TUGboat

Volume 30, Number 1 / 2009

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions

2008 dues for individual members are as follows:

- Ordinary members: $85.
- Students/Seniors: $45.

The discounted rate of $45 is also available to citizens of countries with modest economies, as detailed on our web site.

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, as well as software distributions and other benefits. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is $95 per year, including air mail delivery.

Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

TEX is a trademark of the American Mathematical Society.

Addresses

TEX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Telephone

+1 503 223-9994

Fax

+1 206 203-3960

Web

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

Electronic Mail
(Internet)

General correspondence, membership, subscriptions:
`office@tug.org`

Submissions to *TUGboat*, letters to the Editor:
`TUGboat@tug.org`

Technical support for TEX users:
`support@tug.org`

Contact the Board of Directors:
`board@tug.org`

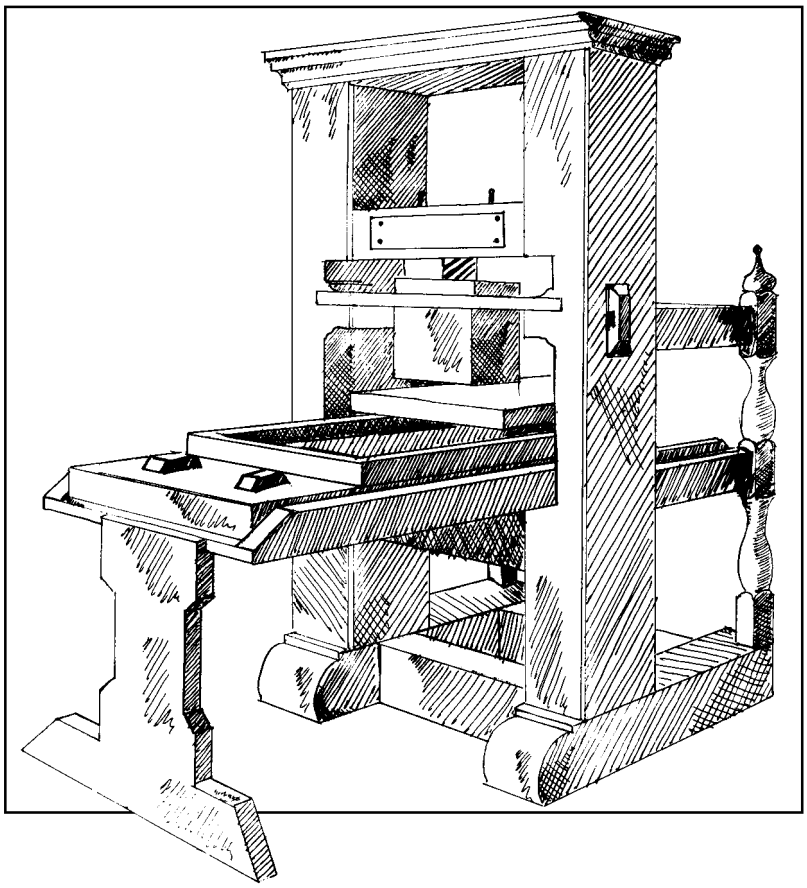
Have a suggestion? Problems not resolved?
The TUG Board wants to hear from you:
Please email `board@tug.org`.

[printing date: July 2009]

Printed in U.S.A.

TUGBOAT

The Communications of the TEX Users Group



Volume 30, Number 1, 2009

TEX Users Group

TUGboat (ISSN 0896-3207) is published by the TEX Users Group.

Memberships and Subscriptions

2008 dues for individual members are as follows:

- Ordinary members: $85.
- Students/Seniors: $45.

The discounted rate of $45 is also available to citizens of countries with modest economies, as detailed on our web site.

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, as well as software distributions and other benefits. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is $95 per year, including air mail delivery.

Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group, as well as providing a discounted group rate and other benefits. For further information, see `http://tug.org/instmem.html` or contact the TUG office.

TEX is a trademark of the American Mathematical Society.

Addresses

TEX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Telephone

+1 503 223-9994

Fax

+1 206 203-3960

Web

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

Electronic Mail
(Internet)

General correspondence, membership, subscriptions:
`office@tug.org`

Submissions to *TUGboat*, letters to the Editor:
`TUGboat@tug.org`

Technical support for TEX users:
`support@tug.org`

Contact the Board of Directors:
`board@tug.org`

Have a suggestion? Problems not resolved?
The TUG Board wants to hear from you:
Please email `board@tug.org`.

[printing date: July 2009]

Printed in U.S.A.

The development of mathematical notation ... was nothing short of revolutionary. ... [T]he notation was universal; it could be understood no matter what your national language was.

Arika Orent
In the Land of Invented Languages
(2009)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 30, NUMBER 1 • 2009
PORTLAND • OREGON • U.S.A.

TUGboat

This regular issue (Vol. 30, No. 1) is the first issue of the 2009 volume year. No. 2 will contain the TUG 2009 (Notre Dame) proceedings and No. 3 will be a joint publication of the EuroTEX 2009 conference in The Hague.

TUGboat is distributed as a benefit of membership to all current TUG members. It is also available to non-members in printed form through the TUG store (`http://tug.org/store`), and online at the *TUGboat* web site, `http://tug.org/TUGboat`. Online publication to non-members is delayed up to one year after an issue's print publication, to give members the benefit of early access.

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 deadline for receipt of final papers for the upcoming proceedings issue is August 19, 2009. More information about this and all conferences are available at `http://tug.org/meetings.html`.

The next regular issue will probably be in spring 2010. As always, suggestions and proposals for *TUGboat* articles are gratefully accepted and processed as received. Please submit contributions by electronic mail to `TUGboat@tug.org`.

The *TUGboat* style files, for use with `plain` TEX and LATEX, are available from CTAN and the *TUGboat* web site. We also accept submissions using ConTEXt. More details and tips for authors are at `http://tug.org/TUGboat/location.html`.

Effective with the 2005 volume year, submission of a new manuscript implies permission to publish the article, if accepted, on the *TUGboat* web site, as well as in print. Thus, the physical address you provide in the manuscript will also be available online. If you have any reservations about posting online, please notify the editors at the time of submission and we will be happy to make special arrangements.

Other TUG Publications

TUG is interested in considering additional manuscripts for publication, such as manuals, instructional materials, documentation, or works on any other topic that might be useful to the TEX community in general.

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 at `tug-pub@tug.org`.

TUGboat Advertising

For information about advertising rates and options, including consultant listings, write or call the TUG office, or see our web pages:
`http://tug.org/TUGboat/advertising.html`
`http://tug.org/consultants.html`

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 should not be considered complete.

METAFONT is a trademark of Addison-Wesley Inc.
PostScript is a trademark of Adobe Systems, Inc.
TEX and $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX are trademarks of the American Mathematical Society.

General Delivery

From the President

Karl Berry

In memoriam

The TEX community has lost two important contributors in 2009. Helmut Kopka, co-author of the renowned book *A Guide to LATEX*, passed away in January. A memorial by his fellow author Patrick Daly appears in this issue. Eitan Gurari, creator of the TEX4ht software package and frequent participant at TUG conferences, unexpectedly passed away in June; he was going to be a featured speaker at the TUG 2009 conference. A profile of Eitan will appear in the proceedings issue.

We will greatly miss our friends and colleagues.

TEX Collection 2009

Work on TEX Live 2009 continues, and anyone willing to try the pretest releases is welcome; details at `http://tug.org/texlive`. Thomas Feuerstack has also prepared a new version of proTEXt for 2009, which can be found at `http://tug.org/protext`.

Among many other changes, this year's release will include a new version of the Computer Modern Type 1 fonts, painstakingly prepared by the AMS to incorporate all of Knuth's glyph shape changes, the hinting previously released by Y&Y, and other fixes.

TEX Live 2009 will also include the new front end TEXworks, written by Jonathan Kew (`http://tug.org/texworks`), a project supported by TUG and other user groups. We expect executables for Windows and Mac OS X to be included in the distribution, while (for painful technical reasons) binaries for other platforms will be downloadable from the TEXworks web site. TEXworks can also be tested now.

The Asymptote graphics program will also be included on at least the major platforms. More about Asymptote is in two articles by John Bowman and his colleagues: one in this issue about its 3D support and a more introductory article in *TUGboat* 29:2 (`http://tug.org/TUGboat/Contents/listauthor.html#Bowman,John`).

Book of TUG interviews

We have finished production of the book of interviews done over the past several years (`http://tug.org/interviews`). It will be publicly available through Amazon and other online (and perhaps physical) stores, and we will also be making it available to TUG members at a large discount. By the time you are reading this, the web site should have the details.

And now that the book is done, regular interviews will resume. Perhaps there will be a volume 2 in the future.

Server hardware

The machine that is currently `tug.org` has served us well for several years, but it has become time to upgrade the hardware. The transition is underway and should be complete — more or less invisibly, we hope — by the fall. It will continue to run GNU/Linux. We will be re-using the current machine for other purposes.

I'd like to take this opportunity to express my appreciation to DAIMI, the computer science department at the University of Aarhus, for hosting `tug.org` for so many years, and more personally to DAIMI senior staff member Michael Glad for much advice and help over the years, and of course to my fellow system administrator, vice-president, and good friend Kaja Christiansen, who made (and makes) the hosting possible.

Conferences

TUG 2009 (`http://tug.org/tug2009`) will be taking place as the issue of *TUGboat* is printed. Its proceedings will be the second issue for 2009.

EuroTEX 2009 (`http://ntg.nl/EuroTeX2009`) will take place at The Hague in The Netherlands; that proceedings will be the third issue.

Looking ahead to 2010, it is probable that the TUG conference will be held in San Francisco around the end of June, while EuroTEX will be in Italy towards the end of the summer. The web page `http://tug.org/meetings` will be updated as plans are finalized.

Joint memberships

We are very pleased that the Italian TEX user group, GuIT, is now offering joint memberships with TUG. The other groups with which we have joint membership agreements are DANTE (German), DK-TUG (Danish), NTG (Dutch), and UK-TUG (United Kingdom). It is great to see such widespread collaboration in our common cause.

◇ Karl Berry
`http://tug.org/TUGboat/Pres/`

Editorial comments

Barbara Beeton

Helmut Kopka, 1932–2009

Helmut Kopka's *LaTeX — Eine Einführung* was one of the first non-English books for LaTeX; it rapidly became a standard reference. Joining forces with Patrick Daly, he revised the book (as *Guide to LaTeX*) for an English audience, where it has gained an even larger following.

Helmut passed away early this year after a short illness. His clear exposition should be a goal for all aspiring technical writers. A remembrance by Patrick Daly appears elsewhere in this issue.

Eitan Gurari, 1947–2009

We recently learned of the sudden and unexpected death of Eitan Gurari, on June 22. Eitan was the creator of TeX4ht, a system used widely for publishing research papers on the Internet. His recent research interests included hypertext processing and Braille production; he was scheduled to give a talk on his Braille work at the upcoming TUG meeting.

Eitan's quiet presence and his contributions to the TeX toolkit will be sorely missed.

A short history of type

Earlier this spring, I attended a lecture at the Museum of Printing in Andover, Massachusetts, entitled "A Short History of Type". The speaker was Frank Romano, Professor Emeritus, Rochester Institute of Technology; Frank occupied the same chair (Melbert B. Cary Distinguished Professor) held previously by Hermann Zapf and currently by Chuck Bigelow.

Over a span of two hours, Frank held the full attention of a small audience describing the winding road from moveable type (Gutenberg and associates, Garamond, Baskerville, and others) through machine typesetting (Mergenthaler, Samuel Clemens (Mark Twain), et al.), film-based phototype (Photon, Compugraphic, Alphatype, ...), and into the digital PostScript era. (He failed to cover the pre-PostScript digital machines, with some of which fortunate early TeXxies spent many hours — an omission to which I called his attention afterwards.)

The lecture was videorecorded by students from a local tech college; I've asked for a copy of the recording, and if it's available in time, I hope to take it with me to the TUG meeting, to share with other attendees.

Frank is scheduled to present a related lecture, "A Short History of Printing", at the Museum on September 25. That's listed in this issue's calendar, with a web link. If you're likely to be in the Andover area at the end of September, by all means sign up! Frank is a delightful speaker, full of fascinating information, and tolerant of off-the-wall questions.

And the Museum itself is chock full of amazing machines and artifacts of the printing industry, including the entire Mergenthaler font library — the original drawings for all the fonts ever produced for Linotype machines. A ongoing development project for the Museum's library will ultimately provide

> web-based public access to electronic records of the Museum's books and ephemera. ... Our goal is that eventually researchers will be able to research and find records for all items in the library and archives. We also aim to provide similar access to records for our collections of artifacts.

For lots more information, go to the Museum's web page, `www.museumofprinting.org`. And visit the Museum; like TUG it's a 501(c)(3) organization, run entirely by volunteers, and needs (and deserves) all the support it can get.

◇ Barbara Beeton
American Mathematical Society
201 Charles Street
Providence, RI 02904 USA
`tugboat (at) tug dot org`

Helmut Kopka, 1932–2009

Patrick W. Daly

In the English-speaking LaTeX world, the name *Helmut Kopka* is most widely associated with my own, as the authors of the *Guide to LaTeX*. In Germany, he is known as the single author of a three-volume set of LaTeX manuals: *Einführung* (Introduction), *Ergänzungen* (Additions), *Erweiterungen* (Extensions).

Helmut's interest in first TEX, and then LATEX, started while he was on an extended stay in the United States, in preparation for a major scientific project on which he was working. This was in the late 80s, when word processing programs were proliferating; Helmut even started working on his own until he was introduced to TEX; recognizing a vastly superior product, he quickly embraced it. He did make his own initial contribution with the DVI driver `dvi2pcl` for the LaserJet printers.

Back at his home institute in Germany, he introduced LATEX as the standard text system for the secretarial work there, at a time when computers were invading the non-scientific offices. He complemented this by writing a series of notes, or lectures, explaining to the secretaries how this system was to be used. These notes later became the basis for his first LATEX textbook, the *Einführung*.

The success of this book in Germany was so great (he once told me that it sold more copies in Germany than Lamport in the world, but I cannot confirm this) that the publisher Addison-Wesley Deutschland considered an English translation. This was where I came into the picture.

Helmut was nothing if not direct. He knew that I was a major user of LATEX, that I was writing *style* files, as packages were called back in the 2.09 days, and as a Canadian was a native English speaker. It was the middle of an Open-House Day in the Institute, we were besieged with thousands of visitors, many from across the recently opened Iron Curtain a mere 20 km away. In the midst of all this, Helmut comes to me and asks if I would be interested in translating his LATEX manual into English, as though this could be done in a day or two. I answered that I would think it over. The rest is LATEX history.

The first edition of *A Guide to LATEX* was very much a translation. While working on it, I was impressed by Helmut's skill at explaining complex ideas very simply, and by the examples he used to illustrate the points. When I was half-way through the translation (it took a year) I realized I could start using it myself as my own reference manual. He was an enthusiastic teacher; a visit to his office with a simple question could result in a fascinating lecture on how METAFONT works. I very much appreciated the material that he had given me to work with.

With the second edition, *A Guide to LATEX 2ε*, I began the rewriting needed to explain the new LATEX version that was about to come out. I consulted Helmut all the time and he incorporated many of my changes into the German equivalent. When I wanted to add an additional appendix, he was hesitant: the original book had 9 chapters and 6 appendices (A–F) and the 7th appendix H would destroy the nice hexadecimal nature of the layout. He did acquiesce in the end.

LATEX was of course only part of Helmut's life. He was born in Dortmund, studied physics in Göttingen with a degree in fluid dynamics, joined the Max-Planck-Institut für Aeronomie in Lindau (am Harz) in 1963, where he started applying advanced computer techniques when high level computer languages were in their infancy. His specialty was now ionospheric physics. In 1974 he became part of a new project called Heating, a very powerful shortwave transmitter designed to perturb and heat electrons in the ionosphere. These active experiments in ionospheric and plasma physics were carried out near Tromsø, in northern Norway, where the sister project, the EISCAT incoherent scatter radar facility, was also located. He was to become a co-leader of this project, financed by the Max Planck Society, the Max-Planck-Institut für Aeronomie, and the German Research Foundation. It was his task to design an antenna and transmission line system which could be realized within the modest budget available. He managed this magnificently through the imaginative use of his physics, mathematics, and computing skills.

Helmut was also very politically engaged. He served as mayor for a few years in his village and even considered going into state politics. He was instrumental in getting a workers' council established in the Institute against the wishes of the director. When the then Ministerpresident of Lower Saxony, Gerhard Schröder, visited the Institute as part of his campaign to become Chancellor of Germany, he insisted on meeting his "old friend Helmut".

Helmut retired from his duties at the Institute in 1997 at which time he began a long battle against cancer, which he ultimately won. He was still a regular visitor to the Institute, coming for lunch every Tuesday with the others from his old group. He enjoyed telling stories about his grandchildren. And he continued to work on his LATEX manuals, being very proud that they now appeared as eBooks.

Helmut Kopka passed away on January 7, 2009, after a short illness; we now mourn a talented, dedicated, affectionate colleague and friend.

◇ Patrick W. Daly
Max-Planck-Institut für
Sonnensystemforschung
37191 Katlenburg-Lindau
Germany
daly (at) mps dot mpg dot de

Software & Tools

DVI specials for PDF generation

Jin-Hwan Cho

Abstract

DVIPDFM(x) manages various PDF effects by means of DVI specials. Appropriate documentation of DVI specials, however, is not easy to find, and exact functionality is not simple to catch without reading the source code of DVI drivers. This paper deals with the DVI specials defined in DVIPDFM(x) that are mainly used for PDF generation. We discuss the features of those specials with some examples, many of which are not documented elsewhere.

