editor of GUTenberg's *Cahiers GUTenberg* and *Lettre GUTenberg*), Christina Thiele (previous editor of TUG's *TTN*), Luzia Dietsche (editor of DANTE's *Die TEXnische Komödie*), Gerard van Nes (editor of NTG's *MAPS*), Wietse Dol (editor of the EuroTEX95 Proceedings), Barbara Beeton (editor of *TUGboat*), and Włodek Bzyl (editor of GUST's magazine). From the various talks it became soon evident that most of the encountered problems were common to most user groups' publications, with in particular often an extreme dependence on a single individual so that when s/he is unavailable the whole production process suffers. It was clear that a production team of a few individuals is the only way out of this situation to ensure that issues can be produced at more or less regular intervals. Other themes were the difficulty of finding authors, and volunteers to proofread, correct and edit the articles. All in all, editing a journal is a non-trivial task and involves the dedication and hard work of a lot of individuals. As Jacques André summed it all up: "An editor is like an organists at Sunday mass: if the music is good, no one hears it; if it is bad, everyone cries." At the end of the session Gerard distributed *MAPS* awards to Christina Thiele for her many years as *TTN* editor and to Mimi Burbank for her hard work organizing the TUG95 Conference.

2 Tuesday July 25th

As Donald Knuth only rarely attends TUG conferences these days it was a real pleasure to have him with us at TUG95 and we gave him the floor for the first part of the morning. After answering the "usual" first question—"when is volume four of *The Art of Computer Programming* coming out?" (it will be published over several years with about 200 pages coming out every six months or so)—he talked at length about TEX and how he would do it in basically the same way if he were to start over today. A more detailed account of Knuth's presentation will be published elseware based on notes taken by Christina Thiele.

After the break John Hobby, the author of MetaPost, gave a live demonstration of his program which is now used by Knuth himself to prepare the graphical material of his books. MetaPost implements a picture-drawing language based on Knuth's METAFONT, but it outputs POSTSCRIPT commands instead. It moreover gives access to all features of POSTSCRIPT and allows easy inclusion of text and graphics (for more details on MetaPost see the user guide available on CTAN or an introductory article in the EuroTEX92 Proceedings, pp. 21–36, Prague, Sep. 1992).

The last set of presentations were about "future systems". First Robin Fairbairns gave an introduction to Unicode. He reviewed the winding road from 6-bit propriety codes for encoding information on computers, to 7-bit ASCII-like codes, then to 8-bit EBCDIC and ISO standards 8859-xx, specific language encodings, the 16-bit East-Asian JIS, GB, KS and Big Five codes, to Unicode and 32-bit ISO 10646 (for a detailed discussion of questions of encoding and multi-linguism see, e.g., *Cahiers GUTenberg* 20, pp.1–53). It is to be noted that Unicode is the internal encoding used by Haralambous and Plaice's Ω program, a 16-bit extension of TEX (*TUGboat* 15(3), pp. 320–324 and 344–352). Jiří Zlatuška then brought us up-to-date on the ε-TEX project, whose first version had just been distributed to developers. Peter Breitenlohner complemented Jiří's talk with some more technical details on ε-TEX. It all sounds like an interesting development.

During the first part of the afternoon Wietse Dol showed how easy it is to use the "plug-and-play" 4allTEX CD-system for PC's. An interactive live presentation showed the ease with which the system can be installed from the CD, and also how applications can be run.

3 Wednesday July 26th

Mark Swift was the first speaker of the day and in his talk *Modularity in LATEX* he explained that LATEX should be built in a highly modular way. In particular an abstraction of functional modules not mapped onto filenames would be an important point, as shown by recent discussions on the `LATEX3` discussion list where the "forced" uniqueness of filenames in the basic LATEX distribution, like `article.cls`, was questioned. The speaker proposed some possible extensions to TEX and discussed his work on the `frankenstein` package, which adds certain kinds of modularity to LATEX.

