TUGboat

Volume 18, Number 1 / March 1997

TEX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published quarterly by the TEX Users Group, P. O. Box 1239, Three Rivers, CA 93271-1239, U.S.A..

1997 dues for individual members are as follows:

- Ordinary members: $60.
- Students: $40.

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in the annual election. A membership form is provided on page 62.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: $70 a year, including air mail delivery.

Periodical-class postage paid at San Francisco, CA, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TEX Users Group, P. O. Box 1239, Three Rivers, CA 93271-1239, U.S.A..

Institutional Membership

Institutional Membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group. For further information, contact the TUG office.

TUGboat © Copyright 1997, TEX Users Group

Permission is granted to make and distribute verbatim copies of this publication or of individual items from this publication provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this publication or of individual items from this publication under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this publication or of individual items from this publication into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the TEX Users Group instead of in the original English.

Some individual authors may wish to retain traditional copyright rights to their own articles. Such articles can be identified by the presence of a copyright notice thereon.

Printed in U.S.A.

Board of Directors

Donald Knuth, *Grand Wizard of TEX-arcana*†
Michel Goossens, *President**
Yannis Haralambous*, *Vice President*
Mimi Jett*, *Treasurer*
Sebastian Rahtz*, *Secretary*
Barbara Beeton
Karl Berry
Donna Burnette
Robin Fairbairns
George Greenwade
Alan Hoenig
Judy Johnson
Patricia Monohon
Arthur Ogawa
Jon Radel
Petr Sojka
Jíři Zlatuška
Raymond Goucher, *Founding Executive Director*†
Hermann Zapf, *Wizard of Fonts*†

* *member of executive committee*
† *honorary*

Addresses

All correspondence, payments, parcels, etc.
TEX Users Group
P. O. Box 1239
Three Rivers, CA 93271-1239 USA

To obtain an address suitable for courier or delivery services, call or email the TUG office.

Telephone

+1 209 561-0112

Fax

+1 209 561-4584

Electronic Mail

(Internet)
General correspondence:
`TUG@mail.tug.org`
Submissions to *TUGboat*:
`TUGboat@ams.org`

World Wide Web

`http://www.tug.org/`
`http://www.tug.org/TUGboat/`

TEX is a trademark of the American Mathematical Society.

When a type design is good, it is not because each individual letter of the alphabet is perfect in form, but because there is a feeling of harmony and unbroken rhythm that runs through the whole design, each letter kin to every other and to all.

Frederic W. Goudy
Type Design: A homily (1961)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 18, NUMBER 1 • MARCH 1997
THREE RIVERS • CALIFORNIA • U.S.A.

TUGboat

During 1997, the communications of the TEX Users Group will be published in four issues. One issue, still to be determined, will contain the Proceedings of the 1997 TUG Annual Meeting.

TUGboat is distributed as a benefit of membership to all members.

Submissions to *TUGboat* are reviewed by volunteers and checked by the Editor before publication. However, the authors are still assumed to be the experts. Questions regarding content or accuracy should therefore be directed to the authors, with an information copy to the Editor.

Submitting Items for Publication

The next regular issue will be Vol. 18, No. 2. The deadline for technical items has passed; reports and similar items are due by May 16. Mailing is scheduled for June. Deadlines for other future issues are listed in the Calendar, page 56.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, Barbara Beeton (see address on p. 3).

Contributions in electronic form are encouraged, via electronic mail, on diskette, or made available for the Editor to retrieve by anonymous FTP; contributions in the form of camera copy are also accepted. The *TUGboat* "style files", for use with either `plain` TEX or LATEX, are available "on all good archives". For authors who have no network FTP access, they will be sent on request; please specify which is preferred. Write or call the TUG office, or send e-mail to `TUGboat@ams.org`.

This is also the preferred address for submitting contributions via electronic mail.

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to `TUGboat@ams.org` or to the Editor, Barbara Beeton (see address on p. 3).

TUGboat Advertising and Mailing Lists

For information about advertising rates, publication schedules or the purchase of TUG mailing lists, write or call the TUG office.

TUGboat Editorial Board

Barbara Beeton, *Editor*
Mimi Burbank, *Production Manager*
Victor Eijkhout, *Associate Editor, Macros*
Alan Hoenig, *Associate Editor, Fonts*
Christina Thiele, *Associate Editor, Topics in the Humanities*

Production Team:
Barbara Beeton, Mimi Burbank (Manager), Robin Fairbairns, Michel Goossens, Sebastian Rahtz, Christina Thiele

See page 3 for addresses.

Other TUG Publications

TUG publishes the series *TEXniques*, in which have appeared reference materials and user manuals for macro packages and TEX-related software, as well as the Proceedings of the 1987 and 1988 Annual Meetings. Other publications on TEXnical subjects also appear from time to time.

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the TEX community in general. Provision can be made for including macro packages or software in computer-readable form. If you have any such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee in care of the TUG office.

Trademarks

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which appear in this issue may not be complete.

MS/DOS is a trademark of MicroSoft Corporation
METAFONT is a trademark of Addison-Wesley Inc.
PC TEX is a registered trademark of Personal TEX, Inc.
PostScript is a trademark of Adobe Systems, Inc.
TEX and AMS-TEX are trademarks of the American Mathematical Society.
Textures is a trademark of Blue Sky Research.
UNIX is a registered trademark of X/Open Co. Ltd.

Notice regarding 1997 TEX Users Group election

[This notice was sent by email on 5 April 1997 to all TUG members for whom TUG has email addresses. If anyone reading this notice here did not receive a copy by email, please let TUG know by sending a message to `tug@mail.tug.org` including your correct, preferred email address. We intend to use email more frequently in the future to communicate with TUG members, and need your help to keep the records up to date.]

The 1997 election for the TEX Users Group Board of Directors has now ended. Since there were fewer nominated candidates than vacant positions, in accordance with the Election Procedures, the candidates who did submit nomination papers are declared elected. No ballots will be circulated. n The open positions, as announced, were President and six members of the Board of Directors.

The following individuals will be taking positions on the Board of Directors for 4-year terms ending in 2001:

Donna Burnette
Mimi Jett
Patricia Monohon
Arthur Ogawa
Petr Sojka

Their statements of candidacy are published in this issue of *TUGboat*.

Four Board members whose terms were due to end in 1998 had agreed that they would be willing for their terms to be extended to 1999 to accommodate a change in the election schedule from annually to every two years (see the election announcement, *TUGboat* 17#3, p.240). Had there been a ballot, their names would have been listed there for confirmation by the membership. However, in the absence of a ballot, it is proposed that, in accordance with the Bylaws (Article VII, Section 10), the incoming President confirm the extension of their terms. These Board members are:

Barbara Beeton
Karl Berry
Judy Johnson
Jiří Zlatuška

There is no candidate for TUG President. This poses a problem. Although the Bylaws and Election Procedures provide for the filling of vacant Board positions by appointment, no provision exists for the absence of a candidate for President. There is not sufficient time left before the annual meeting in July to solicit additional candidates and circulate ballots.

In order to ensure that the necessary decisions are arrived at in a democratic manner, and that the incoming Board has some say in the matters which will affect them most directly, Michel Goossens, the outgoing President, has appointed the new members to the Board for a temporary period effective immediately, to expire when their regular term begins.

It is proposed that individuals wishing to present themselves as candidates for President do so in the following manner, with the election be held during the annual TUG business meeting which will take place during the TUG annual meeting in San Francisco (July 28–August 1).

Any TUG member in good standing who will be present at the annual business meeting may submit a valid nomination form and supporting statement (see the announcement in *TUGboat* 17#3 or on TUG's Web site `http://www.tug.org/nomination-form.ps` no later than Monday, July 28 (the first day of the meeting), and be prepared to present his/her program at the business meeting, (currently scheduled for Thursday, July 31).

All TUG members are reminded that except for certain transactions such as personnel matters, which legally are privileged, Board meetings are open to members as observers. The Board meeting is scheduled to be held on Saturday and Sunday, July 26-27, at a location to be determined; details will be posted at the main meeting site and on the TUG Web pages when available. Candidates for President are encouraged to attend.

Nomination forms and supporting documentation should be submitted to the Elections Committee by one of two methods: (1) by July 15, sent to the address below rather than to the address given in the *TUGboat* announcement, or (2) July 26–28, brought to the meeting and delivered in person to one of the undersigned committee members. If sending forms by mail or fax, please confirm this to the Committee at the email address below; electronic copies of candidates' statements may also be sent to this address any time before the meeting.

In order for this business to be transacted legally, at least 50 TUG members must be present to form a quorum (Bylaws, Article III, Section 6). It is therefore very important for members to attend this meeting, in order to help define the future of our organization.

For the Elections Committee
Sebastian Rahtz, Barbara Beeton

Address for submission of nominations for TUG President:

Barbara Beeton
TUG Elections Committee
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
Fax: +1 401 331-3842
Email: `tug-election@mail.tug.org`

General Delivery

From the President

Michel Goossens

The 1997 Elections are now behind us and it is a real pleasure to congratulate Donna Burnette, Mimi Jett, Patricia Monohon, Arthur Ogawa, and Petr Sojka, who will serve as TUG Board members until Summer 2001. I extend my best wishes to Barbara Beeton, Karl Berry, Judy Johnson, and Jiří Zlatuška, whose terms were extended until Summer 1999. Together they will form the New Board, who will formally take over from the current Board at our Annual Meeting in the Summer. However, to prepare for the changeover and to have a democratic decision base which is as large as possible, I have appointed the incoming members to the Board with immediate effect. I am sure that together we shall be better able to reach the necessary urgent decisions which need to be taken to make TUG serve its membership and the TEX users better.

On the other hand, we had no candidate for TUG President, and as explained in the previous article, we are proposing an *ad hoc* procedure to extend the nomination period and elect a candidate at our Annual Meeting in July. I sincerely hope that one or more good candidates will step forward to take up the Office of TUG President. TUG really needs somebody who can motivate her or his collaborators in an efficient, enthusiastic yet professional way, and can act as a stimulus to guarantee that the Board acts as a united Team to address the problems at hand.

It is with great regret that I have to announce that Mimi Burbank, who has served on the TUG Board for many years, most recently as TUG's Treasurer, resigned from the Board for personal reasons. As well, she will not be assuming the function of Office Manager as proposed in the previous issue of *TUGboat*. I am pleased that Mimi will continue to work on *TUGboat*, since as *TUGboat* Production Manager she is an extremely valuable member of the *TUGboat* Production Team.

To cope with ongoing financial business I have appointed Mimi Jett as interim Treasurer until the new Executive Committee is elected in July at TUG'97. At the same time Art Ogawa became TUG's acting "Office Manager" and Art accepted gracefully to take over a major part of the Office duties. A more definite proposal about the new form the TUG Office will take will be announced soon.

By the time you read this it will be early May, and I apologise for the delay with which you receive this issue. We are doing everything possible to get the next issue of *TUGboat* to you in June. If everything works out as planned, we hope to be able to offer you a (very useful) surprise with that issue.

And last but not least, do not forget TUG'97; i.e., *your* Annual Meeting in July in San Francisco. We look forward to a huge turn-out. We count on your presence!

◇ Michel Goossens
CERN, Geneva, Switzerland
`goossens@cern.ch`

Editorial Comments

Barbara Beeton

Update to PSTricks

Users of PSTricks will know that for some years there has been a 'beta' release of the package, with significant extra facilities. The author, Timothy van Zandt, has had little time to work on the package recently, but he has now agreed to an interim release of PSTricks (PSTricks 97) which merges in all the beta material with the main package, and corrects known bugs. This release has been coordinated by Denis Girou and Sebastian Rahtz.

This release has been installed on CTAN in `graphics/pstricks`; the old release has been moved to `oldstuff/pstricks`.

At the same time, there is a new mailing list for users of PSTricks, moderated by Denis Girou. If you want to subscribe, send a message saying

subscribe

to

`pstricks-request@mail.tug.org`

Quote out of context — *Colophon*

"Prices for complete sets of *Colophon* have been advertised at upward of $10,000, and single specimens of the rarest ... have been listed at several thousand dollars." A rare journal? Some special editions? No, the word omitted from the previous sentence is "species", and the *Colophon* it describes is a slow-moving, wingless stag beetle, whose thirteen species are restricted to the mountaintops of the former Cape Province of South Africa. These rare beetles were given provincial legal protection in 1992, but an attempt to list them on the worldwide roster of endangered species failed, and commercial insect dealers continue to collect and sell them. South African authorities plan to petition again for listing of *Colophon* in a category that offers limited international protection.

Webster's *New International Dictionary*, second edition, defines colophon as

1. Finishing touch. *Obs.*
2. An inscription placed at the end of a book or manuscript, often containing facts relative to its production, as the scribe's, illuminator's, or printer's name, the place and date of publication, etc.; as, from title page to *colophon*.
3. *Print.* An emblem, usually a devices assumed by the publishing house, placed on the title page, shelfback, etc.

Colophon was also an ancient city of Ionia.

However, until I read the book *An Inordinate Fondness for Beetles*[1], I had never become acquainted with the six-legged variety. This book is quite wonderful, with stunning photographs and excellent, informative text. I picked it up to look at the pictures, and became engrossed in the story it had to tell. Recommended, even for the timid.

◇ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
`bnb@ams.org`

Erratum: Amsterdam, 13 March 1996 — Knuth meets NTG members, *TUGboat* 17 (no. 4), pp. 342–355

Barbara Beeton

As often happens, when taking shortcuts, something may go wrong. In this case it appears on page 351 of *TUGboat* 17(4), where in the right column you will find the words *MetaPost* instead of the math symbol $\mp$. This of course is due to a redefinition of `\mp` without paying attention. We apologize.

[1] Arthur V. Evans, Director of the Insect Zoo, Natural History Museum of Los Angeles County, and Charles L. Bellamy, Senior Curator, Coleoptera, Transvaal Museum, Pretoria, South Africa; photography by Lisa Charles Watson, technical advisor Henry Galiano, and illustrations by Patricia Wynne. A Peter N. Nevraumont Book, Henry Holt and Company, New York, 1996. ISBN 3751-9

Dreamboat

$\mathcal{NTS}$ & ε-TEX: a status report

Philip Taylor

The $\mathcal{NTS}$ project project was created at the instigation of Joachim Lammarsch and under the aegis of DANTE e.V. at a DANTE meeting in Hamburg in 1992. The idea of the project was — and is — to perpetuate all that is best in TEX while being free of the constraints which Knuth has placed on the evolution of TEX itself. For that reason, the project was called the $\mathcal{NTS}$ project — $\mathcal{NTS}$ being short for New Typesetting System — to emphasise that we are not violating Knuth's wishes that TEX remain entirely his responsibility: indeed, we have received Knuth's blessing to pursue the ideas of $\mathcal{NTS}$ and ε-TEX, and he went so far as to make some suggestions which, as he put it, "he might have incorporated himself were it not for the fact that he had decided to freeze the evolution of TEX". We are, of course, endeavouring to incorporate those suggestions, although we have not yet fully succeeded.

Before getting into too much technical detail, I should like to explain why the group is called the "$\mathcal{NTS}$ group" yet the project on which this report is centered is called "ε-TEX". During the group's early deliberations, we took advice from Joachim Schrod who proposed that rather than attempt to modify TEX-in-Web, we first re-implement TEX using a modern rapid-prototyping language such as LISP, CLOS or PROLOG. Joachim's philosophy was that TEX-in-Web is essentially too poorly structured to allow for radical change, and a more highly structured implementation, with a well-specified interface between modules, was essential if we were to make more than cosmetic changes to TEX-the-system.

The group agreed that this philosophy was sound, but realised that the effort needed to re-implement TEX from scratch was greater than could be achieved with voluntary labour: if success was to be achieved within a sensible timescale, it was essential that the group be in a position to employ a programmer or small team of programmers who could undertake the re-implementation. As the group had no budget of its own, and as it was financially dependent on the goodwill of DANTE, it was reluctantly agreed that this re-implemention would have to be put on ice until such time as the group had adequate financial resources: in the meantime, the group agreed to pursue a rather more conservative approach, evolutionary rather than revolutionary, and to put all of their efforts into this less radical project.

I am delighted to be able to report that, whilst this paper was being written, the future of the $\mathcal{NTS}$ project became very much more certain: during the February meeting of DANTE in München, the Board and members of DANTE agreed to donate DM 30 000 to the $\mathcal{NTS}$ project, which we believe should enable us to employ a programmer for one year, full time, in the Czech Republic, working on the $\mathcal{NTS}$ project under the direct supervision of Professor Jiří Zlatuška (Dean of the Faculty of Informatics at Masaryk University in Brno).

Despite this very significant change in the status of the $\mathcal{NTS}$ project, the group have not abandoned or even reduced their efforts concerning ε-TEX, and we have agreed a tentative specification for ε-TEX V2 which we will be discussing and endeavouring to implement during the coming months. It is our intention to release ε-TEX V2 one year after the release of ε-TEX V1: that is to say, during the first two weeks of November 1997.

To clarify the distinction, "$\mathcal{NTS}$" refers to a future project to completely re-implement TEX in a modular fashion using a modern rapid-prototyping language, and to investigate each module in turn to see how it might be improved — examples of possible modules include the user interface, the user programming language, the typesetting engine itself, and/or components of the typesetting engine such as the line- and page-breaking algorithms.

"ε-TEX", on the other hand, refers to work-in-progress which seeks to develop the TEX-in-Web implementation in a way which is, and will remain, 100% compatible with TEX itself. The "ε" of "ε-TEX" can be thought of as representing "evolution", "extension", "enhancement" (and perhaps even "European"!); the varepsilon of its typeset form emphasises that it is just a small evolutionary step from TEX itself, not a fundamental paradigm shift.

So, work on $\mathcal{NTS}$ will be started in the near future, but ε-TEX is now! After approximately three years in development and testing, we released ε-TEX at the Hamburg meeting of DANTE towards the end of 1996. A few cosmetic changes were made just subsequent to its official release, but despite the very large number of accesses made on the ε-TEX reference Web site, I am delighted to say that we have received no reports of bugs! This might, of course, simply mean that no-one is actually using ε-TEX, but I hope it means that ε-TEX is, as far as possible, bug free.

What are the features of ε-TEX? First and foremost, ε-TEX is 100% compatible with TEX: if you want to try ε-TEX, you can use it to process all of your legacy documents — it will produce results identical to TEX, right down to compatibility at the level of the `TRIP` test. When you are confident that ε-TEX can do everything that TEX does, you can try its extensions: there are approximately 30 of these, each intended to make the task of programming in TEX somewhat simpler; even when these extensions are enabled — that is, when ε-TEX is operating in so-called "extended mode" — ε-TEX will still process all existing TEX documents in a manner identical to TEX, provided of course that the document does not inadvertently refer to one of the new ε-TEX primitives. And finally, if you need to, you can use ε-TEX in "enhanced" mode, to get access to new features which are simply too radical to be 100% compatible with TEX: in the first release, there is only one enhancement, "TEX--XET", based on the earlier bi-directional typesetting system called "TEX-XET" by Don Knuth and Pierre MacKay. Unlike TEX-XET, TEX--XET requires no variant of DVI, no new pseudo device driver, and performs all of its operations internally.

To summarise, ε-TEX has three modes of operation. These are "compatibilty mode", in which ε-TEX behaves identically to TEX including full `TRIP` compatibility; "extended mode", in which ε-TEX provides approximately 30 new primitives but within which it continues to remain completely compatible with TEX, although it can no longer pass the `TRIP` test because of the presence of new primitives; and "enhanced mode", within which strict compatibility is sacrificed to allow additional features which are in some way fundamentally incompatible with TEX. The choice between compatibility or extended mode is made at the time a format is being generated: once the format is dumped, it contains within it a flag which indicates in which mode it is to operate, and it cannot be used in the alternative mode. The choice between extended and enhanced modes, on the other hand, is made at run time: even if an enhancement is enabled during format-creation, that enhancement will be automatically disabled before the format is dumped, and thus when the format is used the enhancement will initially be disabled. Enhanced mode can be entered only from extended mode, not from compatibility mode.

The extensions of ε-TEX V1

The extensions of the first version of ε-TEX can be classified into six distinct groups, plus a seventh which is accessible only in enhanced mode: these groups are

- Generalisation of the "`\mark`" concept;
- Additional control over expansion;
- Provision for re-scanning previously-read text;
- Environmental enquiries;
- Additional debugging features;
- Miscellaneous primitives;

and

- Bi-directional typesetting: the TeX--XeT primitives.

A brief specification of the elements which compose these seven groups is given below, followed by a provisional specification of our plans for ε-TeX V2.

In ε-TeX V1, we took the initial step towards eliminating from ε-TeX the fixed limits which beset the current TeX language: we generalised the concept of a "`\mark`" into an array "`\marks`", with 256 elements in the first release. All of the related "`mark`" variables were similarly generalised into arrays of 256 elements.

We added a further twenty-three new primitives:

"`\protected`" is a prefix which can be used during macro definition. A macro defined as "`\protected`" will not expand during an "`\edef`", a "`\write`" or any similar operation in which expansion normally occurs. It will, however, expand if it reaches TeX's "stomach" (for example, during the actual typesetting process).

"`\detokenize`" is intended to be used just before a brace-delimited token list (a "general text", in the terminology of *The TeXbook*). It expands to yield a sequence of character tokens of category code 10 or 12 corresponding to the characters which compose the tokens of the unexpanded token list.

"`\unexpanded`" is also intended to be used just before a brace-delimited token list but it expands to yield the actual tokens of the token list. If TeX is performing an "`\edef`" or a "`\write`" or similar operation, no further expansion takes place, but if these tokens reach TeX's "stomach" (for example, during the typesetting process) then they will expand normally.

"`\readline`" is analogous to "`\read`", but treats each character as if it were currently of category code 10 or 12; the text read can then be scanned and re-scanned in different catcode environments using another new primitive "`\scantokens`".

"`\scantokens`" is intended to be followed by a brace-delimited token list, and decomposes the token list into the corresponding sequence of characters as if the token list were written to a file without expansion. It then applies TeX's "`\input`" mechanism to this sequence of characters, re-tokenising them according to the current catcode environment.

"`\currentgrouplevel`" is an internal read-only integer which returns the current group level at the point of call: in other words, it returns the current depth of group nesting.

"`\currentgrouptype`" is an internal read-only integer which returns the type of the innermost group as an integer in the range 0 to 16. These numbers can be converted to text strings using definitions provided in the ε-TeX macro library.

"`\ifcsname`" has the same syntax as TeX's "`\csname`" but is a Boolean "`\if`", yielding "`true`" if and only if the putative control sequence is already known to ε-TeX.

"`\ifdefined`" is analogous to "`\ifcsname`" but takes as parameter a control sequence or active character: it yields "`true`" if and only if the control sequence or active character is already known to ε-TeX.

"`\lastnodetype`" is an internal read-only integer which returns the type of the last node on the current list as an integer in the range −1 to 15 (the upper bound may be increased in a future version). These numbers can be converted to text strings using definitions provided in the ε-TeX macro library.

"`\eTeXversion`" is an internal read-only integer which contains the integral component of the combined version/revision number.

"`\eTeXrevision`" is a primitive which expands to yield a sequence of character tokens of category code 12 representing the fractional component of the combined version/revision number.

"`\showtokens`" is intended to be followed by a brace-delimited token list, and provides a simple way of displaying the contents of a particular element of the "`\marks`" family of arrays; it has many other potential applications.

"`\interactionmode`" provides read/write access to the current interaction mode. Assigning a numeric value sets the associated mode, while the current mode may be ascertained by interrogating its value. Symbolic definitions of these values are provided in the ε-TeX macro library.

"`\showgroups`" causes ε-TeX to display the level and type of all active groups from the point at which it was called.

"`\tracingassigns`", if set to a positive nonzero value, causes ε-TeX to display the value of registers both before and after assignment. Standard TeX displays only the new value, not the old.

"`\tracinggroups`" is an aid to debugging runaway-group problems. If it is set positive and non-zero, ε-TeX traces entry and exit to every group.

"`\tracingifs`" is an aid to debugging the expansion of conditionals. If it is set to a positive non-zero value, ε-TeX traces the flow of control through conditional statements.

"`\tracingscantokens`", when set to a positive non-zero value, causes ε-TeX to display an open-parenthesis and space whenever "`\scantokens`" is invoked; the matching close-parenthesis will be displayed when the scan is complete.

"`\tracingcommands`" is defined in TeX to provide different degrees of verbosity as its value is increased from zero to two; in ε-TeX, we provide greater verbosity and detail when it is set to values greater than two.

"`\everyeof`" is one of Knuth's "possibly good ideas", listed at the end of `tex82.bug`. It is analogous to the other "`\every...`" primitives, and takes as parameter a brace-delimited token list the tokens of which are inserted when the end of a file is reached.

"`\middle`" is analogous to TeX's "`\left`" and "`\right`" primitives: it specifies that the following delimiter is to serve both as a right and left delimiter. It will be set with spacing appropriate to a right delimiter with respect to the preceding atom, and with spacing appropriate to a left delimiter with respect to the succeeding atom.

"`\unless`" allows the sense of all Boolean conditionals to be inverted. For example, "`\unless \ifeof`" yields "`true`" if and only if end-of-file has not yet been reached.

The final class of primitives is composed of those that are accessible only when ε-TeX is operating in enhanced mode. An ε-TeX program enters enhanced mode when it assigns a positive non-zero value to one of the "enhanced state" variables, of which the only one in the first release of ε-TeX is "`\TeXXeTstate`". Once this has been assigned a positive non-zero value, five other primitives become accessible: "`\beginL`", "`\beginR`", "`\endL`", "`\endR`" and "`\predisplaydirection`". If these primitives are used when ε-TeX is not operating in enhanced mode, an error is reported.

"`\TeXXeTstate`" is an internal read/write integer which, when set to a positive non-zero value, enables use of the TeX--XeT primitives;

"`\beginL`" indicates the start of a region which should be set left-to-right;

"`\endL`" indicates the end of a region which should be set left-to-right;

"`\beginR`" indicates the start of a region which should be set right-to-left;

"`\endR`" indicates the end of a region which should be set right-to-left;

"`\predisplaydirection`" is an internal read/write integer which is initialised by ε-TeX to indicate the direction of the last partial paragraph before a maths display; it is used to control the placement of elements such as equation numbers, and it can be directly set to alter this placement when appropriate.

The ε-TeX macro library

Although the majority of the effort in developing ε-TeX has been put into adding new functionality to the TeX language, we also spent a little time in developing a small macro library to accompany ε-TeX. In essence, this is a wrapper for the Plain format source, augmenting the existing definitions where appropriate to support the new primitives.

We also took the opportunity to add two features which we thought might be appreciated by the TeX community at large: we delayed the reading of patterns and exceptions until a natural language handling mechanism had been defined (so now patterns and exceptions are associated with a particular language, rather than being defined in limbo), and we added support for ε-TeX library files so that one can now load modules as an alternative to loading complete files.

The language handling mechanism is not predicated on the use of any particular natural language support system: it can be linked to Babel, for example, or to any other natural language handling system. In addition, hooks are provided so that user code can be threaded before and after language selection and in this way we hope to provide a sufficiently flexible language handling environment to support the needs of the majority of national TeX user groups. This is not to say that we do not foresee a rôle for Omega: on the contrary, it is clear from the Omega discussion list that there is a very real need for a system of Omega's complexity. However, we believe that for typesetting environments in which access is needed only to the major Western languages, ε-TeX will prove sufficient.