1 Introduction

DVI, the output file format of D. E. Knuth's TEX, is not widely used at present compared with PDF, the output format of pdfTEX. It is rather old[1] and obsolete, but it has powerful aspects nonetheless: *simplicity* and *compactness*.

These aspects make it possible to manipulate DVI files in an easy and fast way. Many DVI utilities were developed to convert the DVI format to other file formats including PostScript and PDF. It is also possible to edit DVI files directly by the use of DVIasm [5, 6, 7].

Twenty years ago, at the time PostScript dominated the printing world, nobody expected a new format would replace PostScript. PDF is not eternal either. In future, when a new format surpassing PDF appears, DVI will be the first format in the TEX world that can be converted to the new format. Notice that LuaTEX, considered to be the next generation of pdfTEX, still supports the DVI format.

There are two popular ways to convert DVI to PDF. The first one is a two-way conversion, from DVI to PostScript with DVIPS, and then from PostScript to PDF with a distiller. Adobe Acrobat Distiller is the oldest commercial program, and Ghostscript is the most popular distiller in the TEX world. Mac OS X also has its own distiller.

Adobe designed the pdfmark operator [2] for its distiller to support PDF features that are not expressible using the standard PostScript operators. The pdfmark operator is given in the TEX source by means of a DVI special command. Note that it is not DVIPS but a distiller that processes the pdfmark operator.

[1] DVI was designed by David R. Fuchs in 1979.

Mark A. Wicks' DVIPDFM [11] introduced the other way of converting DVI directly to PDF. He also designed new DVI specials based on the pdfmark operator to support various PDF features. The new specials, however, lacked some functionality in practical use so that not many PDF features could be obtained compared with pdfTEX.

One of the main goal of DVIPDFMx, an extension of DVIPDFM that grew out of the CJK[2] support, was to provide as many PDF features as pdfTEX [3]. DVIPDFMx extended the functionality of some special commands of DVIPDFM, and designed new special commands having a similar functionality of pdfTEX's own primitives. Furthermore, DVIPDFMx has several powerful features not available in DVIPDFM.

- Support 16-bit character sets (CJK encodings and Unicode) with CID-keyed font technology.
- Support various font formats including OpenType, TrueType, etc.
- Use CFF font format for embedded Type1 PostScript fonts so that the size of the PDF output is quite small compared with pdfTEX's output.
- Support extended TEX engines, e.g., Omega, Japanese pTEX, XeTEX (via XDVIPDFMX).[3]

The TODO list of DVIPDFMx had contained one outstanding item for a long time: supporting Till Tantau's beamer package [9], that is widely used for PDF presentation. In fact, this package does not handle DVI specials in a direct way. Instead, the graphics part comes from the same author's PGF package [10], and the other PDF effects come from the hyperref package [8].

DVIPDFMx has supported full functionality of the PGF package since June 2008.[4] Nonetheless, the navigation buttons usually shown in the lower right corner of the presentation still did not work, although they were displayed correctly. The source code[5] implementing the buttons was

```
\def\beamer@linkspace#1{\vbox to7.5pt{}\kern#1}
```

The code above generates an *empty* box that will be surrounded by the two special commands, '`pdf:bann`' (before) and '`pdf:eann`' (after). Unfortunately, neither DVIPDFM nor DVIPDFMx construct any annotation in the case of an empty box. Another special command '`pdf:ann`' must be used instead for

[2] Chinese, Japanese, and Korean.

[3] Upcoming version of DVIPDFMx will support the DVI output generated by LuaTEX.

[4] The DVIPDFMx driver that works with the PGF package included in TEX Live 2008 can be downloaded from `http://project.ktug.or.kr/dvipdfmx/contrib/generic/`.

[5] `http://mirror.ctan.org/macros/latex/contrib/beamer/base/beamerbasenavigation.sty`

this purpose. That was the exact reason why the navigation buttons did not work.

Why did the author of the beamer package make such a mistake? As a matter of fact, it was not his fault because no statement could be found about that functionality in the manual of DVIPDFM [11]. This unhappy story led to this paper.

The author gave a presentation [4] at TUG 2005, in which the different behaviors of DVI specials of DVIPS, DVIPDFM, and DVIPDFMx were discussed. DVI specials for PDF generation, however, were not fully discussed at that time. The main objective of this paper is to bridge this gap.

We will discuss in the following sections the features of DVI specials defined in DVIPDFM for PDF generation, and the extended features given by DVIPDFMx. The author hopes this paper would be useful for package writers who are finding appropriate information on DVI specials.

2 Named PDF objects

There are two kinds of named objects, built-in and user-defined PDF objects.

2.1 Built-in named objects

Built-in objects defined in DVIPDFM(x) are listed in Table 1. We refer to [2, p. 12] and [11, p. 5] for pdfmark and DVIPDFM built-in objects, respectively. Notice that it is not allowed to modify the contents of the last five built-in objects in Table 1.

`@catalog`	catalog dictionary [1, p. 139]
`@docinfo`	(DVIPDFMx only) document information dictionary [1, p. 844]
`@names`	name dictionary [1, p. 150]
`@pages`	root page tree node [1, p. 143]
`@resources`	resource dictionary of current page [1, p. 154]
`@thispage`	current page object [1, p. 145]
`@prevpage`	reference only
`@nextpage`	reference only
`@page`n	reference only
`@xpos`	reference only
`@ypos`	reference only

Table 1: Built-in objects defined in DVIPDFM(x)

2.2 User-defined named objects

Two special commands are provided by DVIPDFM(x) for user-defined objects. One is to define a named object, and the other is to add content to the previously defined object.

- `pdf:obj @name PDFobject` creates a named object that can be referenced later by '`@name`'.

All possible object types for '`PDFobject`' are listed in Table 2. In the case of indirect objects, the object number must be given explicitly, so that this feature is rarely used, especially to specify the objects in a different PDF file.

boolean	`true`, `false`
numeric	`123`, `34.5`, `-.002`
string	`(This is a string)`, `<901FA3>`
name	`/Name1`, `/.notdef`
array	`[3.14 false (Ralph) /Name1]`
dictionary	`<</Key1 (Value) /Key2 3.14>>`
null	`null`
indirect	`12 0 R`
stream	`stream ... endstream`

Table 2: PDF object types [1, p. 51]

It is not simple to construct a stream object with the special command '`pdf:obj`' because the length of the stream object must be specified explicitly, which is quite bothersome. Imagine that you are trying to construct a stream object whose source comes from a file. Is it possible with this special command? Moreover, any stream object requires the keyword '`stream`' followed by an end-of-line marker.[6]

DVIPDFMx, therefore, provides new special commands for stream objects.

- `pdf:stream @name (string) <<dict>>` constructs a stream object the source of which comes from the string object '`(string)`'. The stream dictionary '`<<dict>>`' is optional, and the dictionary entry '`/Length`' is created automatically.

The following two special commands, for instance, construct the same stream object. The stream data of the second object is represented in the ASCII base-85 encoding. [1, p. 70]

```
\special{pdf:stream @name (xxxxxxxx)}
\special{pdf:stream @name (G^+IXG^+IX)
        <</Filter/ASCII85Decode>>}
```

- `pdf:fstream @name (filename) <<dict>>` constructs a stream object in the same way as '`pdf:stream`', but the source of stream data comes from a file '`filename`'.

The following example shows how to include a source TEX file inside the output PDF file. (See [1, p. 637] for more details on the file attachment annotation.)

```
\special{pdf:fstream @myfile (mytest.tex)}
\special{pdf:ann bbox 0 0 10 10 <<
```

[6] An end-of-line marker consists of either a carriage return (`0x0d`) and a line feed (`0x0a`) or just a line feed, and not by a carriage return alone [1, pp. 60–61].

```
/Subtype /FileAttachment /FS <<
  /Type /Filespec /F (mytest.tex)
     /EF << /F @myfile >> >>
/Name /PushPin >>}
```

2.3 Adding content to named objects

We describe the special command for adding content to named objects. The type of the named object must be either array or dictionary.

- `pdf:put @arrayobj object`$_1$ `...` `object`$_n$ appends the n objects at the end of the array object '`@arrayobj`'.
- `pdf:put @dictobj <<dict>>` merges the dictionary object '`<<dict>>`' into '`@dictobj`'. If both the dictionaries have a common key, the old value in '`@dictobj`' will be replaced by the new value in '`<<dict>>`'.

In the following example, the value of the key '`/X`' in the dictionary object '`@name`' is '`@Moon2`'. (See [2, p. 15] for corresponding pdfmark operators.)

```
\special{pdf:put @Moon1
        [ (Earth to Moon) 238855 /mies ]}
\special{pdf:obj @Moon2 []}
\special{pdf:put @Moon2 (Moon to Earth)}
\special{pdf:put @Moon2 238855}
\special{pdf:put @Moon2 /miles}
\special{pdf:obj @name <<>>}
\special{pdf:put @name << /X @Moon1 >>}
\special{pdf:put @name << /X @Moon2 >>}
```

Note that DVIPDFM does not allow adding content to a stream dictionary object, but DVIPDFMx does.

- `pdf:put @streamobj <<dict>>` merges the dictionary object '`<<dict>>`' into the stream dictionary of '`@streamobj`'. The dictionary entries, '`/Length`' and '`/Filter`', in the object '`<<dict>>`' will be ignored.

Finally, DVIPDFM(x) provides the special command `pdf:close @name` to prevent further modifying the content of '`@name`'. After closing the named object, it can only be referenced.

3 Annotations

An annotation is considered as an object with a location on a page. The type of the object is given by the value of the key '`/Subtype`', for instance, '`/Text`', '`/Link`', '`/Sound`', '`/Movie`', etc. (See [1, p. 615] for the list of all annotation types.) The location is given by an array object associated to the key '`/Rect`'. DVIPDFM(x) provides the following special command for annotations.

- `pdf:ann @name width [length] height [length] depth [length] <<dict>>`

  The annotation dictionary is given by '`<<dict>>`' and the location relative to the current position is given by the three dimension parameters, '`width`', '`height`', and '`depth`'.

It is not possible to specify the location in an absolute way. Any value of the key '`/Rect`' in the annotation dictionary '`<<dict>>`' will be ignored if found. It is not allowed to modify the annotation dictionary with '`pdf:put`' command, so '`@name`' must be used as a reference.

Note that DVIPDFMx allows the '`bp`' unit in the dimension parameters, but DVIPDFM does not. Moreover, DVIPDFMx supports the following form.

- `pdf:ann @name bbox [ulx] [uly] [lrx] [lry] <<dict>>`

  The relative location is given by the bounding box consisting of four numbers in '`bp`' units.

The following example shows a movie annotation that enables us to run the movie file '`mymovie.avi`' inside a PDF viewer program.

```
\special{pdf:ann bbox 0 0 360 180 <<
  /Subtype /Movie /Border [1 0 0]
  /T (My Movie) /Movie <<
    /F (mymovie.avi) /Aspect [720 360]
    /Poster true >>
  /A << /ShowControls false >> >>}
```

DVIPDFM(x) provides other special commands for *breakable* annotations, e.g., an annotation broken over several lines or several pages.

- `pdf:bann <<dict>>` begins a breakable annotation. Object name is not allowed for this command.
- `pdf:eann` terminates the previous breakable annotation.

These specials are mainly used for '`/Link`' annotation as the following example shows.

```
\special{pdf:bann << /Subtype /Link
  /BS << /Type /Border /W 0.5 /S /S >>
  /A << /S /URI
    /URI (http://www.tug.org) >> >>}%
http://www.tug.org%
\special{pdf:eann}
```

Warning: No annotation will be constructed if the content between '`pdf:bann`' and '`pdf:eann`' is an empty box. For example:

```
\special{pdf:bann << /Subtype /Link ... >>}
\vbox to 7.5pt{}\kern 10pt
\sepcial{pdf:eann}
```

Annotations constructed by DVIPDFM(x) may happen to be slightly bigger than the expected size. This occurs when the annotation grow size is positive; this value is specified in the configuration file. To

avoid this effect, either modify the configuration file or give '`-g 0`' on the command line when running DVIPDFM(x).

4 Outlines (or bookmarks)

The document outline consists of a tree-structured hierarchy of outline items (sometimes called bookmarks) for which DVIPDFM(x) provides the following special command.

- `pdf:out` n `<<dict>>` adds an outline item to the document. The integer parameter n represents the level of the outline entry (beginning with 1), and '`<<dict>>`' represents the outline item dictionary [1, p. 585].

Note that all the outline items generated by DVIPDFM are closed.[7] The '`bookmarksopen=true`' option of the hyperref package does not work if the PDF output is generated by DVIPDFM.

```
\usepackage[
  dvipdfm,bookmarks=true,bookmarksopen=true
]{hyperref}
```

DVIPDFMx provides two solutions for this problem. The first one is to specify the option '`-O` n' when running DVIPDFMx. Up to level n, the outline entries will be open. The second, and complete, solution is to use this extended special command:

- `pdf:out [-]` n `<<dict>>` The symbol '`[-]`' indicates that the outline item will be closed. On the other hand, '`[]`' without the minus sign indicates that the outline item will be open.

The hyperref package provides a new option '`dvipdfmx-outline-open`' that uses the extended command above. This option enables us to control the open level given by '`bookmarksopenlevel`'.

```
\usepackage[%
  dvipdfmx,bookmarks=true,
  bookmarksopen=true,
  bookmarksopenlevel=1,
  dvipdfmx-outline-open
]{hyperref}
```

5 External objects (or XObjects)

DVIPDFM(x) supports two types of *external objects*,[8] an image XObject and a form XObject.

- `pdf:image @name width [length] height [length] depth [length] (imagefile)` defines an image XObject the source of which comes from the file '`imagefile`'. See [5, p. 216] for complete syntax provided by DVIPDFMx.
- `pdf:bxobj @name width [length] height [length] depth [length]` begins the definition of a form XObject. As with the command '`pdf:add`', DVIPDFMx allows bounding box '`bbox [ulx] [uly] [lrx] [lry]`' for dimension parameters.
- `pdf:exobj` ends the previous form XObject definition.
- `pdf:uxobj @name` displays the image XObject or the form XObject previously defined and associated with '`@name`'. DVIPDFMx allows dimension parameters (same as '`pdf:image`') after '`@name`'.

Typical examples showing how to use image XObjects and form XObjects can be found in [4, pp. 15–16].

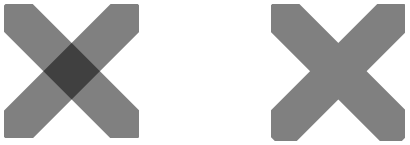


Figure 1: Two form XObjects with opacity 0.5; the right one is a group XObject.

DVIPDFMx extended the command '`pdf:eann`' to support a group XObject.[9] Figure 1 shows the difference between a normal XObject and a group XObject.

- `pdf:exobj <<dict>>` merges the dictionary object '`<<dict>>`' into the type1 form dictionary [1, p. 358] of the previous form XObject and then close the XObject.

The following code draws the right image in Figure 1.

```
\special{pdf:bxobj @group bbox 0 0 50 50}
\special{pdf:code
    15 w 0 0 m 50 50 l S 50 0 m 0 50 l S}
\special{pdf:exobj << /Group
   << /S /Transparency >> >>}
\special{pdf:obj @extgstate
   << /CA0.5 <</CA 0.5>> /ca0.5 <</ca 0.5>> >>}
\special{pdf:put @resources
   << /ExtGState @extgstate >>}
\special{pdf:code /CA0.5 gs /ca0.5 gs}
\special{pdf:uxobj @group}
```

We get the left image if `\special{pdf:exobj}` is used instead of the 4th and the 5th line.

[7] The sign of the value of the key '`/Count`' in the outline item dictionary determines whether the item is open or closed. [1, p. 586]

[8] "A graphics object whose contents are defined by a self-contained content stream, separate from the content stream in which it is used." [1, p. 332]

[9] "A special type of form XObject that can be used to group graphical elements together as a unit for various purposes." [1, p. 360]

6 Raw PDF Operators

This final section deals with writing raw PDF operators in the output. DVIPDFM provides a special command for this feature.

- `pdf:content Operators` adds the list of operators "`Operators`" to the current page at the current location. The operator '`q`', saving the current graphics state, followed by a transformation matrix moving to the current location will be attached to the beginning of the list, and the operator '`Q`' restoring the saved graphics state at the end of the list.

  For instance, the special command

```
\special{pdf:content 10 w 0 0 m 50 50 l S}
```

inserts the following list of operators in the output.

... `q` `1 0 0 1` *x y* `cm` `10 w 0 0 m 50 50 l S` `Q` ...

We sometimes need to insert PDF operators without additional graphics state operators. The author of the PGF package devised a trick:

```
\special{pdf:content Q ... Operators ... q}
```

The first operator '`Q`' and the last operator '`q`' nullify the effects of graphics state operators that are attached.

DVIPDFM*x* provides a new special command instead of the trick above.

- `pdf:literal direct Operators`[10] or simply `pdf:code Operators` plays the same role as '`pdf:content`, but no graphics state operator and no transformation matrix will be added.

Figure 2: The location of '23' in the left image varies according to the location of 1 in the current page.