Bart Wage of Elsevier Science in Amsterdam gave an interesting description of how journals are handled from source copy to printed/electronic document. Text is converted into SGML, figures are kept in various formats (e.g., TIFF, JPEG) and LATEX sources are also translated into SGML using the Elsevier DTD. LATEX allows for easy typesetting, but it has no formal DTD, making extensive tagging somewhat difficult, while SGML allows for a formal DTD, where explicit tagging of all document elements with respect to that DTD is possible. Formal specifications exist for math, tables, bibliographic material, etc.. Elsevier see SGML as an ideal exchange format between different source representations. All documents are translated into SGML and stored in the "Warehouse", which forms the common repository for the various further uses of the documents. This is extremely important for electronic documents, where re-use and structure-awareness are of prime importance. Bart emphasized that a journal is not just a collection of articles, but a real web of cross-links to related topics and references, and the future of publishing lies in the optimization of these facilities for all potential users.

Pierre Mackay then read a paper on *Modern Catalan Typographical Conventions* written by Gabriel Valiente Feruglio, who could not attend. It was an interesting journey in search of typographic rules for scientific Catalan texts. The author complained that no normative typographical conventions existed for his language and then went on to propose a set based on his study of several reference texts. Finally he introduced a set of possible TEX definitions implementing these rules.

After coffee Petr Sojka gave one of the best technical papers of TUG95—Petr got the "Knuth" prize of the Conference for discussing something important that Knuth did not think about when developing TEX—a follow-up of his detailed paper describing the problems of hyphenation with TEX presented at in Gdansk in September 1994 (EuroTEX94 Proceedings pp. 59–68). This time he discussed the problems of hyphenating long compound words, which occur very often in German, Dutch, and the Slavic languages, since in these languages the constituent parts are not signalled by a hyphen or other fill character. This makes it often difficult, if not impossible, to hyphenate words correctly. Therefore Petr suggests extensions to the hyphenation algorithms of TEX to successfully treat such cases and he discussed in a generic way which basic functionalities would be needed. Perhaps something to be implemented in (one of) the "successor(s)-to-TEX", he commented (and Don seemed to agree).

The last talk of the morning was by Sebastian Rahtz, who discussed the translation of LATEX sources into SGML. His presentation was a complement to Bart Wage's earlier that morning. After working on the conversion problem for several months, Sebastian came to the conclusion that the only foolproof way is to use TEX itself to output SGML, a solution implemented by ICPC in Dublin. He is actually using an intermediate approach, (pioneered by Sebastian and myself at CERN), which translates most LATEX commands into SGML by redefinition of macros, and then extracts the text from the dvi file. This system copes with almost all LATEX commands (including math).

The afternoon had presentations by convenors of various working groups. First Norman Walsh presented the work of the `tds` (TEX Directory Structure) working group. He explained the rationale of the choices that have been made, emphasizing that one of the basic constraints had been ISO-9660, which (only) allows for directories eight deep and limited to "8+3" case-insensitive names for files (for DOS users this will sound as a blessing, I am sure). Since not all

TEX engines support an optimized recursive directory search major attention was paid to propose an efficient structure that minimizes losses of efficiency while searching for package and font-related files by TEX. It was emphasized that a production run-time directory structure, like `tds` is different in nature to an archive, like CTAN, and that the two cannot be married easily.

Tomas Rokicki and Michael Sofka then discussed the work in the dvi-standard committee, especially the standardization of the various `\special` commands, that had been discussed by an extremely active interested group of implementors, meeting over several of “working breakfasts and lunches”. I think that real progress was made in this area where a normative syntax had been awaited for too long. I am very grateful for the enthusiasm shown by these people, and am convinced that we shall see their work bear fruit in the near future (Tom told he will be working actively on his program `dvips` over the next few months, so that we can be sure, knowing Tom’s reputation, that many of the hyper- and other goodies discussed during the conference, will become part of this, and other popular dvi-drivers).

After the refreshment break T. V. Raman gave a practical demonstration of his ASTER system, which allows one to “hear” LATEX sources, including mathematical formulas, being read out. His system uses a speech synthesizer via an augmented emacs editor running with Common Lisp, and is able to analyse, decode and then transcribe into audible form well-structured LATEX documents. This last remark is extremely important, since, as already pointed out by Sebastian Rahtz in his talk earlier that day, due to the various (ambiguous) ways that mathematics can be coded in TEX, there exists no automatic way to parse such TEX source into something usable more generally, such as SGML or audible sound. This was the loudest plea yet for using well-structured markup.