Availability

When ε-TeX was announced in late 1996, two reference implementations were available: Peter Breitenlohner's PubliC ε-TeX, which is a Turbo Pascal implementation for the IBM PC family running MS/DOS and Christian Spieler's VMS ε-TeX, a Pascal implementation for the VAX/VMS and

AXP/VMS family of machines. Since ε-TEX's release, it was first ported to the Commodore Amiga and then to the Windows 95/NT environments. During late February 1997, Bernd Raichle announced that his port of ε-TEX to Web2C Version 7 was also available for release; Bernd, who works at Stuttgart University, tells me that Eberhard Mattes is planning to release a combined ε-TEX/ML-TeX for the IBM PC family in the near future.

That summarises the present state of ε-TEX: in the remainder of this paper I will concentrate on the ideas which we are considering for ε-TEX V2.

Ideas for ε-TEX V2

We are looking at facilities for testing that a particular character exists within a given font:

"`\iffontchar`" takes two parameters, a font and a character number, and yields "true" if and only if the character exists within the font. It is easy to define a macro "`\ifchar`" which tests for the existence of a character within the current font.

There are four related primitives for finding the dimensions of a particular character in a given font: "`\fontcharwd`", "`\fontchardp`", "`\fontcharht`", and "`\fontcharic`" each take two parameters, a font and a character number, and return the width, depth, height and italic correction respectively. The effect cannot be achieved by setting a character in a box and then measuring the box because one or more of the dimensions may be negative. As with "`\iffontchar`", it is simple to define macros which performs the analogous tasks for the current font.

We are debating whether to incorporate a control sequence "`\iffont`", which would take as parameter the external name of a font and return "`true`" if and only if a font metric file of the same name can be opened: not all members of the group are convinced of the need for this, so we would welcome your comments on this idea.

In ε-TEX V1, we provided additional diagnostics which report the line number at which a group was opened if it has not been closed at end of program: in V2, we propose to report in addition the name of the file, although implementation differences may prevent us from returning the path to the file. We are also considering reporting if a group mis-match is detected at end of file, although this may be limited to a fairly simple test of group level rather than a full check to ensure that the file is left at the identical group to that at which it was entered.

We hope to avoid many of the problems which currently beset writers of output routines by providing access to the items which are discarded when a page break is taken. These will be made available until the next page break in a new reserved box called "`\pagediscards`". We do not consider it possible at the present time to provide similar access to material discarded during line-breaking.

In a similar area, we hope to make "`\vsplit`" more useful by enabling the programmer to remove the "`\topsplitglue`" which is currently inserted by TEX: we propose to implement this in a very general manner by providing four list destructors, one of which is already present in ε-TEX V1. The four destructors are "`\firstnodetype`", "`\lastnodetype`", "`\removefirstnode`" and "`\removelastnode`". In a future implementation we may allow access to these nodes as well as the option of simply removing them.

In TEX, "`\parshape`" is really a write-only quantity: only the number of entries in the current "`\parshape`" can be subsequently interrogated. In ε-TEX V2, we intend to provide read access to all of the dimensions of "`\parshape`", although we are not yet certain whether we will do this through a single control sequence "`\parshapedimen`" or through a pair of control sequences "`\parshapewidth`" and "`\parshapeindent`". No matter which is implemented, the control sequence will be indexed by an integer to return a particular dimension from the current "`\parshape`".

In a manner analogous to ε-TEX V1's "`\currentgrouplevel`" and "`\currentgrouptype`", we intended to provide in V2 new control sequences for interrogating the current flow of control through conditionals: "`\currentiflevel`" and "`\currentiftype`". We are also considering a third, "`\currentifbranch`", which will enable the programmer to determine whether the thread of expansion is currently the "`*\then`" or the "`\else`" branch, and also to determine if the current "`\if`" has not yet received sufficient tokens to decide which branch to take. It may also be possible to use this control sequence to determine which "`\or`" has been taken in an "`\ifcase`", but we are not yet sure that this is possible.

Again by analogy with ε-TEX V1's "`\showgroups`", we intend to provide "`\showifs`" in V2.

We are considering, but have not yet agreed upon, a new class of alignment, "`\malign`": this, if implemented, would provide a "maths alignment" primitive.

Continuing on the theme of removing fixed limits from ε-TEX, we are looking into the possibility of removing fixed bounds on the number of count registers, dimen registers, skip registers, muskip registers, token-list registers and boxes. If we are successful in implementing these, then we will probably remove the fixed limit on the number of marks at the same time.

During discussions in Brno, Don asked us to consider providing control over the looseness of the last line in a paragraph: although TEX typesets this to its natural length (if "`\parfillskip`" is infinite, as it usually is), Don said that traditionally this line was set to the same looseness as the previous line. We are looking into ways of providing not only these two boundary conditions, but at a continuum between those extremes, possibly through a control sequence "`\finaladjdemerits`".

We are trying to save stack space in ε-TEX V2 by eliminating redundant assignments (that is, values which will be restored to exactly the same value on exit from a loop). No new primitive will be required for this: it is simply an internal optimisation.

We feel that the "lost chars" message which currently goes into the log file is sufficiently important that it should appear on the console as well; accordingly we are extending the semantics of "`\tracinglostchars`" so that a value greater than 1 will cause the message to appear on the console as well as in the log.

We are looking into an idea which would enable a programmer to add material (for example crop marks) to a box being shipped out, even if the output routine has been rendered inaccessible (for example, by a complex format such as LATEX). We will probably implement this through a control sequence "`\outpage`", analogous to "`\output`", the routine associated with which will be entered at the point at which a box is to be shipped out. This will also allow the programmer to overlay the box with material to be placed at fixed points on the page.

We are still discussing the implementation of "`\outpage`", and would welcome your advice as to whether you feel it should be recursive: that is, should a "`\shipout`" called from within an "`\outpage`" automatically invoke a further instantiation of "`\outpage`" unless the outer "`\outpage`" routine has already cleared the "`\outpage`" token-list register?

To allow for formats very different to the Plain/LATEX/AMS-TEX/LAMS-TEX family (for example, ATML[1]), we are considering providing an alternative to the current default of appending "`.tex`" to an otherwise unqualified file specification on an "`\input`", "`\openin`" or "`\openout`" command. We envisage allowing the default extension to be specified explicitly. Here again your comments would be welcome: do you believe that this would be useful?

A recurring proposal, but one which we have still not entirely agreed upon, is the idea of an "`\evaluate`" primitive, which would allow arithmetic to be carried out in ε-TEX's mouth. Although we are all agreed that "`\evaluate`" would be very beneficial, we also realise that to implement it in a way that will guarantee repeatability across all platforms will require that it be implemented without recourse to the host's floating-point arithmetic. This in turn implies either a very limited implementation, or requires considerable time to implement a full floating-point package in software. We are still discussing which of these to adopt (if either) for ε-TEX V2.

During our discussions in Brno, Don asked us to investigate the idea of providing greater control over the spacing of fractions so as to permit, for example, less dependency on the use of kludges such as "`\sub \strut`". We have not yet developed a suitable model for this, but we are continuing to investigate the possibilities.

Finally two fairly major proposals: ML-TEX, and pdfTEX: Mike Ferguson, the author/creator of ML-TEX, has given Bernd Raichle free rein to oversee ML-TEX's future. Bernd and Peter Breitenlohner have made us aware that the present implementation has some deficiencies, particularly in terms of the timing of certain operations. We hope to be able to provide a better implementation, but will ensure that users of ML-TEX are able to contribute to discussions on its specification before any decisions are made.

As to pdfTEX, this is a fairly recent project, undertaken by Han The Thanh at Masaryk University in Brno. We are very impressed with Thanh's work, but are not convinced that the model which he has adopted is necessarily the best way to proceed: in particular, we are concerned that changes of the magnitude required to support his present implementation could introduce subtle bugs into TEX which may be hard to detect and which would adversely affect its stability. Accordingly we are interested in investigating a simpler model which would defer much of the processing to a post-processor similar to Sergey Lesenko's dvi2pdf, but as Thanh has already pointed out, this would be incompatible

[1] "A TEX Markup Language", presented at EuroTEX'95, Papendal.

with adopting a variant of the HZ algorithm[2] which he believes could be incorporated into his present implementation. We are still considering the options in this area, but would very much like to support the concept of pdfTEX in some form.

Finally we would like to express our thanks to all who have made this project possible:

- To Professor Don Knuth, without whom many of us would never have met; for his foresight in creating TEX; and for his willingness to discuss ideas for ε-TEX despite his incredibly busy schedule. And to both him and his wife Jill for making me feel so welcome to join them during their week in the Czech Republic last year.
- To Joachim Lammarsch, for instigating the project;
- To DANTE e.V., for underwriting it;
- To Rainer Schöpf and Joachim Schrod, for their invaluable contributions during the first year of the project;
- To Peter Breitenlohner, without whom ε-TEX simply would not exist: Peter has written virtually all of the Web code, and has been responsible for many of the ideas now in ε-TEX;
- To Bernd Raichle, who has also been responsible for many of the ideas in ε-TEX; and who volunteered to write the `e-TRIP` test, which is a daunting task;
- To Jiří Zlatuška, who as Project Leader for the $\mathcal{NTS}$ project has been very patient in waiting for the project to get the financial backing it needs, and who has in the meantime contributed much to the development of ε-TEX;
- To Friedhelm Sowa, who is continuing to investigate colour and user interfaces, and who acts as treasurer for the project;
- To all the members of DANTE, for making me so welcome each time I attend their meetings, for their courtesy in speaking to me in English, and for their encouragement whenever I try to speak a few faltering words of German. And for their magnificent donation of DM 30 000 to the $\mathcal{NTS}$ project, which will enable the project to finally become a reality.

Philip Taylor,
March 1997.

[2] Electronic Publishing, Vol 6(3), 283–288 (September 1993) "About micro-typography and the hz-program", Herman Zapf.

Software & Tools

A GNU Emacs editing mode for METAFONT and METAPOST sources

Ulrik Vieth

Abstract

This article announces the release of `meta-mode.el`, a GNU Emacs editing mode for METAFONT and METAPOST source files. `meta-mode.el` provides a number of features commonly found in GNU Emacs editing modes for programming languages, such as automatic indenting of source code, syntactic highlighting, symbol completion for partially-typed keywords, motion commands to move to the beginning or end of the enclosing environment, or commands to comment out or reindent environments, regions, or buffers. An interface to running METAFONT or METAPOST in a shell buffer from within Emacs is currently under development and may be integrated into `meta-mode.el` later.

1 Introduction

The GNU Emacs[1] editor [1, 2] is one of the most widely used editors on Unix systems and some other platforms. As one of its most remarkable features it provides a vast number of specialized editing modes for a large variety of text formatting or programming languages. While support for editing TeX or LaTeX files has been included in Emacs for many years, either as part of the standard GNU Emacs distribution or through the optional AUC-TeX package [3], no such mode-specific editing support existed so far for the somewhat esoteric programming languages of METAFONT and METAPOST.

When I started using METAPOST on a regular basis in early 1995, shortly after I had completed my first port of METAPOST, integrating it into the Web2C/Kpathsea distribution, I wasn't too much concerned about the lack of editing support since I was primarily interested in getting acquainted with METAPOST so that I could get some data plotted. During the course of time, however, especially after I started gaining a little experience with Emacs Lisp programming [4, 5], I became increasingly unhappy about being stuck with fundamental mode for editing METAPOST sources in Emacs.

[1] While this article only refers to GNU Emacs, most of it should be applicable to the XEmacs editor as well. Although `meta-mode.el` was developed exclusively under GNU Emacs, it was tried to make sure that everything will also work under XEmacs.

So it happened one day in January 1997 that I began asking myself (and also our local Emacs guru) what it would take to write a new major mode for editing METAFONT or METAPOST sources. After consulting the *GNU Emacs Lisp Manual* [5] the task turned out to be simpler than expected and pretty much straightforward. On the following weekend, when I had some spare time, I sat down to begin writing what was to become `meta-mode.el`. By coincidence, it happened to be February 1, 1997, exactly twenty years after the day on which genesis of TEX took place according to Don Knuth's own account.[2] I suppose I couldn't have chosen a better date to embark on this project nor a better way to celebrate this very special anniversary ...

2 Overview of `meta-mode.el`

2.1 Installation

From the technical point of view `meta-mode.el` is a contributed Emacs Lisp package that first needs to installed in a place where it can be found by Emacs, i. e. either in a personal or system-wide Emacs Lisp library directory listed in the `load-path` variable. To activate the features provided in `meta-mode.el` the package then needs to be loaded, which is most easily arranged for by adding a few lines of Lisp code like these

```
(autoload 'metafont-mode "meta-mode"
  "Major mode for editing Metafont sources" t)
(autoload 'metapost-mode "meta-mode"
  "Major mode for editing MetaPost sources" t)
(setq auto-mode-alist
      (append '(("\\.mf\\'" . metafont-mode)
                ("\\.mp\\'" . metapost-mode))
              auto-mode-alist))
```

to the personal or system-wide Emacs startup file to have `meta-mode.el` autoloaded at the first time a METAFONT or METAPOST source file is opened.

2.2 Initialization

Once `meta-mode.el` is loaded, the above code has the effect of invoking an Emacs Lisp function called `metafont-mode` or `metapost-mode` whenever a '`.mf`' or '`.mp`' file is loaded, which then proceeds to set up everything necessary when entering the new editing mode. Much of this initialization code is actually identical for both METAFONT and METAPOST mode as far as it concerns routine tasks needed for every Emacs editing mode, such as setting up a syntax table or installing a keymap and a pull-down menu for the mode-specific functions. However, it turned out to be necessary to have two separate initialization functions to be able to take care of subtle differences between the two modes, most notably, perhaps, when it comes to the list of known symbols for the completion function or the list of shell commands to generate proof sheets.

Following the usual Emacs conventions, both of these initialization functions provide hook variables `metafont-mode-hook` and `metapost-mode-hook` to allow adding extra setup or customization code to the individual modes if desired. In addition, there is also a `meta-common-mode-hook` that applies to both modes as well as a `meta-mode-load-hook` that is evaluated when `meta-mode.el` is first loaded.

2.3 Features

Once the general framework for a major mode is in place, adding more features and mode-specific functions becomes relatively easy since they can be conveniently added one by one as needed. The functionality currently implemented in `meta-mode.el` can be summarized in the following areas:

- automatic indenting of source code
- syntactic highlighting (a.k.a. fontification)
- completion for partially-typed keywords
- other miscellaneous editing functions

Additional functionality for running METAFONT or METAPOST and related commands for producing proof sheets in a shell buffer from within Emacs is currently under development and may be included into `meta-mode.el` later. At present, a preliminary test version is implemented in a separate Emacs Lisp package, tentatively called `meta-buf.el`, which may be integrated with `meta-mode.el` by making clever use of the various hook variables discussed above. For example, the load hook may be used to load `meta-buf.el` at the same time `meta-mode.el` is loaded while the common mode hook may be used to make the functions provided in `meta-buf.el` available in the keymap.

2.3.1 Indentation

The default keymap used in `meta-mode.el` maps the `TAB` key to a function that reindents the current line using an appropriate indent level computed automatically. Furthermore, the `RET` key also reindents the current line before jumping to the appropriate indent level on the next line. This allows you to blindly type arbitrary META[3] code, terminating each line with `RET` as you type, and get a nicely

[2] Donald E. Knuth, *The Errors of TEX*, reprinted as Chapter 10 of *Literate Programming*, p. 249.

[3] We will henceforth use the term "META" whenever we discuss features that are applicable to both METAFONT and METAPOST.

indented source file from which the grouping level and the control flow of conditionals and loops is immediately apparent.

At present, `meta-mode.el` recognizes all standard META language constructs, including **if** … **fi**, **for** … **endfor** and **def** … **enddef** blocks, as well as common variants like **forever**, **forsuffixes**, **vardef**, or even **mode_def**. In addition, it also recognizes standard macros introducing block structures such as **beginchar** … **endchar** in METAFONT as well as **beginfig** … **endfig** and **begingraph** … **endgraph** in METAPOST.

Furthermore, occurrences of **begingroup** and **endgroup** are also considered, although this might actually be the wrong thing to do if these are used unbalanced across different macro definitions. Users should therefore be aware that it may occasionally be necessary to adjust the indentation of their source files manually in some unusual cases.

Much of the Emacs Lisp code used in the indentation function in `meta-mode.el` was adopted from AUC-TEX's `latex.el`, which actually had a somewhat simpler job since it only had to look out for `\begin`… `\end` environments or lonely `\items` while we have to handle a wider variety of META language constructs. Nevertheless, most of the AUC-TEX code could be put to a very good use. For example, the code that previously used to outdent `\items` could be adapted to handle occurrences of **elseif** and **else** within **if** … **fi** blocks. It would have been possible to apply the same logic to **exitif** and **exitunless** in the middle of **forever** … **endfor** blocks, but this idea was rejected since it appeared too different from common coding style.

In any event, `meta-mode.el` allows easy customization of the kinds of META language constructs recognized by modifying the default regular expressions, either by using `M-x edit-options` or by writing a few lines of Lisp code to put in the personal `~/.emacs` startup file. Some familiarity with Emacs regular expressions will be unavoidable, however, to customize `meta-mode.el` at this level.

2.3.2 Fontification

Font Lock mode is a minor mode provided in GNU Emacs which allows modification of the appearance of a variety of major editing modes for different programming or text formatting languages. The basic idea is to have a number of differently colored text faces, which are used to highlight various language elements consistently throughout all editing modes, such as keywords, function or variable names, references to external filenames, etc. In addition, certain language elements are also highlighted on the basis of their syntactic properties, such as comment lines or quoted strings.

Since Font Lock mode is an optional package it needs to be loaded and activated first. With recent versions of GNU Emacs this has become very easy, as it is possible to turn on Font Lock mode as well as optional Font Lock support packages globally with just two lines of Lisp code:

```
(global-font-lock-mode t)
(setq font-lock-support-mode 'lazy-lock-mode)
```

Once Font Lock mode is globally activated like this, it will automatically apply to any new editing mode that supports it. In order to take advantage of fontification when writing a major mode such as `meta-mode.el`, it suffices to set up a few syntactic variables and put together a list of regular expressions that match the various language elements we wish to have highlighted.

While putting together a regular expression to match a list of keywords is fairly easy, writing good patterns to match macro definition headers presents quite a challenge since we have to cope with the rich variety of language constructs that are available in the META languages. For instance, we have to be aware that there are not only straightforward unary macro definitions introduced by **def** or **vardef**, in which the name of the function follows immediately after the definition keyword, but also binary operator macro definitions introduced by **primarydef**, **secondarydef**, or **tertiarydef**, in which the name of the function is embedded in between the parameter arguments. Furthermore, function or variable names don't necessarily have to consist of alphabetic characters or underscores; they might just as well consist of non-word symbols such as '`$`' or '`@`' or operator symbols such as '`**`' or '`&&`'.

If this isn't enough, another complication arises when it comes to parsing seemingly straightforward variable declarations that involve a list of comma-separated arguments of arbitrary length. To handle this case a simple regular expression isn't enough; instead, it is necessary to write a special-purpose utility function to match the arguments.

While all this has caused many headaches during the development of `meta-mode.el`, it appears that the Font Lock patterns currently implemented are good enough to handle most common cases, as can be verified by loading `plain.mf` or `plain.mp` into GNU Emacs and turning on fontification.

Finally, it should be noted that there was one more case that required special attention, namely TEX code embedded in between **btex** … **etex** or **verbatimtex** … **etex** in METAPOST sources.

From the point of view of syntactic highlighting it seemed best to treat this embedded TeX code just like a quoted string as it isn't interpreted in any way by METAPOST itself, but just passed on to MakeMPX for typesetting. However, to ensure proper parsing this interpretation also made it necessary to retain the meaning of escape character for the backslash, although this doesn't agree with its usual meaning of **relax** in the META languages.

2.3.3 Symbol Completion

Automatic completion of partially-typed keywords or filenames is a concept found throughout most parts of GNU Emacs as well as in some modern Unix shells. The basic idea is to save keystrokes by allowing one to type just the first few letters and perform completion on pressing `M-TAB`, resulting in partial completion and a display of all possible matches if no unique match is found.

An appreciable side-effect of symbol completion is that it provides a way of spell checking keywords in a programming language, which helps to avoid some of the most annoying compilation errors.

To implement symbol completion when writing a new major mode it takes two things: a completion function that does the actual job, and a list of known symbols that are offered for completion.

As for the completion function implemented in `meta-mode.el`, there is little to say. It was directly adopted from AUC-TeX's `latex.el`, but the framework was considerably simplified since it appeared unnecessary to support multiple completion lists for different kinds of symbols in META mode, whereas it did make sense to have them in LaTeX mode.

As for the list of known symbols, there is a slightly more interesting story to tell: The idea was to have one comprehensive list of symbols for each of METAFONT and METAPOST which should include all primitives and macros defined in `plain.mf` or `plain.mp`, optionally augmented by the macros defined in standard packages, such as `graph.mp` or `boxes.mp` in the case of METAPOST. So what's the best method to get a complete list of primitives? The answer is simple: Use the source, Luke!

I eventually ended up with a little bit of Unix shell hackery along the lines of

```
$ grep '^primitive("[a-zA-Z]*"' {mf,mp}.web \
  | sed 's/primitive(\("[a-zA-Z]*"\).*/\1/' \
  | sort > {mf,mp}_prim.list
```

to extract the information about primitives directly from the `WEB` sources. Unfortunately, extracting the corresponding information from the macro definition headers in `plain.mf` and `plain.mp` didn't work out quite as well and required a little editing to fix up the extracted list, but this didn't matter too much since it had to be done only once anyway.

In any case, the resulting completion lists in `meta-mode.el` should be fairly comprehensive and might actually serve to give a good overview of what commands are available. Thus, if you ever wanted to know what tracing options exist, just type '`trac`' followed by `M-TAB` twice and see for youself. As this example illustrates, typing '`trac`' is sufficient to get a partial completion to '`tracing`', whereupon typing another one or two letters will be enough to resolve the remaining ambiguities.

In comparison to the completion in AUC-TeX it should be mentioned that `meta-mode.el` doesn't currently provide any context-sensitive completion, nor does it prompt the user to fill in the arguments where applicable. Instead, it just offers any known symbols for completion that match, regardless of whether they would make any sense in that context. Given the versatility of the completion function, it would certainly be possible to implement some of this by preparing a more involved completion list and some supporting functions if desired, but there are no such plans for the near future. After all, one might reasonably assume that users of METAFONT or METAPOST will be programmers who may be expected to know what they are doing, whereas authors of TeX documents don't necessarily have to be TeX macro programmers, and thus might require a little more help.

2.3.4 Miscellaneous Functions

As usual in Emacs editing modes for programming languages `meta-mode.el` also provides a small number of basic editing functions that are adapted to the mode-specific semantics. For instance, there are motion commands to move to the beginning or end of the previous or next "environment", or commands to apply the mode-specific indentation function or the standard Emacs comment-region function to each line in an "environment", a region, or a buffer.

As for what kinds of META language elements constitute an "environment", a somewhat different set of regular expressions is used than in the indentation function. Only the outermost block structures such as **beginchar** ... **endchar** in METAFONT or **beginfig** ... **endfig** in METAPOST are taken into account for this purpose, whereas conditionals and loops are disregarded. In addition, definition blocks such as **def** ... **enddef** and variants thereof are also considered as defining an "environment" for the convenience of editing more extensive macro packages. However, this may lead to problems if local macro definitions are nested inside **beginchar** ... **endchar** blocks, in which case a command on an environment might be incorrectly applied to the inner block rather than the outer one. Unfortunately, there doesn't seem to be a general solution to this other than modifying the default regular expressions.

2.3.5 Keybindings

Most of the mode-specific editing functions provided in `meta-mode.el` are bound to fairly standard keybindings also used in Emacs editing modes for other programming languages. For example, `M-C-a` and `M-C-e` are bound to the motion commands applicable to environments while `M-a` and `M-e` are retained for the motion commands applicable to sentences, primarily for use in comment paragraphs. Likewise, `M-C-q` reindents an environment of META code while `M-q` is retained as the function to refill text in a comment paragraph.

A complete listing of mode-specific keybindings in `meta-mode.el` can be obtained using the Emacs help system. Furthermore, if Emacs is run under a windowing system such as X11, all mode-specific editing commands are also accessible from a pull-down menu entitled "Meta" that gets installed in the menubar when entering METAFONT or METAPOST mode. This menu lists all available mode-specific editing commands along with their corresponding keybindings.

3 Availability

In the past, preliminary versions of `meta-mode.el` have been made available by postings to the Usenet newsgroup `gnu.emacs.sources` and the METAFONT mailing list at `metafont@ens.fr`. As of version 1.0, `meta-mode.el` has been uploaded to CTAN where it has found a place in the `tex-archive/support/emacs-modes/` directory.

Shortly after releasing one of the early test versions, I was contacted by Richard Stallman about signing a copyright transfer agreement to the Free Software Foundation to allow the integration of `meta-mode.el` into the GNU Emacs distribution, which I have done now. Therefore, readers may look forward to finding `meta-mode.el` as the default METAFONT or METAPOST editing mode when the next version of GNU Emacs eventually arrives.

While the functionality provided in version 1.0 of `meta-mode.el` is pretty stable now, development of some additional features will continue. Most importantly, there are plans to implement an interface to allow running METAFONT or METAPOST in a shell buffer from within Emacs. In order to ensure stability, however, such development will be confined to add-on packages such as `meta-buf.el` that may be merged back into `meta-mode.el` from time to time, when the new features have proved stable.

4 Acknowledgements

A number of features implemented in `meta-mode.el` have been significantly influenced by features found in various Emacs editing modes for other programming or text formatting languages, among them, in particular, the AUC-TEX package, from which I drew much of the indentation and symbol completion functions. Emacs Lisp code was borrowed and adapted to the new purposes wherever possible, thereby sharing all the good ideas in the true free software tradition while at the same time avoiding to reinvent the wheel unnecessarily. Putting it all together and supplying the necessary knowledge about META language features to write appropriate font lock patterns and regular expressions, however, is the main ingredient for which I take responsibility entirely myself. I hope METAFONT and METAPOST users using any flavor of Emacs will enjoy it!

References

[1] Debra Cameron, Bill Rosenblatt, and Eric Raymond. *Learning GNU Emacs.* O'Reilly & Associates, Inc., September 1996.

[2] Richard Stallman. *GNU Emacs Manual.* Free Software Foundation, August 1996. 12th edition, for Emacs version 19.34.

[3] Kresten Krab Thorup. GNU Emacs as a front end to LATEX. *TUGboat*, 13(3):304–308, October 1992.

[4] Robert J. Chassell. *Programming in Emacs Lisp — An Introduction.* Free Software Foundation, October 1995. edition 1.04.

[5] Bil Lewis, Daniel LaLiberte, Richard Stallman, and GNU Manual Group. *GNU Emacs Lisp Reference Manual.* Free Software Foundation, June 1995. edition 2.4, for Emacs Version 19.29.

◇ Ulrik Vieth
Heinrich-Heine-Universität Düsseldorf
Institut für Theoretische Physik II
Universitätsstraße 1
D-40225 Düsseldorf
Germany
`vieth@thphy.uni-duesseldorf.de`
URL: `http://www.thphy.uni-duesseldorf.de/~vieth/`

Philology

The Traditional Arabic Typecase, Unicode, TEX and METAFONT

Yannis Haralambous[*]

1 Introduction

The first Arabic book, a 5 × 11 cm volume titled كتاب صلات السواعي (*Book of the prayer of hours*), was printed in 1514 by Grégoire de Grégoire in Venice and Fano, under the protection of Pope Leo the 10th [1, p. 18 – 19]. It took about two centuries for Arabic book printing to move to the East: in 1727 the Ottoman printing agency was founded in Constantinople and started printing using Dutch types and technology [8, p. 156]. A similar institution was founded in Cairo in 1821.