Consider the following code, labelled Listing 1. Which image in Figure 2 does this code generate?

```
\def\bpic{\special{pdf:content q}}
\def\epic{\special{pdf:content Q}}
\def\myop#1{\special{pdf:content Q #1 q}}
1\bpic2\myop{.5 G 10 w 0 0 m 100 100 l S}3\epic4
```

Listing 1: Which image in Figure 2 is the result of this code, produced by DVIPDFM(*x*)?

The macro `\bpic` in Listing 1 nullifies the effect of the operator '`Q`' that will be attached after '`q`', and the macro `\epic` nullifies the effect of the list '`q 1 0 0 1` *x y* `cm`' that will be attached before '`Q`'.

Most people may choose the right-hand image in Figure 2 as the result of Listing 1, if they remember the fact that special commands are considered nothing by TEX. However, the answer is the left-hand image. The reason is that the transformation matrix in the macro `\bpic` still has an effect on the characters '2' and '3'. The effect will be nullified by the macro `\epic`.

To produce the right-hand image, DVIPDFM*x* provides the following new special commands.

- `pdf:bcontent` starts a block that works in the same way as '`pdf:content`' except that all text between this command and '`pdf:econtent`' will be placed in the right position.
- `pdf:econtent` ends the current block.

Moreover, '`pdf:bcontent`' and '`pdf:econtent`' can be nested.

Finally, we can get the right-hand image in Figure 2 as the result of Listing 2 following, produced by DVIPDFM*x*.

```
\def\bpic{\special{pdf:bcontent}}
\def\epic{\special{pdf:econtent}}
\def\myop#1{\special{pdf:code #1}}
1\bpic2\myop{.5 G 10 w 0 0 m 100 100 l S}3\epic4
```

Listing 2: The right-hand image in Figure 2 is the result of this example produced by DVIPDFM*x*.

References

[1] Adobe Systems, Inc., *PDF Reference*, 6th edition (Version 1.7, November 2006). `http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf`.

[2] Adobe Systems, Inc., *pdfmark Reference* (Version 8.0, November 2006). `http://www.adobe.com/devnet/acrobat/pdfs/pdfmark_reference.pdf`.

[3] Jin-Hwan Cho, *DVIPDFMx, an extension of DVIPDFM*, TUG 2003. Hawaii, United States. `http://project.ktug.or.kr/dvipdfmx/doc/tug2003.pdf`.

[4] Jin-Hwan Cho, *Practical Use of Special Commands in DVIPDFMx*, TUG 2005, International Typesetting Conference. Wuhan, China. `http://project.ktug.or.kr/dvipdfmx/doc/tug2005.pdf`.

[5] Jin-Hwan Cho, *Hacking DVI files: Birth of DVIasm*, The PracTEX Journal (2007), no. 1, and TUGboat 28:2, 2007, 210–217. `http://tug.org/TUGboat/Articles/tb28-2/tb89cho.pdf`.

[10] The idea of '`pdf:literal direct`' came from the primitive '`\pdfliteral direct`' of pdfTEX.

[6] Jin-Hwan Cho, *Handling Two-Byte Characters with DVIasm*, The Asian Journal of TEX **2** (2008), no. 1, 63–68. http://ajt.ktug.kr/assets/2008/5/1/0201cho.pdf.

[7] Jin-Hwan Cho, *The DVIasm Python script*. http://mirror.ctan.org/dviware/dviasm/.

[8] Heiko Oberdiek, *The hyperref package* (Version 6.78f, August 2008). http://mirror.ctan.org/macros/latex/contrib/hyperref/

[9] Till Tantau, *The beamer package* (Version 3.07, March 2007). http://mirror.ctan.org/macros/latex/contrib/beamer/.

[10] Till Tantau, *PGF, A Portable Graphics Format for TEX* (Version 2.00, February 2008). http://mirror.ctan.org/graphics/pgf/.

[11] Mark A. Wicks, *DVIPDFM User's Manual* (Version 0.12.4, September 1999). http://gaspra.kettering.edu/dvipdfm/dvipdfm-0.12.4.pdf.

◇ Jin-Hwan Cho
Department of Mathematics
The University of Suwon
Republic of Korea
chofchof (at) ktug dot or dot kr

Ancient TEX: Using XETEX to support classical and medieval studies

David J. Perry

Abstract

This article provides a brief background on Unicode and OpenType and then explains how they have become important to scholars in classics and medieval studies. XETEX, with its support for Unicode and OpenType, now makes TEX a good choice for scholars working in these fields — particularly on Windows and Linux, where OpenType support is not readily available otherwise.

1 The movement toward Unicode

(If you already have a good understanding of Unicode, you can skip ahead to section 2, or to 2.2 if you don't need an introduction to OpenType.)

Unicode is a project designed to make it possible to use all the living languages of the world, and many historical ones, in an efficient and standardized way. It is developed by the Unicode Consortium, a group that includes software companies, institutions such as universities and governmental agencies, and individuals. The Unicode Standard is developed in coordination with the international standard ISO-10646, known as the Universal Character Set; all characters added to one are also added to the other. (These two projects were begun separately in the late 1990s, but soon were merged since it was not beneficial to have two competing standards.)

ISO-10646 is essentially a list of characters. The Unicode Standard provides additional help to those who need to write software using various scripts; for instance, Unicode provides a bidirectional algorithm to integrate left-to-right and right-to-left scripts as well as guidance about how to work with scripts such as Arabic and the various Indic scripts that have complex shaping requirements. For more information, see the web site of the Unicode Consortium: `http://www.unicode.org`.

During the last 15 years or so, Unicode has become more and more important. All the major computer operating systems (Microsoft Windows, Linux, and Apple's Mac OS X) have been Unicode-based for some time, and much software has been written that takes advantage of Unicode.

Unicode is based on the *character/glyph model*. Under this system, Unicode encodes *characters*, basic phonemic or semantic units. It does not concern itself with the fact that these characters may appear in different forms on a page; the exact shape that a character assumes in a given context is referred to as a *glyph*. Two examples will clarify this distinction.

1. The character LATIN SMALL LETTER A may appear as a, a, a, *a* or as many other shapes, depending on the typeface and style (italic, bold, small capitals, etc.) chosen by the author or designer.
2. In Arabic, letters take on different shapes depending on whether they are the first letter in a word, appear in the middle of a word, or come as the last letter of a word. Unicode encodes one general set of Arabic letters, corresponding to the forms used in isolation (as when a reference book shows "the Arabic alphabet" in a table). In order to display Arabic properly, software must take a string of these basic Arabic letters and apply the correct forms as called for by the context.

The character/glyph model enables Unicode text to be stored in an efficient and permanently valid form. In the case of the Latin script, it would obviously be impossible and undesirable to attempt to encode permanently every different letter shape. For Arabic, the same text may be processed at the present time on a Windows system using OpenType or on a Mac using AAT, or new technologies may be developed for other computer systems in the future; but the underlying text remains valid.

For scholars in fields such as classics, biblical studies, and medieval studies, Unicode provides two important, related advantages:

- the ability to mix different scripts and languages easily in one document
- a standardized, internationally recognized, and permanent set of characters

A biblical scholar, for instance, might need to use ancient Greek, Hebrew, and Latin, along with one or more modern languages. While it has been possible for some time to mix languages on most computer systems, this was not always easy, particularly if one wanted to mix right-to-left and left-to-right scripts.

The case of ancient Greek provides a good example. It requires three accents, two breathing marks, a special form of the letter iota written below other vowels, and a few additional signs. Neither Apple nor Microsoft ever created any standard for ancient Greek, so each font maker set up his own system of matching Greek letters to various positions in the Latin alphabet and their corresponding keystrokes. (Prior to Unicode, users could access no more than 256 characters at one time, so a single font could not support, e.g., Latin and Greek.) By the time it became practical to use Unicode Greek (about

1996), there were several Greek fonts in use by classicists, each different from the others. Exchanging text with colleagues was difficult unless they happened to be using the same font. Without the appropriate font (or at least a table stating what Greek letters were mapped to what Latin ones), the meaning of a given text could be entirely lost. The situation with biblical Hebrew was similar.

This is very different from the situation in mathematics; the development of TEX and its adoption as a standard early in the personal computer era meant that mathematicians did not feel the same urgency as classicists did to move to Unicode.

Unicode changed the multilingual landscape. Classicists and biblical scholars eagerly adopted Unicode Greek and Hebrew, for they recognized the advantages of a standardized format that was internationally recognized and not dependent on the use of a particular font. Unicode fonts can contain more than 64,000 characters, although most contain far fewer. Therefore one can potentially use a font that contains Greek or Cyrillic letters designed to harmonize with the Latin forms; the text looks good and one need not worry about switching fonts.

2 The importance of OpenType

All is not perfect in the marriage of scholarship and Unicode, however. Classicists and medievalists have embraced Unicode because we appreciate its many benefits and because we do not want to be left out as the computing world becomes more Unicode-centric, but the character/glyph model is not a perfect fit for our needs. There are three important issues for which Unicode by itself does not provide a good solution: glyph variants, unusual combinations of diacritical marks and base letters, and non-standard ligatures. OpenType provides a solution for all these issues. Before discussing how scholars can use OT to address their specific needs, we give some background about OT in general.

2.1 OpenType basics

The OpenType specification was created jointly by Microsoft and Adobe. It provides many different tools that enable a string of Unicode characters to be displayed in ways are linguistically appropriate and typographically attractive. These tools are referred to as *features*.

Some features are used to render a string of Unicode characters in ways that are required for text to be considered correct by users. For Arabic, OT provides features to replace the basic letters with the forms needed if a letter is the first or last in a word, as explained above. An Arabic-capable word processor applies these features automatically as the user types, so the user does not have to worry about them; the resulting text displays in normal Arabic fashion. The font developer must do what is required to ensure that the features operate correctly. In the case of Arabic, this means putting additional glyphs into the font for initial, medial and final forms and setting up tables so that when, for instance, the application calls for the word-initial form of a letter, it can locate the proper glyph to use.

Another example: the Serbian language may be written in either the Latin or the Cyrillic script. When using the latter, Serbians employ a few letter shapes that are slightly different from those used in Russia. OT fonts can contain a feature that specifies which shapes to use for which language. There is no question that the same alphabet — Cyrillic — is used for both languages, and it would be very undesirable to encode the Serbian shapes separately. OT makes it possible to have standard Unicode text displayed appropriately for Serbian or Russian readers.

Other OT features are used to provide high-quality typography in scripts such as Latin, Greek, and Cyrillic that, unlike Arabic or Indic scripts, do not require complex processing. An OT font can contain true small capitals, various varieties of numbers (lining numerals, oldstyle ["lowercase"] numerals, and both proportionally spaced and monospaced versions of either style), ligatures (fi, ff, etc.), and many other typographic refinements. These features, unlike those required for correct display of Arabic, usually do not display unless specifically requested by the user. An application that supports high quality typography via OT must provide an interface for this purpose.

In short, OT is a two-headed beast. Microsoft originally adopted it as a means to get Unicode text to display properly in languages that have complex script requirements. Adobe has been more interested in the typographic possibilities of OT in standard scripts and has promoted its use by releasing OT versions of Adobe fonts and by providing access to OT features in programs such as the advanced InDesign page layout program.

We should note that Mac OS X includes a technology called AAT (Apple Advanced Typography) that does many of the same things as OT, both to implement complex scripts and to provide high-quality typography in standard scripts. AAT has not met with great success, partly because it is more difficult for font developers to create AAT than OT fonts. In response, Apple has enhanced OS X (beginning with version 10.4) so that it now processes and displays many features found in OT fonts.

OT font files are cross-platform (Mac, Windows,

Figure 1: Shapes of the centurial sign.

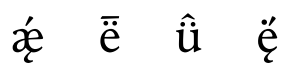


Figure 2: Some medieval combinations of diacritics.

Unix); the basic text will always display properly, but implementation of advanced OT typographical features is up to the application.

2.2 OpenType and scholars

As mentioned above, there are three areas in which Unicode does not adequately meet the needs of classicists and medievalists. Let's look at each in turn.

Some characters appear in shapes that vary considerably, depending on when and where the text was created. For instance, Roman inscriptions often contain a symbol that represents the word *centurio* (centurion, the Roman equivalent of a sergeant) or *centuria* (century, a military unit of 100 men). This centurial sign may take the shapes shown in Figure 1, which are referred to as *glyph variants*.

The centurial sign was recently accepted into Unicode. This is good because the character can now be stored in electronic texts in such a way that its identity will always be understood. But what if the editor wants to display the same shape as found on the original stone, when that is not the same as the Unicode reference glyph? Recall that under the character/glyph model, Unicode does not normally encode variant shapes for characters. An OT font can contain a number of alternate glyphs for a character, using the Stylistic Alternates feature. After entering the standard Unicode value for the centurial sign, the user can apply the Stylistic Alternates feature and select the desired glyph shape. This is a neat solution to a difficult problem. If a character from the Private Use Area were used to print the variant, its value might be lost if the proper font was not available in the future or if the text was copied and pasted into another application. (The Private Use Area is a range of codepoints that will never be defined by Unicode, i.e., they will always be officially left empty. Users can create customized fonts and put non-Unicode characters in the PUA for their own purposes. While the PUA can be useful, it is inherently unstable and characters in it should never be used in texts intended to have a long life, such as electronic editions of literary works.)

Medieval manuscripts contain dozens of combinations of letter plus diacritical mark(s) that are not used in any modern language and therefore are not directly supported by any operating system; see a few examples in Figure 2. (These examples are taken from the Character Recommendation of the Medieval Unicode Font Initiative, `http://www.mufi.info/`.) A few such combinations are also needed for ancient Roman inscriptions. Unicode provides all the needed diacritics in the Combining Diacritical Marks and Combining Diacritical Marks Supplement ranges. However, if a user simply types a base letter followed by a diacritic, there is no guarantee that the diacritic will be centered or otherwise placed appropriately over the base. Furthermore, good typographic practice is to replace the normal dotted i with the dotless 'ı' before applying an accent above the i. OpenType fonts can be set up to handle proper placement of diacritics and the substitution of dotless i as needed. (The original design of Unicode envisioned that operating systems would be able to place any combining diacritic appropriately and automatically. This vision is taking a very long time to be realized. Mac OS X was the first to attempt it, by looking at the widths of the characters in the font. The results are frequently acceptable, though some combinations need manual adjustment. Windows Vista has now taken some very limited steps to implement combining diacritics. But for now, and probably for some time to come, we need to rely on information built into each font in order to get diacritics working properly.)

Finally there is the matter of ligatures. These are found in ancient Greek and Roman inscriptions and even more frequently in medieval manuscripts. They were used to save space on stones and to save time for scribes. OpenType supports the standard f ligatures used in modern printing (fi, fl, ff, ffi, and ffl) through its Standard Ligatures feature. It also provides a feature called Historical Ligatures. An OT font designed to support epigraphy could include an entry in the Historical Ligatures feature to replace the letters NT with the ligature commonly found in Roman inscriptions, if the user applied this feature to a run of text.

It should be emphasized that even if an alternate glyph or an historical ligature is presented to the reader via OT features, the underlying Unicode text is not changed. This is important in regard to searching and reusing text. A user, for instance, might not know about all the varying shapes of the Roman centurial sign; even if he or she did know them all, it is not desirable to require multiple searches in order to cover all possibilities. If the user enters the standard Unicode value for the centurial sign

when searching, the proper results will be returned, regardless of which glyph is shown in the document.

Likewise, a user can copy some text that is displayed with unusual ligatures and paste it into an application that cannot handle OT features. The underlying letters will be shown, not some random characters, so that the text is still meaningful, even if not displayed in its historical form.

2.3 Software support for OT features

So it seems that classicists and medievalists now have a good solution to many of their needs, using OT features for display on top of Unicode text. The problem is that support for OT has been slow in coming. Mac users are best off. The word processor Mellel was developed around OT (rather than AAT) and provides good support. Some of Apple's own applications, such as the word processor Pages, include a Typography palette that provides access to AAT or OT features, whichever a specific font offers. Both Mellel and Pages are reasonably priced. The high-end page layout programs Adobe InDesign and Quark Express (v7 or later) offer outstanding Unicode and OT support, but are prohibitively expensive for many users.

On Windows, support for high-end typography is provided only by InDesign and Quark Express. Windows Vista includes some APIs that make it easier for software developers to access OT features, but so far developers have not taken advantage of them — including those responsible for Microsoft's own Office suite. The situation is equally bleak in the Linux world. Neither OpenOffice nor Scribus yet supports OT features on any platform.

This situation is very frustrating to scholars. We need to use Unicode, for the reasons explained above, and we understand that the character/glyph model just does not allow for glyph variants or unusual ligatures or diacritic combinations to be encoded. OT does provide a solution that works, but software support is extremely limited, particularly for Windows and Linux users.

What does the TEX world offer for our needs?

3 XeTEX brings it all together

3.1 XeTEX basics

Released in 1994 by Jonathan Kew, XeTEX was originally available for Mac OS X and then was ported to Unix and Windows. It extends the functionality of TEX and LATEX in three important ways.

- XeTEX provides direct Unicode support. Users can mix scripts, use large fonts, and access any Unicode character, as explained above. They can also use the standard methods to which they are accustomed when entering text. For example, if a Windows system is set up to handle polytonic Greek or Hebrew as well as English, the user can employ the icon in the system tray or the normal ALT-LEFT SHIFT combination to switch easily between languages and their associated keyboard layouts.
- XeTEX allows users to take advantage of OT and AAT features that may be present in a font.
- XeTEX enables users to access all fonts installed on the system without the need to create special configuration files for each font.