Just preceding the Conference dinner the editors of the various TEX-related magazines had a second meeting to discuss ways of improving communication. It was decided to write a short overview of the experiences of each team for *TTN*, to provide cross-references to each other’s publications on the user groups’ respective WWW pages, where tables of contents of the magazines will be posted (in fact, GUTenberg already decided to make freely available on the Internet via WWW, all articles—initially in POSTSCRIPT form only—of the *Cahiers* and *Lettre GUTenberg*). It was also proposed that all non-English publications try and provide an abstract in both the local language and in English, so that these abstracts can be published in *TUGboat* (or elsewhere). Editors were also asked to signal potential articles that might be interesting for translation into English and publication into *TUGboat* (of course, editors can translate articles from *TUGboat* into their national language also!). Presently, CSTUG, GUST, GUTenberg, NTG, TUG, and UKTUG are working on a TEXART CD-ROM that will make all publications of those user groups available on this electronic medium (and on the Web). It was also agreed that each author should be asked permission to reprint her/his article(s) in this way. During the meeting Gerard had the pleasure to offer the third *MAPS* award to Barbara Beeton, the most senior and long-standing editor present for her 16 years of efforts to make *TUGboat* an example of the typographic quality that can be achieved with TEX.

At the end of the conference dinner in the evening a set of books written (and dedicated) by Donald Knuth were put up for sale to the highest bidder. Even Lamport’s LATEX book got a nice inscription by the hand of Knuth. After I let the the volumes of the “Art of Computer Programming” escape I concentrated on the “big one”, namely the five volumes of *Computers & Typesetting*, who became mine for the nice little round sum of $700. Adorned with the dedication of Donald Knuth this will remain one of the treasures of my personal library! The sales were an outstanding success and about $1800 were collected towards the funding of the EuroTEX bus which will take participants from Russia and central Europe to the EuroTEX95 Conference in Papendaal (the Netherlands). Many thanks to Addison-Wesley, who donated the books.

4 Thursday July 27th

In his presentation T. V. Raman gave an overview of ASTER—an Audio System For Technical Readings—the system he demonstrated the day before. ASTER renders LATEX documents in an audible way, so that visually impaired persons can “listen” to their contents. Raman emphasized the importance of the use of clear generic markup for the input source document to ease the extraction of structural logical information that can be easily translated into an internal representation. ASTER then renders information by applying rendering rules written in AFL—Audio Formatting Language—to the internal representation. In a sense AFL is to audio formatting what POSTSCRIPT is to visual formatting (although AFL is by far not as complex). As a conclusion he emphasized that one needs a semantic-oriented DTD to produce a high-quality audio document. Since no such completely general DTD can be constructed, one has to use the facilities provided by LATEX and its hyperTEXt extensions.

Mark Doyle next reviewed the purpose and history of the Los Alamos preprint server, which is one of the first (and more successful) document servers on the Web. In fact it started in the area of (theoretical) High Energy Physics and took place in close collaboration with CERN (where WWW was "invented"). Today several tens of thousands of preprints are available online and over 20000 users visit the server each day. Although at present most documents are only available as (mainly TEX) source and standard POSTSCRIPT, they are now producing PDF versions that include cross-references to other documents on the Web using the `hyperTEX` tool and PDF hypertext links. In this way cross-references to other documents can be easily instantiated.

During the next half-hour I gave an introduction to Nikos Drakos' tool `latex2html` and showed how by simple customization the visual quality of the output HTML files can be substantially improved. I went on the show how the `latex2html` system also allows for interconnecting separate documents. I ended with a few examples of HTML3 output generated by an ad-hoc program developed at CERN and viewed with the HTML3-capable `arena` browser.

After the break Sebastian Rahtz showed how with his `hypertex` package (sharing some code with the `HyperTEX` package discussed earlier by Mark Doyle) it is easy to turn LATEX documents into hyper-documents. Their "hyper" contents can be enriched by adding supplementary information about LATEX's cross-references via `\special` commands. These are picked up and translated into PDF's `pdfmark` commands by Mark Doyle's "hypertext" `dvihps` program, an extension to Tom Rokicki's `dvips` program. Tom stated that these extensions will end up, in one form or another, in the forthcoming upgrade of standard `dvips`.