Undoubtedly a script like the Arabic one, having deep roots in calligraphy, was rather difficult to adapt to typography, a technique where strict standardization and repetition of forms is necessary. When Aldus Manutius created the first *italic* font in 1501, out of manuscript calligraphic forms, he made a certain number of choices — and these choices became a standard for occidental typography. Similar choices had to be made for Arabic: calligraphy had to be "tamed", so that the results would be homogeneous, reproducible, and flexible enough to be pleasant to the eye.

This standardization took place in 1906, in Cairo, when the ألطابع الأماري (*’Almaṫāb‘ al’amārya*) typecase is defined. This typecase (see fig. 1), divided in four parts (as opposed to "upper" and "lower" case of the Occident), uses a total of 470 characters. Astonishing as it may seem, this typesetting system has been kept in use until today: books typeset in a traditional way, all around the Arabic world, are still using the same set of characters, and the same conventions and rules.[1] In fig. 1, the reader can see the four parts of this typecase.

The reader knowing the technical limitations of computerized typesetting can already imagine the effect of computers on the Arabic script: not being able to cope with the complexity of the Cairo typecase, the computer industry has tried (and was finally able) to impose new standards of simplified typesetting,[2] most of the time covering only the fundamental properties of Arabic script, without any typographical enhancement. Was it the computers, which have simplified Arabic printed script, or was it a deeper change in Arabic society and mentality? This is hard to say; nevertheless, even today, commercial computer typesetting systems are — a few isolated exceptions apart — unable to reach the typographic quality of *’Almaṫāb‘ al’amārya*. In fig. 2, one can see different samples of printed Arabic material, showing the evolution and simplification of Arabic script; these examples are extreme cases: the first one is taken from a scholarly book printed in Lebanon (it contains almost all ligatures of the *’Almaṫāb‘ al’amārya* typecase), the second from a technical book printed in East Germany (a fewer number of ligatures), and the third from a daily newspaper printed in the U.K. (almost no ligatures).

This paper describes the author's solution to this problem: الامل(Al-Amal), a typesetting system based on TEX (actually TEX--XET), emulating the *’Almaṫāb‘ al’amārya* typecase. This system (already presented in [6] and [7]), has been recently extended to the complete set of Unicode Arabic alphabet characters; problems and open questions arising from this extension are discussed at the end of the paper.

2 The Cairo typecase

Arabic letters have contextual forms, depending on surrounding letters in the same word: a typical three letter word will start with a letter in initial form, followed by a letter in medial form and, finally, by a letter in final form (the hypothetical word consisting of three times the letter 'gha' is written غغغ). A fourth form is used for isolated letters (this is also the form used in crosswords or Scrabble-like games, where letters have to placed in boxes, indepedently of their context). Some letters appear only in isolated and final form (and sometimes even only in isolated form), so that the letters immediately following them must be written in initial (or isolated) form, although they are located inside the word.

These are the basic contextual rules of the Arabic script: they are independent of style and medium, and are applied in all cases, without exception; they are as basic as the dot on the Latin lowercase 'i', or the horizontal bar of the 't'.

But besides these contextual forms, *’Almaṫāb‘ al’amārya* also combines letters into *ligatures*, not

[*] The author would like to thank Michel Goossens, for — among other things — having given him access to [11], an extremely exciting book which has motivated this and forthcoming developments.

[1] In [4, p. 102 – 103], a book published in 1880 (!) the reader can find 30 rules for typesetting Arabic, which are still *strictly* applied today by traditional typographers.

[2] For more information on the Arabic script and the computer see also [3] and [10].

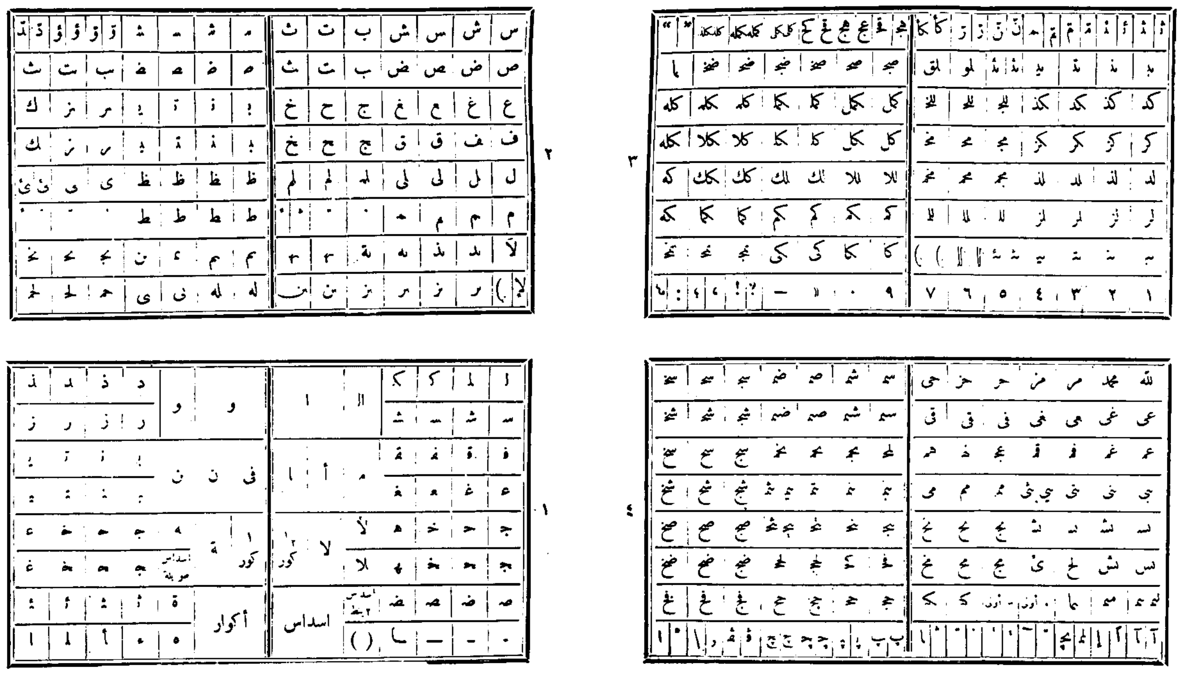


Figure 1: The Cairo typecase — cases 1 (left, bottom), 2 (left, top), 3 (right, top) and 4 (right, bottom)

unlike the ‘f’ + ‘i’ → ‘fi’ phenomenon in Latin alphabet typesetting. In fig. 3, the reader can compare the same text with and without ligatures. The first case is more-or-less the best one can obtain from a standard commercial Macintosh and Windows-based Arabic typesetting system. The text of the second figure is typeset in الامل (Al-Amal), a typesetting system developped by the author, and based upon the combined use of a Lex/Yacc preprocessor, TeX and METAFONT; it follows the traditional typesetting rules of the *’Almaṫāb‘ al’amārya* typecase.

3 Typesetting rules for the Cairo typecase

In this section we give a short description of the most important ligatures and variant characters found in the Cairo *’Almaṫāb‘ al’amārya* typecase, and their use. This set of rules is described in [4, p. 102 – 103] and has been confirmed by careful examination of various printed texts of different origin. In fig. 3 the reader can compare the same text (actually the text of the first example of fig. 2) typeset via the Al-Amal system, with and without *’Almaṫāb‘ al’amārya* ligatures.

In the following we start by giving the mandatory ligatures (those that are part of every font), then we give the second level ligatures (those that are characteristic of the Cairo typecase), and finally we give the variant characters (form changes applied to single characters). By “foo-like”, where “foo” is some common Arabic letter, we mean all characters having the same base form as letter “foo” but different dots and other diacritics (for example, the ba-like family is the set of characters ب, ت, ث, ط, etc.).

After each entry of our list we give the coordinates of the type positions in the Cairo *’Almaṫāb‘ al’amārya* typecase (see fig. 1). These coordinates are notated in the following chess-like way: (a) the number of case (1 – 4), (b) the column (A – N, counting from right to left[3] with a subscript when the slot is splitted into two parts, the first part being the one on the right), (c) the row, (1 – 8 counting bottom to top).

3.1 Mandatory ligatures

L1. a lam-like letter followed by an alif-like letter: لا, جميلا, الأثنين, etc. 1E3, 1E4, 1F3, 2A2, $2A_1 1$.

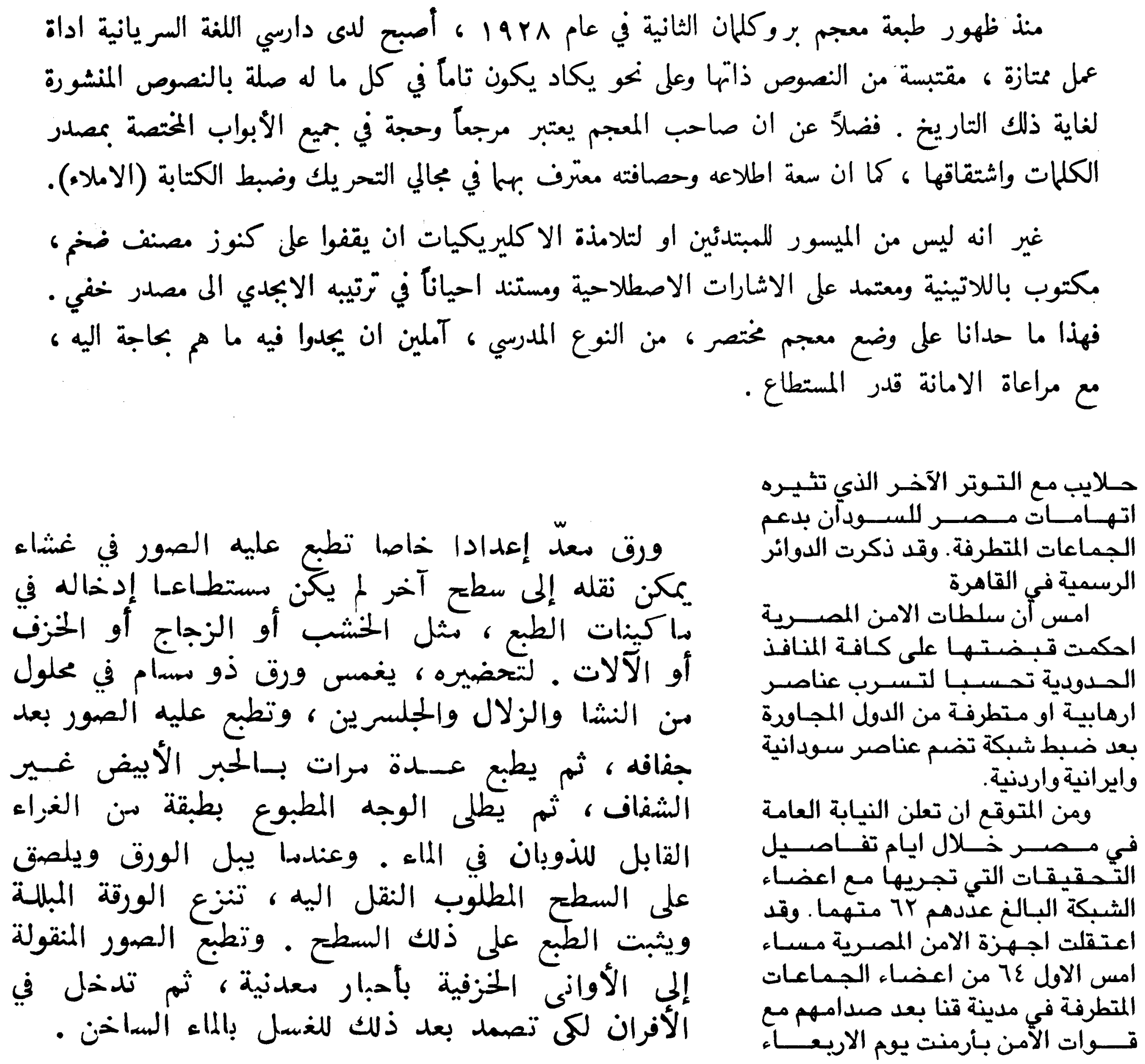
L2. the second part of the word *Allah* (God): الله. 4A8.

3.2 Typographical ligatures

L3. a lam-like letter followed by meem: لماح, سلمى, علم, لم, etc. 1B8, 2E4, 2F4, 2G4.

L4. a ba-like letter followed by a ra-like one: ثبر, فتر, جير, etc. 2B1, 2C1, 2D1, 2E1.

[3] Remember, we are reading from right to left!



منذ ظهور طبعة معجم بروكلمان الثانية في عام ١٩٢٨ ، أصبح لدى دارسي اللغة السريانية اداة عمل ممتازة ، مقتبسة من النصوص ذاتها وعلى نحو يكاد يكون تاماً في كل ما له صلة بالنصوص المنشورة لغاية ذلك التاريخ . فضلاً عن ان صاحب المعجم يعتبر مرجعاً وحجة في جميع الأبواب المختصة بمصدر الكلمات واشتقاقها ، كما ان سعة اطلاعه وحصافته معترف بهما في مجالي التحريك وضبط الكتابة (الاملاء).

غير انه ليس من الميسور للمبتدئين او لتلامذة الاكليريكيات ان يقفوا على كنوز مصنف ضخم ، مكتوب باللاتينية ومعتمد على الاشارات الاصطلاحية ومستند احياناً في ترتيبه الابجدي الى مصدر خفي . فهذا ما حدانا على وضع معجم مختصر ، من النوع المدرسي ، آملين ان يجدوا فيه ما هم بحاجة اليه ، مع مراعاة الامانة قدر المستطاع .

ورق معدّ إعدادا خاصا تطبع عليه الصور في غشاء يمكن نقله إلى سطح آخر لم يكن مستطاعا إدخاله في ماكينات الطبع ، مثل الخشب أو الزجاج أو الخزف أو الآلات . لتحضيره ، يغمس ورق ذو مسام في محلول من النشا والزلال والجلسرين ، وتطبع عليه الصور بعد جفافه ، ثم يطبع عــدة مرات بـالحبر الأبيض غـير الشفاف ، ثم يطلى الوجه المطبوع بطبقة من الغراء القابل للذوبان في الماء . وعندما يبل الورق ويلصق على السطح المطلوب النقل اليه ، تنزع الورقة المبللة ويثبت الطبع على ذلك السطح . وتطبع الصور المنقولة إلى الأوانى الخزفية بأحبار معدنية ، ثم تدخل في الأفران لكى تصمد بعد ذلك للغسل بالماء الساخن .

حـلايب مع التـوتر الآخـر الذي تثـيـره اتـهــامــات مــصـــر للســـودان بدعم الجماعات المتطرفة. وقد ذكرت الدوائر الرسمية في القاهرة
امس أن سلطات الامن المصـــرية احكمت قبـضـتـهـا على كـافـة المنافذ الحـدودية تحـسـبـا لتـسـرب عناصـر ارهابيـة او مـتطرفـة من الدول المجاورة بعد ضبط شبكة تضم عناصر سودانية وايرانية واردنية.
ومن المتوقع ان تعلن النيابة العامة فـي مــصـــر خـــلال ايام تفـاصــيـل التـحـقـيـقـات التي تجـريها مـع اعضـاء الشبكة البـالغ عددهم ٦٢ متـهمـا. وقد اعـتقلت اجـهـزة الامن المصـرية مسـاء امس الاول ٦٤ من اعـضـاء الجـمـاعـات المتطرفة في مدينة قنا بعد صدامهم مع قـــوات الأمن بـأرمنت يوم الاربعــــاء

Figure 2: Samples of printed Arabic: Beirut 1963 (top), Leipzig 1981 (left), London 1992 (right).

L5. a ba-like letter followed by a final noon-like one: فنن, فلسطين, پختن, etc. 2F1, 2K2.

L6. a lam-like letter followed by a ha-like one: 2H1, 2I1.

L7. a ba-like letter followed by a ya-like one: سبي, سكنى, أخواتى, etc. 2J1, 2K1.

L8. a gim-like letter followed by meem: حمن, جم, etc. 2L1.

L9. a lam-like letter followed by a gim-like, and eventually a meem: لج, لجام, لحمة, etc. 2M1, 2N1.

L10. a ba-like letter followed by a ha-like: بهم, تهلل, پهلوى, etc. 2F2, 2G2.

L11. a ba-like letter followed by meem: تمتع, ثمل, منهر, غنهة, مقيم, غيم, قا, خائمتم, etc. 2H2, 2I2, 2J2.

L12. a ba-like letter followed by a gim-like one, and eventually a meem: بجاثة, تخت, ثخن, بچه, يخ, تحمل, etc. 2L2, 2M2, 2N2.

L13. a lam-like letter followed by a ya-like one: إلي, etc. 2C4, 2D4.

L14. a kaf-like letter followed by an alif-like, or a lam-like, or a final kaf-like: كأب, كلاسك, كلبى, etc. Such a two-letter ligature can be extended to a three-letter or even four-letter ligature, by adding a ya-like letter, or a ha-like letter, or lam-like letter, or lam-alif-like ligature, etc.: كلى, كلام, كلما, etc. 3H2, 3I2, 3L4, 3M4, 3H5,

منذ طهور طبعة معجم بر وكلمان الثانية فى عام ١٩٢٨ء أصبح لدى دارسى اللغة السريانية اداة عمل
ممتازةء مقتبسة من النصوص ذاتها وعلى نحو يكاد يكون تاماً فى كل ما له صلة بالنصوص المنشورة
لغاية ذلك التاريخ. فضلاً عن ان صاحب المعجم يعتبر مرجعاً وحجة فى جميع الأبواب المختصة بمصدر
الكلمات واشتقاقهاء كما ان سعة اطلاعة وحصافته معترف بهما فى مجالي التحريك وضبط الكتابة
(الاملاء).
غير انه ليس من الميسور للمبتدئين او لتلامذة الا كلير يكيات ان يقفوا على كنوز مصنف ضخمء
مكتوب باللاتينية ومعتمد على الاشارات الاصطلا حية ومستند احياناً فى ترتيبه الابجدي الى مصدر
خفى. فهذا ما حدانا على وضع معجم مختصرء من النوع المدرسيء آملين ان يجدوا فيه ما هم بحاجة ليهء
مع مراعاة الامانة قدر المستطاع.

منذ طهور طبعة معجم بر وكلمان الثانية في عام ١٩٢٨ء أصبح لدى دارسي اللغة السريانية اداة
عمل ممتازةء مقتبسة من النصوص ذاتها وعلى نحو يكاد يكون تاماً في كل ما له صلة بالنصوص
المنشورة لغاية ذلك التاريخ. فضلاً عن ان صاحب المعجم يعتبر مرجعاً وحجة في جميع الأبواب
المختصة بمصدر الكلمات واشتقاقهاء كما ان سعة اطلاعة وحصافته معترف بهما في مجالي
التحريك وضبط الكتابة (الاملاء).
غير انه ليس من الميسور للمبتدئين او لتلامذة الا كلير يكيات ان يقفوا على كنوز مصنف ضخمء
مكتوب باللاتينية ومعتمد على الاشارات الاصطلا حية ومستند احياناً في ترتيبه الابجدي الى
مصدر خفي. فهذا ما حدانا على وضع معجم مختصرء من النوع المدرسيء آملين ان يجدوا فيه ما
هم بحاجة ليهء مع مراعاة الامانة قدر المستطاع.

Figure 3: Samples of text typeset with Al-Amal, with (top) and without (bottom) Cairo typecase ligatures

3I5, 3J5, 3K5, 3L5, 3M5, 3N5, 3L6, 3M6, 3N6, 3K8, 3M8.

L15. a ba-like letter, followed by a meem and a gim-like letter: 3L2, 3M2, 3N2.

L16. a lam-like letter followed by a lam-like letter and eventually by a meem or a gim-like letter: 3E3, 3F3, 3G3, 3E6, 3F6, 3G6.

L17. a kaf-like letter followed by a meem and eventually other letters: لكمة, لكم, كم, بكان, etc. 3H3, 3I3, 3J3, 3K3, 3L3, 3M3, 3H6, 3I6, 3J6, 3K6, 3L6, 3M6.

L18. a meem followed by a gim-like letter and eventually a meem: مجد, مخ, محمل, etc. 3E4, 3F4, 3G4, 3E5, 3F5, 3G5.

L19. a ṡad-like, ha-like, fa-like or kaf-like letter followed by a gim-like one: صحفى, ضج, فخل, فجر, فخر, etc. 3H7, 3I7, 3J7, 3K7, 3L7, 3M7, 3H8, 3I8, 3J8, 4I3.

L20. a ba-like, or lam-like, followed by meem, or a meem followed by a ba-like, followed by meem, or a lam-like followed by two meems: $4C_11$, 4A2, 4B2.

L21. a sin-like, or ṡad-like, or fa-like, or ayn-like, or gim-like, followed by a gim-like and eventually by a meem: جحفل, حجج, خجلان, سجع, شج, صحفى, ضج, عجل, عج, غجر, عجمى, فخمة, etc. TABLE 4, COLUMNS L – N, ROWS 2 – 8 AND COLUMNS H – K, ROWS 2 – 4.

L22. a lam-like letter or lam-meem-like ligature, followed by a gim-like letter, and eventually a meem: لج, لجام, لجة, لمح, لحمة, etc. 4C3, 4H6.

L23. the name “Mohammad” محمد 4B8.

3.3 Variant forms

V1. an initial ba-like letter in front of a sin-like, ṡad-like, ayin-like, waw-like or ha-like one grows higher: بسمة, تعليق, ثورة, نهزة, يونان, etc. 2H6, 2I6, 2J6, 2K6, $3A_18$, $3B_18$.

V2. a medial ba-like letter between two ba-like letters, or in front of a sin-like letter grows higher:

تتبع, ثبت, بءيس, تستيت, تقييش, متيسر, etc. 2H5, 2I5, 2J5, 2K5, $3A_28$, $3B_28$, 3A7, 3B7, 3C7, 3D7, 3E7.

V3. an initial or medial gim-like letter in front of an alif-like or lam-like letter takes a rounder closed form: جليل, حاجت, چاي, etc. 1K3, 1K4, 1L3, 1L4, 1M3, 1M4.

V4. an initial meem in front of a ra-like letter, a ha-like letter or a ya-like letter gets smaller and non-hollow: مربع, مهل, أمي, etc. 4C8, 4D8.

V5. a ra-like letter following a gim-like, ṫa-like, ayn-like, fa-like, kaf-like, ha-like letter or a meem, takes a more calligraphic form: حرف, طرطر, ظرف, عربي, غزل, قرمة, كريم, گربه, هرع, etc. 2L5, 2M5.

4 Porting the Cairo case to Unicode

The first plane of ISO 10646-1, also known as Unicode, provides characters for the following languages: Arabic (modern and classical), Farsi, Urdu, Pashto, Sindhi, Ottoman Turkish, Baluchi, Kashmiri, Kazakh, Lahnda, Dargwa, Uighur, Turkic, Berber, Hausa, Malay, Adighe, Ingush, Kirghiz [12].[4]

Similarly to European languages which have diacritized letters of the Latin alphabet to adapt them to their phonetic needs, the languages stated in the previous paragraph have added diacritics to the letters of the basic Arabic alphabet. There is a slight difference though: historically, Arabic alphabet was first written without dots;[5] so in a sense, dots are already “diacritics”. It is only natural that these languages have first tried to use new combinations of dots and letter forms: almost every combination of basic form and sets of one, two, three, or even four dots, over or under the word has been used to obtain new characters.

The author has expanded the الامل system to cover all these characters derived from the basic Arabic script; in fig. 4 the reader can see an example of Sindhi text (kindly provided to the author by Prof. Aqha, Univ. of Illinois) typeset in Al-Amal. In most of the cases, the extension to Unicode has been a straightforward task. Nevertheless, in some cases the fact of applying a ligature or even just a contextual form similar to those of the basic Arabic alphabet brought up ambiguities. These will be discussed below.

4.1 Cases where contextuality leads to confusion between characters

1. **Letters fa and qaf.** In basic Arabic, letters fa ف (and its artificial derivative va ڤ) and qaf ق have different forms: the former is longer and flatter, while the latter is rounder and deeper. This difference is visible only in the isolated and final forms: compare ففف ف and ققق ق. Since these letters differ mainly in the number of dots (one for fa, two for qaf, three for va), the shape difference is of minor importance, and in some modern Arabic typefaces it is totally ignored.

   The problems arise with Unicode characters `06A7` (ARABIC LETTER QAF WITH DOT ABOVE) and `06A8` (ARABIC LETTER QAF WITH THREE DOTS ABOVE), which use the basic shape of letter qaf, and have the same number of dots as fa and va. These characters are used in Maghribi Arabic. In initial and medial forms, as well as in ligatures involving these forms, they are indistinguishable from the basic Arabic letters fa and va.

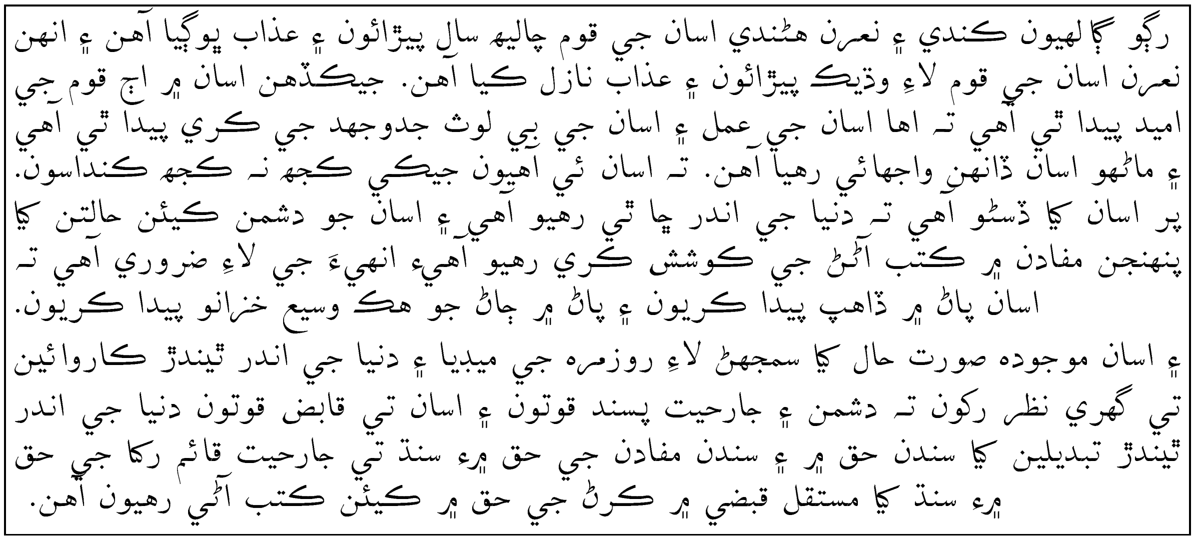
2. **Letters ta, noon and ya.** In basic Arabic, letters ta ت and noon ن have different forms: the former is longer and flatter and the latter is rounder and deeper. Once again, the difference can only be seen in isolated and initial forms: compare تتت ت and ننن ن. Since these letters differ mainly in the number of dots (one above for noon, two above for ta, etc.) the shape difference is of minor importance.

   Unicode characters `06BB` (ARABIC LETTER RNOON), `06BD` (ARABIC LETTER NOON WITH THREE DOTS ABOVE) use the letter form of the Arabic noon and the dots of the Urdu letter tteh and the Arabic letter tha. These letters are used in Sindhi and Malay. Their initial and medial forms, as well as all ligatures involving initial and medial forms are indistinguishable from the Urdu and Arabic counterparts.

   The situation is even more complicated since the Arabic letter ya ي shares the same initial and medial forms as ba, noon and friends: ي ييي. Nevertheless, the isolated and final forms of this letter are significantly different from

[4] This set of characters is quite complete; nevertheless, the author encountered characters not provided in Unicode, in four cases: for typesetting the Qur'ān, a ba-like letter without dot is needed [2, p. 102–103] (one new character), for typesetting old manuscripts, all characters are needed without dots (2 new characters, in ba-like and qaf-like forms), Salem Chaker's proposal for the transcription of Berber into Arabic script [5] (one new character), and Ahmed Lakhdar's proposal for the writing of African languages [9] (7 new characters and 6 new diacritics).