3.2 Encouraging new users to try XeTEX

Until the creation of XeTEX, TEX was not an ideal choice for classicists and medievalists. Their world is becoming more Unicode-centric, and they are hoping that OT will solve many of the problems that Unicode presents for their work. Furthermore, they very often need special fonts — after all, support for ancient epigraphy or medieval manuscripts is not a concern to most font makers — and such fonts are nowadays all Unicode-based. Being able to use installed Unicode system fonts without the complicated configuration process previously required by TEX removes an important barrier for new users. Since support for advanced OT typography in standard scripts is available only in a very small number of expensive applications under Windows and not at all in Linux except for XeTEX, those who have a real need for OT features should seriously consider using XeTEX.

New TEX users, and old hands who advise them, should be aware of the following:

- XeTEX is now included in most TEX distributions, so users will already have it.
- To take full advantage of XeTEX, a Unicode-based text editor or integrated environment is necessary; some of those still in use in the TEX world can handle only ASCII, such as WinEdt and TEXnicCenter (the latter will be Unicode-capable in v.2, according to its web site); Texmaker handles Unicode but knows nothing about XeTEX yet.
- Jonathan Kew and others are now developing TEXworks, an easy-to-use integrated environment for document creation that fully supports XeTEX. While it has not yet been officially released, working versions can be obtained from the project's web site: `http://www.tug.org/texworks/`. Alain Delmotte has written an introductory manual for TEXworks and also provides up-to-date binaries for those who do

not wish to compile the software themselves; see http://www.leliseron.org/texworks/. I used TEXworks to prepare this article, so it is certainly functional, albeit with a few rough edges. I regard it as the best choice for beginners with XETEX at the present time.

- For those who are willing to work with a plain text editor, Notepad (bundled with Windows) and BabelPad (at http://www.babelstone.co.uk/Software/BabelPad.html) will do the job; the latter is particularly Unicode-friendly.
- I have written an article intended for scholars in classics and medieval studies who want to begin using XETEX; it is available from http://scholarsfonts.net. Experienced TEX users, especially those who have read this article, will not find much new there, but they might want to pass it along to colleagues who seek aid in using TEX. It does contain more information about the fontspec package and OT, including a table that sorts out the names of features. (fontspec uses names that do not exactly match the standard OT names, which can be confusing.)

3.3 Using XETEX

Using XETEX is not difficult. You need to add a few packages to your preamble: fontspec, xunicode, xltxtra, and perhaps polyglossia. The first, fontspec, is very important because it helps XETEX select fonts and is the only practical way to apply OT or AAT features. Documentation for it is included and will be accessible to those experienced with TEX; newcomers will find it a bit tricky. The second, xunicode, enables users to employ traditional TEX shortcuts such as --- for an em-dash; neither it nor xltxtra requires any action on the user's part once added to the preamble.

To add support for language-specific hyphenation and punctuation, use the polyglossia package; see its documentation for the various options, which should be understandable by anyone with a basic knowledge of TEX. It is a replacement for the babel package, which should not be used with XETEX.

If you are using TEXworks, you can start a new file by using File / New from Template ... and choosing one of the XELATEX templates. This will get you fontspec and other packages you need.

3.4 Some samples

To conclude this article, we will provide some samples of what can be done with XETEX and OT features. The following is by no means a complete illustration of what OT can do, but it will, we hope, whet the appetites of readers to explore OT further. All samples make use of Junicode, a font for medievalists described in section 4 below. OT features are called through the fontspec package. In these examples I used fontspec's \addfontfeature{} command, which provides an easy way to apply features to short runs of text. There are other ways, such as setting defaults in the preamble if you want a feature to be used throughout a document.

Keep in mind that even though some of these samples look unusual, the underlying text consists of regular letters and numbers, and (for instance) a PDF file containing such text can be easily searched without inputting any special characters. The first three samples illustrate OT features that are helpful for setting high-quality text in any Latin-script language, while the rest are specific to medieval studies.

3.4.1 Oldstyle numerals

The following code produces the result shown in Figure 3.

```
default numbers: \quad 1234567890 \\
{\addfontfeature{Numbers=OldStyle}
with Oldstyle on: \quad 1234567890 }
```

default lining numbers: 1234567890
with Oldstyle feature on: 1234567890

Figure 3: Lining versus oldstyle figures.

3.4.2 Fractions

The following code produces the result shown in Figure 4.

```
Without fractions:
\quad 1/2 \quad 2/5 \quad 2/3 \quad 7/8 \\
{\addfontfeature{Fractions=On}
With fractions on:
\quad 1/2 \quad 2/5 \quad 2/3 \quad 7/8 }
```

Without fractions: 1/2 2/5 2/3 7/8
With fractions on: ½ ⅖ ⅔ ⅞

Figure 4: Creation of true typographical fractions.

3.4.3 Small capitals

Many OT fonts contains properly designed small capitals. (This is not the same as the "small capitals" found in programs like Microsoft Word, which are scaled-down capitals that do not follow traditional design principles for small caps.) OT provides a feature to invoke small capitals and another that changes only uppercase letters to small caps. The latter is useful for abbreviations that are typed in caps but look better as small caps when mixed in running text. The following code produces the result shown in Figure 5.

```
quick brown fox
{\quad \addfontfeature{Letters=SmallCaps}
quick brown fox}\\
the NATO alliance
{\addfontfeature{Letters=UppercaseSmallCaps}
\quad the NATO alliance }
```

quick brown fox QUICK BROWN FOX
the NATO alliance the NATO alliance

Figure 5: True small capitals.

3.4.4 Historical forms and historical ligatures

OT's Historical Forms feature allows the user to turn on shapes that are appropriate only in historical contexts, such as the long s and its ligatures, which were used in English through the 18th century. Junicode uses the Historical Ligatures feature to access ligatures found in medieval manuscripts. Note that one can turn on more than one feature at a time in fontspec by separating the features with a comma. The following code produces the result shown in Figure 6.

```
same silly distant \quad AA aa AY ay ag al}
{\addfontfeature{Style=Historic,
  Ligatures=Historical}
same silly distant  \quad AA aa AY ay ag al}
```

same silly distant AA aa AY ay ag al
ſame ſilly diſtant Ꜳ ꜳ Ꜽ ꜽ ag al

Figure 6: Historical forms and historical ligatures applied to text.

3.4.5 Language-specific features

The letters thorn and eth were used in Old English and are still employed in modern Icelandic. Junicode's default is to use the Old English shapes. Those who prefer the Icelandic forms can access them as shown here. The following code produces the result shown in Figure 7.

```
Default Old English shapes: \\
\quad {\Large Þþ Ðð}  \\
\addfontfeature{Language=Icelandic}
Icelandic shapes now used:
\quad {\Large Þþ Ðð}
```

Default Old English shapes: Þþ Ðð
Icelandic shapes now used: Þþ Ðð

Figure 7: Use of language-specific forms.

4 Resources

To learn more about Unicode, OpenType, and XƎTEX, an excellent place to start is Michel Goossens's *The XƎTEX Companion: TEX Meets OpenType and Unicode*, currently available at `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`. Written with an eye toward those who already have some familiarity with TEX, it provides more in-depth information than what is found in this article.

The web site of the Unicode Consortium, `http://www.unicode.org`, offers a great deal of information, including the entire text of *The Unicode Standard* in downloadable PDF form.

Here are some options if you want to experiment with the advanced typographical features of OpenType:

- The Junicode font by Peter Baker, freely available from `http://junicode.sf.net/`. The zip download includes some documentation that was created with XƎTEX.
- Linux Libertine by Philipp Poll (freely available from `http://linuxlibertine.sf.net/`) is another nice font family with many OT features; despite its name, it also works on Windows and Mac OS X.
- TEX Gyre is a project to update and extend the fonts distributed with the open-source Ghostscript page description language. It includes a number of fonts, each in OpenType and Type 1 formats. The OT versions contain many features for advanced typography, all of which are identified in the documentation. Latin Modern does the same for TEX's Computer Modern fonts. See `http://www.gust.org.pl/tex-gyre` and `http://www.gust.org.pl/lm`, respectively.
- If you have access to any of Adobe's Pro fonts (Warnock Pro, Minion Pro, etc.), these also contain OT features. Adobe's online font catalog at `http://www.adobe.com/type/` shows what features are included in the various fonts they sell (not all fonts have all features).

If you are curious about how characters, particularly scholarly ones, get added to Unicode, you can look at the proposals for medieval characters prepared by the Medieval Unicode Font Initiative at `http://www.mufi.info/` or at my proposals for classical Latin characters at `http://scholarsfonts.net/latnprop.html`.

◇ David J. Perry
Rye High School
Rye, New York
USA
hospes02 (at) scholarsfonts dot net
http://www.scholarsfonts.net

TEXonWeb

Jan Přichystal

Abstract

This article describes a web application TEXonWeb which allows using the (LA)TEX typesetting system without needing installation on a local computer. TEXonWeb is simple, with a text area where the user can write the source code of his document and then click the button to get resulting PDF or PostScript output. This article briefly summarizes the features and capabilities of TEXonWeb.

1 Introduction

One of the main aims of the TEXonWeb application is to provide a simple interface for document processing using the typographic system (LA)TEX. Many users would like to produce high-quality documents but are not familiar with the non-trivial (LA)TEX installation and configuration. Also, sometimes users are in a situation where they cannot use their own computer and have to work for example in an Internet café. This is a time when they can use TEXonWeb. The only thing needed is a web browser.

2 First steps

TEXonWeb can be used with any web browser supporting JavaScript and cascading style sheets. We recommend Mozilla Firefox or Internet Explorer. Upon visiting the web address `http://tex.mendelu.cz/en`, the user sees a simple page with a text area in which the template of a LATEX document is entered. He can immediately start to work and write text and TEX or LATEX commands. There are no limitations on length or complexity of documents, or on the TEX commands available.

TEXonWeb can be used in two modes — anonymous and authenticated. Anonymous access is designated for very simple and short documents with no other included parts. Here the user just types a document and presses a button to get PDF or PostScript. No special features or tools are available.

More users create their own accounts and work in authenticated mode. To do this, a user follows the 'Create account' link and provides a login name and password. If the login name is not yet used by another user, a new account is created and the user can log in. This mode is designated for repeated usage of TEXonWeb. Users can typeset more complex documents, store them the on server, set up working space and use supporting tools (spell-checker, table wizard, etc.).

3 User interface

The most important part of the application is the text area which acts as an editor, where the user can type the source code of his document. Under the editor window, there are buttons 'PDF' and 'PostScript' which produce the document in the corresponding format. Next to these there is a 'Log file' button for viewing the log file created by processing the document. See Fig. 1.

Below these buttons are option menus to set how the document should be processed. The first item determines whether the plain or LATEX format is used. The second item defines if the document is processed one, two or three times (e.g., if generating a table of contents). The last item, if checked, returns the document in `.zip` format (for slow Internet connections).

If the user is logged in, a user menu and toolbar are above the editor window. The user menu consists of these items:

- File — options to open and save files stored on the server, upload and download files from the local computer and process documents into PDF or PostScript formats.
- Edit — options for the usual undo, redo, copy and paste actions.
- Settings — options to toggle syntax highlighting, toolbar and detailed setting of user interface.
- Styles — templates for standard documents such as letters, wall calendars and business cards.
- Tools — spell checker, table wizard or inserting non-breaking spaces.
- Help — Documentation of TEXonWeb.

The set of tools simplifies document editing. A spell checker highlights misspelled words, while another tool can insert non-breaking spaces in suitable places. There are also wizards for inserting code of more complex components such as tables or pictures. The user can use these interactive tools to define the properties of the object being inserted without knowing the exact syntax of LATEX commands.

There is also a toolbar for interactive insertion of LATEX commands. The user just clicks an icon and the code appears in the editor window. Commands are divided into related sets — undo/redo, font settings, headings, paragraph settings, lists, spacings, math symbols, etc.

4 Implementation of TEXonWeb

The TEXonWeb application runs on a common IBM PC compatible with dual-core processor and 4 GB RAM. The server runs the Linux-based CentOS

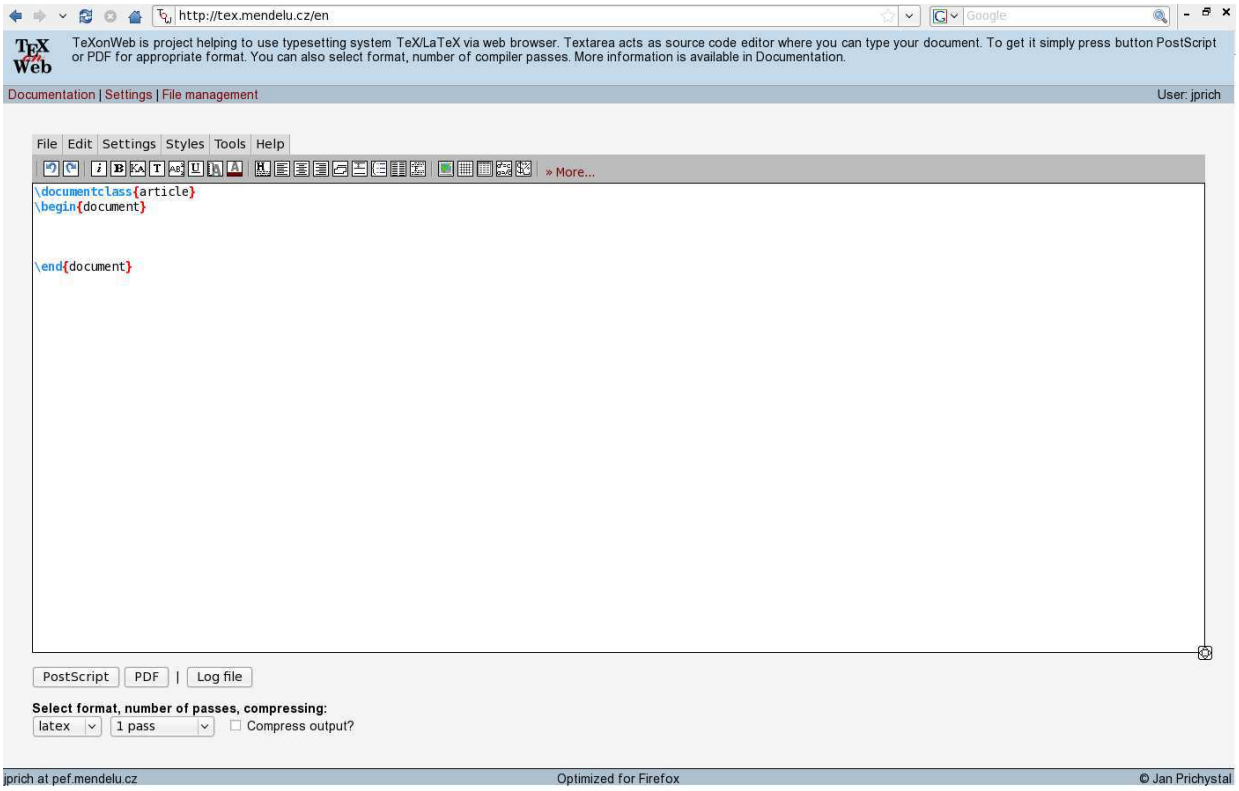


Figure 1: TEXonWeb main page

operating system with Apache web server. The application is programmed in Perl and JavaScript. The core of the application is a Perl module and the user interface creates a few Perl scripts. The toolbar and other parts of the user interface are implemented in JavaScript. The TEX processor comes from the `tetex` package provided in CentOS. Installation is located in a special directory and runs under `chroot`. This precaution makes attacking the system more difficult. For example, including or viewing system files are not allowed. We also enforce a limitation on the size of uploaded documents.

How does TEXonWeb work? The user types the source code of the document and clicks the button for translation. A Perl function sends the source code to the server, placing a `\nonstopmode` command at the beginning. This provides non-interactive translation, omitting prompting when an error occurs. Then PDF or PostScript is generated depending on the options set in the 'Settings' page or directly on the main page. The resulting file is returned to the web browser for the user to view. If an error makes translation impossible, the user is notified to view the log file.

5 Conclusion

TEXonWeb is not intended as a full substitution of specialized (LA)TEX editors installed on a local computer. A web page could not offer such comfort in writing documents. But in some situations it could be useful. For instance, TEX beginners who want to try how TEX works may find it interesting.

TEXonWeb is still being developed and new functions and options are being added. Currently we are working on multilingual support, document templates, editor improvements and support of the Opera web browser.

You can try TEXonWeb at the url `http://tex.mendelu.cz/en`.

◇ Jan Přichystal
Zemědělská 1
Brno, 613 00
Czech Republic
`jprich (at) pef dot mendelu dot cz`
`http://akela.mendelu.cz/~jprich/`

Typography

Typographers' Inn

Peter Flynn

1 The electronic book

For years (seems like centuries) we have seen forecasts that the electronic book is just round the corner, and soon we'll be able to let the trees grow in peace because there won't be any more demand for printing onto paper.

About 10 years ago I did a TV interview about the launch of some new e-book products and an impending software-only release from Microsoft. The marketing droids were out in force, predicting the immediate demise of the printed page, so my whines about 'it's the file format, stupid' went unheard.

A decade has come and gone, and we still keep hearing that e-paper and e-ink are where it's at. In 2001 (I think), at the TEX Users Group meeting at the University of Delaware, we even had a presentation from IBM about their research into e-paper, which was fascinating. It seems to be tantalizingly close each time, but never quite seems to make it.