The afternoon started by a second presentation of the *4AllTEX* system, and, as always, there was great admiration amongst all those present for the ease with which it is possible to "plug and play", i.e., start to setup and run the system without much ado. It became all the more evident that such a CD-ROM for Unix is a real need, and a recurring proposal for the next great thing that TUG should come up with (and we are surely thinking about a way get this done).

During the next hour I gave an introduction to SGML using HTML as an example of a DTD, and showed that it is not difficult to understand the structure and syntax of a DTD, and from there to figure out the various possible document elements, their attributes and the entities that are available to the user. Work on other DTD's for mathematics and tables were briefly mentioned, as were a few tools for authoring and checking SGML documents. I came away with the feeling that at the end of my talk most of the audience had a more balanced view about what SGML is, and what it is not. I therefore hope that my presentation will also contribute to eliminate most of the artificial animosity between the SGML and TEX worlds. As Sebastian, myself and a few of the other speakers tried to show, SGML is about structure, and TEX about typesetting, and the two tools are therefore complementary and both useful.

Chris Hamlin, in the last scheduled talk of the day, described the production work at the American Physical Society, and, as expected, it is similar in content, form, and structure to what we had heard by other speakers (at this conference Elsevier, or at other conferences Springer, OUP, etc.), namely a mixture of TEX and other word-processor inputs are accepted by the production team. The proportion of TEX sources varies wildly between publications (between almost nothing in the chemical journals to well over 50–60% in some of the physical journals). Various house styles are available, and at present ways are being investigated to translate the inputs into SGML to take full advantage of electronic publishing tools.

The last part of the afternoon was for the TUG Business meeting.

5 Friday July 28th 1995

Already Friday. It seemed as though the Conference only just started, but the bags at the sides of the room and the now-empty vendor's room made us realize that we were only here for another few hours.

The morning started with a paper submitted by Jonathan Fine, but read by Alan Hoenig in Jonathan's absence. The title was *New perspectives in TEX Macros* and dealt with a possible way of combining the advantages of both SGML and TEX. His TEX macro package SIMSIM takes SGML and style files as input and generates pages formatted with TEX as output. SIMSIM comes with an SGML parser and the style files are used to link TEX actions to SGML events. The SIMSIM system also offers a programming environment for writing TEX macros and style files. At present issues of performance were not addressed directly but on sample documents the speed was comparable to that of LATEX. All by all an interesting idea, and I look forward to see Jonathan's finished product soon.

Sergey Lesenko then told us about his *t1part* tool that partially embeds Type1 POSTSCRIPT font files into a document. The principle is to include the POSTSCRIPT code for only those characters that are actually referenced. This can result in huge savings in size if one uses only a few characters from many fonts (the procedure is based on the same model that includes only the necessary Type3 bitmaps for characters built with METAFONT). Tom Rokicki and Sergey have been working together over the last few months and this facility will be built into the "next" version of Tom's `dvips`. I mentioned that Basil Malushev has a somewhat similar utility `fload`, that uses the publicly available ghostscript program to make a map of all referenced fonts and then includes only the characters needed. Basil's approach can be used for any kind of POSTSCRIPT file, so that it is complementary to Sergey's which is well integrated with TEX and needs no supplementary external program. During the discussion there were some interesting remarks on copyright issues connected with including Type1 fonts inside documents. It was felt that, although partial font loading would make pirating fonts less effective, it does not mean that all font vendors would agree to let us include their fonts in this way in files distributed electronically (on CD-ROMs or the Internet). To be continued ...

A more technical talk, on METAFONT this time, was Jeremy Gibbons' presentation *Dotted and dashed lines in METAFONT*. He showed that drawing evenly spaced dotted and dashed lines in METAFONT is a non-trivial task, and he proposed several solutions to make it possible. He introduced the notions *evenly spaced in time* as opposed to *equally spaced in space* and went on to show that they are far from identical, since points can move at different "speeds" in space as they progress along a path evenly in time. Using recursive adaptive refinement techniques he showed how one can solve the problem in METAFONT. His procedure can be extended to allow for dashed, or alternating dashes and dots. As recursive techniques have the unwanted feature that they can overflow the stack, Jeremy also proposed a solution based on an iterative non-adaptive technique that, although perhaps less elegant and automatic, does the job almost equally well. At the end of his talk he showed several attempts at drawing an attractive muskrat, the logo of the *Mississippi Muskrats* jazz band he used to play in.