[5] Take for example letters ب ('b'), ت ('t'), ث ('th' like in 'thought'); they differ only by the number and position of dots. Originally, these letters were all written without dots, and the reader had to guess their pronounciation from the context(!).



رڳو ڳالهيون ڪندي ۽ نعرن هڻندي اسان جي قوم چاليهه سالِ پيڙائون ۽ عذاب ڀوڳيا آهن ۽ انهن
نعرن اسان جي قوم لاءِ وڌيڪ پيڙائون ۽ عذاب نازل ڪيا آهن. جيڪڏهن اسان ۾ اڄ قوم جي
اميد پيدا ٿي آهي تہ اها اسان جي عمل ۽ اسان جي ٻي لوث جدوجهد جي ڪري پيدا ٿي آهي
۽ ماڻهو اسان ڏانهن واجهائي رهيا آهن. تہ اسان ئي آهيون جيڪي ڪجهہ نہ ڪجهہ ڪنداسون.
پر اسان کيا ڏسڻو آهي تہ دنيا جي اندر ڇا ٿي رهيو آهي ۽ اسان جو دشمن ڪيئن حالتن کيا
پنهنجن مفادن ۾ ڪتب آڻڻ جي ڪوشش ڪري رهيو آهيء انهيءَ جي لاءِ ضروري آهي تہ
اسان پاڻ ۾ ڏاهپ پيدا ڪريون ۽ پاڻ ۾ ڄاڻ جو هڪ وسيع خزانو پيدا ڪريون.
۽ اسان موجوده صورت حال کيا سمجهڻ لاءِ روزمره جي ميڊيا ۽ دنيا جي اندر ٿيندڙ ڪاروائين
تي گهري نظر رکون تہ دشمن ۽ جارحيت پسند قوتون ۽ اسان تي قابض قوتون دنيا جي اندر
ٿيندڙ تبديلين کيا سندن حق ۾ ۽ سندن مفادن جي حق ۾ء سنڌ تي جارحيت قائم رکا جي حق
۾ء سنڌ کيا مستقل قبضي ۾ ڪرڻ جي حق ۾ ڪيئن ڪتب آڻي رهيون آهن.

Figure 4: Sindhi text typeset in Al-Amal

those of the ba and noon letter shapes. Once again, in basic Arabic the number and position of dots is sufficient for determining the letter (ya carries two horizontaly aligned dots below).

Unicode character 067B (ARABIC LETTER BEEH) has the form of ba and carries two vertically aligned dots below; this is also the case of 06D0 (ARABIC LETTER E) which carries the same set of dots, but has the form of an Arabic ya. Furthermore, 06D1 (ARABIC LETTER YEH WITH THREE DOTS BELOW) carries three dots below, exactly as does Arabic letter tha: the former has the letter form of a ya, while the latter the one of a ba.

3. **Arabic and Sindhi letters kaf.** In Arabic, the letter kaf is written with an oblique ascender stroke in initial and medial form, and with a hamza-like diacritic in final and isolated form. Sindhi uses a kaf-like letter 06A9 (ARABIC LETTER KEHEH) which has oblique ascender strokes in all forms and no hamza-like diacritic. This letter is indistinguishable from the Arabic kaf, in initial and medial forms, as well as in all ligatures involving these forms.

4.2 Cases where ligatures obstruct proper diacritization of characters

1. The Pashto ring (as in ټ) is incompatible with the ba-like + gim-like ligature (for example ټج) and the initial/isolated ba-like + meem ligature (for example ټم). Either the ring must be designed like "a drop that hangs" — a dubious esthetic result — or the ligature must be broken. The author has tried to design a ligature of isolated form ټم, but the result is not entirely satisfying.
2. The Uighur character ٵ 0675 (ARABIC LETTER HIGH HAMZA ALEF) can hardly take part in a lam-alef-like ligature: the hamza would be too far to the right.[6]
3. The fact that *'Almaṭāb' al'amārya* ligatures have been designed without taking into account Indic characters, makes many ligatures with non-standard dots ambiguous: is ڤچ the combination of ڥ and ح (06A5 ARABIC LETTER FEH WITH THREE DOTS BELOW and the standard Arabic ḫah) or of ڡ and چ (06A1 ARABIC LETTER DOTLESS FEH and 0686 ARABIC LETTER TCHEH)? Theoretically, one can distinguish them by slightly moving the dots to the right in the former case (ڥح vs. ڡچ); but still the two forms are very close graphically, and it may be difficult to the reader to distinguish them at first sight.

5 Technical details

5.1 Preprocessing

The extended Al-Amal system consists of four modules, as shown in fig. 5:

1. re-encoding into the (extended) Unicode encoding;
2. standard contextual analysis and processing of the mandatory ligatures;

[6] Not to mention the fact that in the Qur'ān one finds a lam-alef ligature with a central hamza, not included in Unicode.

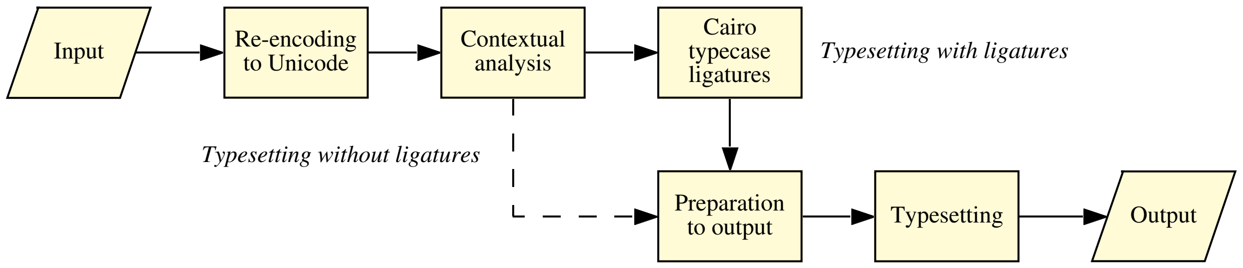


Figure 5: The Al-Amal internal structure

3. Cairo typecase ligatures processing (optional); and
4. output preparation (conversion into TEX code).

The first three modules are independent of TEX. To avoid ligatures one simply removes module 3 from the processing chain. Preprocessors have been written in C, using Flex and Bison tools: writing a grammar for Arabic ligatures avoids tedious pattern matching.

5.2 The fonts

The Al-Amal have been designed in the METAFONT language, to benefit from the maximum possibilities of optical scaling. Many ligatures have been split in several parts and are re-combined by TEX (this is essentially the task of the preprocessor module). One can consider these fonts as *glyph containers*, providing glyphs which TEX combines into characters and ligatures. This approach has allowed minimization of storage space and time needed for design the font. The author was able to produce all possible Cairo typecase ligatures on the Unicode Arabic character set, using only six 8-bit (partially filled) font tables,[7] consisting of less than 1500 glyphs. See tables 1 – 6.

References

[1] Josée Balagna, *L'imprimerie arabe en occident*, Maisonneuve & Larose, Paris 1984.

[2] Syed Barakat Ahmad, *Introduction to Qur'anic Script*, Curzon Press, London, 1985.

[3] Joseph D. Becker, *Arabic Word Processing*, Communications of the ACM, **30** (7), 1987.

[4] Claus Faulmann, *Das Buch der Schrift, enthaltend die Schriftzeichen und Alphabete aller Zeiten und aller Völker des Erdkreises*, Vienna, 1880 (reprint by Franz Greno, Nördlingen, 1985).

[5] Yannis Haralambous, *Un système TEX berbère*, Études et documents berbères, **11** 43 – 54, 1991.

[6] Yannis Haralambous, *Towards the revival of traditional Arabic typography... through TEX*, Proceedings of the EuroTEX92 conference, Prague, 1992

[7] Yannis Haralambous, *Typesetting the Holy Qur'ān with TEX*, Proceedings of the 2nd International Conference on Multilingual Computing (Latin and Arabic script), Durham, 1992.

[8] Klaus Lagally, *ArabTEX — Typesetting Arabic with Vowels and Ligatures*, Proceedings of the EuroTEX92 conference, Prague, 1992

[9] Ahmed Lakhdar-Ghazal, *Caractères arabes diacritiques selon l'ASV-CODAR (pour imprimer les langues arabes)*, Institut d'Études et de Recherches pour l'Arabisation, Rabat, 1993.

[10] Pierre MacKay, *Typesetting problem scripts*, BYTE 11, **2** 1986, 201 – 218.

[11] Roland Meynet, *L'écriture arabe en question*, Dar el-Maghreb Éditeurs, Beyrouth, 1971.

[12] The Unicode Consortium, *The Unicode Standard*, Version 1.0, Vol. 1, Addison-Wesley, 1991.

◇ Yannis Haralambous[8]
Atelier Fluxus Virus, 187, rue Nationale, F-59 800 Lille, France
`yannis@pobox.com`
URL: `http://pobox.com/~yannis`

[7] In a forthcoming implementation of Al-Amal to Ω, these fonts will be merged into one 16-bit (virtual) font, and contextual analysis, as well as Cairo typecase ligatures, will be handled by Ω Translation Processes.

	"x0	"x1	"x2	"x3	"x4	"x5	"x6	"x7
"1x	ج	ح	ـحـ	ح	خ	خـ	ـخـ	خ
	د	ذ	ـذ	ر	ز	س	ـسـ	س
"2x	-	!				٪		
	)	(			ء	–	.	\
"3x	٠	١	٢	٣	٤	٥	٦	٧
	٨	٩	:	؛	»		«	؟
"4x	ـ	ا		ـئے	ضـ	ے	ئے	غـ
	ـحـ	ـہـ		گ	ﷲ	ﷲ	ٹ	ـٹ
"5x	ـٹـ	ـثـ	ـک	ـشـ	ط	ث	ـث	ـقـ
	ـفـ	صـ	ظ	ـثـ	کـ	ک	ق	ق
"6x	ـعـ	ـڈ	ـبـ	چ	ـد	ـۀ	ـفـ	ـجـ
	ـمـ		ژ	ک	ـا	ـہـ	ـنـ	
"7x	ـپـ	ـقـ	ـر	ـسـ	ـتـ	ؤ	ـقـ	و
	ؤ	ـيـ	ـز	ـر	ـئـ	ـتـ	ۀ	
"8x	ش	ـشـ	ش	ص	ـصـ	ص	ض	ـضـ
	ض	ط	ـط	ط	ظ	ظ	ـظ	ع
"9x	عـ	ع	غ	ـغـ	غ	ف	ـفـ	ف
	ق	ـقـ	ق	ك	ک	ك	ل	لـ
"Ax	ل	مـ	ـمـ	م	ن	نـ	ن	ه
	هـ	ـهـ	ـه	ـهـ	و	ى	يـ	
"Bx	ى	ي	ي	ڤ	ـڤـ	ڤ	پ	ـپ
	پ	چ	ـچ	چ	ژ	گ	گ	گ
"Cx	ڭ	ـڭ	ڭ	ڭ	ٹ	ـڈ	ـٹـ	ٹ
	ڈ	ـڈ	ڑ	ـڑ	ں	ں	ے	لا
"Dx	ﻼ	ﻸ	ﻸ	ب	ـد	ـبـ	ب	ف
	ـه	ـمـ	ـى	و	ق	ـمـ	ف	ق
"Ex	ـه	ـد	ـبـ	ـتـ	ـثـ	ـنـ	ـيـ	ـپـ
	ـڈ	ـد	ـبـ	ـتـ	ـثـ	ـذ	ـيـ	ـپـ
"Fx	ـڈ	ـجـ	ـحـ	ـخـ	ـچـ	ـر	ـز	ـژ
	آ	آ	لآ	ﻶ	ک	ک	ن	ن
	"x8	"x9	"xA	"xB	"xC	"xD	"xE	"xF

Table 1: Table of the amal0-10 font (Basic glyphs).

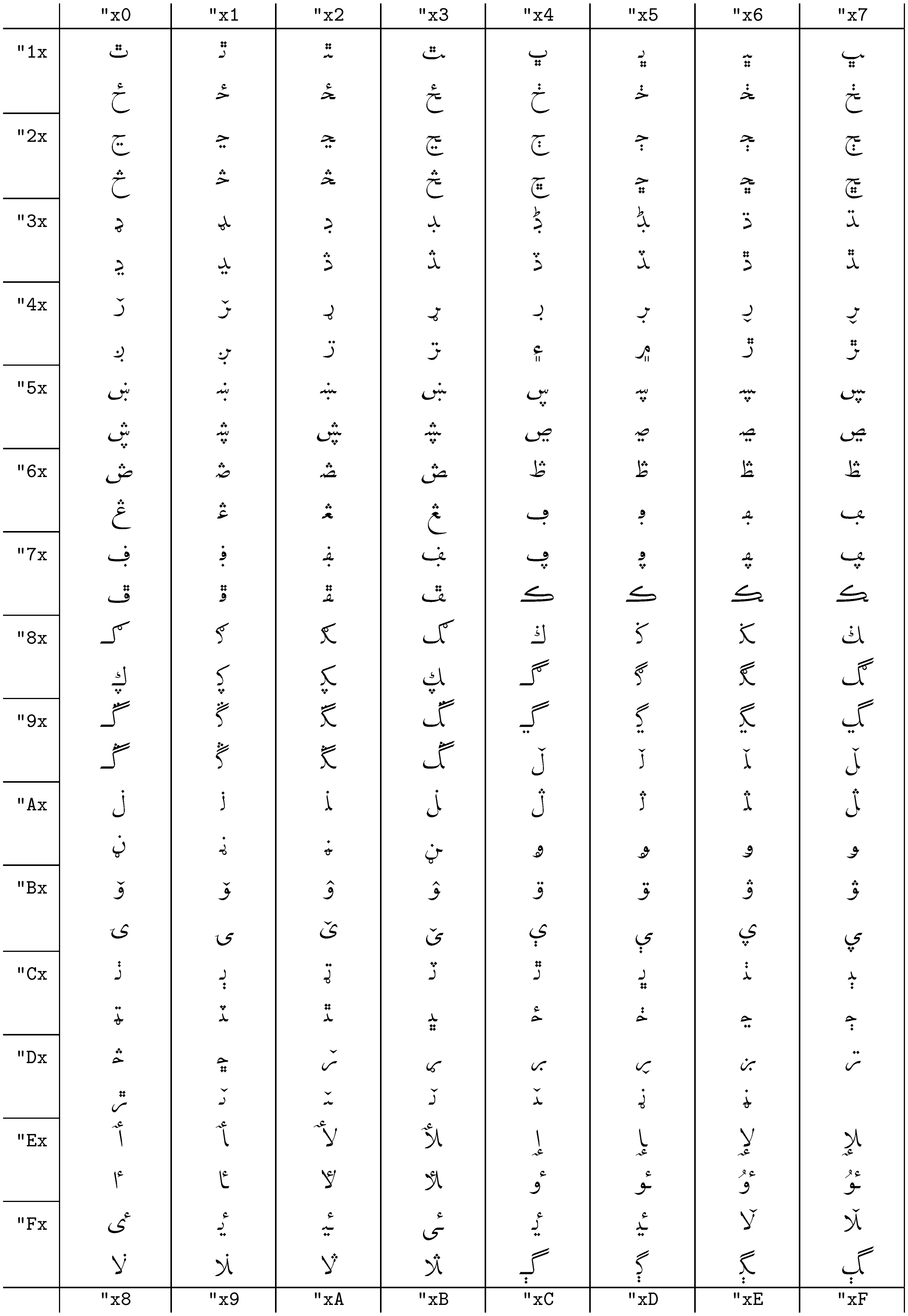

"x0
"x1
"x2
"x3
"x4
"x5
"x6
"x7
"1x
"2x
"3x
"4x
"5x
"6x
"7x
"8x
"9x
"Ax
"Bx
"Cx
"Dx
"Ex
"Fx
"x8
"x9
"xA
"xB
"xC
"xD
"xE
"xF

Table 2: Table of the `amal1-10` font (Ligatures I).

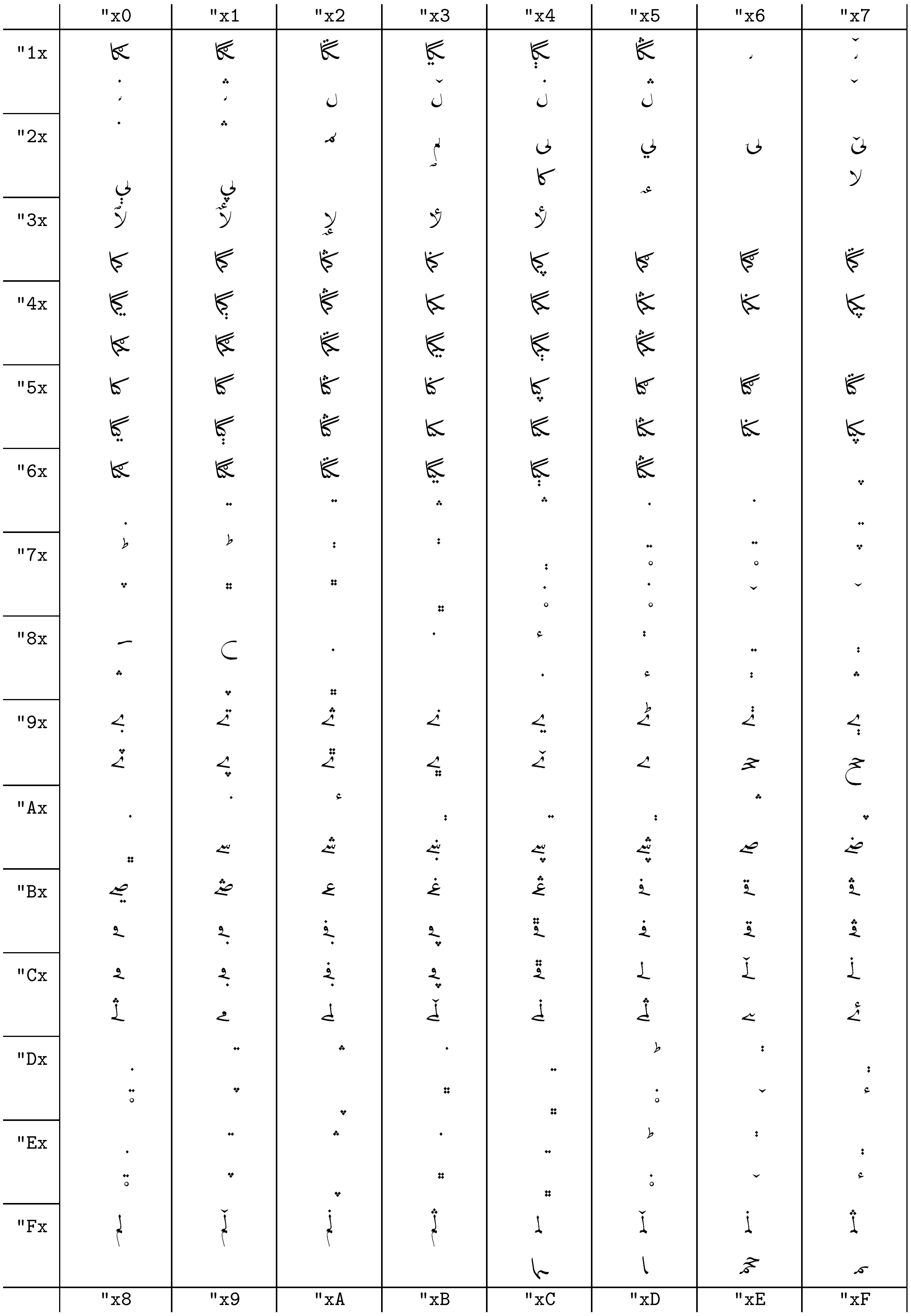

"x0
"x1
"x2
"x3
"x4
"x5
"x6
"x7
"1x
"2x
"3x
"4x
"5x
"6x
"7x
"8x
"9x
"Ax
"Bx
"Cx
"Dx
"Ex
"Fx
"x8
"x9
"xA
"xB
"xC
"xD
"xE
"xF

Table 3: Table of the `amal2-10` font (Ligatures II).

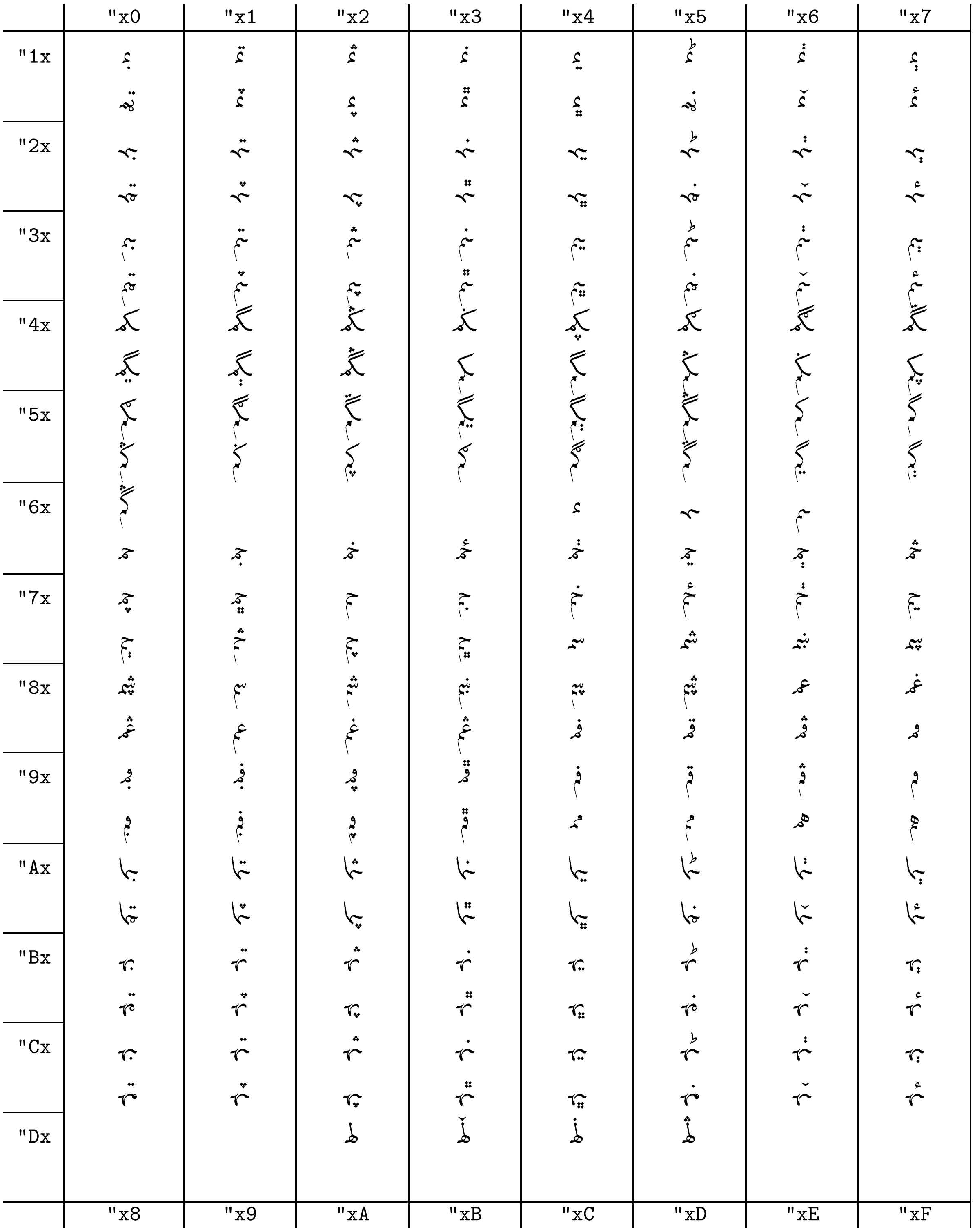

"x0
"x1
"x2
"x3
"x4
"x5
"x6
"x7
"1x
"2x
"3x
"4x
"5x
"6x
"7x
"8x
"9x
"Ax
"Bx
"Cx
"Dx
"x8
"x9
"xA
"xB
"xC
"xD
"xE
"xF

Table 4: Table of the `amal3-10` font (Ligatures III).

	"x0	"x1	"x2	"x3	"x4	"x5	"x6	"x7
"1x	·	ط	˘	°	·	˘	:	¨
	ۛ	⁘			لی	لٚی	لی	ڷی
"2x	بں	تں	ثں	نں	یں	ٹں	تں	پں
	ٿں	پں	ٿں	ییں	بر	ئں	ں	
"3x	·	ط	·	ۛ				
	بی	تی	° ثی	نی	یی	ٹی	تی	پی
"4x	ٿی	پی	ٿی	پی	ٮٚی	ئی	ی	
	بی	تی	ثی	نی	یی	ٹی	تی	پی
"5x	ٿی	پی	ٿی	پی	ٮٚی	ئی	ی	
	سی	شی	ښی	ڛی	ڜی	سی	شی	ښی
"6x	ڛی	ڜی	صی	ضی	صی	ضی	صی	ضی
	ڝی	ضی	فی	قی	ڨی	ٯی	ڢی	ڣی
"7x	ڥی	ڦی	فی	قی	ڨی	ٯی	ڢی	ڣی
	ڥی	ڦی	کی	گی	ڳی	کی	ڮی	ګی
"8x	ڳی	گی	ڮی	ڲی	ڴی	کی	گی	ڴی
	کی	ڮی	ګی	گی	گی	ڲی	ڲی	ڴی
"9x	¨	˘	:	ۛ	ء			
		˘			ء			
	"x8	"x9	"xA	"xB	"xC	"xD	"xE	"xF

Table 5: Table of the `amal4-10` font (Ligatures IV).

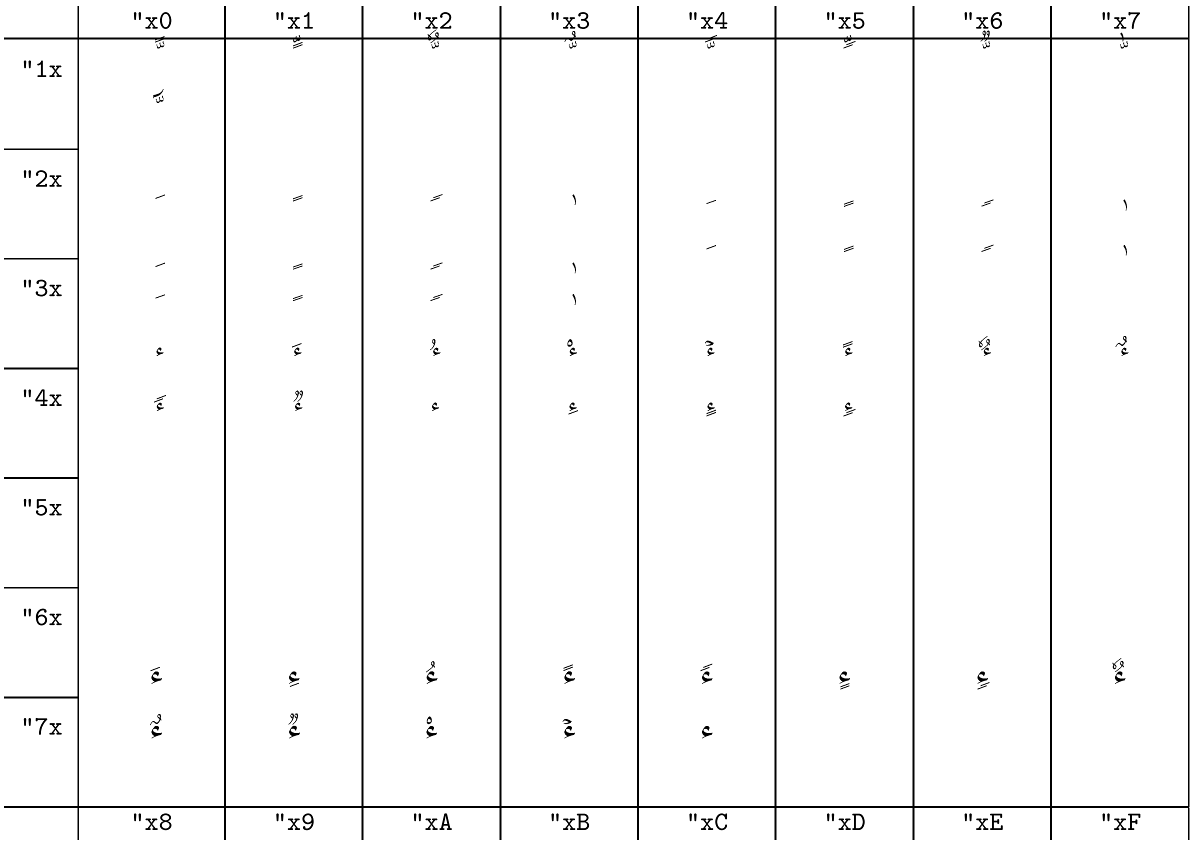


Table 6: Table of the `amalf-10` font (Vowels and diacritics).