The latest device is the Kindle, and it has garnered a growing and eager following, with a wireless connection that works, and a good number of titles coming out from publishers who would previously have dismissed the technology. But it suffers from poor interface design, poor provision of typefaces, and the proprietary tie-in to Amazon. Amazingly, it accepts file formats other than its native AZW (a variant of HTML), including Word, PDF, and Mobi... but not the one format that is designed for the job, the Open Publication Structure (OPS, successor to the Open eBook or OEB format of unhappy memory).

As I write this, publishers and manufacturers are meeting at the Digital Book 2009 conference in New York, run by the International Digital Publishing Forum, who manage OPS, trying to identify the business case for e-books. It's notable that the sponsors are the manufacturers: the publishers are nowhere to be seen. It's all about workflow and Digital Right Management (DRM) — not a whisper about typefaces or formatting.

So where does this leave those of us who set type? It's easy to create nice PDFs with LATEX, and they can be done for the precise dimensions of the device's screen with great accuracy. But if you want to read your e-book on several devices (desktop, laptop, handheld, e-book reader, or even your cellphone), you need a separately-optimized version for each.

Enter reflowable PDF, which will let the text content of a PDF document behave like HTML in your browser: change the shape of the window, and all the text reformats automatically to fit. After the TEX Users Group meeting in Cork last year, there was an impromptu session on this which hasn't progressed very far (details in the mailing list at `http://lists.ucc.ie/xml-tex-pdf.html`).

This isn't the perfect solution; it's fine for novels and other books consisting of continuous, uninterrupted text, but it isn't easy to make it work for mathematics or for books with chunks of code. TEX systems, on the other hand, are nothing if not programmable, so I'm asking anyone working in this area to consider joining the mailing list and sharing their thoughts. Wouldn't it be nice if the solution came from the TEX field?

2 Breaking the mold

When did you last design a whole book, from end to end?

At the TEX Users Group meeting in San Diego two years ago I was generously presented with a copy of Valerie Kirschenbaum's wonderful book *Goodbye Gutenberg* [1]. It's 400 pages of rich color, with each double-page spread separately designed and drawn (or typeset). It's fascinating, and you can have almost as much fun with it as you can with the *Très Riches Heures* of the Duc de Berry (*c.* 1415).

The author's aim is to rescue books from the slough of black-and-white reproduction, where every page is the same layout as the others, and to return to the creativity of the era before printing, where pages could differ. She asserts that print and layout technology is now at a stage where this can be done with little or no increase in cost.

In respect of a book designed to illustrate her purpose, she succeeds admirably, although she shows considerable naivety in her assumptions about typesetting and presswork costs, and ignores completely the need for consistency in reference books and heavily-structured documents. She is right, of course, that book 'design' has been in decline for decades (with a few notable exceptions), and that technology has indeed advanced to the point where what she proposes is *technically* feasible — if not financially — like Heyerdahl and Severin in the field of exploration, she has actually done it.

So what do we do when faced with yet another publisher's Compositor's Specification? I've had three in the last year which have appeared to have

been written (or drawn) by a teenager with 15 minutes' experience of Word. Inconsistent, inaccurate, and inappropriate; and in one case accompanied by an equally inaccurate PDF supposed to be an example of the output. I'm not sure where the publishers get these from, but it's clear that at least some 'designers' have only the vaguest idea of how text gets from the author's fingertips onto the printed page. In any event they appear not to have actually looked through the book to see what kinds of things they need to provide for, so you get specs with no information about how they want figures to look; what to do with second or third level lists; or how to format the endnotes.

Perhaps we should after all start to think about redesigning the book. After all, if the publishers (with the occasional honorable exception) cannot now be bothered to design even the whole of their own books, perhaps they would give us free rein to do the job for them. We surely can't be any worse at it than the authors.

3 RIOTING TYPOGRAPHERS RAMPAGE ONLINE!

The TYPO-L mailing list, which I refer to from time to time, is populated by well over 100 ladies and gentlemen of the industry, who conduct themselves with a decorum becoming to their profession, and occasionally venture to submit or answer a question, trusting that their colleagues will do the same for them on another occasion. I happen to be the List Owner of this happy band of typophiles, and I hardly ever have cause to intervene, except to fix the occasional glitch or to update a member's email address. Sometimes days or even a week can go past without a message, and then there is a small burst of activity over some common topic.

During April, however, over 13,000 lines of email were exchanged, much of it to celebrate or disparage '50 Years of Stupid Grammar Advice', as the topic was named, after an article by Geoffrey Pullum [2], in which he discusses the 'limp platitudes [and] inconsistent nonsense' in Strunk & White's [in]famous *The Elements of Style*.

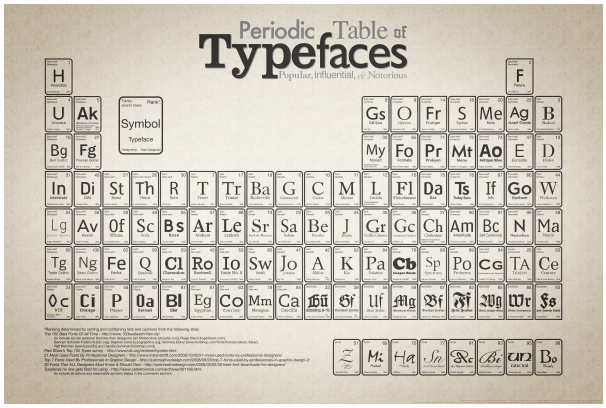
Something obviously touched a raw nerve somewhere, something only understandable by those who have had to suffer the insistence of learned academics stubbornly insisting on wholly inappropriate matters of style, or had to undo the depredations of unlearned students whose heads had been stuffed with outdated regulations. As I have mentioned before, we get called upon to exercise much more than LaTeX, and often have to deal with orthographic and syntactic errors when there is no-one else qualified or experienced enough to correct them.

As many posters pointed out, S&W contains a wealth of useful material as well as useless. It follows a kind of 80/20 rule (or is it 90/10?) which covers most aspects of most things rather than all aspects of everything, and is intended as a general guide rather than the absolute prescription as which it is often mistakenly presented. While I usually reserve my own venom for the trifling foolishnesses of the MLA in their placement of punctuation, I count myself lucky to have been spared the worst of Messrs Strunk & White by fortune of having been born outside their ambit, so I kept schtumm for most of it.

If you're interested in what typesetters talk about behind the authors' and publishers' backs, you can join the list and read the discussion in the archives at `http://listserv.heanet.ie/typo-l.html`

4 Periodic table of typefaces

Thanks to Michael Brady for pointing this out in the TYPO-L mailing list: `http://www.behance.net/Gallery/Periodic-Table-of-Typefaces/193759`



They also do a nice 25.5″ × 17″ print.

References

[1] Valerie Kirschenbaum. *Goodbye Gutenberg.* Global Renaissance Society, New York, 2005.

[2] Geoffrey K Pullum. 50 Years of Stupid Grammar Advice. *The Chronicle Review*, Apr 2009.

◇ Peter Flynn
Textual Therapy Division, Silmaril Consultants, Cork, Ireland
Phone: +353 86 824 5333
`peter (at) silmaril dot ie`
`http://blogs.silmaril.ie/peter`

Fonts

OpenType math illuminated*

Ulrik Vieth

Abstract

In recent years, we have seen the development of new TEX engines, XETEX and LuaTEX, adopting OpenType font technology for providing Unicode typesetting support. While there are already plenty of OpenType text fonts available, both from the TEX community and from commercial font suppliers, there is little support for OpenType math fonts so far. Ironically, it was left to Microsoft to develop a *de facto* standard for OpenType math font information and to provide the first reference implementation of a full-featured OpenType math font.

In order to develop the much-needed math support for Latin Modern and TEX Gyre fonts, it will be crucially important to develop a good understanding of the internals of OpenType math tables, much as it is necessary to develop a good understanding of Appendix G and TEX's `\fontdimen` parameters to develop math support for traditional TEX fonts. In this paper, we try to help improve the understanding of OpenType math internals, summarizing the parameters of OpenType math fonts as well as illustrating similarities and differences between traditional TEX math fonts and OpenType math fonts.

1 Background on OpenType math

In recent years, the TEX community has been going through a phase of very significant developments. Among the most important achievements, we have seen the development of new TEX engines, XETEX and LuaTEX, providing support for Unicode and OpenType font technology. At about the same time we have also seen the development of new font distributions, Latin Modern and TEX Gyre, provided simultaneously in Type 1 format as a set of 8-bit font encodings as well as in OpenType format.

Together these developments have enabled TEX users to keep up with current trends in the publishing industry, providing users of the new TEX engines with a comprehensive set of free OpenType fonts and enabling them to take advantage of the many offerings by commercial font suppliers.

As far as text typesetting is concerned, support for OpenType font technology in the new TEX engines is already very advanced, supporting not only traditional typographic features of Latin alphabets, but also addressing the very complex and challenging requirements of Arabic typography.

However, when it comes to math typesetting, one of the traditional strongholds of TEX, support for Unicode and OpenType math is only just beginning to take shape.

Ironically, it was left to Microsoft to develop the first system to offer support for Unicode math. When Microsoft introduced support for math typesetting in Office 2007 [1, 2], they extended the OpenType font format and commissioned the design of Cambria Math [3] as a reference implementation of a full-featured OpenType math font.

Fortunately for us, Microsoft was smart enough to borrow from the best examples of math typesetting technology, thus many concepts of OpenType math are not only derived from the model of TEX, but also go beyond TEX and introduce extensions or generalizations of familiar concepts.

While OpenType math is officially still considered experimental, it is quickly becoming a *de facto* standard, as it has already been widely deployed to millions of installations of Microsoft Office 2007 and it is also being been adopted by other projects such as the FontForge [4] font editor and independent font designs such as Asana Math [5].

Most importantly, support for OpenType math has already been implemented or is currently being implemented in the new TEX engines, thus adopting OpenType math for the development of the much-needed Unicode math support for Latin Modern and TEX Gyre obviously seems to be a most promising choice of technology.

2 Design and quality of math fonts

When it comes to developing math fonts, designing the glyph shapes is only part of the job. Another part, which is equally important, is to adjust the glyph metrics of individual glyphs and to set up the global parameters affecting various aspects of glyph positioning in math typesetting.

As we have discussed at previous conferences, the quality of math typesetting crucially depends on the fine-tuning of these parameters. Developing a good understanding of these parameters will therefore become an important prerequisite to support the development of new math fonts.

In the case of traditional TEX math fonts, we have to deal with the many `\fontdimen` parameters which have been analyzed in Bogusław Jackowski's paper *Appendix G Illuminated* and a follow-up paper by the present author [6, 7].

* First published in *Biuletyn GUST* 25 (2009), pp. 7–16, proceedings of the BachoTEX XVII conference. Reprinted with permission.

In the case of OpenType math fonts, we need to develop a similar understanding of the various tables and parameters and how the concepts of OpenType math relate to the concepts of TEX.

3 Overview of the OpenType font format

The OpenType font format [8] was developed jointly by Adobe and Microsoft, based on elements of the earlier PostScript and TrueType font formats by the same vendors. The overall structure of OpenType fonts consists of a number of tables, some of which are required while others are optional [9].

In the case of OpenType math, the extension of the font format essentially consists of adding another optional table, the so-called MATH table, containing all the information related to math typesetting. Since it is an optional table, it would be interpreted only by software which knows about it (such as the new TEX engines or Microsoft Office 2007), while it would be ignored by other software.

Unlike a database table, which has a very rigid format, an OpenType font table can have a fairly complex structure, combining a variety of different kinds of information in the same table. In the case of the OpenType MATH table, we have the following kinds of information:

- a number of global parameters specific to math typesetting (similar to TEX's many `\fontdimen` parameters of Appendix G)
- instructions for vertical and horizontal variants and/or constructions (similar to TEX's charlists and extensible recipes)
- additional glyph metric information specific to math mode (such as italic corrections, accent placement, or kerning)

In the following sections, we will discuss some of these parameters in more detail, illustrating the similarities and differences between traditional TEX math fonts and OpenType math fonts.

4 Parameters of OpenType math fonts

The parameters of the OpenType MATH table play a similar role as TEX's `\fontdimen` parameters, controlling various aspects of math typesetting, such as the placement of limits on big operators, the placement of numerators and denominators in fractions, or the placement of superscripts and subscripts.

While a number of parameters are specified in TEX through the `\fontdimen` parameters of math fonts, there are other parameters which are defined by built-in rules of TEX's math typesetting engine. In many such cases, additional parameters have been introduced in the OpenType MATH table, making it possible to specify all the relevant parameters in the math font without relying on built-in rules of any particular typesetting engine.

In view of the conference motto, it is interesting to note that the two new TEX engines, XETEX and LuaTEX, have taken very different approaches how to support the additional parameters of OpenType math fonts: While XETEX has retained TEX's original math typesetting engine and uses an internal mapping to set up `\fontdimen` parameters from OpenType parameters [10], LuaTEX has introduced an extension of TEX's math typesetting engine [11], which will allow it to take full advantage of most of the additional OpenType parameters.[1]

For font designers developing OpenType math fonts, it may be best to supply all of the additional OpenType parameters in order to make their fonts as widely usable as possible with any typesetting engine, not necessarily limited to any specific one of the new TEX engines.

In the following sections, we will take a closer look at the various groups of OpenType parameters, organized in a similar way as they are presented to font designers in the FontForge font editor, but not necessarily in the same order.

We will use the figures from [6, 7] as a visual clue to illustrate how the various parameters are defined in TEX, while summarizing the similarities and differences between OpenType parameters and TEX parameters in tabular form.

4.1 Limits on big operators

In TEX math fonts, there are five parameters controlling the placement of limits on big operators (see figure 1), which are denoted as ξ_9 to ξ_{13} using the notation of Appendix G.

Two of them control the default position of the limits (ξ_{10} and ξ_{12}), two of them control the inside gap (ξ_9 and ξ_{11}), while the final one controls the outside gap above and below the limits (ξ_{13}).

In OpenType math fonts, the MATH table contains only four parameters controlling the placement of limits on big operators. Those four parameters have a direct correspondence to TEX's parameters (as shown in table 1), while the remaining TEX parameter has no correspondence and is effectively set to zero.[2]

[1] More precisely, while XETEX only provides access to the OpenType parameters as additional `\fontdimen`s, LuaTEX uses an internal data structure based on the combined set of OpenType and TEX parameters, making it possible to supply missing values which are not supported in either OpenType math fonts or traditional TEX math fonts.

[2] Considering the approach taken in other circumstances, it is very likely that if there were any such correspondence,

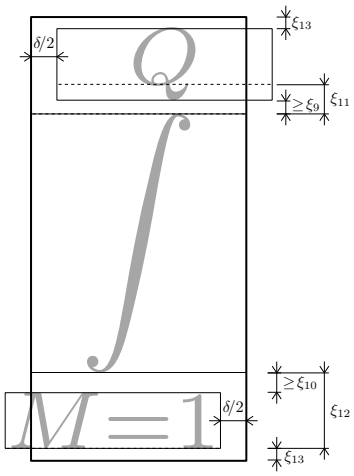


Figure 1: TEX font metric parameters affecting the placement of limits above or below big operators.

OpenType parameter	TEX parameter
UpperLimitBaselineRiseMin	ξ_{11}
UpperLimitGapMin	ξ_9
LowerLimitGapMin	ξ_{10}
LowerLimitBaselineDropMin	ξ_{12}
(no correspondence)	ξ_{13}

Table 1: Correspondence of font metric parameters between OpenType and TEX affecting the placement of limits above or below big operators.

4.2 Stretch stacks

Stretch stacks are a new feature in OpenType math fonts, which do not have a direct correspondence in TEX. They can be understood in terms of material stacked above or below stretchable elements such as overbraces, underbraces or long arrows.

In TEX, such elements were typically handled at the macro level and effectively treated in the same way as limits on big operators.

In LuaTEX, such elements will be implemented by new primitives using either the new OpenType parameters for stretch stacks (as shown in table 2) or the parameters for limits on big operators when using traditional TEX math fonts.

4.3 Overbars and underbars

In TEX math fonts, there are no specific parameters related to the placement of overlines and underlines.

there might actually be two parameters in OpenType instead of only one, such as UpperLimitExtraAscender and LowerLimitExtraDescender. In LuaTEX's internal data structures, there are actually two parameters for this purpose, which are either initialized from TEX's parameter ξ_{13} when using TEX math fonts or set to zero when using OpenType math fonts.

OpenType parameter	TEX parameter
StretchStackTopShiftUp	ξ_{11}
StretchStackGapAboveMin	ξ_9
StretchStackGapBelowMin	ξ_{10}
StretchStackBottomShiftDown	ξ_{12}

Table 2: Correspondence of font metric parameters between OpenType and TEX related to stretch stacks.

OpenType parameter	TEX parameter
OverbarExtraAscender	$(= \xi_8)$
OverbarRuleThickness	$(= \xi_8)$
OverbarVerticalGap	$(= 3\,\xi_8)$
UnderbarVerticalGap	$(= 3\,\xi_8)$
UnderbarRuleThickness	$(= \xi_8)$
UnderbarExtraDescender	$(= \xi_8)$

Table 3: Correspondence of font metric parameters between OpenType and TEX affecting the placement of overlines and underlines.

Instead, there is only one parameter controlling the default rule thickness (ξ_8), which is used in a number of different situations where other parameters are expressed in multiples of the rule thickness.

In OpenType math fonts, a different approach was taken, introducing extra parameters for each purpose, even supporting different sets of parameters for overlines and underlines. Thus the MATH table contains the following parameters related to overlines and underlines (as shown in table 3), which have only an indirect correspondence in TEX.