The last scheduled talk was by Robin Fairbairns. After explaining the principles of the POSTSCRIPT multiple master Type1 font format, Robin showed how a crude first system of using these fonts with TEX was set up. All font instances are expressed in function of weights with respect to the master designs. These weights are calculated by the POSTSCRIPT interpreter from the design parameters via the POSTSCRIPT operator ConvertDesignVector. One has to use version 3.x of the ghostscript program to extract the weights, which were then used to generate the Adobe Font Metrics (AFM) instances from the AFM files for the master designs. Then Rokicki's afm2tfm program was run to generate corresponding tfm files needed by TEX, while a header file was also defined to allow dvips to actually specify the font instances from the weightvector. This set up was used to typeset the last issue of UKTUG's magazine *Baskerville* in Minion, one of Adobe's Multiple Master fonts.

The morning ended with the "Closing Ceremony" and the announcements of the TUG95 prize winners.

Christina Thiele, as vendor and public-relations liaison thanked the various companies who had vendor booths or otherwise contributed to the TUG95 conference, in particular Addison-Wesley for the books they donated (and that were put on for sale on the Wednesday evening for the EuroTEX95 bus, and still a few left for another sales idea we are playing with for spicing the TUG96 bursary. Stay tuned to *TUGboat* or *TTN*!).

Mimi Burbank, as Chair of the Organizing Committee, thanked all the people at SCRI who helped her financially, organizationally, by providing PC's or a printer. She also thanked the extremely efficient hotel staff for their never-ending devotion to a job well-done.

Sebastian Rahtz, the chair of the Programming Committee, then announced the prize winners for TUG95. Just before coffee that morning all participants were asked to write down an ordered list of the four papers they liked most, and on the basis of that list it was Raman who was selected as best presentation, best paper, and most important contribution to the TEX world (and humanity, one person wrote). The Knuth prize, for the paper discussing something that he "forgot" in his TEX program, went to Petr Sojka for his work on hyphenation. Other prizes went to Richard Kinch for MetaFog (who put his prize copy of Textures up for sale, so that the EuroTEX bus got again somewhat more money to take home), Alan Hoenig for his marvellous Poetica work, Jeremy Gibbons for his entertaining and erudite explanation of METAFONT, and Sergey Lesenko and Tom Rokicki for partial font downloading, and work on dvi standards. Many thanks are due to the fine TEX vendors Blue Sky Research, Y&Y, PCTEX, and Richard Kinch, who generously donated copies of their products for the prizes.

Of course we did not forget our friends from NTG, without which this conference would not have been the same. Their *4AllTEX* CD-ROM was one of the highlights at this conference (they sold about 40 copies, and the remaining 60 were taken to the TUG office for selling them to the North American TUG community). Their "presence", good humour, the organization of the book auction, the coordination of the TEX-ED initiative and the hundred or so photos they took made them a memorable and unforgettable part of this meeting. Therefore a signed copy of the METAFONT book was given to Wietse Dol. In a gesture underlining their dedication to TEX and TUG Wietse then offered TUG the two golden (original) CD-masters of the third edition of *4AllTEX* that just appeared. I had the pleasure to receive them in name of TUG and I promised that they would be framed and displayed in a prominent place in the TUG office in San Francisco. The gifts were concluded with the UK TEX Users Group and TUG presented 2 bottles of wine, and 2 boxes of chocolates, to Don Knuth, maintaining the '2' theme begun by NTG's 2 CDs at the start of the conference.

Finally it was my duty to formally end the TUG95 meeting, and after thanking Knuth for his presence, which made this 2^{2^2}th meeting even more special, I re-iterated the thanks to all vendors, SCRI and the hotel staff, for their display of (southern) American hospitality. Then I invited all participants to the next (17th) TUG annual meeting in 1996 in Dubna (Russia, 150 kms north of Moscow, on the Volga River), where from July 28th to August 1st TUG96 will be hosted by the Joint Institute of Nuclear Research.