A Medieval Icelandic manuscript[*] The making of a diplomatic edition

Andrea de Leeuw van Weenen

In November 1993 my edition of the Icelandic Homily Book[1] was published by the Stofnun Árna Magnússonar á Íslandi (SAM)[2] after having been 'in press' for a period of 19 years. If it had not been for TeX, this period might easily have been extended indefinitely. Looking back, work on the Icelandic Homily Book can be divided into three stages: the scholarly work, the attempts at printing before TeX, and the typesetting with TeX.

1 The scholarly work

My involvement with this edition, or with Old Icelandic scholarship in general, came about almost by accident. I arrived in Iceland in 1971 with my husband, who had taken a temporary job at the University of Reykjavík, and my two small sons; my knowledge of Icelandic at that time could easily find place in half a column of this journal. In order to escape the drudgery of diaper laundry I enrolled in the "Icelandic for foreigners" program at the university (Háskóli Íslands), where I became enthralled in my second year by the secrets of paleography and Old Icelandic grammar. So when I had passed my examination for the Bacc. Phil. Islandicae degree, I looked around for something useful in that direction to occupy me in my third and final year in Iceland.

A suggestion by Helgi Guðmundsson, associate professor of Icelandic at the University of Reykjavík, to write a doctoral thesis and to choose an edition with a thorough grammatical analysis as the topic did not strike me as a realistic option. I had majored in mathematics, so would have to go a long way before getting to a doctorate in a completely different field. Nevertheless, he insisted that shortcuts could be found and that preparing such an edition while I had the right resources was a sensible thing. Although I did not believe him at the time, he turned out to be right. Anyway, I let myself be talked into this undertaking and after some consultations with the SAM, I choose the Icelandic Homily Book from the three or four manuscripts that the institute and Helgi deemed suitable and most urgent.

Apart from some fragments, the Icelandic Homily Book is the oldest extant Old Icelandic manuscript, dating from around 1200 and containing on its 102 parchment leaves (204 pages) some 60 sermons. By its age alone, this manuscript is of the greatest interest for the study of the Old Icelandic language; but it is also considered to be an example of good style.

Work on the transcription started in the summer of 1973. At first I worked from a set of photographs, later I was able to use the manuscript itself.[3] After the first year the transcription with the critical apparatus was finished, and the introduction, which was going to concentrate on orthography and morphology, was well under way. Meanwhile, the staff at the institute had been keeping an eye on my work, and had offered to publish the edition in one of their series as a combined facsimile[4] and diplomatic edition.[5] I gladly accepted their offer, but should perhaps have sensed the problems that would develop afterwards when the project meeting was nearly exclusively devoted to the choice of paper, rather than to editorial principles, deadlines, special requirements, and the like.

I left Iceland in 1974 with the promise that typesetting the transcription would start next week. Famous last words. During the next two years I finished writing the introduction and fulfilled the requirements of the University of Utrecht for a masters degree in Old Germanics. As typesetting in Iceland still had not started, I typed the introduction, pasted the needed corrections into the transcription and handed the thesis in as typescript, thinking that it might well be several more years before the book got printed; but I never suspected that it would take 17 more years — or that I would have to be my own typesetter.

[*] An earlier version of this article appeared in MAPS 14 (1995) pp. 31–34.

[1] Andrea de Leeuw van Weenen, ed., *The Icelandic Homily Book, Perg. 15 4^o in the Royal Library, Stockholm*, Íslensk Handrit/Icelandic Manuscripts, Series in quarto vol. III, Stofnun Árna Magnússonar á Íslandi, Reykjavík 1993. pp. 436 + 204 plates.

[2] This institute in Reykjavík keeps most of the existing Icelandic manuscripts and is devoted to their study and publication.

[3] The Royal Library loaned the manuscript to Iceland and later kindly granted permission for it to be taken out of its binding to be photographed for the facsimile edition.

[4] This required photographing the whole manuscript. For this it had to be taken out of its binding. This was done in 1975. The delay in printing had also delayed the rebinding. In fact, I found that, in February 1996, the manuscript was still in loose pages.

[5] The edition is a diplomatic one. That is, it aims at reproducing the manuscript. In a diplomatic edition of the strictest kind, even the abbreviation marks would be reproduced. Here the abbreviations are expanded in italics. The facing facsimile allows the reader to observe the originals of such abbreviation marks.

2 Typesetting, the years before TeX

In 1974 all typesetting on Iceland was still done in lead. The transcription for my project required a number of unusual characters, and it turned out that not only did the typesetting firms not have these characters, they did not exist in the Monotype catalogue. So they would have to be specially cut for this publication. The various firms that were approached were understandably reluctant to invest in this, as there was no guarantee that the characters could be used for other books. These negotiations took several years — in the small Icelandic community, firms could be approached only one at a time, and most took their time to think the proposition over. In 1979 the news came that one firm had purchased phototypesetting machinery of the matrix variety and that they were willing to start work on the transcription. Slowly, the proofs started to come. But with them came a surprise.

I had believed that proofreading would be my responsibility, but now I found that proofs of books to be published by the Stofnun Árna Magnússonar[6] were habitually read by three independent readers — the editor of the edition, one of the senior staff members[7] and a junior staff member. Additionally, the proofreading did not mean comparing the proofs with the typescript but with the manuscript or the photographs, thereby checking not only the work of the typesetter but also that of the editor. This meant that proofreading took quite some time. For the SAM staff, it was one of the many jobs they had to do besides their own research. When we disagreed about a reading there were lengthy discussions by mail, which usually were only resolved during one of my visits to Iceland. So, when in 1983, we were finally in agreement about the corrections to be made and sent the corrected proofs to the typesetter, it was a very unpleasant surprise when we were told that he had just got himself a new phototypesetting machine and could not convert the material he had on punch tapes to this new machine. But he would have the thing typeset anew as soon as he could. In the end this took a year.

And so the whole circus started again in the autumn of 1984: proofreading in triplicate. There were fewer cases to discuss between us three; on the other hand, the work went a lot slower. I was both in the final stage of another project and taking up a new job which required a lot of reading up — if there had ever been any feeling of urgency about the book in Iceland, that had now certainly gone. So it was early 1989 when the marked proofs were returned for the second time to the typesetter. But when I arrived in Reykjavík some months later, I found that the machinery had again been replaced and that the typesetter was planning to start from scratch. Again.

By this time I had about 10 years' experience with computers and I was quite sure that conversion was possible. Moreover, I had at some stage requested and obtained copies of the typesetting files. Admittedly, it had not been easy to decipher these,[8] but I had copies on DOS disks of the original files and conversions (via a SNOBOL4 program) of these files to ASCII, where the typesetting codes had been removed. At this stage the Stofnun Árna Magnússonar was as opposed as I was to going through the whole troublesome procedure again — it was becoming clear to us that, with the methods of the institute, we would always be limping behind the continuous advances in technology.

So the disks were sent to Iceland and in due course of time new proofs arrived. But after the initial joy that conversion to the new machine was possible, a closer look brought great disappointment. The font used looked decidedly irregular and the kerning of the high 's' (ſ) was absolutely ugly. But even worse, many errors had crept in. A systematic study of the errors identified on the first few pages brought me to the conclusion, later confirmed: a conversion program had been written, and where it produced erroneous results, rather than correcting and rerunning the program, they had opted for manual correction of the output file, but such corrections had not been carried out very systematically.

Considering the state of affairs and the possibilities for correcting the files, I decided that the best thing would be to get my hands on their files and repair them by comparison with mine. As this required only a physical conversion to DOS disks, it seemed easy enough. Unfortunately, this could not be done in Iceland, but had to be handled in Denmark by the manufacturer of the machinery, and after some phoning and explaining, two disks arrived, which were not too difficult to decipher. As soon as I had corrected a couple of pages, I returned the disk, and waited with some optimism for a corrected proof. No such thing — only a panicky fax that the disk could not be read. Some weeks of multilateral discussion followed between the institute and the typesetter in Reykjavík, the

[6] Det Arnamagnæanske Institut in Copenhagen follows the same policy.

[7] In my case, Stefán Karlsson, now director of the Institute and professor at the University of Iceland.

[8] The original 8-inch disks were written in a proprietary format. It required the help of a specialized publishing house (Brill) to convert the files into DOS format.

technical staff of the manufacturer of the typesetting machine in Denmark, and myself in Leiden (the Netherlands). This discussion was not made any easier by the lack of a common language. In the end, it became clear that the lack of expertise on the Icelandic end, combined with the distances involved, made it highly unlikely that the problem would ever be solved.

By this time, 1990, I had gained some experience with TEX and METAFONT, enough at least to be confident that the job could be done, and luckily *not* enough to foresee all the problems involved. Moreover, I had already keyed in the apparatus, together with all the points raised in connection with them in 10 years of correspondence, and even made a few METAFONT characters needed there. So I wrote a letter to Iceland enumerating the possibilities open to us; these ranged from typesetting from scratch (for the third time) via various methods involving conversion to the new machine, to doing it myself with TEX. I outlined the disadvantages and advantages and the fact that, in my opinion, some methods were so impractical and relied so much on factors beyond our control that I was not willing to cooperate in them. Probably the members of the staff of the Stofnun Árna Magnússonar were then about as fed up with the whole thing as I was, so they agreed that I should have a go with TEX.

The edition had now been 'in press' for 15 years. This time was rather evenly divided in three periods: trying to find a suitable typesetting firm/system (1974–1979), first phototypesetting system (1979–1983), second phototypesetting system (1983–1989).

3 Typesetting, the years with TEX

3.1 Picking up the pieces

Apart from the transcription, which by now had gone through two failed typesetting attempts, the book was also to have an extensive introduction (215 pages in the finished product). Again the transcription was tackled first, and this time on the base of the machine-readable version produced in the second attempt. This had been converted to a simple ASCII-based encoding scheme of my own devising and had been used for searching, concordancing, etc. The features of LATEX were irrelevant for this part, so plain TEX was used to produce this part. As I had to write a conversion program[9] anyway to convert the files in my code to TEX files, I could easily include line numbering as well. The apparatus existed already in TEX form and needed only minor corrections, so the relevant part could be inserted after each page automatically. And so, while EDMAC was available, it was not considered.

Proofreading of the transcription could be minimal, as the original ASCII files had been corrected and only the correctness of the conversion and the working of the TEX macros needed to be checked.

In the previous stages, no attempt at typesetting the introduction had been made. Over the years I had had serious discussions with Stefán Karlsson about the arrangement of some of the quires. As this involved drawing and redrawing the figures depicting those quires, I had at some stage done the necessary drawings within LATEX's picture environment. As a result I adopted LATEX for the production of the introduction; however, neither LATEX's book style nor the NTG's `boek` style were to the liking of the institute. I therefore had to write my own style file — or rather, to fiddle with `boek.sty` and its attached files to get the required results. A small surprise was having to define a new `strutbox`, as the normal one suits 10pt text only.

Some of the many tables in the introduction would only fit in landscape. As they all required a full page, I took the easy way out and produced them separately.

The introduction required even more special characters than the transcription, as the various abbreviation marks, which are expanded in italics in the transcription, have to be represented. On the other hand, it had been decided already in 1974 that the survey of the characters occurring in the manuscript should contain drawings by hand of the various characters and their variants. Here small gaps were to be left, to be filled in by hand in the final 1270dpi copy.

3.2 Design constraints

The edition of the Icelandic Homily Book had two components: the introduction (written after the body of the text had been produced), and the text itself. For the text, there were to be a series of notes at the bottoms of pages (the critical and paleological apparatus), line numbers in the left margins, occasional margin notes in the right, complex font combinations throughout, and forced line and page breaks. The introduction would be different in structure, with section headings at various levels, footnotes, tables, and numerous citations from the text. It also had a preface, table of contents, and a bibliography.

The book had to appear in a series and was planned as a combined facsimile and diplomatic edition, with photographs and transcription on facing

[9] Again, the conversion was handled by a SNOBOL4 program.

pages (see illustrations). This meant that both page breaks and line breaks in the transcription were predetermined by the manuscript, not by the software. As well, presentation of the manuscript required a large paper size (30.3 × 23.3mm), which in turn meant using a 12-point font. It came therefore as an unpleasant surprise that the Computer Modern fonts which I wanted to use were significantly wider than the fonts used previously, and, more to the point, that the resulting lines did not fit the given page width. After much hesitation I decided to decrease the width of the characters by about 10%. As the line breaks are determined by the manuscript, I could have set the `\hsize` to a rather arbitrary large value had it not been for the biblical references which occasionally had to appear in the right margin. Setting `\hsize` to 175mm and setting the references flush right in the line resulted in only one or two places where line and reference clashed. In these cases a solution was found by moving the reference one line down.

The paperheight too was not unproblematic. Some manuscript pages had many more lines than others, and there was a critical apparatus that also had to be accommodated as a whole at the foot of the page and could not be allowed to float to the next page. If I chose a page height that would fit all pages, the majority would look ugly, as they would have far too large a gap between text and apparatus. So after some experiments I choose a page length that fitted most pages with the apparatus at the bottom of the page. The few overlong pages had a special page height and the apparatus directly following the text.

After some experimentation `\vsize` was set to 270mm and pages arranged as follows: first a headline containing the folio number (this is suppressed in the illustration), then the body of the text, then the apparatus part. For normal pages this took the form: `\vfill`, text of the apparatus, and finally 46.8mm vertical white space. For overlong pages, the apparatus followed the text after a 2.6pt gap and was followed by a `\vfill`. In both cases the apparatus was printed with a linewidth of 145mm.

TEX's habit of stretching and shrinking spaces, much as I value it elsewhere, did not improve the readability of this text, so I disabled it by redefining `\fontdimension`'s 2, 3, and 4 for all relevant fonts (roman, italic, bold, and bold italic). As well, the distances between the lines were set to a fixed value.

Otherwise, the style file for the transcription consisted only of macros to arrange the fonts at various sizes into families, and shorthands for the special characters. All other coding, for example, the `\llap` for the line numbers, were put directly into the TEX files by the conversion program.

3.3 Font issues

The next problem concerned the special characters that had caused us difficulties right from the beginning: ſ aꝛ ǫ ꜵ, to name a few, and of course þ. The latter could be taken from the Icelandic font, but the others had to be made with METAFONT. Some were easily constructed: the high 's' (ſ) only required removing the horizontal stroke from the 'f', and of course the introduction of quite a few new ligatures. Others, however, required adding a diacritic to a character: ǫ, ỏ, ǫ̉, ỏ́. Still others required more METAFONT skills: aꝛ or Ǝ. I must stress that I was, and am, far from mastering METAFONT, and I remember with embarrassment the time that I had produced a version of an 'o' with a squiggle (ǫ) that looked acceptable in isolation, but different when inserted in a font. Only when I made a test font with just two copies of this character did I realise that the 'o' was drawn with the pen inherited from the previous character, and that I had introduced a smaller pen for the tail part.

Finally, I made roman, italic, bold, and bold italic fonts which consisted of the same characters as their cm counterparts, minus the Greek letters, but with the addition of the special characters and the small capitals needed (see below). In the part of the fonts taken over from the cm fonts I made a small change to the character ø — not to its shape, but to its height. The height of this cm character is the height not of the 'o', but of the diagonal stroke. This results in the accent above ø standing higher than that over 'o': ǿ ó. By reducing the height of the ø to the height of 'o', the accents come at the same height: ǿ ó.

The various fonts were produced in a 300dpi version for proofreading and a 1270 dpi version for the final production. The parameetrs were taken from the cm fonts as well, apart from the necessary adaptation mentioned above regarding the width of characters.

Besides the problem of designing special characters, there were also problems with the integration of various typefaces. Due to the diplomatic nature of the transcription, roman characters, italics, and small capitals can occur within a single word, and this poses problems.

The transcription follows the manuscript in its use of small capitals, which are employed to indicate double consonants.[10] Normally smallcaps are larger

[10] This was one of the methods used by medieval Icelandic scribes to put as much text as possible on the expensive

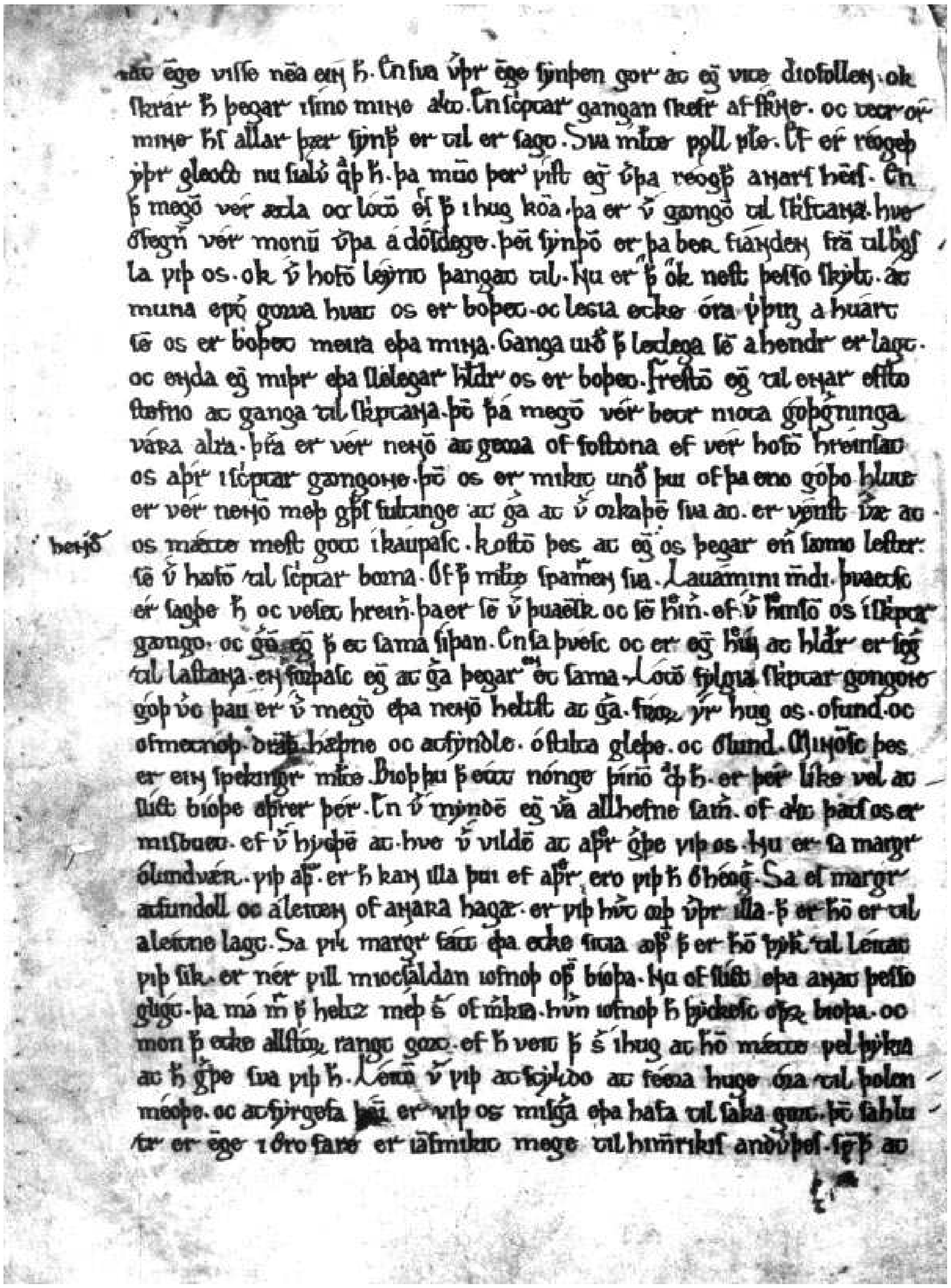

aþ ēge viſſe nēa eiṅ h. En ſua v́þr ēge ſynþen gor at eḡ vitt dioꝼollen. ok
ſkrar h þegar iſmo mine aḱo. En ſcptar gangan ſkeſt aſ ſkine. oc veir oꝼ
mine hſ allar þær ſynþ er til er ſago. Sua mlte poll ꝑlo. Eꝼ eꝼ régeþ
ẏþr gleoꝺ nu ſialꝟ q̄þ h. þa muo þer ꝟiſt eḡ v̄þa reogþ aнarſ heīſ. En
þ megō vér æla oc lotō eſ þ ihug kōa. þa er v̄ gangō til ſkſtana. hve
ōſegn̄ vér monū v̄þa á dōīdege. þeī ſynþō er þa bera ſiānden ſrā til bǵſ
la viþ os. ok v̄ hoſō leynto þangat til. hu er þ ōk neſt þeſſo ſkylt. at
muna epṓ gorua huat os er boþet. oc leſta ecke óra v́þin a huart
ſē os er boþet meira eþa minna. Ganga unð þ loꝺega ſē a hendr er lagt.
oc enda eḡ miþr eþa ſleiegar hldr os er boþet. ſreſtō eḡ til eнar eſtto
ſteſno at ganga til ſkptaнa. þō þa megō vér betr niota goþḡninga
vára alra. þra er vér neнō at gera oꝼ ſoſtona eꝼ vér hoſō hreinſat
os aþr iſcptar gangone. þō os er mikit unð þui oꝼ þa eno goþo hlute
er vér neнō meþ gþſ ſultinge at ga at v̄ ōrkaþē ſua at. er venſt ꝟæ at
· neнō os mætte meſt gott ikaupaſc. koſtō þes at eḡ os þegar en̄ ſamo leſter.
ſē v̄ hoſō til ſcptar borna. Eꝼ þ mlte ſpamen ſua. Lauamini m̄di. þuaeſc
er ſagþe h oc veſet hrein̄. þaer ſē v̄ þuaeſk oc ſē hin̄. eꝼ v̄ hinſō os iſkptar
gango. oc ḡā eḡ þ et ſama ſiþan. En ſa þueſc oc er eḡ hin̄ at hldr er ſeḡ
til laſtaнa. en ſnþaſc eḡ at ḡa þegar et ſama. Lotō ſylgia ſkptar gongoꝛ
goþ v́c þau er v̄ megō eþa neнō heltſt at ḡa. ſrar v́r hug os. oſund. oc
oſmetnoþ. drāp. hæþne oc atſyndle. óſtilta gleþe. oc ōlund. Minnōſc þes
er ein ſpekingr mlte. Bioþ þu þ oðrū nónge þinō q̄þ h. er þer líke vel at
ſliſt bioþe aþrer þér. En v̄ mynde eḡ vā allhefne ſam̄. oꝼ alt þætſ os er
miſbuen. eꝼ v̄ hyḡþē at. hve v̄ vildē at aþr ḡþe viþ os. hu er ſa margr
ōlundvǽr. viþ aþ̄. er h kan illa þui eꝼ aþr ero viþ h ōhǽgǩ. Sa eſ margr
atſundoll oc áleiten oꝼ aнaʀa hagar. er viþ hv́t meþ v̄þr illa. þ er hō er til
aleitne lagt. Sa viſ margr ſatt eþa ecke ſtua aþ þ er hō þyk̄ til leitat
viþ ſik. er nér viſl mioc ſialdan iofnoþ oþ̄ bioþa. hu eꝼ ſliſt eþa aнat þeſſo
glígt. þa má m̄ þ helz meþ s̄ oꝼ m̄kia. hv́n iofnoþ h hyckeſc oþꝛ bioþa. oc
mon þ ecke allſtor rangt gert. eꝼ h veit þ s̄ ihug at hō mætte vel þykia
at h ḡþe ſua viþ h. Lerā v̄ viþ at ſkyldo at ſema huge óra til þolen
móþe. oc at ſyrgeſa þeī er viþ os miſgā eþa haſa til ſaka gert. þō ſahl
tr er ēge iōro ſare er iaſmikit mege til himrikiſ andv̄þeſ. ſeḡ þ at

Figure 1: Photograph

at e*n*ge viſſe ne*m*a éiN h*aN*. En ſva ver*þ*r e*n*ge ſynþen gor at e*i*g*e* vite diofolleN. ok
ſkrár h*aN* þegar iſíno miNe aLt. En ſcr*i*ptar gangan ſkefr af ſkr*ó*Ne. oc tecr ór
miNe h*an*ſ allar þær ſynþ*er* er til er ſagt. Sva m*ǽ*lte pǫll p*oſt*ole. Ef ér réogeþ I Cor 11.31
yþr gleoct nu ſiálv*er* q*uaþ* h*aN*. þa mu*n*o þer víſt e*i*g*e* ver*þ*a réogþ*er* aNarſ hei*m*ſ. En
107 þ*at* mego*m* vér ætla oc lóto*m* | ø\ſ/ſ þ*at* i hug ko*m*a. þa er *vér* gængo*m* til ſkr*i*ftaNa. hve
ðfegn*er* vér monu*m* ver*þ*a á dó*m*ſdege. þei*m* ſynþo*m* er þa beR fiáNdeN fra*m* til br*i*gſ-
la viþ os. ok *vér* hofo*m* léynt þangat til. Nu er ok þ*at* neſt þeſſo ſkylt. at
muna ept*er* gorva hvat os er boþet. oc leGia etke óra v*ir*þing a huárt
ſe*m* os er boþet meira eþa miNa. Ganga uNd*er* þ*at* letlega ſe*m* a hendr er lagt.
oc eNda e*i*g*e* miþr eþa ſlélegar heldr os er boþet. Freſto*m* e*i*g*e* til eNar eſſto
ſtefno at ganga til ſkr*i*ptaNa. þ*ui*at þá mego*m* vér betr niota góþg*er*ninga
váRa alra. þ*eir*ra er vér neNo*m* at geora of foſtona ef vér hofo*m* hréinſat
os áþr i ſcr*i*ptar gængoNe. þ*ui*at os er mikit und*er* þui of þa eno góþo hlute
er vér neNo*m* meþ g*u*þſ fultinge at g*e*ra at *vér* orkaþe*m* ſva at. er vę́nſt v*ǽ*re at
os mǽtte meſt gott í káupaſc. Koſto*m* þes at e*i*g*e* /heNde\ os þegar en*er* ſæmo leſter.
ſe*m* *vér* hæfo*m* til ſcr*i*ptar borna. Of þ*at* mælte ſpam*aþ*reN ſva. Lauámini mu*n*di. Þvaetſc Is 1.16
ér ſagþe h*aN* oc veſet hrein*er*. Þa er ſe*m* *vér* þuaem*ſ*k oc ſe*m* hrein*er*. ef *vér* hr*e*inſo*m* os íſkr*i*ptar
gængo. oc g*e*ro*m* e*i*g*e* þ*at* et ſama ſíþan. En ſa þvéſc oc er e*i*g*e* hr*é*iN at heldr er ſeg*er*
til laſtaNa. eN forþaſc e*i*g*e* at g*e*ra þegar \eN/ et ſama. Lóto*m* fylgia ſkr*i*ptar gongoNe
góþ v*er*c þau er *vér* mego*m* eþa neNo*m* heltſt at g*e*ra. ſnor*um* ýr hug os. ofund. oc
ofmetnóþ. dra*m*b. hǽþne oc atfyndle. ó ſtilta gleþe. oc ðlund. MiNo*m*ſc þes
er eiN ſpekingr m*æ*lte. Bioþ þu þ*at* éitt nónge þíno*m* q*uaþ* h*aN*. er þér líke vel at -
ſlíct bióþe aþrer þér. En *vér* mynde*m* e*i*g*e* v*er*a allhefne ſam*er*. of aLt þatſ os er
miſbuet. ef *vér* hyGþe*m* at. hve *vér* vilde*m* at aþrer g*er*þe viþ os. Nu er ſa margr
ólundvǽR. viþ aþ*ra*. er h*aN* kaN illa þui ef aþ*rer* ero viþ h*aN* ð héog*er*. Sa eſ margr
atfundoll oc á leiteN of aNaRa hagæ. er viþ hv*er*t orþ v*er*þr illa. þ*at* er h*onom* er til
a léitne lagt. Sa viL margr fátt eþa etke ſitia æþ*rom* þ*at* er h*onom* þyk*er* til léitat
viþ ſik. er nér vill mioc ſ\i/aldan iofnoþ oþ*rom* bióþa. Nu of ſlíct eþa aNat þeſſo
glígt. þa má m*aþr* þ*at* heltz meþ *sér* of m*er*kia. hv*er*n iofnoþ h*aN* þyckeſc oþr*um* bioþa. oc
mon þ*at* etke allſtór*um* rangt gort. ef h*aN* veit þ*at* *sér* ihug at h*onom* mǽtte vel þykia
at h*aN* g*er*þe ſva viþ h*aN*. Léito*m* *vér* viþ at ſcyLdo at féora huge óra til þolen-
méoþe. oc at fyrgefa þéi*m* er viþ os miſg*e*ra eþa hafa til ſaka gort. þ*ui*at ſa hlu-
tr er e*n*ge i ðro fare er ia*m*ſmikit mege til hime*n*rikiſ andv*ir*þeſ. ſe*m* þ*at* at

2 fkr*ó*Ne] ɔ: ſkr*ó*Ne 7 ok þ*at*] *T*. 10 heldr] *Add* en (*Vr*). 11 þá] < þ*at*. 20 ſnor*um*] n < v, *correction indicated by an accent*. 21 ofmetnóþ] *Accent suspect, L* ofmetnoþ. 26 hagæ] æ < a. 28 nér] *Del* (*SK*). 31 féora] o < e.