It is interesting to note that the introduction of additional parameters in OpenType math fonts provides for greater flexibility of the font designer to adjust the values for best results.

While TEX's built-in rules always use a fixed multiplier of the rule thickness regardless of its size, OpenType math fonts can compensate for a larger rule thickness by using a smaller multiplier.

An example can be found when inspecting the parameter values of Cambria Math: In relative terms the inside gap is only about 2.5 times rather than 3 times the rule thickness, while the latter (at about 0.65 pt compared to 0.4 pt) is quite a bit larger than in typical TEX fonts.

Obviously, making use of the individual OpenType parameters (as in LuaTEX) instead of relying on TEX's built-in rules (as in XETEX) would more closely reflect the intention of the font designer.

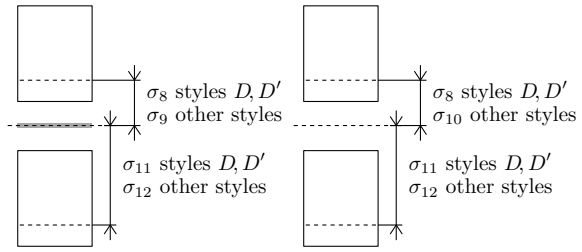


Figure 2: TEX font metric parameters affecting the placement of numerators and denominators in regular and generalized fractions.

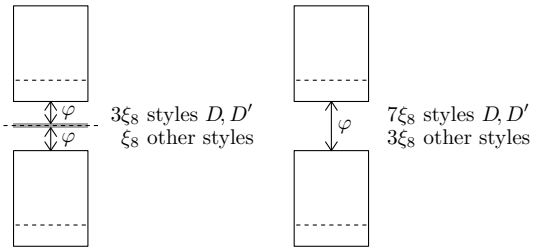


Figure 3: TEX's boundary conditions affecting the placement of numerators and denominators in regular and generalized fractions.

4.4 Fractions and stacks

In TEX math fonts, there are five parameters controlling the placement of numerators and denominators (see figure 2), which are denoted as σ_8 to σ_{12} using the notation of Appendix G.

Four of them apply to regular fractions, either in display style (σ_8 and σ_{11}) or in text style and below (σ_9 and σ_{12}), while the remaining one applies to the special case of generalized fractions when the fraction bar is absent (σ_{10}).

Besides those specific parameters, there are also a number of parameters which are based on built-in rules of TEX's math typesetting engine, expressed in multiples of the rule thickness (ξ_8), such as the thickness of the fraction rule or the inside gap above and below the fraction rule (see figure 3).

In OpenType math fonts, a different approach was once again taken, introducing a considerable number of additional parameters for each purpose. Thus the MATH table contains 9 parameters related to regular fractions and 6 more parameters related to generalized fractions (also known as stacks).

As shown in table 4, there is a correspondence for all TEX parameters, but this correspondence isn't necessarily unique, since the same TEX parameter is used for multiple purposes in fractions and stacks. Obviously, font designers of OpenType math fonts should be careful about choosing the values of OpenType parameters in a consistent way.

Analyzing the font parameters of Cambria Math once again shows how the introduction of additional parameters increases the flexibility of the designer to adjust the parameters for best results: In relative terms, `FractionDisplayStyleGapMin` is only about 2 times rather than 3 times the rule thickness. Similarly, `StackDisplayStyleGapMin` is only about 4.5 times rather than 7 times the rule thickness. In absolute terms, however, both parameters are about the same order of magnitude as in typical TEX fonts.

4.5 Superscripts and subscripts

In TEX math fonts, there are seven parameters controlling the placement of superscripts and subscripts (see figure 4), which are denoted as σ_{13} to σ_{19} using the notation of Appendix G.

Three of them apply to superscripts, either in display style (σ_{13}), in text style and below (σ_{14}), or in cramped style (σ_{15}), while the other two apply to the placement of subscripts, either with or without a superscript (σ_{16} and σ_{17}).

Finally, there are two more parameters which apply to superscripts and subscripts on a boxed subformula (σ_{18} and σ_{19}), which also apply to limits attached to big operators with `\nolimits`.

Besides those specific parameters, there are also a number of parameters which are based on TEX's built-in rules, expressed in multiples of the x-height (σ_5) or the rule thickness (ξ_8), most of them related

OpenType parameter	TEX parameter
`FractionNumeratorDisplayStyleShiftUp`	σ_8
`FractionNumeratorShiftUp`	σ_9
`FractionNumeratorDisplayStyleGapMin`	$(= 3\,\xi_8)$
`FractionNumeratorGapMin`	$(= \xi_8)$
`FractionRuleThickness`	$(= \xi_8)$
`FractionDenominatorDisplayStyleGapMin`	$(= 3\,\xi_8)$
`FractionDenominatorGapMin`	$(= \xi_8)$
`FractionDenominatorDisplayStyleShiftDown`	σ_{11}
`FractionDenominatorShiftDown`	σ_{12}
`StackTopDisplayStyleShiftUp`	σ_8
`StackTopShiftUp`	σ_{10}
`StackDisplayStyleGapMin`	$(= 7\,\xi_8)$
`StackGapMin`	$(= 3\,\xi_8)$
`StackBottomDisplayStyleShiftDown`	σ_{11}
`StackBottomShiftDown`	σ_{12}

Table 4: Correspondence of font metric parameters between OpenType and TEX affecting the placement of numerators and denominators.

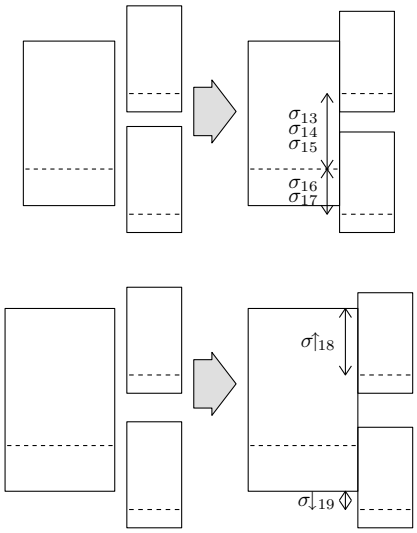


Figure 4: TEX font metric parameters affecting the placement of superscripts and subscripts on a simple character or a boxed subformula.

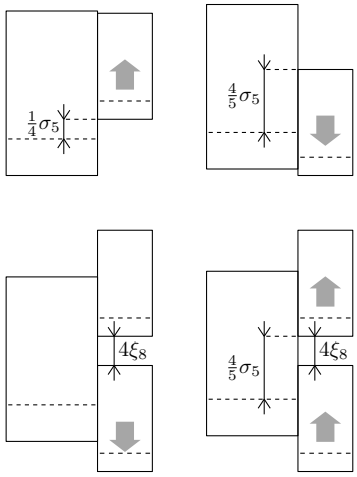


Figure 5: TEX font metric parameters affecting the placement of superscripts and subscripts in cases of resolving collisions.

to resolving collisions between superscripts and subscripts or adjusting the position when a superscript or subscript becomes too big (see figure 5).

In OpenType math fonts, we once again find a number of additional parameters for each specific purpose, as shown in table 5.

It is interesting to note that some of the usual distinctions made in TEX were apparently omitted in the OpenType MATH table, as there is no specific value for the superscript position in display style, nor are there any differences in subscript position in the presence or absence of superscripts.

While it is not clear why there is no correspondence for these parameters, it is quite possible that there was a conscious design decision to omit them, perhaps to avoid inconsistencies in alignment.

OpenType parameter	TEX parameter
SuperscriptShiftUp	σ_{13}, σ_{14}
SuperscriptShiftUpCramped	σ_{15}
SubscriptShiftDown	σ_{16}, σ_{17}
SuperscriptBaselineDropMax	σ_{18}
SubscriptBaselineDropMin	σ_{19}
SuperscriptBottomMin	$(= \frac{1}{4}\sigma_5)$
SubscriptTopMax	$(= \frac{4}{5}\sigma_5)$
SubSuperscriptGapMin	$(= 4\,\xi_8)$
SuperscriptBottomMaxWithSubscript	$(= \frac{4}{5}\sigma_5)$

Table 5: Correspondence of font metric parameters between OpenType and TEX affecting the placement of superscripts and subscripts.

4.6 Radicals

In TEX math fonts, there are no specific parameters related to typesetting radicals. Instead, the relevant parameters are based on built-in rules of TEX's math typesetting engine, expressed in multiples of the rule thickness (ξ_8) or the x-height (σ_5).

To be precise, there are even more complications involved [6], as the height of the radical rule is actually taken from the height of the radical glyph rather than the default rule thickness to account for effects of pixel rounding in bitmap fonts.

In OpenType math fonts, we once again find a number of additional parameters for each purpose, as shown in table 6.

While there is a correspondence for all of the parameters built into TEX's typesetting algorithms, it is interesting to note that OpenType math has also introduced some additional parameters related

OpenType parameter	TEX parameter
RadicalExtraAscender	$(= \xi_8)$
RadicalRuleThickness	$(= \xi_8)$
RadicalDisplayStyleVerticalGap	$(= \xi_8 + \frac{1}{4}\sigma_5)$
RadicalVerticalGap	$(= \xi_8 + \frac{1}{4}\xi_8)$
RadicalKernBeforeDegree	e. g. $^5/_{18}$ em
RadicalKernAfterDegree	e. g. $^{10}/_{18}$ em
RadicalDegreeBottomRaisePercent	e. g. 60 %

Table 6: Correspondence of font metric parameters between OpenType and TEX affecting the placement of radicals.

to the placement of the degree of an n th root ($\sqrt[n]{x}$), which is usually handled at the macro level in TEX's format files `plain.tex` or `latex.ltx`:

```
\newbox\rootbox
\def\root#1\of{%
  \setbox\rootbox
  \hbox{$\m@th\scriptscriptstyle{#1}$}%
  \mathpalette\r@@t}
\def\r@@t#1#2{%
  \setbox\z@\hbox{$\m@th#1\sqrtsign{#2}$}%
  \dimen@=\ht\z@ \advance\dimen@-\dp\z@
  \mkern5mu\raise.6\dimen@\copy\rootbox
  \mkern-10mu\box\z@}
```

As shown in the listing, the definition of the `\root` macro contains a number of hard-coded parameters, such as a positive kern before the box containing the degree and negative kern thereafter, expressed in multiples of the font-specific math unit. In addition, there is also a raise factor expressed relative to the size of the box containing the radical sign.

Obviously, the extra OpenType parameters related to the degree of radicals correspond directly to the parameters used internally in the `\root` macro, making it possible to supply a set of font-specific values instead of using hard-coded values expressed in multiples of font-specific units.

In LuaTEX, this approach has been taken one step further, introducing a new `\Uroot` primitive as an extension of the `\Uradical` primitive, making it possible to replace the processing at the macro level by processing at the algorithmic level in LuaTEX's extended math typesetting engine [11].

4.7 General parameters

The final group of OpenType parameters combines a mixed bag of parameters for various purposes. Some of them have a straight-forward correspondence in TEX (such as the math axis position), while others do not have any correspondence at all. As shown in table 7, there are some very noteworthy parameters in this group, which deserve some further explanations in the following paragraphs.

(Script)ScriptPercentScaleDown
These OpenType parameters represent the font sizes of the first and second level script fonts relative to the base font. In TEX math fonts, these parameters do not have a correspondence in the font metrics. Instead they are usually specified at the macro level when a family of math fonts is loaded.

If a font family provides multiple design sizes (as in Computer Modern), font loading of math fonts in TEX might look like the following, using different design sizes, each at their natural size:

OpenType parameter	TEX parameter
ScriptPercentScaleDown	e. g. 70–80 %
ScriptScriptPercentScaleDown	e. g. 50–60 %
DisplayOperatorMinHeight	?? (e. g. 12–15 pt)
(no correspondence)	σ_{20} (e. g. 20–24 pt)
DelimitedSubFormulaMinHeight	σ_{21} (e. g. 10–12 pt)
AxisHeight	σ_{22} (axis height)
AccentBaseHeight	σ_5 (x-height)
FlattenedAccentBaseHeight	?? (capital height)

Table 7: Correspondence of font metric parameters between OpenType and TEX affecting some general aspects of math typesetting.

```
\newfam\symbols
\textfont\symbols=cmsy10
\scriptfont\symbols=cmsy7
\scriptscriptfont\symbols=cmsy5
```

If a font family does not provide multiple design sizes (as in Y&Y MathTime), font loading of math fonts will use scaled-down versions of the base font:

```
\newfam\symbols
\textfont\symbols=mtsy10 at 10pt
\scriptfont\symbols=mtsy10 at 7.6pt
\scriptscriptfont\symbols=mtsy10 at 6pt
```

The appropriate scaling factors depend on the font design, but are usually defined in macro packages or in format files using higher-level macros such as `\DeclareMathSizes` in LATEX.

In OpenType math fonts, it will be possible to package optical design variants for script sizes into a single font by using OpenType feature selectors to address the design variants and using scaling factors as specified in the MATH table.[3]

The corresponding code for font loading of full-featured OpenType math fonts in new TEX engines might look like the following:

```
\newfam\symbols
\textfont\symbols="CambriaMath"
\scriptfont\symbols="CambriaMath:+ssty0"
  scaled <ScriptPercentScaleDown>
\scriptscriptfont\symbols="CambriaMath:+ssty1"
  scaled <ScriptScriptPercentScaleDown>
```

If the font provides optical design variants for some letters and symbols, they will be substituted using the `+ssty0` or `+ssty1` feature selectors, but the scaling factor of (Script)ScriptPercentScaleDown will be applied in any case regardless of substitutions.

[3] As discussed in [12], there are many issues to consider regarding the development of OpenType math fonts besides setting up the font parameters. One such issue is the question of font organization regarding the inclusion of optical design variants into the base font.

DisplayOperatorMinHeight
This OpenType parameter represents the minimum size of big operators in display style. While TEX supports only two sizes of operators, which are used in text style and display style, OpenType can support multiple sizes of big operators and it needs an additional parameter to determine the smallest size to use in display style.

For font designers, it should be easy to set this parameter based on the design size of big operators, e. g. using 14 pt for display style operators combined with 10 pt for text style operators.

DelimitedSubFormulaMinHeight
This OpenType parameter represents the minimum size of delimited subformulas and it might also be applied to the special case of delimited fractions.

To illustrate the significance, some explanations may be necessary to point out the difference between the usual case of fractions with delimiters and the special case of delimited fractions.

If a generalized fraction with delimiters is coded like the following

```
$ \left( {n \atop k} \right) $
```

the contents will be treated as a standard case of a generalized fraction, and the size of delimiters will be determined by taking into account the effects of `\delimiterfactor` and `\delimitershortfall` as set up in the format file.

As a result, we will typically get 10 pt or 12 pt delimiters in text style and 18 pt or 24 pt delimiters in display style. For typical settings, the delimiters have to cover only 90 % of the required size and they may fall short by at most 5 pt.

If a generalized fraction with delimiters is coded like the following

```
$ {n \atopwithdelims() k} $
```

the contents will be treated as a delimited fraction, and in this case the size of delimiters will depend on the `\fontdimen` parameters σ_{20} and σ_{21} applicable in either display style or text style.

As a result, regardless of the contents, we will always get 10 pt delimiters in text style and 24 pt delimiters in display style, even if 18 pt delimiters would be big enough in the standard case.

While DelimitedSubFormulaMinHeight may be the best choice of the OpenType parameters to supply a value for TEX's `\fontdimen` parameters related to delimited fractions, it will be insufficient by itself to represent the distinction between display style and text style values needed in TEX. (Unless we simply assume a factor, such as $\sigma_{20} = 2\,\sigma_{21}$.)

In the absence of a better solution, it may be best to simply avoid using `\atopwithdelims` with OpenType math fonts in the new TEX engines and to redefine user-level macros (such as `\choose`) in terms of `\left` and `\right` delimiters.

(Flattened)AccentBaseHeight
These OpenType parameters affect the placement of math accents and are closely related to design parameters of the font design.

While TEX assumes that accents are designed to fit on top of base glyphs which do not exceed the x-height (σ_5) and adjusts the vertical position of accents accordingly, OpenType provides a separate parameter for this purpose, which doesn't have to match the x-height of the font, but plays a similar role with respect to accent placement.

In addition to that, OpenType has introduced another mechanism to replace accents by flattened accents if the size of the base glyph exceeds a certain size, which is most likely related to the height of capital letters. At the time of writing, support for flattened accents has not yet been implemented in the new TEX engines, but it is being considered for LuaTEX version 0.40 [11].

In view of these developments, font designers are well advised to supply a complete set of values for all the OpenType math parameters since new TEX engines working on implementing full support for OpenType math may start using them sooner rather than later.

So far, we have discussed only one aspect of the information contained in the OpenType MATH table, focusing on the global parameters which correspond to TEX's `\fontdimen` parameters or to built-in rules of TEX's math typesetting algorithms.

Besides those global parameters, there are other data structures in the OpenType MATH table which are also important to consider, as we will discuss in the following sections.

5 Instructions for vertical and horizontal variants and constructions

The concepts of vertical and horizontal variants and constructions in OpenType math are obviously very similar to TEX's concepts of charlists and extensible recipes. However, there are some subtle differences regarding when and how these concepts are applied in the math typesetting algorithms.

In TEX, charlists and extensible recipes are used only in certain situations when typesetting elements such as big operators, big delimiters, big radicals or wide accents. In OpenType math fonts, these concepts have been extended and generalized, allowing them to be used also for other stretchable elements such as long arrows or over- and underbraces.