During the afternoon Alan Hoenig gave a partical introduction to the use of virtual fonts. He showed how they can be used to create new characters as various combinations of glyphs and rules. He described how Alan Jeffrey's `fontinst` package allows one to easily install POSTSCRIPT font families. In his usual pedagogical approach Alan made it all sound as though it is extremely simple and straightforward, and all fifty participants to this last "event" of the conference came away with the feeling there were ready to generate some virtual fonts themselves.

VII Malcolm’s Gleanings

Malcolm Clark

1 Indefatigable

There hardly seems a month goes by when I do not pick up a journal or magazine with an article by one of LaTeX’s most indefatigable proselytisers, Allan Reese. This month it was *Axis* (a rather specialised journal for ‘Academic Computing and Information Systems’). In a rather fetchingly titled article, ‘¡Hoja! Herr Böll, Ça va?’ he champions the use of LaTeX as a suitable medium for email in order to accommodate all those annoying foreign languages which have accents. Well, we’ve been here before, but it is still good to see Allan determinedly and relentlessly spreading the word. An accolade to that man.

2 Synchronicity

In the last *Baskerville* I extolled GUTenberg and their *Cahiers*, noting that their next volume would be on character coding. This volume arrived on my desk a week or so ago. It began with a brief *In memoriam* to Cathy Booth. Cathy attended several of the early GUTenberg meetings, and had many friends in the french-speaking TeX community. It was a very thoughtful and touching gesture to dedicate the volume to her. A similar event was perpetuated at the TUG conference, where Sebastian Rahtz presented a prize for best paper to T. V. Raman in her name. Personally, I can think of few people more worthy than T. V. to receive any sort of prize – and on top of that, he’s a really nice person.

3 MIME

Our out going and retiring chair, Chris Rowley, points out that `x-dvi` is already a MIME type. In theory this should mean that you can include a `dvi` file with an email message and the recipient will be able to read it. I’d be reluctant to contemplate this outside the arcane world of Unix. In any case, what I had in mind was to have a simple viewer which exploited Adobe’s Multiple Master fonts, rather than (say) Computer Modern. It would be interesting to know if this MIME type is used in real life, or whether it merely represents good intentions. In passing, it was serendipitous that the last *Baskerville* was printed in Adobe Minion, one of the two main Multiple Master fonts. But at least it almost demonstrated that LaTeX and Multiple Masters can co-exist.

4 What I did on my summer hols

I’ve stopped going to international TeX conferences, and I no longer do much TeX or LaTeX teaching (the former on moral and ethical grounds: those and the fact that you have to tell your audience to suspend disbelief for the first morning: “you do *what*?”), but I was inveigled into giving a course on LaTeX 2ε, the only true LaTeX, in Malaysia. Imagine bringing latex to Malaysia. It is an ironic footnote that the rubber plantations there are being cut down, or the timber being used for wood, rather than for the harvesting of latex. This course turned out to be one of the most enjoyable that I have taught. Besides the pleasure of teaching a really nice bunch of people (and by and large, TeXies tend to be in that category) this was virtually the first time I have had access to teaching facilities where I didn’t have to waste half the first day teaching people how to use the operating system and its interaction with TeX. I had two teaching rooms: one with pcTeX for Windows, the other with *Textures* 1.7.5. Since I had to do a little of the software installation (pcTeX, for example, doesn’t come with all the bits and pieces I might have expected), I can also vouch for the relative robustness and ease of the installations. pcTeX is only a couple of years behind *Textures – i.e.*, where *Textures* was maybe two or three years ago. They should almost catch up within a year or so. Naturally there is a catch: both these implementations cost money. Since I now have copies of both these implementations I’ll say more about their limitations and ease of use at some future point, but the issue here is that some of the pain and difficulty of learning LaTeX was removed ‘at a a stroke’ by the availability of a responsive and easy to use point and click version of TeX.