Figure 2: Transcription

than the corresponding romans, as can be seen when I use this strategy for English and write coRect for 'correct', plaNing for 'planning'. This makes the page look very jumpy, so I scaled down the small-caps. However, this was not completely successful. Even with the large number of parameters for the cm fonts, there seemed to be a relationship between the thickness of various strokes. I feel that a small capital that has to fit within a word should be parameterized in a different way, but for that task I lacked the time.

The transcription also has italics and romans mixed within words. I had thought that the italic correction would take care of that problem, but it did not. So I had to figure out experimentally the amount of kerning needed for each pair of roman-italic and italic-roman that occurred. Again, this can certainly be improved upon by someone with a designer's eye. I can only say that this kerning is a great improvement over the results without the kerning. As the TEX files for the transcription pages were produced by a conversion program, these explicit kernings had already been automatically inserted.

And while all of this had been resolved for the actual transcription pages, the introduction still only existed as a typescript and contained thousands of quoted words from the transcription. I was not looking forward to typing in all those explicit kernings, so I decided to solve the kerning problem by combining romans and italics in a single font while taking care of the kernings in the ligature tables. The roman and italic smallcaps which occurred within the transcription were placed in this same font. This arrangement meant that italics could not be accessed by the usual `\it` command, but via macros: `\ia` for italic-a, and so on. Since at most only one or two consecutive italic characters occur, this made the typing not too onerous.

4 Conclusion

The flexibility of TEX and METAFONT have made it possible to produce a publication which might otherwise never have made it to the printing press, as another typesetting + proofreading cycle would probable have taken even longer and left us even further behind in the technology race.

Looking back, the first typesetting and its proofreading were completely wasted. The second typesetting was not, although in a very round-about way, since it yielded a computer-readable, and after further conversions, TEXable text. Nor was the second proofreading a waste, since all errors spotted were corrected in the machine-readable text.

I found it possible for somebody who is far from being a TEXpert to produce this rather complicated edition with the support of the ever helpful TEX community. In particular, I would like to thank Kees van der Laan and Piet van Oostrum, who were both very helpful, providing me not only with TEX tricks but also with their explanations. The final result looks far better than two of the attempts before TEX, and certainly as good as the third.

If I were to be confronted with the same problem now, I would certainly opt for TEX again. Also, many of the other decisions would be taken in essentially the same way. The only decision I might reconsider is the choice of LATEX for the introduction. I found the relations between the macros of LATEX itself and its various style files hard to understand. It was therefore difficult to achieve the requested design and format changes.

The example of the Homily Book has served to persuade a number of colleagues that TEX can rescue their work as well. In most of those cases a book has been produced with a word-processing program, the publisher wants camera-ready copy, but the requirements of the publisher (and sometimes even that of simple readability, or of the conventions in the specific field) cannot be met by the word-processing program. Up to now TEX has always provided the necessary functionality.

◇ Andrea de Leeuw van Weenen
Department of Comparative Linguistics
Postbus 9515
2300 RA Leiden
LeeuwvW@rullet.LeidenUniv.nl

parchment. Another was the frequent use of abbreviation signs.

Book Review

Book review: *Writing With TEX* and *TEX & LATEX: Drawing & Literate Programming*, by Eitan Gurari

Michael D. Sofka

Eitan M. Gurari, *Writing with TEX*. McGraw-Hill, New York, 1994, ISBN 0-07-025207-6.

Eitan M. Gurari, *TEX & LATEX: Drawing & Literate Programming*. McGraw-Hill, New York, 1994, ISBN 0-07-911616-7.

In this pair of books Gurari has written primers for using TEX and LATEX for writing, programming and drawing figures. The books are short and to the point, with most commands being introduced and explained in one or two paragraphs. If you are new to TEX, or do not like to experiment, these books may not be for you. If, on the other hand, you are looking for a short and comprehensive review of TEX, or if you want to draw figures and write programs in TEX, Gurari is a good choice.

Writing With TEX is a TEX reference manual. It starts with a three page *Getting Started*[1] chapter which explains how to run TEX. Most of the remaining book is divided into two parts. Chapters 2–10 cover *TEX as a Formatting Language.* This includes simple text, *Fonts*, *Layout of Pages* and *Mathematical Formulas* (each in five pages!), inserts, boxes, and basic programming constructs such as *Groups* and scanning. The concepts and commands introduced, however, are only the essentials needed for formatting.

Chapters 11–16 is called *TEX as a Programming Language.* It covers *Macros*, *Data Types*, *Selectors* and *Auxiliary Files* (file I/O), each of which is a basic programming unit. These chapters are a little longer, averaging ten pages each. The programming is rounded out with two chapters covering macros and character codes in more detail. This includes `\let`, `\edef`, `\csname`...`\endcsname`, token expansion, active characters, changing character codes and other subtleties of writing TEX macros.

Chapter 17 is titled *Environments For Writing.* Macros are described for basic document structures such as `\Chapter` and `\Section`, as well as environment blocks, lists, cross-references, table of contents, figures and bibliographies. LATEX[2] is introduced at the end of this chapter as an example style library.

The book ends with eight appendices. The first six are divided by tasks and cover TEX commands in more detail. Tables and output routines are addressed here, as well as a more about symbols, boxes and penalties. Appendix G is a complete catalog of TEX commands, each with a short explanation and example. The final appendix is a short bibliography of TEX books, newsletters and electronic resources.

I have always been of the opinion that as a general rule a programming language should be describable in about 100 pages. TEX is complex enough that a short reference is difficult. In *Writing With TEX*, however, Gurari has made a decent 230-page attempt. This economy of language, however, comes at the price of ease of learning and depth of knowledge. If you have already learned the basics of TEX and wish to know more, there are other books which go into the details of macros, line and page breaking and output routines. If you are new to TEX, and not skilled with programming, Gurari offers little help. On the other hand, *Writing With TEX* is a decent TEX reference. Gurari's explanations are clear, concise and independent, which is what a reference manual should be.

TEX & LATEX: Drawing & Literate Programming is a reference manual for two macro packages written by Gurari: DraTEX for drawing pictures, and ProTEX for literate programming. The book title is misleading since it is really about drawing and literate programming (and specifically about DraTEX and ProTEX). Very little of the book deals generically with TEX or LATEX.

The book begins with a short *Getting Started* chapter which explains how to run TEX and LATEX. Chapters 2–5 cover the basics of TEX (including *Mathematical Formulas* and *LATEX*) in a whirlwind 20 pages. Given how peripheral this is to the actual macro packages, I wonder why these chapters are included — it is unlikely somebody will purchase *TEX & LATEX* who is not already familiar with TEX. Still, Gurari is good at providing short and accurate explanations, and I dare say the five page LATEX chapter is enough to get started.

Chapters 6–11 cover DraTEX primitives. These include macros for line and curve drawing, painting and clipping, coordinate systems (including three-dimensional viewpoints), repetition and data paths, creating objects with tables of data, and arithmetic operations. DraTEX is indeed an impressive package which does all of its drawing in TEX. This does,

[1] When chapter titles appear appear as part of a description they will be italicized.

[2] Version 2.09

however, impose some limits on what can be drawn. For example, rotated text is not possible without special fonts or `\special` support, and the clipping and three dimensional commands are limited.

Chapters 12–20 cover *High-Level Drawing Facilities*. These are drawing templates provided by AlDraTEX, a macro package based on DraTEX. The templates include macros for pie, XY and bar charts. There are three chapters introducing diagramming, a chapter on tree diagrams and another on labeled graphs. Gurari has authored a textbook on computational theory, and the graphs and trees provided by AlDraTEX seem sufficient for such a book.

Chapters 21 and 22 cover *Literate Programming* using ProTEX.[3] Literate programming is a programming style invented by Donald Knuth, and used for writing TEX, METAFONT and the suite of programs used for font management. Literate programs consist of a mixture of code and code documentation. A set of filters converts the input file into either the source code for the program, or a typeset document describing the operation of the program. Gurari's ProTEX is a macro package which uses TEX as that filter. ProTEX's output is a typeset document and auxiliary files consisting of the code described in the document. The code may be Pascal, C or any other language including TEX.

The book ends with five appendices. The first two cover the *Implementation of ProTEX*, and the use of *PostScript Figures Within TEX*. ProTEX implementation details are provided so that users can adapt ProTEX to their own needs. PostScript is introduced as a standalone page description language, and as a way to supplement the abilities of DraTEX. In this mode it is similar to Timothy van Zandt's PSTricks.

Appendix C is a *Catalog of Commands* covering TEX, LATEX, DraTEX, AlDraTEX and ProTEX. Each command is presented with an example of its output. It would be helpful if the commands were cross-referenced with their explanation in the book.

The last two appendices are a bibliography and information on acquiring the macro packages. The bibliography includes references on drawing and literate programming. The macros are available on a disk which accompanies the book, but updates and examples are online.

Regardless of the systems on which you work, there are a variety of affordable (frequently free) drawing programs available. For complex figures these are usually a better choice. For simple figures, or if you like the idea of providing a single, portable source document containing both text and figures, DraTEX and AlDraTEX are a good choice.[4]

DraTEX and ProTEX are available on CTAN and at `ftp.cis.ohio-state.edu` in the directory `/pub/tex/osu/gurari/` (they have been updated since the book and disk were published). This includes an `Examples.tex` file, which is a ProTEX file of examples from the book. There is no better way to sample the abilities of ProTEX and DraTEX then to compose this file. The result is a typeset description of the examples, and 68 example figures and exercises.

On the whole, both books are well written and cover a lot of material in few pages. There are, however, no answers provided for the exercises so expect to spend time experimenting. I recommend these books only for experienced programmers who need a concise TEX reference manual, or who would like to use a TEX based drawing or literate programming package.

◇ Michael D. Sofka
Computing Information Services
Rensselaer Polytechnic Institute
Troy, New York 12180-3590
`sofkam@rpi.edu`

[3] This is the third package I am aware of called ProTEX. The other two are *Pro*fessional typesetting systems based on TEX.

[4] You will, however, require a version of TEX compiled with a large main memory array, and a fairly fast computer. I ran the samples on an RS/6000-250 with Web2c, and a 100Mhz Pentium with MikTEX. On both systems performance was acceptable.

Tutorials / Surveys

Typesetting mathematics for science and technology according to ISO 31/XI

Claudio Beccari

Abstract

Mathematicians set mathematics into type differently from physicists and engineers; the latter require some particular tricks in order to satisfy ISO 31/XI and to distinguish similar symbols that have different meanings. The LaTeX 2$_\varepsilon$ commands to implement such tricks are shown and explained.

1 Introduction

As D.E.Knuth points out very well in *The TeXbook* [7], the strength of TeX and its derived dialects (among which LaTeX 2$_\varepsilon$ outclasses all others) lies in the ability to typeset beautiful mathematics; the variety of symbols, the shape of the operators, the spacing, both vertical and horizontal, the sizes of first and second order sub and superscripts make (LA)TeX the best software available today for typesetting mathematics. Of course (LA)TeX does a wonderful job also with plain text, tables, cross referencing, indexing, and so forth, but other programs may perform well with the latter tasks; what other programs really cannot do is the excellent work with mathematics; all this is not surprising since TeX was created by a mathematician for typesetting mathematics, first of all in his own books.

In this paper I do not discuss how to typeset mathematics, since (LA)TeX takes care of most of it; very seldom the author needs minor corrections of a formula, and when this happens it is usually to correct some spacing when slanted operators are too close or too far away from the symbols they precede or follow, so that the slanting shape of the operator requires some degree of manual intervention in order to fix the spacing. Several such cases are dealt with both in *The TeXbook* and in Lamport's LaTeX Handbook [8].

Nor do I discuss the aesthetics of a typeset formula, where several factors should follow one another in such a way that the formula profile is as smooth as possible, without valleys and peaks. Typesetting a complicated formula requires both mathematical knowledge and a sense of aesthetics, but requires also, especially in didactic books, that the relevant parts of the formula are highlighted in the proper way coming to a compromise between readability and abstract typesetting rules.

I will discuss here those few tricks that physicists and engineers, *not mathematicians*, must know in order to satisfy the international regulations and to distinguish similar symbols with different meanings and, ultimately, in order to cope with the ISO regulations [1] and the recommendations issued by the International Union of Pure and Applied Physics (IPU) [6].

2 Upright and sloping letters

The main and possibly the only difference between "mathematical" vs. "physical" mathematics lies in the use of upright and sloping letters. Scientists and technologists (should) use upright letters much more often than mathematicians.

In math mode LaTeX 2$_\varepsilon$ chooses normal letters from the "math italics" alphabet, which includes also the Greek lowercase ones. Both Latin and Greek letters have a sloping shape, the former being in italics, the latter just sloping to the right with a slope angle that matches the one of the italics characters. Just the uppercase Greek letters (by default) are upright, but it would be very easy, although unusual, to set them with a sloping shape because they appear in the "math italic" font, and in all the "text italic" and other "slanted" fonts.

In the following I will call "roman" the upright shape and "italic" the one that TeXies are used to associate with math italics. The choice of the word "roman" is not chance, since sans-serif characters are not suited for physical mathematics because several signs are not easy to distinguish in the absence of serifs: compare I and l for example; you cannot tell which one is "upper case I" and which one "lower case l".

Sans-serif upright characters may be used in technical and/or physical texts in order to mark objects that cannot be confused with mathematical symbols, for example for the names of points in the description of geometrical figures, technical objects, experimental setups, and the like. Therefore sans-serif upright letters never appear in a mathematical formula of a physicist or an engineer, while mathematicians use sans-serif fonts to represent certain structures in category theory.[1] As a partial exception, sans-serif *sloping* uppercase letters are allowed to indicate tensors of the second rank, but this is the

[1] Thanks to the reviewer for this information; although he/she does not specify it, nevertheless I suppose that they are sans-serif *sloping* fonts.

only exception mentioned in the IPU recommendations [6] and stated in the ISO regulations [1, clause 11-10.14] for using sans-serif fonts.

2.1 Italic symbols

According to the ISO regulations and the IPU recommendations, italic symbols should be used only to denote those mathematical and physical entities that may assume different values, typically those symbols that play the role of physical variables, but also those physical "constants" that are not really constant, because better measuring techniques may produce updated values.

Among such constants there is for example the elementary electric charge (the charge of the proton) $e = 1.602 \cdot 10^{-19}\,\mathrm{C}$, that is considered constant until better measures will add other significant digits; the same holds true for such constants as the velocity of light c, the Planck constants h and $\hbar$, the Boltzmann constant k, and so on.

Every physical variable is represented by *one* italic letter with as many modifiers as needed such as subscripts, superscripts, primes, etc. [6, clause 1.2.1]. There are a few exceptions to this rule, represented by the dimensionless parameters (such as the Mach number, the Euler number, and so on) that are specified with a two-letter symbol by the ISO regulations [2]; for example the Mach number is represented by Ma, the Euler number by Eu; when such two-letter symbols are used, equations must be written with special care so as to make sure that Ma does not represent the product of the physical variables M and a. For the names of the nuclides, that may consist of two letters, see the next section.

In pure mathematics two- or three-letter names are used in applications such as, for example, the name of the Galois field with n elements[2] that is represented with $\mathrm{GF}(n)$. But such applications are not substantially different from what the ISO regulations say about the names of special functions [1, clauses from 11.11.1 to 11.11.21].

In the domain of Computer Science as well as in Electronics authors and typesetters make frequent use of multiletter symbols,[3] but this tradition is evidently in contrast with the ISO and IPU statements and should be abandoned.

Sub and superscripts must be set in italics when they represent physical quantities or mathematical variables [6, clause 1.2.2] otherwise they should be set in roman type; for example: C_T, where T represents "temperature"; M_i, where i is a summation index; but R_E where 'E' distinguishes an object such as the "emitter".

[2] Thanks again to the reviewer for pointing out this topic.
[3] Please notice the difference between the *symbol* of a physical quantity and the *name* of an operator or a function.

2.2 Roman symbols

Any other symbol that was not dealt with in the preceding subsection must be set in roman font; the list of such "roman" entities is surprisingly long and, unfortunately, little known although ISO regulations and IPU recommendations are quite clear on this subject.

1. Numbers must be set in roman type (LaTeX 2ε does this by default).
2. Numerical constants must be set in roman type; this is perhaps the most neglected rule, but it applies to $\mathrm{e} = 2.718\,281\,8\ldots$, the base of natural logarithms, to the "imaginary unit" that mathematicians and physicists call 'i', while most engineers call 'j', and so on. The ISO regulations [1, clause 11-8.1] allow both symbols for the imaginary unit, but both must be set in roman type. The reason behind this is to avoid confusion between the base of natural logarithms and the elementary electric charge, between the imaginary unit and the instantaneous current i or the instantaneous current density j whose symbols are recommended by the IPU [6, sections 7.5 and 8.5].

   Anybody can notice that 'e' and 'i' or 'j' are universally typeset in italic font in both mathematical and physical or technological texts; this observation gives a measure of how much the rule I am speaking about is ignored, at least by physicists and engineers.

   The same rule should apply to numerical constants represented by Greek letters, such as $\pi = 3.141\,592\,6\ldots$, but it is necessary to get by with it because it is difficult to have both the upright and the sloping Greek fonts; fortunately enough the frequency of such symbols (except π) is not high. If both upright and sloping Greek fonts were available, an upright π should indicate the numerical constant "3.1415...", while a sloping π should indicate the physical constant "3.1415... rad" corresponding to the angle of 180°.
3. Physical units must be set in roman type with the additional requirement that a line break cannot take place between the measure and the unit of measure. In order to emphasize the unitary nature of a physical constant, the measure and the unit of measure should be separated by an unbreakable *thin* space, rather than by a regular interword (unbreakable) space. Besides

being quite reasonable, this point copes with the high penalty that TeX introduces in a formula within a product of terms; if TeX has to break a formula across lines, it does so close to a binary or a relation operator (although very reluctantly), not within the product of terms, and a physical quantity *is* the product of the measure times the unit of measure.

4. Mathematical operators indicated with letters must be set in roman type; LaTeX 2ε already does this for a number of predefined operators, such as 'lim' or 'sin'. The package amsmath adds some more predefined operators, but, most important of all, it adds a couple of commands for using other operators. We will discuss this topic in a following section.

   It is worthwhile to mention that operators do not require that their argument be enclosed within parentheses if the operand is made up of no more then two objects (2π being counted as one object), but proper spacing is required on both sides; so you simply write $\sin\omega t$ and do not need to write $\sin(\omega t)$, but you should write $\sin(2\pi ft)$; in any case parentheses are required when a formula contains several operators with their arguments.

5. The names of special functions such as 'erf' (error function), or 'Ei' (exponential integral), or 'E' (incomplete elliptic integral of the second kind), ..., are treated by ISO 31/XI the same as the names of operators [1, clauses from 11.11.1 to 11.11.21] and should be typeset in roman type, although their argument(s) must always be indicated within parentheses. See the following sections for what concerns the definition of new operators.

6. A particular operator, the operator of differentiation, should be set in roman type, but under the LaTeX 2ε point of view, it should behave differently from other operators as concerns spacing. The use of roman type for the differential operator is another example of a highly neglected rule, albeit the ISO regulations [1, clause 11-5.15] are explicit on this point.

   The "house style" of the majority of publishing companies, where the differential operator is a common italic '*d*', was evidently set up under the influence of the tradition of pure mathematical typesetting before the ISO regulations were published; now many years have elapsed since their publication so that the ISO regulations should be widely applied, and it is surprising (it surprises me, at least) that, while the modern world is so attentive to international standards, this particular one is almost completely neglected.

   The ISO prescription and the IPU recommendation concerning the differential operator are not illogical, since in physical and technological mathematics it is essential to distinguish different mathematical objects within a formula, taking into account the nomenclature recommendations for physical quantities; it is really necessary to distinguish 'd' from '*d*' when the latter indicates one of the many physical quantities whose symbol is recommended to be *d*: thickness, diameter, relative density, lattice plane spacing, degeneracy of vibrational mode, etc.

7. Sub and superscripts that do not represent physical quantities or mathematical variables should be set in roman type; the amsmath package makes available the command `\text` for setting in roman type any word or sentence within mathematics, also in sub and superscripts. In general, though, the problem is to distinguish the type of sub and superscripts in order to decide how they should be typeset. In the books I wrote I estimated that more than two thirds of the subscripts I used had to be set in roman type, while the absolute majority of superscripts were mathematical expressions; although just a few books have no statistical value, the experience I gained allows me to say that this percentage of roman subscripts should be considered typical, so that roman subscripts are almost the default case in physics and technology.

8. Chemical symbols coincide with the names of the nuclides if they carry proper sub and superscripts; with respect to typography, particles and quanta are treated as the nuclides. They are made up of one or two letters and must be set in roman type [6, section 4]; chemical equations are in general quite complicated so that for setting into type a book on chemistry it is better to use specialized packages; the last one that was described in *TUGboat* [10] is an excellent example; the *LaTeX Companion* [9] describes another package ChemTeX [11] for the same purpose.

   For simpler chemical formulas that do not require the graphic facilities of LaTeX 2ε, the author can typeset chemical equations with the usual math commands provided he/she remembers to switch to roman type for the symbols of the chemical elements.

9. Roman numbers are seldom used in physics and technology, although they find their way in

chemistry where they may represent the spectrum of a z-fold ionized atom or, in superscript position, the oxidation number; in both cases they are set in roman type [6, clause 4.5].

Roman numbers used for enumerated lists or for numbering front matter pages do not pertain to mathematics; they are generally set according to the publishing house style. As a matter of personal taste, in these situations I prefer small-caps roman numerals to italic ones.

3 Other typesetting recommendations

While typesetting "physical" mathematics it is convenient to remember that scientific and technical papers have a worldwide readership and the observance of the ISO regulations and IPU recommendations is essential for making one's text easily understood by readers of different language and culture. Here are some hints:

1. The decimal separator is the "decimal comma" in the rest of the world [6, clause 3.2] and the "decimal point" in the English speaking countries; it is necessary to be consistent with this rule and to avoid mixed separators (see the following point).
2. Before and after the decimal separator digits may be grouped in triplets separated only by a *thin space*, not by commas (USA) or lowered or raised points (Europe) [6, clause 3.5]; in general Europeans are more used to the North American habit of dividing triplets with commas, than North Americans with European habits. In both cases a professional typesetter of a scientific or technical text uses only thin spaces so as to avoid the trouble of interpretation to readers of different cultures. Triplets may be avoided when before or after the decimal separator there are just four digits: that is, you should write USD 1900 (or $1900) preferably to USD 1 900, while writing USD 1,900 should be absolutely avoided: in Europe it means "one US dollar and 90 cents"!

   In this paper the North American decimal separator is being used all over; for Europeans I recall that the point in the role of the decimal separator is allowed only in numerical constants written within a (section of a) programming language.
3. When a number is less than unity in absolute value, this should be explicitly indicated with a 'zero' preceding the decimal separator [6, clause 3.2]; therefore it is necessary to write 0.123 and -0.456 in place of .123 and $-.456$ respectively.
4. The International System of Units (SI) [3] is the only legal system of units to be used in those countries that undersigned the specific bill. In practice the SI is the only system of units accepted by the scientific community, although engineers do it more readily than physicists; the latter sometimes indulge in using one of the old cgs systems, more for the possibility of skipping some constants in the formulas dealing with electromagnetic phenomena, than for a real need of using centimetres/centimeters and grams.

   The SI establishes the symbols for the fundamental and derived units and the legal decimal prefixes; some non-SI units keep their value, while some others are "outlaw". In any case every legal or accepted unit has its own symbol and only that symbol must be used; you write 3h 12min 45s, not 3hrs 12mins 45secs; you write "a conductance of 25 mS", not "a conductance of 25 m℧".
5. The symbols of the physical units are *symbols*, not abbreviations, therefore they must never carry the abbreviation point after them and do not change in the plural [6, clause 2.1.2] so that it is correct to write 7.25 cm while it is wrong to write either 7.25 cm. or 7.25 cms with the abbreviation point or with the plural of the symbol.

   Unit symbols must always be used when they accompany the measure and they must be set after the measure; the full spelled name (with lower case initial even if the unit has an uppercase symbol) should be used in the text when preceded by general specifications such as "several", "few" or when the unit of measure is called by name; you write "electric bulbs are labeled with the operating voltage in volts and the power rating in watts", not "electric bulbs are labeled with the operating voltage in V and the power rating in W".

   As for the plurals of the unit names (not of the unit symbols that are invariant as mentioned above), every language has its own rules and every country its own regulations; beware not to use a plural form of another language.[4]
6. Proper scientific prose does not use the physical quantity symbols in text mode, it rather uses the full spelled name possibly followed by the symbol; you write "space s and time t are the only physical quantities necessary to deal with

[4] This is not a common problem in English speaking countries, but it is a problem in other countries where English plurals are often used.

kinematics", not "s and t are the only physical quantities necessary to deal with kinematics".