5.1 Vertical variants and constructions

Big delimiters When typesetting big delimiters or radicals TEX uses charlists to switch to the next-larger vertical variants, optionally followed by extensible recipes for vertical constructions. In OpenType math, these concepts apply in the same way.

It is customary to provide at least four fixed-size variants, using a progression of sizes such as 12 pt, 18 pt, 24 pt, 30 pt, before switching to an extensible version, but there is no requirement for that other than compatibility and user expectations.[4]

Font designers are free to provide any number of additional or intermediate sizes, but in TEX they used to be limited by constraints such as 256 glyphs per 8-bit font table and no more than 16 different heights and depths in TFM files. In OpenType math fonts, they are no longer subject to such restrictions, and in the example of Cambria Math big delimiters are indeed provided in seven sizes.

Big operators When typesetting big operators TEX uses the charlist mechanism to switch from text style to display style operators, but only once. There is no support for multiple sizes of display operators, nor are there extensible versions.

In OpenType math, these concepts have been extended, so it would be possible to have multiple sizes of display style operators as well as extensible versions of operators, if desired.

While LuaTEX has already implemented most of the new features of OpenType math, it has not yet addressed additional sizes of big operators, and it is not clear how that would be done.

Most likely, this would require some changes to the semantics of math markup at the user level, so that operators would be defined to apply to a scope of a subformula, which could then be measured to determine the required size of operators.

In addition, such a change might also require adding new parameters to decide when an operator is big enough, similar to the role of the parameters `\delimiterfactor` and `\delimitershortfall` in the case of big delimiters.

5.2 Horizontal variants and constructions

Wide accents When typesetting wide math accents TEX uses charlists to switch to the next-larger horizontal variants, but it doesn't support extensible recipes for horizontal constructions.

As a result, math accents in traditional TEX fonts cannot grow beyond a certain maximum size, and stretchable horizontal elements of arbitrary size have to be implemented using other mechanisms, such as alignments at the macro level.

In OpenType math, these concepts have been extended, making it possible to introduce extensible versions of wide math accents (or similar elements), if desired. In addition, new mechanisms for bottom accents have also been added, complementing the existing mechanisms for top accents.

Over- and underbraces When typesetting some stretchable elements such as over- and underbraces, TEX uses an alignment construction at the macro level to get an extensible brace of the required size, which is then typeset as a math operator with upper or lower limits attached.

While it would be possible to define extensible over- and underbraces in OpenType math fonts as extensible versions of math accents, the semantics of math accents aren't well suited to handle upper or lower limits attached to those elements.

In LuaTEX, new primitives `\Uoverdelimiter` and `\Uunderdelimiter` have been added as a new concept to represent stretchable horizontal elements which may have upper or lower limits attached. The placement of these limits is handled similar to limits on big operators in terms of so-called 'stretch stacks' as discussed earlier in section 4.2.

Long arrows In TEX math fonts, long horizontal arrows are constructed at the macro level by overlapping the glyphs of short arrows and suitable extension modules (such as − or =). Similarly, arrows with hooks or tails are constructed by overlapping the glyphs of regular arrows and suitable glyphs for the hooks or tails.

In OpenType math fonts, all such constructions can be defined at the font level in terms of horizontal constructions rather than relying on the macro level. However, in most cases such constructions will also contain an extensible part, making the resulting long arrows stretchable as well.

In LuaTEX, stretchable long arrows can also be defined using the new primitives `\Uoverdelimiter` as discussed in the case of over- and underbraces. The placement of limits on such elements more or less corresponds to using macros such as `\stackrel` to stack text on top of a relation symbol.

5.3 Encoding of variants and constructions

In traditional TEX math fonts, glyphs are addressed by a slot number in a font-specific output encoding. Each variant glyph in a charlist and each building block in an extensible recipe needs to have a slot of its own in the font table. However, only the entry points to the charlists need to be encoded at the

[4] At the macro level these sizes can be accessed by using `\big` (12 pt), `\Big` (18 pt), `\bigg` (24 pt), `\Bigg` (30 pt).

macro level and these entry points in a font-specific input encoding do not even have to coincide with the slot numbers in the output encoding.

In OpenType math fonts, the situation is somewhat different. The underlying input encoding is assumed to consist of Unicode characters. However these Unicode codes are internally mapped to font programs using glyph names, which can be either symbolic (such as `summation` or `integral`) or purely technical (such as `uni2345` or `glyph3456`).

With few exceptions, most of the variant glyphs and building blocks cannot be allocated in standard Unicode slots, so these glyphs have to be mapped to the private use area with font-specific glyph names. In Cambria Math, variant glyphs use suffix names (such as `glyph.vsize<n>` or `glyph.hsize<n>`), while other fonts such as Asana Math use different names (such as `glyphbig<n>` or `glyphwide<n>`).

For font designers developing OpenType math fonts, setting up vertical or horizontal variants is pretty straight-forward, such as

```
summation : summation.vsize1 summation.vsize2 ...
integral  : integral.vsize1  integral.vsize2  ...
```

or

```
tildecomb : tildecomb.hsize1 tildecomb.hsize2
```

provided that the variant glyphs use suffix names.

Setting up vertical or horizontal constructions is slightly more complicated, as it also requires some additional information which pieces are of fixed size and which are extensible, such as

```
integral : integralbt:0 uni23AE:1 integraltp:0
```

or

```
arrowboth :
  arrowleft.left:0 uni23AF:1 arrowright.right:0
```

It is interesting to note that some of the building blocks (such as `uni23AE` or `uni23AF`) have Unicode slots by themselves, while others have to be placed in the private use area, using private glyph names such as `glyph.left`, `glyph.mid`, or `glyph.right`.

Moreover, vertical or horizontal constructions may also contain multiple extensible parts, such as in the example of over- and underbraces, where the left, middle, and right parts are of fixed size while the extensible part appears twice on either side.

6 Additional glyph metric information

Besides the global parameters and the instructions for vertical and horizontal variants and constructions, there is yet another kind of information stored in the OpenType MATH table, containing additions to the font metrics of individual glyphs.

In traditional TEX math fonts, the file format of TFM fonts provides only a limited number of fields to store font metric information. As a workaround, certain fields which are needed only in math mode are stored in a rather non-intuitive way by overloading fields for other purposes [13].

For example, the nominal width of a glyph is used to store the subscript position, while the italic correction is used to indicate the horizontal offset between the subscript and superscript position.

As a result, the nominal width doesn't represent the actual width of the glyph and the accent position may turn out incorrect. As a secondary correction, fake kern pairs with a so-called skewchar are used to store an offset to the accent position.

In OpenType math fonts, all such non-intuitive ways of storing information can be avoided by using additional data fields for glyph-specific font metric information in the MATH table.

For example, the horizontal offset of the optical center of a glyph is stored in a `top_accent` table, so any adjustments to the placement of math accents can be expressed in a straight-forward way instead of relying on kern pairs with a skewchar.

Similarly, the italic correction is no longer used for the offset between superscripts and subscripts. Instead, the position of indices can be expressed more specifically in a `math_kern` array, representing cut-ins at each corner of the glyphs.

7 Summary and conclusions

In this paper, we have tried to help improve the understanding of the internals of OpenType math fonts. We have done this in order to contribute to the much-needed development of math support for Latin Modern and TEX Gyre fonts.

In the previous sections, we have discussed the parameters of the OpenType MATH table in great detail, illustrating the similarities and differences between traditional TEX math fonts and OpenType math fonts. However, we have covered other aspects of OpenType math fonts only superficially.

For a more extensive overview of the features and functionality of OpenType math fonts as well as a discussion of the resulting challenges to font developers, readers are also referred to [12].

In view of the conference motto, *TEX: at a turning point, or at the crossroads?*, it is interesting to note that recent versions of LuaTEX have started to provide a full-featured implementation of OpenType math support in LuaTEX and ConTEXt [14, 15], which differs significantly from the implementation of OpenType math support in XETEX [10].

In this paper, we have pointed out some of these differences, but further discussions of this topic are beyond the scope of this paper.

Acknowledgments

The author once again wishes to thank Bogusław Jackowski for permission to reproduce and adapt the figures from his paper *Appendix G Illuminated* [6]. In addition, the author also wishes to acknowledge feedback and suggestions from Taco Hoekwater and Hans Hagen regarding the state of OpenType math support in LuaTEX.

References

[1] Murray Sargent III: Math in Office Blog.
http://blogs.msdn.com/murrays/default.aspx

[2] Murray Sargent III: High-quality editing and display of mathematical text in Office 2007.
http://blogs.msdn.com/murrays/archive/2006/09/13/752206.aspx

[3] John Hudson, Ross Mills: Mathematical Typesetting: Mathematical and scientific typesetting solutions from Microsoft. Promotional Booklet, Microsoft, 2006.
http://www.tiro.com/projects/

[4] George Williams: FontForge. Math typesetting information.
http://fontforge.sourceforge.net/math.html

[5] Apostolos Syropoulos: Asana Math.
http://mirror.ctan.org/fonts/Asana-Math/

[6] Bogusław Jackowski: Appendix G Illuminated. *TUGboat* 27(1), 2006, pp. 83–90, Proceedings of the 16th EuroTEX Conference 2006, Debrecen, Hungary.
http://www.gust.org.pl/projects/e-foundry/math-support/tb87jackowski.pdf

[7] Ulrik Vieth: Understanding the æsthetics of math typesetting. *Biuletyn GUST*, pp. 5–12, 2008. Proceedings of the 16th BachoTEX Conference 2008, Bachotek, Poland.
http://www.gust.org.pl/projects/e-foundry/math-support/vieth2008.pdf

[8] Microsoft Typography: OpenType specification. Version 1.5, May 2008.
http://www.microsoft.com/typography/otspec/

[9] Yannis Haralambous: Fonts and Encodings. O'Reilly Media, 2007. ISBN 0-596-10242-9
http://oreilly.com/catalog/9780596102425/

[10] Will Robertson: The unicode-math package. Version 0.3b, August 2008.
http://github.com/wspr/unicode-math/

[11] Taco Hoekwater: LuaTEX Reference Manual. Version 0.37, 31 March 2009.
http://www.luatex.org/svn/trunk/manual/luatexref-t.pdf

[12] Ulrik Vieth: Do we need a 'Cork' math font encoding? *TUGboat*, 29(3), 426–434, 2008. Proceedings of the TUG 2008 Annual Meeting, Cork, Ireland.
http://tug.org/TUGboat/tb29-3/tb93vieth.pdf

[13] Ulrik Vieth: Math Typesetting: The Good, The Bad, The Ugly. *MAPS*, 26, 207–216, 2001. Proceedings of the 12th EuroTEX Conference 2001, Kerkrade, Netherlands.
http://www.ntg.nl/maps/26/27.pdf

[14] Taco Hoekwater: Math extensions in LuaTEX. *MAPS* 38, Voorjaar 2009, pp. 22–31 (as "Math in LuaTEX 0.40"). First presented at BachoTEX 2009.

[15] Hans Hagen: Unicode math in ConTEXt MkIV. *MAPS* 38, Voorjaar 2009, pp. 32–46. First presented at BachoTEX 2009.

◇ Ulrik Vieth
Vaihinger Straße 69
70567 Stuttgart
Germany
ulrik dot vieth (at) arcor dot de

A closer look at TrueType fonts and pdfTEX

Hàn Thế Thành

Abstract

Explanations and examples of using TrueType fonts directly with pdfTEX, especially the complications regarding encodings and glyph names.

1 Glyph identity in Type 1 vs. TrueType

The most common outline font format for TEX is Type 1. The TrueType format is rather different from Type 1, and getting it right requires some extra work. In particular, it is important to understand how TrueType handles encoding and glyph names (or more precisely, glyph identity).

We start with Type 1, since most TEX users are more familiar with it. In the Type 1 format glyphs are referred to by names (such as '`/A`', '`/comma`', and so on). Each glyph is identified by its name; so, given a glyph name, it is easy to tell whether or not a Type 1 font contains that glyph. Encoding with Type 1 is therefore simple: for each number n in the range 0 to 255, an encoding tells us the name of the glyph that should be used to render (or display) the charcode n.

With TrueType the situation is not that simple. TrueType does not use names to refer to glyphs, but rather so-called "indices": each glyph is identified by an index, not a name. These indices are simply numbers that differ from font to font. The TrueType format handles encodings by a mechanism called "cmap", which (roughly) consists of tables mapping from character codes to glyph indices. A TrueType font can contain one or more such tables, each corresponding to an encoding.

2 Glyph names vs. Unicode in TrueType

Because glyph names are not strictly necessary for TrueType, they are not always available inside a TrueType font. Given a TrueType font, one of the following cases may arise.

- The font contains correct names for all glyphs. This is the ideal situation and is indeed often the case for high-quality Latin fonts.
- The font contains wrong names for all or most of its glyphs. This is the worst situation that often happens with poor-quality fonts, or fonts converted from other formats.
- The font contains no glyph names at all. Newer versions of Palatino fonts by Linotype (v1.40, coming with Windows XP) are examples of this.
- the font contains correct names for most glyphs, and no names or wrong names for a few glyphs. This happens from time to time.

One may wonder how the situation can be so complex with glyph names in TrueType and still get anything typeset correctly. The reason is that Type 1 fonts rely on correct names to work properly. Thus, if a glyph has a wrong name, it gets noticed immediately. In contrast, as mentioned before, TrueType does not use names for encoding. So, if glyph names in a TrueType font are wrong or missing, it is usually not a big deal and can easily go unnoticed.

The potential problem with using TrueType in pdfTEX is that we TEX users are accustomed to the Type 1 encoding convention, which relies on correct glyph names. Furthermore, most font tools rely on this convention and all encoding files (`.enc` files) use glyph names. But, as explained above, glyph names in TrueType are not reliable. If we encounter a font that does not have correct names for its glyphs, we need to do some more work.

If glyph names are not correct, we need another way to refer to a glyph in TrueType fonts. The most reliable way seems to be via Unicode: usable TrueType fonts must provide a correct mapping from Unicode value to glyph index.

Since version 1.21a pdfTEX has supported the naming convention 'uniXXXX' in encoding (`.enc`) files. This makes sense only with TrueType fonts. When pdfTEX sees for example '`/uni12AB`', it

- reads the ⟨*unicode*⟩→⟨*glyph-index*⟩ table from the font, and
- looks up the value '12AB' in the table, and if found then uses the relevant glyph index.

The ttf2afm utility does the same lookup when it sees names like 'uni12AB'.

3 Using TrueType in pdfTEX

Let's review the minimal steps to get a TrueType font working with pdfTEX:

- Generate an afm from the TrueType font using ttf2afm. Example:

```
ttf2afm -e 8r.enc -o times.afm times.ttf
```

- Convert afm to tfm using any suitable tool — afm2tfm, fontinst, afm2pl, etc. Example:

```
afm2tfm times.afm -T 8r.enc
```

- Define the needed map entry for the font. Example:

```
\pdfmapline{%
  +times TimesRoman <8r.enc <times.ttf}
\font\f=times
\f Hello this is Times.
```

(The font name 'TimesRoman' used in the map line is declared inside the `times.ttf` file.)

The above deals with the easiest case: when glyph names are correct.

Now let us consider a font where we cannot rely on glyph names: Palatino version 1.40 from Linotype, for example. Let us assume that we want to use the T1 encoding with this font. So we put `pala.ttf` and `ec.enc` in the current directory before proceeding further.

First attempt:

```
ttf2afm -e ec.enc -o pala.afm pala.ttf
```

However, since the names in `ec.enc` are not available in `pala.ttf` — in fact there are no names inside this font — we get a bunch of warnings:

```
Warning: ttf2afm (file pala.ttf): no names
  available in 'post' table ...
Warning: ttf2afm (file pala.ttf): glyph 'grave'
  not found
...
```

and the output `pala.afm` will contain no names at all, but instead weird entries like 'index123'. Furthermore, glyphs are not encoded:

```
C -1 ; WX 832 ; N index10 ; B 24 -3 807 689 ;
```

We try again, this time without any encoding:

```
ttf2afm -o pala.afm pala.ttf
```

Since this time we did not ask ttf2afm to reencode the output afm, we get only the first warning:

```
Warning: ttf2afm (file pala.ttf): no names ...
```

but the afm output is the same as in the previous attempt. This is not useful, since there is little we can do with names like 'index123'.

So we try to go with Unicode:

```
ttf2afm -u -o pala.afm pala.ttf
```

This time we get different warnings, such as:

```
Warning: ttf2afm (file pala.ttf): glyph 108 has
multiple encodings (the first one being used):
uni0162 uni021A
```

At first sight it is hard to understand what ttf2afm is telling us with this message. So let us recap the connection between glyph name, glyph index and Unicode value:

- TrueType glyphs are identified internally by an index, not a name.
- ⟨*glyph-name*⟩→⟨*glyph-index*⟩ is optional, and the information may be wrong, if present. Likewise ⟨*glyph-index*⟩→⟨*glyph-name*⟩.
- ⟨*unicode*⟩→⟨*glyph-index*⟩, on the other hand, is (almost) always present and reliable.
- ⟨*glyph-index*⟩→⟨*unicode*⟩ is not always reliable, and need not even be a mapping, since there can be more than one Unicode value mapping to a given glyph index. That is, given a glyph index, there may be no corresponding Unicode value, or there may be more than one. If there is none, the glyph index will be used ('index123', for example). Now suppose that there are more than one, as in the example above, where 0162 and 021A are both mapped to glyph index 108

  In sum, we have asked ttf2afm to print glyphs by Unicode, and ttf2afm cannot know for sure which value to use. Hence it outputs the first Unicode value and issues the warning.