Naturally, since the predominant language of Malaysia, Melayu (or Bahasa Malay) is not English, they wanted

LaTeX to reflect the difference. Fortunately it is a Latin script and they had the very great sense to spell it phonetically, with no accents. One difficulty is that many words form their plural by doubling the word, separated by a hyphen: *e.g.*, it is as if the plural of sheep were sheep-sheep. Fine, but how will TeX hyphenate this? The answer is that it won't. By default, TeX does not add hyphens to words which already contain a hyphen. That's one of the reasons that the Cork encoding has a 'link-hyphen'. With eight bit encoding, therefore, it can be done. The other problem, of language localisation (turning 'Appendix' to 'Lampiran' or 'Contents' to 'Kandungan') was wonderfully easy thanks to Babel. In there you will find `bahasa.dtx`. This is a language conversion which claims to handle Bahasa, which is closer to an Indonesian 'dialect' of Malay. I gained a lot of kudos from a quick hack of what will become `melayu.dtx`. There really is lots to be said, not just for LaTeX 2ε, but the comprehensive installation which accompanies it.

5 When shall I rebuild LaTeX 2ε?

Since LaTeX 2ε pops up every six months now, there is an interesting question developing: when should you download it from the archive and rebuild your installation?. There is no point taking it as soon as it is released, since you know that the first patch will be released about 26 hours after the announcement. Experience seems to indicate that patch level 3 represents LaTeX 2ε approaching its asymptote. Having said that, I couldn't build that patch level on my Mac (despite everything I said earlier . . .).

6 ε-TeX escapes

ε-TeX, the first stage in a new improved (but not called) TeX has been sort of released. DOS versions were distributed at the TUG conference in St Petersburg Beach. There are also unconfirmed rumours that a VMS Alpha version is lurking somewhere south of the Thames. But what is it all for? Can anyone point me to a description of the real and tangible benefits which will accrue from its adoption? I know how the committee structure works, and who does the work, thanks to Phil Taylor's *TUGboat* notes, but I don't yet know what real world typesetting problems it will solve, at a stroke.

7 The empty vessel

. . . giveth a greater sound than the full barrel[12]. Goodbye special directors: having been implicated in the creation of TUG's special directors[13], it's good to see that TUG has now dispensed with them. The special directors were the chairs or presidents (in the case of those whose boots grew) of the 'older' European users groups. When they were adopted onto the TUG Board it was a useful way to try to redress the very strong US-ocentric focus which TUG had. Now I hear that the Americans are complaining that TUG is a European dominated organisation which just happens to have its headquarters in San Francisco (this week).

Barbara Beeton's hold on *TUGboat* has been reduced. It will have been obvious that one of TUG's major problems, and one of the reasons its membership has fallen to unsustainable levels, has been the persistent non-appearance of *TUGboat*. Some of the responsibility for this has to laid at the feet of Barbara (one of the nicest people I could ever hope to meet, whose knowledge of TeX is legendary, whose standards are Knuth-like, and whose skills in the art of delegation are limited). The production of *TUGboat* will now be based in Florida. Editorial matters remain within Barbara's control. Incoming President Michel Goossens has placed the production of *TUGboat* as a top priority for TUG.

As predicted, Florida was hot and sweaty. Some say it was the sweatiest on record, but I guess they didn't go partying in the flesh pots of Santa Barbara. Mind you, not much partying went on at St Petersburg Beach – and no bowling.[14]

Just as this column was the first to reveal that Michel Goossens (of the LaTeX *Companion* fame) was to be the next TUG President, it can also reveal that Sebastian Rahtz is the new Secretary, Mimi Jett Treasurer and the startling Judy Johnson is Vice-President (she startled me!). With the Rahtz–Goossens dynamic duo at the helm we can expect some interesting developments at TUG. At the very least, *TUGboat* should start to appear regularly.

You may wonder how Sebastian manages to accomplish so much. A carefully inspection of his name is revealing:

[12]John Lyly, 1579, Euphues, the Anatomy of Wit.
[13]one of the allegations of my influence on TUG which is based in fact.
[14]Clearly Malcolm's sources did not tell him about the 'Three Dancing TeXxies', Tom Rokicki, Petr Sojka and Michael Cohen — Editor