7. Connected to the previous item, the scientist and the engineer should choose the correct names for physical quantities; the IPU recommendations [6] offer a very wide nomenclature list in English and French for virtually every quantity of interest in pure and applied physics; for every quantity a preferred literal symbol is indicated and the names of the quantities should be used in a consistent way, rather than inventing new names and new symbols for old objects simply because of specialized technical jargons.
8. The physicist's and the engineer's equations are relationships between physical quantities; the latter are groups made up of a first term that represents the measure, and a second term that represents the unit of measure; although they are not commutative, they behave as factors of a multiplication. The mathematical operations performed on physical quantities operate in the usual way on the numerical parts and in an algebraic way on the units of measure, giving rise in some instances to derived units. If a physical equation contains a physical constant not represented by a symbol, this constant must contain both elements: measure and unit of measure. You write $Z_0 = 377\,\Omega/\sqrt{\epsilon}$, not $Z_0 = 377/\sqrt{\epsilon}$.

   For the same reason it is not necessary, or better, it is definitely wrong to write the units of measure after a coherent physical equation; see also the next item.
9. Measure equations should be absolutely avoided in professional scientific texts; measure equations were somewhat popular before the SI was universally adopted; now they should not be used any more. They survived in those countries where the "English system of units" is being used, but, since scientifically speaking this traditional system of units is "illegal",[5] measure equations have no reason to be used any more.
10. Since every physical quantity is a group formed by the measure and the unit of measure, it is wrong to specify the unit of measure after the symbol of a physical quantity; you write "a periodic function of period T", not "a periodic function of period T seconds".

[5] The United States was one of the last countries to adopt the SI in the late fifties, but even after four decades have elapsed, in that country the SI seems to have difficulties in replacing the traditional units.

11. Physicists seem to like the powers of 10; engineers prefer the decimal prefixes established by the SI. Such prefixes, representing positive and negative powers of 1000 (in addition to several that represent the first few positive and negative powers of 10), cover an extremely wide range, from 10^{-18} to 10^{18}, so that powers of 10 can be easily avoided. When choosing the right prefix it is necessary to remember that integer values of the measures are preferred and that zeroes just after the decimal separator should be avoided; you write 32 pF rather than 0.032 nF. This rule holds true everywhere, except in tabular formats where the unit of measure is specified in every column-head and is valid for all the table entries of that column [3, section 4].
12. Connected with the previous item there is the question of the number of significant digits; while leading zeroes (between the decimal separator and the first nonzero digit) are allowed only in tables, trailing zeroes should be avoided unless they are significant because they carry information on the measure precision: in physics 7.25 m is different from 7.250 m because the former quantity is precise to ± 0.5 cm while the latter, to ± 0.5 mm. Therefore physicists and engineers should be careful to use just the number of significant digits (inclusive of the trailing zeroes) that are compatible with the experimental or technological data they are referring to.
13. I need not emphasize that prefixes as well as units of measure should be spelled correctly, paying particular attention to uppercase and lowercase letters and to the operators between units; nevertheless the following errors are quite common: 'K' (kelvin) in place of 'k' (kilo); 'M' (mega) in place of 'm' (milli) or vice versa; 'mμ' (milli-micro) in place of 'n' (nano) [6, clause 2.2.2]; 'u' in place of 'μ'; 'hz' and 'db' in place of 'Hz' and 'dB'; '°K' in place of 'K'; 'kwh', 'KWH', 'KWh', 'kw/h' (awful!) in place of 'kWh' (or even better 'kW h' — notice the thin space); 'kc' (kilocycles) in place of 'kHz' or, at least, even if it is not fully SI, 'kc/s' (kilocycles per second).

    Decimal prefixes should be attached to the following unit without interposing any space; compound units may have a thin space between them or (exceptionally) a raised dot [6, clause 2.3.1]; either separator is mandatory when units may be confused with prefixes; positive or negative powers refer to the unit *inclusive of its* *decimal prefix*; negative powers, although allowed, are preferably avoided: 'kW h' is better than 'kW · h' and better than 'kWh'; 'm/s^2' is better than 'm s^{-2}' and both mean something different from 'ms^{-2}' (meters per square second versus the inverse of square milliseconds) [6, clause 2.2.3].

14. As mentioned above, the math italic typeface is used for normal variables and roman typeface is used for letters and "adjectives" that do not represent variables. Other typefaces are seldom used; the IPU recommendations [6] state that sloping sans-serif characters may be used for second rank tensors and boldface italic characters for vectors (lowercase) and matrices (uppercase).
15. Currency units are not part of the SI, and are standardized by other regulations [5]; they are generally made up of three uppercase letters, the first two of which are the two-letter code for the nation while the third one is the initial of the currency name; therefore you write USD, not US$, for a text that should be read abroad, and keep the symbol $ just for the national readership — after all, many countries use the dollar as the national currency (besides the United States, there are Canada, Australia, Hong Kong, and many others), but in general they are not equivalent to one another; the same holds true for the UK pound £ and the Italian lira,[6] whose international symbols are GBP and ITL respectively.

    In contrast to other units, currency units precede the monetary value but the rule of the unbreakable thin space keeps its validity. Trailing zeroes such as 'USD 1900.00' may be required in order to specify also the unit submultiples for legal and/or commercial purposes.

4 LaTeX 2$_\varepsilon$ new commands

Here I will describe some very simple LaTeX 2$_\varepsilon$ new commands intended to help in the typesetting of "physical" mathematics. In general I will use the command `\providecommand*`. This instruction behaves just as `\newcommand*` except that it defines the new command only if it is undefined; if the command already existed the new definition is silently ignored. In this way if you make yourself several packages that include such definitions, the new commands are defined just once and you do not have to be concerned about anything else; on the contrary if you use `\newcommand*`, you receive error messages when LaTeX 2$_\varepsilon$ executes them if the command being defined already exists, and if you use `\renewcommand*` the error messages show up the first time LaTeX 2$_\varepsilon$ executes it. I prefer the asterisk form because I get better diagnostic messages in case I forget some closing brace.

1. Some commands for special units and a prefix are very handy; the following definitions are valid both in text and in math mode:

```
\DeclareMathAccent{\ring}%
           {\mathalpha}{operators}{"17}
\providecommand*{\angs}%
{\ensuremath{\smash{\mathrm{\ring A}}}}
\providecommand*{\ohm}%
         {\ensuremath{\mathrm{\Omega}}}
\providecommand*{\degree}%
                  {\ensuremath{^\circ}}
\providecommand*{\celsius}%
       {\ensuremath{\mathrm{^\circ C}}}
\providecommand*{\micro}%
                     {\ensuremath{\mu}}
```

   The first declaration adds the ring accent in math mode; LaTeX 2$_\varepsilon$ has the `\r` command for the ring accent in text mode, but in math mode the ring accent is missing so that it is necessary to define it. At the present state of the LaTeX 2$_\varepsilon$ art, mathematical signs, letters, accents, . . . , are taken from the traditional knuthian cm fonts, and accents are set over the accented letter, especially a capital one, a little too high; with the dc fonts accented letters are made up of just one glyph and the positioning of the accent has been carefully adapted to each letter by the font designer. Maybe in the future, when new 256-glyph math fonts are available, the position of the accents will be more precise. This is why I introduced the `\smash` command in the definition of the ångström unit,[7] otherwise in text mode a line containing such a unit would force some interline leading, resulting in a nonuniform line spacing. The chance of interfering with the previous line descenders is small but not zero. See the following paragraph.

   The `\ring` command is useful not only for setting the unit symbol 'Å', but also for setting

[6] The UK pound and the Italian lira have similar symbols: £ and ₤; attentive people notice that the UK pound symbol has one stroke across, while the Italian lira has two strokes, but if you ask Italians, the great majority don't notice the difference.

[7] The ångström unit is tolerated by the SI in the specialized field of light and optics; the name of the unit, in contrast to other SI units derived from personal names containing diacritics, maintains its diacritics [3, annex A, clause 6.3.1] [4, section IV.2].

a variety of 'physical' variables in telecommunications when they refer to the analytical signal.

The remaining definitions introduce handy commands for common units that may be used in both modes thanks to the `\ensuremath` command. There is a drawback with these definitions, that is the newly defined units are set in roman medium upright type everywhere and do not change family or series in text mode.

The `\ohm` definition is a little redundant,[8] because `\Omega` by default is defined as a symbol, not as a letter, so that with LaTeX 2ε it is not subject to the change in the math version (either normal or bold); but if you redefine it or use a package where it is redefined as a letter from the `\mathnormal` alphabet, the above definition guarantees that the unit Ω is set in roman type anyway.

2. The following macro lets you set the units, whichever they are, in roman type, both in text and math modes, with the proper unbreakable thin spacing:

```
\providecommand*{\unit}[1]{%
              \ensuremath{\mathrm{\,#1}}}
```

When you use `\unit` in text mode, of course, you must not leave any space between the measure and the `\unit` command.

3. Numerical constants represented by Latin letters may be treated by the following macros:

```
% The number 'e'
\providecommand*{\eu}%
               {\ensuremath{\mathrm{e}}}
% The imaginary unit
\providecommand*{\iu}%
               {\ensuremath{\mathrm{j}}}
```

In the imaginary unit definition a physicist would probably substitute 'j' with 'i', but this is not the point, because the above definition is just an example of how the imaginary unit should be defined. A more sophisticated definition might refer to math operators (see below) so that proper spacing is left around such letters (especially around the imaginary unit). In several sciences, though, a common expression is the combination of 'e' raised to some imaginary power; if you use 'j' for the imaginary unit, its descender butts against the 'e' irrespective of what definition you use (and in practice this occurs even if you stick to the tradition of using italic fonts); this is one of the rare occasions where the typesetter should correct the almost perfect setting performed by TeX.

4. LaTeX 2ε provides an internal command for setting text superscripts, but we need commands that let you set sub and superscripts with the proper math sizes; the package amsmath provides the command `\text` for setting text in math mode, and this command chooses the right size according to the sub or superscript commands. Here I introduce a couple of commands that are good for single word sub and superscripts:

```
\providecommand*{\ped}[1]{%
              \ensuremath{_\mathrm{#1}}}
\providecommand*{\ap}[1]{%
              \ensuremath{^\mathrm{#1}}}
```

These definitions, that work in both modes, can be handy also for setting text superscripts such as 2^{nd}, 3^{rd}. The names are abbreviations of the Latin words *pedex* and *apex* that refer to the foot or the head position respectively.

5. The LaTeX 2ε package amsmath provides a couple of commands for *using* a sequence of letters as an operator; they are `\operatorname` and `\operatornamewithlimits`; here I introduce a couple of new commands that *define* new operators.

But before going on, it is better to recall a couple of details about operators. (La)TeX deals with two kinds of operators: the "log-like" and the "lim-like" ones; the former treat subscripts and superscripts as regular ones, that is (La)TeX sets them in smaller size at the right and lowered or raised respectively, while the latter accept sub and superscripts as limits, so that in math display mode subscripts are set underneath the operator and superscripts above it.

Moreover (La)TeX sets spaces at the left and at the right of the operator in different ways according to what kind of mathematical object falls close to the operator; such different spacings are described in detail in chapter 18 of *The TeXbook*. Therefore to set perfect (physical) mathematics it is advisable to define operators in the proper way, but owing to the two kinds of operators it is necessary to specify in the definition whether sub and superscripts should be used as limits or not:

```
\providecommand{\newoperator}[3]{%
      \newcommand*{#1}{\mathop{#2}#3}}
\providecommand{\renewoperator}[3]{%
    \renewcommand*{#1}{\mathop{#2}#3}}
```

[8] In any case, even if `\ohm` was simply `\let` to `\Omega`, it would be shorter to spell and clearer to understand.

Here the first argument is the operator name, the second is its full description and the third is the TeX declaration `\limits` or `\nolimits`. A couple of examples are necessary: according to the ISO regulations the integer part of a decimal number is called "ent" [1, clause 11-4.15], so that the following definition is required

```
\newoperator{\ent}%
                 {\mathrm{ent}}{\nolimits}
```

According to the ISO regulations [1, clauses 11-8.2 and 11-8.3] the real part and the imaginary part of a complex number are "Re" and "Im", not $\Re$ and $\Im$; therefore we need the following redefinitions:

```
\renewoperator{\Re}%
                 {\mathrm{Re}}{\nolimits}
\renewoperator{\Im}%
                 {\mathrm{Im}}{\nolimits}
```

6. The differential operator must be treated in a slightly different way, because it is a special operator that requires different spacings on the left and on the right; moreover being made up of just one letter, it is necessary to use a little TeX trick in order to guarantee that its mathematical axis lines up properly. The differential operator should be spaced as an operator on the left, while on its right it should receive different spacings, and in particular it should not be spaced from its argument. A possible way of achieving this result is to define it as an operator that contains a negative spacing on the right; in order to cope with possible exponents some tests must be performed along the line and this requires some low level TeX programming.

```
\makeatletter
\providecommand*{\diff}%
        {\@ifnextchar^{\DIfF}{\DIfF^{}}}
\def\DIfF^#1{%
        \mathop{\mathrm{\mathstrut d}}%
            \nolimits^{#1}\gobblespace}
\def\gobblespace{%
            \futurelet\diffarg\opspace}
\def\opspace{%
        \let\DiffSpace\!%
        \ifx\diffarg(%
            \let\DiffSpace\relax
        \else
            \ifx\diffarg[%
               \let\DiffSpace\relax
            \else
               \ifx\diffarg\{%
                   \let\DiffSpace\relax
               \fi\fi\fi\DiffSpace}
```

With respect to (total and partial) derivatives it might be useful to define some commands (with the order as an optional argument) by means of the usual `\providecommand*` construct:

```
\providecommand*{\deriv}[3][]{%
    \frac{\diff^{#1}#2}{\diff #3^{#1}}}
\providecommand*{\pderiv}[3][]{%
                \frac{\partial^{#1}#2}%
                    {\partial #3^{#1}}}
```

The first *optional* argument is the derivative order, the second is the function being derived, and the third the derivation variable. Partial high order mixed derivatives require a more sophisticated definition, or direct setting with the `\frac` command.

With the help of these new commands it becomes very easy to typeset the following equations with the assurance that spacings are right and the differential operator is correctly set in roman type; a sample of source code follows each equation.

$$a\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + b\frac{\mathrm{d}y}{\mathrm{d}x} + c = f(x)$$

(`a\deriv[2]{y}{x}+...`)

$$\frac{\mathrm{d}(\log y)}{\mathrm{d}x} = \frac{1}{y}\frac{\mathrm{d}y}{\mathrm{d}x}$$

(`\deriv{(\log y)}{x}=...`)

$$\mathrm{d}z(x,y) = \frac{\partial z}{\partial x}\,\mathrm{d}x + \frac{\partial z}{\partial y}\,\mathrm{d}y$$

(`...=\pderiv{z}{x}\diff x+...`)

$$\mathcal{L}[f(t)] = \int_{0-}^{\infty} \mathrm{e}^{st} f(t)\,\mathrm{d}t$$

(`...\eu^{st}f(t)\diff t`)

7. The prefix "femto" (that stands for 10^{-15}) turns out to be rather frequent in microelectronics where it is generally used as a prefix for the unit 'farad'; the couple 'fF' requires some kerning so that the lowercase 'f' does not butt against the uppercase 'F'; up to now[9] the available math fonts do not consider this couple for implicit kerning information, therefore the typesetter should remember to insert an italic

[9] At the time of writing, the latest version of dc fonts is 1.3 and has a date of June 1996; this release of the dc fonts copes with many new kernings that take place with SI decimal prefixes, but this doesn't help much in math mode because here cm fonts are used; their latest version was refined in 1992, but the latest available source METAFONT files on the CTAN archives carry the date of June 1995; let's hope that the new math fonts with 256-glyph encoding will be available soon.

correction between the lowercase ‘f’ and any uppercase non-slanting-left-side letter that follows; examples: ‘fF’ (`f\/F`), ‘fW’ (`f\/W`), ‘fH’ (`f\/H`), but ‘fA’ (`fA`).

At present a `\femto` macro would solve the kerning problem, but in the future such kernings should find their way into the very definitions of the various fonts that are used in math mode.

5 Examples

In addition to the examples shown in the previous section, I report here some more examples where the various features described above show the difference between ‘mathematical’ and ‘physical’ equations. Between parentheses you find the source code for some relevant part of the example.

1. The Euler equation involves the five most important mathematical constants:
$$\mathrm{e}^{\mathrm{j}\pi}+1=0$$
(`\eu^{\iu\pi}`)
Actually in this case, as pointed out in the previous section, the typesetter should introduce a small correction to the spacing:
$$\mathrm{e}^{\,\mathrm{j}\pi}+1=0$$
(`\eu^{\,\iu\pi}`)
2. A component list:
$$\begin{array}{ll} R_\mathrm{B}=2.2\,\mathrm{k\Omega} & R_\mathrm{F}=180\,\mathrm{k\Omega}\\ h_\mathrm{ie}=1.5\,\mathrm{k\Omega} & h_\mathrm{fe}=100\\ C_\mathrm{E}=3.3\,\mathrm{nF} & R_\mathrm{C}=4.7\,\mathrm{k\Omega}\end{array}$$
(`R\ped{C}=4.7\unit{k\ohm}`)
3. The equations of a common emitter stage with a feedback resistor on the emitter:
$$\begin{aligned} I_\mathrm{b} &= (1/h_\mathrm{ie})(V_\mathrm{b}-V_{R_\mathrm{E}})\\ V_{R_\mathrm{E}} &= (1+h_\mathrm{fe})R_\mathrm{E}I_\mathrm{b}\\ V_\mathrm{c} &= -h_\mathrm{fe}R_\mathrm{C}I_\mathrm{b}\end{aligned}$$
Notice in particular the term V_{R_E} whose LaTeX source code is (`V_{R\ped{E}}`).
4. A table

Stub length in mm	Decay time in ns
0.04	798.72
1.20	19 836.07
2.28	17 356.74
3.36	15 468.04
3.96	14 747.75

(`14\,747.75`)
5. An equation with complex variables:
$$\mathrm{Re}[F(\sigma+\mathrm{j}\omega)]>0 \qquad \forall\,\sigma>0 \text{ and } \forall\,\omega$$
(`\renewoperator\Re{\mathrm{Re}}{\nolimits}`)
6. Some text with physical quantities: “The heat sink with a thermal resistance θ_sa of 3.3 °C/W is supposed to maintain the temperature below the transistor maximum junction operating temperature when it operates in an environment at 60 °C.”
(`$\theta\ped{sa}$ 3.3\unit{\celsius/W} 60\unit{\celsius}`)
7. Use of ångströms: a wave length of 5500 Å (`5500\unit{\angs}`).

6 Conclusion

I have been using the above commands for several years; before the advent of LaTeX 2ε I had put together similar commands for LaTeX 2.09, but the difference lies simply in the easier way of defining them with the new version of LaTeX. With the help of such commands I have found it very simple to cope with the ISO regulations and IPU recommendations at the point that now I feel somewhat handicapped if I have to write anything scientific on a computer where the LaTeX implementation is lacking such commands.

At the same time they are so few and simple that I did not consider the possibility of making up a short package file (where the documentation and the insertion driver would be much larger than the useful lines) to be submitted to the CTAN archives. Anybody can copy such commands from paper and/or use them as guidelines for making one's own set of useful commands, possibly performing better than the simple ones that I suggested.

But, most important of all, I warmly suggest to pay attention to the international regulations and recomendations; the results are worth the little extra effort.

References

[1] “Mathematical sign and symbols for use in physical sciences and technology — ISO 31/11” *ISO Standards Handbook N.2*, International Organization for Standardization, Geneva 1982, 2nd ed.

[2] “Dimensionless parameters — ISO 31/12” *ISO Standards Handbook N.2*, International Organization for Standardization, Geneva 1982, 2nd ed.

[3] “SI units and recommendations for the use of their multiples and of certain other units — ISO 1000” in *ISO Standards Handbook N.2*, International Organization for Standardization, Geneva 1982, 2nd ed.

[4] *Le Système International d'Unités*, Bureau International des Pois et Mesures, Pavillon de Breteuil, Sèvres, France, 1991, 6th ed.

[5] "Codes for the representation of currencies and funds — ISO 4217" International Organization for Standardization, Geneva 1990.

[6] "Symbols, units and nomenclature in physics" in *CRC Handbook of chemistry and physics*, R. C. Weast, M. J. Astle and W. H. Beyer, editors, CRC Press Inc., Boca Raton, Florida, 65th edition, 1985, pp. F259–F293.

[7] Knuth D.E., *The TEXbook*, Addison-Wesley Publishing Company, Reading, Mass., 1984.

[8] Lamport L., *A document preparation system, LATEX, User guide & reference manual*, Addison-Wesley Publishing Company, Reading, Mass., 1986 (second edition).

[9] Goossens M., Mittelbach F. and Samarin A., *The LATEX Companion*, Addison-Wesley Publishing Company, Reading, Mass., 1994.

[10] Fujita S., "XΥMTEX for drawing chemical structural formulas" in *TUGboat*, vol. 16, n. 1, pp. 81–89, March 1996.

[11] Haas R.T. and O'Kane K.C., "Typesetting Chemical Structure Formulas with the Text Formatter TEX/LATEX" in *Computers and Chemistry*, vol. 11 n. 4 pp. 251–271, 1987.

◇ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
`beccari@polito.it`

LaTeX

A LaTeX Tour, part 3: mfnfss, psnfss and babel

David Carlisle

1 Introduction

This third installment of my tour covers three more distributions that are supported via the standard LaTeX bug report mechanism described in Part 1.

The mfnfss distribution provides LaTeX support for some popular Metafont produced fonts, that do not otherwise have any LaTeX interface.

The psnfss distribution consists of LaTeX packages giving access to PostScript fonts.

The third distribution in this part of the tour is babel, which provides LaTeX with multi-lingual capabilities.

2 The Mfnfss Distribution

The mfnfss distribution is something of a 'collecting point' for files in the distribution that have not got anywhere else to go.

2.1 Font Packages

These packages provide LaTeX interfaces to some publicly available fonts. They do *not* provide the fonts themselves, which are available from the `fonts` tree in the standard CTAN archives.

`pandora` The 'Pandora' family of fonts designed by Nazneen N. Billawala is an alternative to the standard 'Computer Modern' fonts of Knuth. The family consists of a full range of text fonts, including sans-serif and slanted.

`oldgerm` The old German fonts designed by Yannis Haralambous. There are three styles of text font, Schwabacher, Fraktur and Gothic. (The terms 'Fraktur' and 'Gothic' tend to be used interchangeably by English speaking mathematicians such as the present author, but the fonts in this collection have clearly distinguishable styles.)

There is also a font of 'initials', highly ornate uppercase letters, suitable for use as the first letter of a section. If you wish to use this in 'drop caps' style you may also want to use one of the contributed packages available on CTAN such as drop, or dropping, that automate the setting of a suitable paragraph shape and inserting the initial letter at the correct size.[1]

2.2 T1 Encoded 'Concrete' Fonts

Note: The following two files require the old release 1.1 of the dc fonts. Hopefully there will soon be an officially supported release of T1 encoded 'concrete' fonts based on the recently released ec fonts. At that time these files will probably be withdrawn from this mfnfss distribution.

`dccr.mf` Metafont source file used by the output files from `dccrstd.tex` to generate Concrete Roman fonts in T1 encoding.

`dccrstd.tex` TeX file used in the generation of Concrete Roman fonts in T1 encoding. It will produce a number of `.mf` files corresponding to Concrete Roman fonts in different sizes. By modifying the table inside this file further Metafont driver files can be generated. The `.fd` files for the Concrete Roman fonts can be produced with `cmextra.ins` which is part of the LaTeX base distribution.

3 The Psnfss Distribution

With the release of LaTeX 2_ε, LaTeX gained inbuilt support for the use of alternative font families in documents, and in particular for the use of scalable font formats such as Type 1 (PostScript) or TrueType.

The collection of packages, coordinated by Sebastian Rahtz, known as psnfss offers convenient interfaces to most of the more common font sets.

Most of the files here relate to font files renamed to a consistent naming scheme, promoted and maintained by Karl Berry. This encodes the font vendor, and details of the font such as its weight, style and encoding into a compact name that usually fits in the eight letter filenames used by some common filesystems. More information about the font naming scheme can be found on CTAN in `info/fontname`. It should be noted however that the packages themselves, such as the times package, do *not* depend on any particular font naming convention. LaTeX isolates packages from the details of the external font files by the use of '`fd`' (Font Descriptor) files which map the LaTeX 'NFSS' model of fonts to the external font metric files.

In principle, there is no real need for packages to load text fonts into LaTeX. For example, once the font metrics and font descriptor files for Times Roman (which is ptm in the Karl Berry Naming Scheme) are installed, then one could in principle switch to Times Roman in a LaTeX document by simply specifying `\fontfamily{ptm}\selectfont`. Normally one would instead want to assign the new font to one of the 'default' LaTeX families, Roman, as used by `\rmfamily`, Sans Serif (`\sffamily`) and Typewriter or Monospace (`\ttfamily`).

[1] The `fd` files provided here load the original `yinit` font. The CTAN archives also contain '`yinitas`', a modified version of this font.

The support for PostScript fonts is split into two. The CTAN `fonts/psfonts` area contains material that is mainly automatically generated from the Adobe font metric files that are distributed with all Type 1 fonts. This includes the font metrics themselves, the Font Descriptor files, the 'map' files used to make fonts known to the dvips driver, and some basic packages to declare single fonts to LaTeX. This is supplemented in `macros/latex/packages/psnfss` by the 'hand written' packages of the psnfss collection that load popular *combinations* of font families, or deal with mathematics.

This section refers at various points to PostScript or Type 1 fonts, but in fact the TeX support for these fonts applies equally well to True Type, or other scalable formats. As long as TeX has access to the font metrics, the font format does not matter (to TeX; it matters to the driver you use to print the DVI file).

3.1 Psfonts

The CTAN psfonts area primarily contains the font metric and LaTeX font descriptor files, organised by font vendor, as outlined below. The basic format of the file structure is the same for each font family, so only the top level directories are given here, except for the Adobe Times family, which is further expanded as an example.

3.1.1 Font Vendors

The font subdirectories of `fonts/psfonts` are:

`adobe` Fonts sold by Adobe, or built into PostScript devices.

`bh` Fonts designed by Bigelow and Holmes, these are mainly sold through Y&Y.

`bitstrea` Bitstream fonts.

`monotype` Monotype fonts.

`textures` Textures Fonts for the Blue Sky Research Macintosh TeX implementation.

`urw` Fonts distributed by URW.

`xadobe` Adobe 'expert' font sets.

`xmonotype` Monotype 'expert' font sets.

Each of the vendor directories contains subdirectories corresponding to the font families supported by the psfonts distribution. (Using the tools provided one can generate TeX support files for most other text fonts, the selection here is really just a set of examples.)

The subdirectories of the adobe directory are:

`agaramon` Adobe's rendition of a Garamond serif Roman family. (Commercial.)

`avantgar` Avant Garde sans serif (built into most PostScript devices).

`baskervi` Baskerville, a commercially available serifed Roman family.

`bembo` Bembo, a commercially available serifed Roman family.

`bookman` Bookman (built into most PostScript devices).

`centaur` Centaur, a commercially available serifed Roman family.

`courier` Courier (built into all PostScript devices).

`garamond` Garamond 3. Another Garamond serif Roman family. (Commercial.)

`gillsans` Gill Sans, a commercially available sans serif family.

`helvetic` Helvetica (built into all PostScript devices).

`nbaskerv` ITC New Baskerville, another variant on the Baskerville theme. (Commercial.)

`ncntrsbk` New Century Schoolbook (built into most PostScript devices).

`optima` Optima, a commercially available sans serif family.

`palatino` Palatino serifed Roman family (built into most PostScript devices).

`symbol` Symbol (built into all PostScript devices).

`times` Times Roman (built into all PostScript devices).

`univers` Univers, a commercially available sans serif family.

`utopia` Utopia, a commercially available serifed Roman family.

`zapfchan` ITC Zapf Chancery. A script font built into most PostScript devices.

`zapfding` ITC Zapf Dingbats. A symbol font built into most PostScript devices.

All the directories corresponding to a font family look essentially the same, each with the following subdirectories.

`dvips` Contains the 'map' file for the dvips driver program. This file can be appended to `psfonts.map` or used via a configuration file to tell dvips where to find the specified fonts. A suitable configuration file is included in the directory.

Other drivers will need similar information, but perhaps in a different format.

`tex` This directory contains the font descriptor files which must be placed in the input path for LaTeX, so that LaTeX has available the information about the available fonts. For some font families this directory would also contain a LaTeX package that assigns the fonts to one of the standard LaTeX font families, such as `\sffamily`. Some packages, such as times, are not distributed here as they would clash with the packages distributed as part of psnfss, as described below.

`tfm` The font metrics, in 'tfm' format. These files contain all the information about letter sizes, ligatures, and kerning that TeX needs to typeset text.

There are several files, as each font in the original family is made available in several encodings, the two main ones being the 'Classic' TeX encoding used by Computer Modern. This is known as OT1 in LaTeX, and as '7t' in the Karl Berry font naming scheme used here. Similarly the files with names ending in '8t' relate to fonts encoded to the eight bit 'Cork' encoding, known as T1 in LaTeX.

`vf` The virtual fonts. Most (but not all) drivers handle the re-encoding of the original fonts to the encodings that TeX expects by means of the virtual font mechanism. Some special fonts, such as Zapf Dingbats are not re-encoded, and so do not have a `vf` directory.

There is one very important thing to note about the above list. *There are no fonts!* Almost all of the `fonts/psfonts` area of CTAN is concerned with providing mechanisms for using fonts that you have obtained *elsewhere.* The fonts may be built in to your printer, or may be purchased separately. There are a few freely available Type 1 fonts. In such cases there will be an additional directory, `type1`, which contains the font files (normally in 'pfb' format).

3.1.2 Standard PostScript Fonts

In addition to the above directories, the psfonts area contains two zip files. If you need the files and have not got unzip (or pkunzip or winzip or...) then you can get a copy of unzip from the CTAN support area.

`lw35nfss` This zip archive expands to the subset of the `psfonts/adobe` tree that corresponds to the 'Standard 35' PostScript fonts as used in Adobe Laserwriter printers. If you are only interested in using fonts built into your printer, and not in using downloaded fonts, then just get this file rather than the large collection of metrics in `psfonts/adobe`.

`lw35pk` This zip archive contains bitmap fonts for the 'Standard PostScript fonts' in the usual PK format understood by most dvi drivers. This enables documents using Type 1 fonts to be previewed with dvi previewers that can not use outline font formats. (For example xdvi or the emtex drivers).

3.1.3 Tools and Extra Packages

There are a few remaining directories in psfonts.

`ts1` The LaTeX textcomp package and related utilities for accessing fonts in the 'text companion' encoding known as TS1 in LaTeX. These include the TC fonts that are distributed with the EC fonts, and suitably re-encoded fonts from the standard Type 1 font sets. This encoding contains many non alphabetic symbols that should match the current text font (rather than the math font). It includes currency symbols, superior digits, dagger signs, etc.

`mathcomp` A contributed package for using the text companion fonts in math mode.

`tools` The source for the scripts and utilities used for generating all these files.

3.2 Standard Psnfss Packages

By contrast to the packages and font descriptor files in the psfonts distribution, the psnfss distribution contains 'hand written' files. These are either used to set up popular *combinations* of the 'standard' fonts, or load alternative font sets for mathematics. Due to the nature of mathematics fonts, these latter packages are typically much more complicated internally than the one or two line packages that load text fonts. For the user, however, this complexity should not be apparent.

The first set of packages (all generated from the source file `psfonts.dtx`) load combinations of the Basic Adobe PostScript font set into LaTeX.

`times` As one might guess, this declares Times Roman as `\rmfamily`. For mainly historical reasons, this package also declares Helvetica as `\sffamily` and Courier as `\ttfamily`, so effectively ensuring that all text (but not mathematics) is set in the basic PostScript font set.

This is a convenience for the user who wants to replace all the text fonts by references to the basic Adobe fonts. It is an advantage to do this if you want to produce device independent and small PostScript documents for distribution. The disadvantage is that Times Roman, Helvetica and Courier, despite being the 'standard PostScript combination' look particularly horrible if placed next to each other at the same nominal size, as done by this package. Helvetica has a much larger 'x-height' (the height of the lower case letters) than Times Roman, so if sans serif and Roman text are mixed in-line, then the sans serif looks much too big. (This is not so much of a problem if the sans serif is only used for headings.) Courier is just too 'wide' when placed alongside Times Roman, which is a particularly compact font.

To partially compensate for these problems, the pslatex package (written by me, but currently distributed as a contributed package, not part of the core LaTeX distribution) is an alternative to the times package. It loads Helvetica scaled by 90% and loads Courier by way of a virtual font that condenses it by scaling the horizontal direction (only) by 85%. pslatex also contains a copy of the mathptm package (see below) so installs a Times-Italic based font set for use in mathematics.

`palatino` Declares Palatino as `\rmfamily`, and Helvetica and Courier as `\sffamily` and `\ttfamily`.

`helvet` Declares Helvetica as `\sffamily`. (Does not change the other families.)

`avant` Declares Avant Garde as `\sffamily`. (Does not change the other families.)

`newcent` Declares New Century Schoolbook as `\rmfamily`, Avant Garde as `\sffamily` and Courier as `\ttfamily`.

`bookman` Declares Bookman Roman as `\rmfamily`, Avant Garde as `\sffamily` and Courier as `\ttfamily`.

`chancery` Declares Zapf Chancery as `\rmfamily`.

The above packages only affect *text* fonts, not mathematics. `psfonts.dtx` contains one special package, written by Alan Jeffrey, which does affect the math setup.

`mathptm` This package uses a set of virtual files that use various built in or freely available fonts to make a set of fonts suitable for replacing the standard Computer Modern Math fonts. In the current release, bold fonts (and so the LaTeX `\boldmath` command) are not supported. The pslatex package referred to above contains an essentially verbatim copy of mathptm.

One may use mathptm as an example of the coding needed to make virtual fonts for mathematics based on other text italic fonts. How successful this will be depends to a certain extent how visually compatible are the symbols that are gathered from the various 'real' fonts that are used by the virtual math fonts. There are often good reasons for making such fonts (the main one being that documents using freely available fonts may be more easily placed on the Web in PostScript form), however the result is never likely to be as good as using fonts that have symbols that are *designed* to be visually compatible. For mathematics use within TEX, that currently restricts use to Computer Modern, or the commercial MathTime or Lucida Bright font sets described below.

The `psfonts.dtx` source file contains one other package:

`pifont` This declares the Zapf Dingbats font which contains an assorted mixture of symbols, and also defines new user level commands to access these symbols. See the package documentation, or *The LATEX Companion* for details.

3.3 Freely Available Type 1 Text Fonts

The next set of packages are contributed by Peter Dyballa. In fact these are just one-line packages loading the appropriate font. Most of the code is in the `fd` files which are generated from the same source file.

`charter` Defines `\rmfamily` to use Bitstream Charter.

`nimbus` Declares URW Nimbus Roman-Regular and URW Nimbus Sans-Regular as `\rmfamily` and `\sffamily`. These are essentially free clones of Times Roman and Helvetica.

`utopia` Defines `\rmfamily` to use Adobe Utopia-Regular.

3.4 Commercial Text Fonts

The following packages are generated from the source file `adobe.dtx`. They are a rather random selection from the large catalogue of fonts sold by Adobe.

`garamond` Garamond as `\rmfamily`, Optima as `\sffamily` and Courier as `\ttfamily`.

`basker` Baskerville as `\rmfamily`.

`mtimes` Monotype[2] Times as `\rmfamily`.

`bembo` Bembo as `\rmfamily`, Optima as `\sffamily` and the ever popular Courier as `\ttfamily`.

3.5 Adobe Lucida

The following two packages relate to the original Lucida font set, designed by Bigelow and Holmes and sold by Adobe. They are generated from the `alucida.dtx` source file.

[2] Not sure why this is generated from *adobe* source file.

`lucid` Declares Lucida Roman and Lucida Sans as the Roman and sans serif families, and Adobe Courier again as the monospaced font.

`lucmath` Lucida has a matching set of mathematics fonts suitable for TEX use. This package makes the required definitions to make these known to LATEX.

3.6 Lucida Bright

A newer and more extensive Lucida family, also designed by Bigelow and Holmes but in this case sold by Y&Y, is known as 'Lucida Bright' and 'Lucida New Math'. The LATEX support described here was written by Sebastian Rahtz and myself.

`lucidabr.dtx` This package (replacing the earlier lucidbrb and lucidbry packages) changes the LATEX defaults for both text and mathematics to use the Lucida Bright and Lucida New Math font collections. It has numerous options to control different aspects of the package and to control which of the fonts to use. (Lucida Bright contains several font families, including 'fax' and 'casual' etc, as well as variant forms of the math italic alphabet.)

The LATEX package and the font descriptor files for the math fonts are generated from this source file. The font descriptor files for the Lucida text fonts in the standard LATEX encodings are available from the psfonts area (in the bh) directory, after Bigelow and Holmes, the creators of these fonts.

The TEX support and font metrics are freely available, but the fonts themselves must be purchased separately.

`lucidabr.ins` LATEX installation file for Lucida Bright using the standardised 'Karl Berry' font names.

`lucidabr.yy` Alternative installation file. Use this instead of `lucidabr.ins` if you plan to install the fonts with their original font names, as sold by Y&Y. (In this case you do *not* need the `fd` files from the psfonts area.)

`lucidabr.txt` Introduction and installation guide for this package.

3.7 MathTime

The MathTime fonts are produced by Michael Spivak 'TEXplorators'. They are sold by Y&Y. The LATEX support was written by Frank Mittelbach and myself.

`mathtime.dtx` The mathtime package is mainly concerned with mathematics setup, although it selects Times, Helvetica and Courier as the text fonts if they have not already been set by another package. The MathTime mathematics fonts are specially designed to match Times Roman, but blend quite well with other text fonts that are of a similar weight. Computer Modern mathematics tends to look very 'light' if used with font families other than Computer Modern. The package has several options to control the font choices made.

`mtfonts.fdd` The source for the font descriptor files for MathTime mathematics fonts.

`mathtime.ins` Installation file. Note that this file may be edited in a couple of places depending on whether or not you have the extended 'MathTime Plus' font set which includes bold math support.

`mathtime.txt` Introduction and installation guide for this package.

3.8 Documentation and Other Files

`readme.txt` General introduction.

`psnfss2e.tex` User level documentation on the use of these packages.

`test0.tex` Testing accents and other encoding specific commands are working correctly using PostScript fonts.

`test1.tex` Test document that uses most of the 'Standard 35' fonts.

`pitest.tex` Test of the pifont package.

`mathtest.tex` Test of the mathptm package.

`makefile` Unix 'make' utility to automate installation of the packages.

`allpspk` Unix script that makes a test document using a specified font family and then uses dvips and its associated scripts to generate 'pk' versions of the fonts.

`makepk` Unix script that calls allpspk on some common fonts.

3.9 Psnfssx

Recently the psnfss collection has aquired a close cousin, psnfssx, distributed as a contributed package from `macros/latex/contrib/supported/psnfssx`. This contains some lesser used or nonstandard packages, related to PostScript support. Of particular interest might be the ly1 files (contributed by myself) in that directory which provide the LaTeX support for the 'texnansi' encoding promoted by Y&Y by way of an `LY1` option to the fontinst package.

This psnfssx collection also contains some obsolete versions of packages formerly in psnfss; this material is provided for historical interest only. Use at own risk!

4 The Babel Distribution

The babel package is distributed from `latex/packages/babel` and is supported via the LaTeX bug reporting address, but has origins predating the current LaTeX release. As well as supporting LaTeX it contains support for plain TeX (and formats such as AMSTeX or eplain that are based on plain). Primarily babel is the work of Johannes Braams, with contributions for specific language files by numerous people.

Babel consists of a 'kernel' that extends LaTeX with a mechanism for switching between specified languages. Part of this kernel (related to hyphenation) must be loaded when the LaTeX format is made to get the full benefit of hyphenation tables for multiple languages. For each language, or related group of languages, supported by babel there exists a language-specific code file. This will offer translations of the fixed text strings used in the standard LaTeX classes, such as 'Table of Contents', 'Figure', etc., and may also offer language-specific 'shorthands' that make typing common constructs easier (for example the german option provides the construct '`"ff`' to produce 'ff' that would hyphenate to 'ff-f' if it fell at the end of a line). The language file may also modify the typesetting to support the normal conventions of that language. For example the french option modifies the spacing around punctuation marks in text.

4.1 Babel Kernel

`babel.sty` The main interface to babel. The user specifies all languages to be used in a document as options to this package, the last option specified is the default language for the document. So for example

```
\usepackage[french,german]{babel}
```

would enable the use of French and German conventions within the document, with the default language being German.

`hyphen.cfg` The standard LaTeX interface to hyphenation. When the LaTeX format is being made, this file is input if it exists, to setup the required hyphenation patterns. In the base LaTeX distribution there is no such file, and so a default action is taken which loads the original TeX patterns for American English. The babel distribution provides this configuration file (generated from `babel.dtx`) which defines some core functionality, and then reads `language.dat` to specify which hyphenation files to load.

`language.dat` This file must be edited to specify which language hyphenation files to load, and the name of the external file which contains the hyphenation table for each such language (and optionally a second external file, typically containing hypenation exceptions). Note that hyphenation files *must* be specified here, and so loaded when the format is made. This is a restriction of the underlying TEX system. Documents using other languages not specified here may still be processed, and babel will translate any fixed text strings, but it will not be able to correctly hyphenate that language. A default hyphenation will be used (most likely English) which may or may not be suitable depending how far the language differs from English.

`switch.def` This file is also generated from the same `babel.dtx` source. If babel is used as a package but was not used when the format was made, then the core functionality normally provided by `hyphen.cfg` will not be present. The package will detect this, and so input this file to provide the necessary definitions.

4.2 Language-Specific Files

The implementation of the language-specific code for each language within babel is contained in files with extension '`.ldf`' (language definition files). These are not directly input by the user, but specified as options to the babel package. Normally the option name is the same as the file name, except where noted below. Some similar languages or dialects are supported by the same external file, and some options are available in more than one name; such aliases are noted in parentheses in the list below.

Most languages also have a file with extension `.sty`; however this is just offered for compatibility with older versions of Babel and of LATEX, or for use with plain TEX based formats. In normal LATEX usage only the `.ldf` file is used.

`bahasa` Support for the Bahasa language.

`basque` Support for the Basque language.[3]

`breton` Support for the Breton language.

`catalan` Support for the Catalan language.

`croatian` Support for the Croatian language.

`czech` Support for the Czech language.

`danish` Support for the Danish language.

`dutch` The dutch and afrikaans options.

[3] Not in the current release, planned for babel 3.7.

`english` The american (USenglish) and british (UKenglish) options. The option english refers to either British or American English, depending on the local installation.

`esperant` The esperanto option.

`estonian` Support for the Estonian language.

`finnish` Support for the Finnish language.

`frenchb` Support for the French language (the corresponding options are french (frenchb) or francais. If the french option is used then `french.ldf` will be used (from the GUTenburg french package) if it is available.

`galician` Support for the Galician language.

`germanb` The austrian and german (germanb) options.

`kannada` Support for the Indian language, Kannada.[3]

`irish` Support for the Irish Gaelic language.

`italian` Support for the Italian language.

`lsorbian` The lowersorbian option.

`magyar` The magyar (hungarian) options.

`norsk` Support for the Norwegian languages with options norsk, nynorsk.

`polish` Support for the Polish language.

`portuges` The brazil (brazilian) and portuges (portuguese) options.

`romanian` Support for the Romanian language.

`sanskrit` Support for the Sanskrit language, transliterated to latin script.[3]

`scottish` Support for the Scottish Gaelic language.

`slovak` Support for the Slovakian language.

`slovene` Support for the Slovenian language.

`spanish` Support for the Spanish language.

`swedish` Support for the Swedish language.

`turkish` Support for the Turkish language.

`usorbian` The uppersorbian option.

`welsh` Support for the Welsh language

Babel version 3.6 sees the welcome (re)introduction of support for non-latin scripts. It is probably fair to say that this support is still more experimental than the support for latin scripts. One problem, not directly under babel 'control', is that the TEX encodings for Greek and Cyrillic (corresponding to T1 for European Latin scripts) have not yet been finalised or agreed. Currently babel uses two 'locally defined' encodings, LWN and LGR.

`greek` The greek option, which utilises the 'kd' Greek fonts.

`russianb` The russian option, which utilises the 'LH' fonts.

Two separate packages are currently in preparation which will be distributed, together with suitable fonts and hypenation tables, from CTAN. These will extend babel with options for the Ethiopian and Ukrainian languages.

4.3 Compatibility Files

The distribution contains the following two source files which generate files which enable the use of babel with formats based on plain TeX (and also the old LaTeX 2.09 release).

`bbcompat` The source for compatibility mode files. Most languages are provided with a 'package' with extension `.sty`. This just inputs the corresponding language definition file and should never be needed using the normal LaTeX interface.

`bbplain` The source for the `plain.def` file allowing the use of babel with plain TeX.

4.4 Installation Script and Font Descriptor Files

`babel.ins` Unpacks the babel distribution from the documented source files

`cyrillic.fdd` Font descriptor files for Cyrillic fonts in 'LCY' encoding.

`greek.fdd` Font descriptor files for Greek fonts in 'LGR' encoding.

4.5 Documentation

4.5.1 ASCII Text Files

`00readme.txt` The distribution guide.

`install.txt` How to install Babel.

`install.mac` How to install Babel with OZTeX.

`CyrillicFonts.txt` Further notes on the Cyrillic installation.

`GreekFonts.txt` Further notes on the Greek installation.

4.5.2 TeX Documents

`tb1202` The source of the original article that appeared in *TUGboat*, Volume 12 (1991), No. 2.

`tb1401` The source of an update article that appeared in in *TUGboat*, Volume 14 (1993), No. 1.

`tb1604` The source of an update article that never appeared in *TUGboat*, but was presented at EuroTeX 1995, Arnhem.

4.6 Example File

`language.skeleton` An example file that can be used to build new language definition files from scratch.

5 Coming Soon

Part 4 of this tour will describe the files of the amsfonts and amslatex distributions of packages produced by the American Mathematical society.

◇ David Carlisle
Department of Computer Science
Manchester University
Oxford Road
Manchester, England M13 9PL UK
`carlisle@cs.man.ac.uk`

Late Breaking News

Production Notes

Mimi Burbank

Once again we're late! That is the bad news. The good news is that the next issue of *TUGboat* should be reaching your desk in about a month with a present included therein.

We include in this issue another article from the TUG'96 Conference, by Yannis Haralambous. This article requires special typesetting capabilities unavailable to the production team at this time, and so we asked the author to typeset it for us.

Output The final camera copy was prepared at SCRI on several of the local platforms to test the new *TeX Live* setup (see below): Alphastation 2100/500 running OSF1 v3.2; IBM RS6000 workstation, running aix 4.1.1; IBM SP2 RS6000 node, running aix 3.2.5; Silicon Graphics SGI workstation, running Irix 5.3; and a Dual Pentium Pro 200, running Linux i686. We used the *TeX Live* setup (Version 2), which is based on Karl Berry's *Web2c* TEX implementation version 7.0; we also worked with Thomas Esser's teTEX package, loaded from the *TeX Live* (Version 1) CD-ROM. PostScript output at 600dpi was produced using Radical Eye Software's dvipsk 5.66a and printed on a QMS 860 printer.

I am so pleased to announce that *TeX Live* (Version 2) is "alive and doing well!"

Coming Next Issue

Well, we have a gift for you in the next issue. The *TeX Live* (Version 2) CD-ROM is being pressed as this issue goes to press, and will be included in the next issue of *TUGboat*, along with the documentation! The CD-ROM contains a very large TDS-compatible tree of macros, fonts and documentation, as well as a runnable TEX system for a dozen varieties of Unix; also included is a system for win32 (Windows 95 or NT) and packages for Windows 95, DOS and Macintosh ready to install.

We also plan to add some material which was held over from the last issue of *TEX and TUG News* (1994), a series of abstracts from *Die TEXnische Komödie*, and *Les Cahiers GUTenberg*, as well as a manual by B. Jackowski on "METAFONT: practical and impractical applications".

TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home

TUG'97

TEX Comes Home

TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home
TEX Comes Home

July 28th through August 1st

Lone Mountain Conference Center

San Francisco, California

We would like to extend an invitation to TEX users around the world to join us in one of the most beautiful and exciting cities in the world. The conference will be held at The University of San Francisco Lone Mountain Conference Center, located on a 55-acre hilltop refuge with views of the Pacific Ocean, San Francisco Bay and the dramatic downtown skyline, and is a short walk from Golden Gate Park. Economical guest housing on campus is available to conference attendees.

The facility is about a 15-20 minute bus ride from the center of San Francisco, between town and the ocean beach. A single bus will take passengers from the ferry terminal building on the Embarcadero through town, past the University and to the beach.

The center is very well equipped, with several lecture halls and dining facilities. Shuttle transportation is available to take conference attendees who are staying on campus to the center. Computer accounts will be given out to all conference attendees.

Please visit our TUG'97 Home Page home page at `http://www.tug.org/tug97`, and note that a preliminary schedule has been posted, together with links for travel and lodging information. Anyone lacking a web browser is welcome to contact the Conference Committee at `tug97@mail.tug.org` for information via email.

We hope to arrange bursary funds for support of students and those participants who demonstrate need. For bursary requests, send email to `bursary@mail.tug.org` or by postal mail to TUG'97 Bursary Committee, 135 Center Hill Road, Plymouth, MA 02360 USA.

Deadlines — Revised

Submission of Abstracts	March 14, 1997
Notification of acceptance to authors	March 21, 1997
Preliminary Papers Due	May 2, 1997
Preprint Deadline	May 16, 1997
Camera Ready Copy Deadline	June 20, 1997

New members of the TUG Board

As announced earlier in this issue, the number of candidates for open positions on the Board of Directors was fewer than the number of positions, hence, according to the Election Procedures, the candidates who did submit nomination papers were declared elected. Since there will therefore be no ballot, the biographies and personal statements that would have appeared on the ballot are presented here, to introduce these individuals to the membership.

The new Board members are listed in alphabetical order. Their terms of office all extend to the annual meeting in the year 2001.

Barbara Beeton
For the Elections Committee

Donna Burnette

SCRI
Florida State University
Tallahassee, FL 32306-4052
U.S.A.
Internet: `donna@scri.fsu.edu`

Biography:

My job over the past 10 years at the Supercomputer Computations Research Institute (SCRI) has evolved from TEXnical typing, to TEX software installation and support, to full support of VMS systems and their software. We support a large community of research scientists, university faculty and administrative staff. I have written and deciphered many TEX programs during this period and become quite knowledgeable in the interrelationships between TEX and its utilities (dvips, xdvi, etc.). I designed and implemented the TEX programs required to maintain our TEX publications database and provide information on a variety of disciplines in many different formats. I designed the program that generates Florida State University masters and doctoral theses compliant with their graduation requirements. Word of this spread quickly, so I routinely get requests from other departments to copy it to their systems. I have continually provided my TEX expertise to those wishing to design and distribute their own documents and have assisted with the installation/upgrade of Unix TEX systems prior to the teTeX distributions and the creation of the *TUGboat* Production Group work on installation generation here locally.

Personal statement:

As a board member I would be interested in actively involving members of our TEX community in extending the installation standardization process to include the VMS operating system.

Mimi Jett

ETP Harrison
2906 N.E. Glisan
Portland, OR 97232
U.S.A.
Internet: `mimi@etp.com`

Biography:

President and CEO of ETP Harrison, a full-service electronic publishing firm located in Portland, Oregon. We began the company as a provider of *troff* typesetting services, but moved into TEX in 1987 at the request of textbook publishers. With the help and support of the TEX Users Group we found the resources necessary to become a serious provider of composition and related services. Prior to starting this business, I began several other businesses, including Oregon Serigraphics (1973); The Renaissance Press (1977); The Resource (1983); and Marketing Communications (1985), all related to print or advertising. ETP Harrison employs 27 people and 6 freelancers full time. I have served on numerous boards, including the TUG board in 1992–93.

Personal statement:

TUG has been a vital contributor to our success, and I am eternally grateful. The benefits of membership have been questioned as long as I can remember, and I would like to contribute the time and resources to help define, and enhance, the value of the organization. Only through a thriving population will the group continue to grow and contribute to the success of others needing the support of the TEX Users Group.

I feel that my experience in management and leadership qualify me to serve as a member of the Board of Directors, and wish to serve in that capacity.

Patricia Monohon

[not delivered for publication]

Arthur Ogawa

40453 Cherokee Oaks Drive
Three Rivers, CA 93271
U.S.A.
Internet: `ogawa@teleport.com`

Biography:

PhD Physics, UC Berkeley, I began working with TEX in my job at the Stanford Linear Accelerator Center in about 1982. After a two-decade long career as an experimental physicist, I decided in 1987 to work as a publishing consultant and have run TEX Consultants since that time. I presently live and work in Three Rivers, California, USA.

I have assisted numerous people worldwide in using TEX and have worked with a number of commercial firms in setting up and managing TEX-based publishing enterprises. For more personal information, please see `http://www.teleport.com/~ogawa`.

Personal statement:

I decided to offer my services on the TUG board because I believe I can make a positive contribution to TUG's future. TUG is an organization worth turning around, and my intention is to do just that—*with your help.*

The most serious indicator of TUG's troubles is its declining membership, so my goal is to address that issue squarely. I firmly believe that TUG can accomplish this task by realigning its services and benefits with its current and future membership.

I say that TUG can do this, not that *I* can do this, simply because no person can singlehandedly achieve this goal. So, I will be relying on your participation, not just as a member of TUG, but as a TUG activist and volunteer.

TUG needs your help to identify what member benefits TUG should offer in order to be truly relevant to the needs of TEX users worldwide. Then your assistance will be needed in providing those benefits, whether through the vehicle of *TUGboat*, training classes, the annual conference, our website, or other activities. I believe we can make TUG once again the vehicle for volunteers to better the lot of TEX users everywhere.

If you share my goal of revitalizing TUG through its own membership, please commit your energies to doing what is needed to make TUG the success it once was. I invite your input: my email address is `mailto:ogawa@teleport.com`.

TUG's meaning is only what we give it, and its very justification lies in what it gives its members.

Petr Sojka

Faculty of Informatics
Masaryk University Brno
Burešova 20
602 00 Brno, Czech Republic
Internet: `sojka@informatics.muni.cz`

Biography:

I studied Math/Computer Science at Masaryk University Brno, where I work now as an Assistant Professor. I am currently president of CSTUG, the zech and Slovak TEX Users Group.

Personal statement:

My potential activities for TUG could be based on my experience gained in

– revitalization of CSTUG
– managing/consulting/leading various publication and other projects based on TEX
– maintenance of TEX-related services
– participation in organizing scientific conferences

I expect to help for better communication/exchange between LUGs in Central/East Europe and the rest of the world.