If all we want to do is to use `pala.ttf` with the T1 encoding, probably the easiest way is to create a new enc file `ec-uni.enc` from `ec.enc`, with all glyph names replaced by Unicode values. (This simple approach does not handle ligatures; see below.) This can be done easily enough by a script that reads the AGL (Adobe Glyph List, `http://www.adobe.com/devnet/opentype/archives/glyphlist.txt`) and converts all glyph names to Unicode.

Assuming that we have such a `ec-uni.enc`, the steps needed to create the tfm are as follows:

```
ttf2afm -u -e ec-uni.enc -o pala-t1.afm pala.ttf
afm2pl pala-t1.afm pltotf pala-t1.pl
pltotf pala-t1.pl
```

We could then use the font in pdfTEX as follows:

```
\pdfmapline{+pala-t1 <ec-uni.enc <pala.ttf}
\font\f=pala-t1
\f This is Palatino in the T1 encoding.
```

4 General solutions for fontinst et al.

If we want to do more than just using `pala.ttf` with T1 encoding, for example processing the afm output with fontinst for a more complex font setup, then we must proceed slightly differently. Having an afm file where all glyph names are converted to the 'uniXXXX' form, as we have done above, is not very useful for fontinst. Instead, we need an afm file with AGL names, do our processing, and then convert back to 'uniXXXX'. We can do this as follows.

- Generate the afm with 'uniXXXX' glyph names:

  ```
  ttf2afm -u -o pala.afm pala.ttf
  ```
- Convert that `pala.afm` to `pala-agl.afm`, so that `pala-agl.afm` contains only AGL names. A script similar to the one mentioned above can do this.
- Process `pala-agl.afm` with fontinst or whatever else is desired.
- In the final stage, when we have the tfm's from fontinst (et al.), plus the map entries (from fontinst or created manually), we need to replace the encoding by its counterpart with the

'uniXXXX' names, since that is what the actual TrueType font requires. For example, if fontinst tells us to add a line saying

```
pala-agl-8r <8r.enc <pala.ttf
```

to our map file, we need to change that line to

```
pala-agl-8r <8r-uni.enc <pala.ttf
```

where `8r-uni.enc` is derived from `8r.enc` by converting all glyph names to the 'uniXXXX' form.

The encoding files distributed with the TEX Gyre fonts cover just about everything a typical TEX user needs. Those encodings have been converted to the 'uniXXXX' form for your convenience and are available at `http://tug.org/fontname`, with names such as `q-ec-uni.enc`.

5 Disappearing glyphs and final tips

Another problem that happens from time to time is being sure that a glyph exists inside a font but we don't get that glyph in the pdfTEX output.

The likely cause is the glyph being referenced by different names at the various stages the process of creating support for the font, e.g., the tfm, vf, enc and map files. For example, the names 'dcroat', 'dbar', 'dslash' and 'dmacron' can all refer to the same glyph in a TrueType font. In general, the origin of a glyph name can come from several sources:

- the individual font itself;
- a predefined scheme called "the standard Macintosh ordering of glyphs" (unfortunately the TrueType specifications by various companies (Apple, Microsoft and Adobe) are not consistent in this scheme and there are small differences, for example 'dmacron' vs. 'dslash');
- the result of the ⟨*unicode*⟩→⟨*glyph-name*⟩ conversion, according to the AGL.

In such situations, probably the easiest and most reliable way to get the glyph we want is to use a font editor like FontForge (`http://fontforge.sf.net`), look into the font to discover the specified Unicode for the glyph and then use the 'uniXXXX' form to instruct ttf2afm and pdfTEX to pick up that glyph.

Finally, another way to get a problematic TrueType font to work with pdfTEX is to forget all of the above and simply convert the font to Type 1 format using FontForge. While it sounds like a quick hack, there is nothing necessarily wrong with this; it can be a simple and effective workaround.

◇ Hàn Thế Thành
River Valley Technologies
`thanh (at) river-valley dot org`

The Open Font Library

Dave Crossland

A large number of redistributable and modifiable fonts now exist and are useful for all computer users. But most people are unaware of this, and have no way to easily browse and use these fonts. The Open Font Library, at `http://www.openfontlibrary.org`, is a project that aims to enable this by visually showcasing them on the web.

The project has three aims:

1. To be a website for graphic designers and everyday users to browse and download free software fonts, akin to proprietary font vendors' websites.
2. To introduce type designers not deeply involved in the free software community to it, and help them to share their fonts with the community, such as GNU/Linux distributions and TEX Live, through a single, simple process.
3. To be a central place to link webfonts from, using the upcoming CSS3 font linking technology. This is already available in webkit-based browsers like Midori, Safari, and on Android phones, and will be in Opera 10 and Mozilla Firefox 3.5.

The imminent arrival of web fonts is an important window of opportunity for free software fonts because it is where they demonstrate immediate usefulness. There will never be any question over whether a free software font can be used on the web — if it is hosted in the Open Font Library, it can simply be linked to directly.

There was a campaign to apply DRM to web fonts at `http://www.fontembedding.com`, but that seems to not have been successful. The Open Font Library is well positioned to offer advice to balance that debate and showcase what web font linking can do.

The project started in 2004 but had very little development effort on a community basis, essentially because font developers are not web developers. This means that, despite already collecting nearly 150 fonts, it did not seem like a worthwhile cause to ambitious type designers and graphic designers.

However, the long term viability of the project has always been assured since its hosting is provided by the Oregon State University's Open Source Lab (`http://osuosl.org`).

So in September 2008 I sought sponsorship to relaunch the website by hiring British freelance web developers, and brothers, Ben and James Weiner. With Karl Berry's help, TUG formed a "Free Font Fund" to take care of the administrative overheads of this effort, and make donations to the fund tax deductible for patrons in the USA. By the end of October, the fund totalled US$12,000 and patrons included TUG, the TEX-based typesetting company River Valley Technologies, Prince XML and Mozilla.

In order to raising the site's profile, the Weiners originated a totally new visual identity for the site, so that it appeals to the international graphic design community, yet without losing the inclusive free software community attitude. They also wrote compelling copy for graphic designers, explaining what the site is about. This included documenting how to contribute fonts, and the licensing issues that typically vex type designers who contribute to the free software community.

Ben Weiner customised the ccHost (`http://wiki.creativecommons.org/CcHost`) content management system as part of this work, and integrated MediaWiki to cover the site's documentation needs.

Ed Trager is an expert in Asian writing systems, and he worked hard to develop new programs that signpost what is available in the library. His 'Font Playground' is an interactive AJAX tool to generate enticing previews of each font using an on-screen keyboard that supports all major writing systems. He also developed a Unicode coverage analysis tool, Fontaine (`http://fontaine.sf.net`).

The site has been in public beta at `http://openfontlibrary.fontly.org` for some time now, and I hope that it will go live by the time this sees print. Comments are very welcome.

I would like to thank again the generous patrons: OSU-OSL, TUG, River Valley Technologies, Prince XML and Mozilla.

◇ Dave Crossland
University of Reading, UK
`dave (at) lab6 dot com`
`http://www.openfontlibrary.org`

Bibliographies

Managing bibliographies with LaTeX

Lapo F. Mori

Abstract

The bibliography is a fundamental part of most scientific publications. This article presents and analyzes the main tools that LaTeX offers to create, manage, and customize both the references in the text and the list of references at the end of the document.

1 Introduction

Bibliographic references are an important, sometimes fundamental, part of academic documents. In the past, preparation of a bibliography was difficult and tedious mainly because the entries were numbered and ordered by hand. LaTeX, which was developed with this kind of document in mind, provides many tools to automatically manage the bibliography and make the authors' work easier. How to create a bibliography with LaTeX is described in section 2, starting from the basics and arriving at advanced customization. Bibliographic styles for both the list of references and in-text-citations are analyzed in section 3. The last two sections (4 and 5) analyze two of the most powerful packages available: natbib and BibLaTeX.

2 Bibliography with LaTeX

There are two main ways to compose a bibliography with LaTeX: automatically with the BibTeX program that uses external bibliographic databases (section 2.2), or manually with the `thebibliography` environment that allows to include all the bibliographic information in the source `.tex` file (section 2.3). Regardless from the strategy chosen, citations can be added to the text with the same commands, as shown in section 2.1.

2.1 References in the text

Citations can be added to the text with the command `\cite{key}` (and its variants), where `key` corresponds to the citekey field in the `.bib` file (if using BibTeX, section 2.2) or the key of `\bibitem` (if using the `thebibliography` environment, section 2.3). When compiling the source, `\cite{key}` is linked to the respective `\bibitem` and substituted by the appropriate reference (numbered, author-year, or footnote depending on the style chosen).

Multiple citations can be added by separating with a comma the bibliographic keys inside the same `\cite` command; for example

```
\cite{Goossens1995,Kopka1995}
```

gives

(Goossens et al., 1995; Kopka and Daly, 1995)

Bibliographic entries that are not cited in the text can be added to the bibliography with the `\nocite{key}` command. The `\nocite{*}` command adds all entries to the bibliography.

2.2 Automatic creation with BibTeX

BibTeX is a separate program from LaTeX that allows creating a bibliography from an external database (`.bib` file). These databases can be conveniently shared by different LaTeX documents. BibTeX, which will be described in the following paragraphs, has many advantages over the `thebibliography` environment; in particular, automatic formatting and ordering of the bibliographic entries.

2.2.1 How BibTeX works

BibTeX requires:

1. one or more bibliographic databases `.bib`;
2. a bibliographic style `.bst`;
3. that the `.tex` file contains: commands that specify what style `.bst` and database `.bib` to use, and citations in the text with the `\cite` and similar commands, as in:

```
\documentclass{...}
...
\begin{document}
See \citet{Kopka1995}.
..
\bibliographystyle{plainnat}
\bibliography{database}
\end{document}
```

4. that the document is compiled in the following order (let's assume that the principal file is called `document.tex`):

```
latex document
bibtex document
latex document
latex document
```

The first time LaTeX runs, the `\bibliographystyle` command writes the name of the style `.bst` to be used into the `.aux` file, each `\cite` command writes a note into the `.aux` file, and the `\bibliography` command writes into the `.aux` file the name of the `.bib` database(s) to be used. At this stage LaTeX does not substitute the `\cite` in the text: the `.dvi`

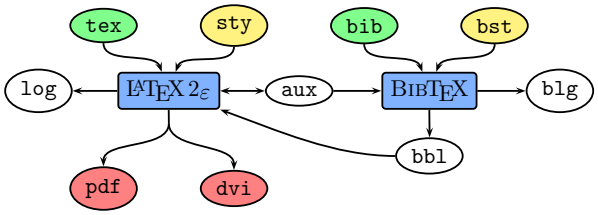


Figure 1: Data flow between LaTeX and BibTeX to create a `.pdf` or a `.dvi` (■) from the data files `.tex` and `.bib` (■) and the style files `.sty` and `.bst` (■).

shows question marks for every `\cite` and the `.log` file reports warnings for unknown references.

BibTeX will then read the `.aux` file, the bibliographic style `.bst` and `.bib` database(s) specified. For every `\cite` entry, it will read the information contained in the `.bib` files and format it according to the style `.bst`. At the end it will order the entries according to the style `.bst` and will write the result into the `.bbl` file. The `.bbl` contains the bibliographic entries organized in a `thebibliography` environment; the entries follow the format specified by the `.bst` style and have the information contained in the `.bib` database. This file will be read every time that the source `.tex` file is compiled.

When running LaTeX again, it will produce a warning due to the fact that the references are not defined yet and will substitute the `\bibliography` command with the content of the `.bbl` file. It will also write a note into the `.aux` file for every `\bibitem`.

At the next run, LaTeX will find the definition for every `\cite` in the `.aux` file and all the citations in the text will be resolved. The `.blg` file is a log file created by BibTeX, similar to the `.log` file for LaTeX.

This flow, represented in Fig. 1, has to be repeated if the citations in the text, the `.bib` database or the `.bst` style are modified.

2.2.2 Structure of a bibliography database

The `.bib` files are databases that store the information for each bibliographic entry. Here is an example:

```
@BOOK{Kopka1995,
  title = {A Guide to {\LaTeX} -- Document
       Preparation for Beginners and Advanced Users
       },
  publisher = {Addison-Wesley},
  year = {1995},
  author = {H. Kopka and P. W. Daly},
}
```

Each type of entry (`BOOK`, `ARTICLE`, `PROCEEDINGS`, etc.) offers many fields in addition to those shown in the example (`title`, `publisher`, `year`, `author`). Further details can be found in Patashnik (1998).

2.2.3 Rules for using BibTeX

The syntax of BibTeX entries is very intuitive. The following paragraphs will discuss the most common rules, all the details can be found in Patashnik (1998).

Capital letters. The BibTeX styles control capital and lower case letters, especially for the titles. This behavior is very convenient because it ensures a uniform format for the entries but could cause problems in some specific situations such as acronyms, chemical formulæ, etc. In these cases the user needs to enclose in braces the letters whose capitalizations should not be changed by BibTeX, as in the following example for "CO":

```
  title = {{CO} pollution},
```

The `.bst` style can be used to automatically change the capitalization behavior for titles (e.g. title style or sentence style) instead of manually enclosing all the titles in braces.

Commands. Since BibTeX modifies the capitalization depending on the `.bst` style, some LaTeX commands inside titles may not work. For example, if the sentence style is in use and one of the titles contains the command `\LaTeX`, this will be converted to `\latex` and will give the following error:

```
! Undefined control sequence.
```

This also can be avoided by enclosing such commands in braces: `{\LaTeX}`.

Accents and special characters. A similar problem arises with the commands for accents and special characters such as "ö" (`\"{o}`), "ç" (`\c{c}`), "ñ" (`\~{n}`), etc. Again, enclosing the command in braces solves the issue:

```
  title = {Writing the curriculum vit{\ae}},
```

Name lists. When more than one name is present in the `author` or `editor` fields, they have to be separated by ␣`and`␣ (a space before and after):

```
  author = {Author1 and Author2},
```

Each name has four parts: Name, von, Surname, Jr. The surname is mandatory, all other parts are optional. BibTeX accepts two different syntaxes for names:

- **Name von Surname**: for example "Pico della Mirandola" has to be written as `Pico della Mirandola`.
- **von Surname, Jr, Name**: for example "Pico della Mirandola II" has to be written as `della Mirandola, II, Pico`.

The second form is more general because the first one cannot be used when the Jr field is present or when the surname contains more than one word and the von part is not present. The `.bst` style takes care of abbreviating (or not) names. If the author wants to use abbreviated names, the `.bib` database can also have abbreviated names. For example:

```
  author = {Mori, L. F.},
```

If the list of authors or editors is too long, it can be ended by `and others` that will then be formatted by the style as "et al."

```
  author = {Conte, G. B. and Pianezzolla, E. and
       Chiesa, P. and Rossi, G. and others},
```

URL. The standard BibTeX styles do not provide a field for web addresses. A solution with these styles is to use the `howpublished` field:

```
@MISC{...,
  ...,
  howpublished = {\url{http://...}},
}
```

A better solution is to use styles that provide the `url` field, such as `plainnat` and `abbrevnat` that come with the natbib package (section 4) or those that come with the babelbib package (section 2.7.3). Moreover, the `url` field is also provided by the custom styles created with `makebst` (section 3.2.2). In all these cases, the syntax is:

```
@ARTICLE{...,
  ...,
  url = {http://...},
}
```

Problems might arise when a web site address is long and close to the write margin. When the document is compiled with pdfLaTeX, the driver can break the links over several lines. On the other hand, when the document is compiled with the `dvips` driver (LaTeX → `.dvi` → PostScript → `.pdf`), the breakurl[1] package must be loaded to support breaking links into several parts while retaining the hyperlink.

Months. BibTeX provides macros to automatically manage the month name specified with the `month` field; these macros automatically manage the full form (March) and abbreviated form (Mar.) and the language (Marzo) depending on the `.bst` style in use. In order to take advantage of these macros, the month has to be written in the abbreviated English form.[2]

[1] The breakurl package was written by Vilar Camara Neto and the last version was released in 2009.

[2] The abbreviated English form consists in the first three letters of the month: `jan`, `feb`, `mar`, `apr`, `may`, `jun`, `jul`, `aug`, `sep`, `oct`, `nov`, `dec`.

2.2.4 Creating and managing bibliography databases

As should be clear from the example in section 2.2, the `.bib` file syntax is so intuitive that these files can be easily written with any text editor. Several programs, however, are dedicated to the creation and management of `.bib` databases, such as BibTeXMng[3] (shareware for Windows), BibDesk[4] (open-source for Mac OS X), KBibTeX[5] (open-source for Linux), Pybliographer[6](open-source for Linux), bibliographer[7] (freeware for Linux), Bibwiki[8] (Internet based), cb2Bib[9] (freeware for Windows and Linux), Zotero[10] (open-source multi-platform plugin for Firefox), and JabRef[11] (multi-platform open-source). The last three programs are particularly interesting and will be discussed in the next paragraphs.

cb2Bib. cb2bib (which stands for "clipboard to BibTeX") is a program for extracting bibliographic information from unformatted sources such as `.pdf` files, web sites, and email. cb2bib reads the content of the clipboard and process it according to predefined patterns.[12] If no predefined format pattern is found, cb2bib can still be used for manual data extraction.

Zotero. Zotero is an open-source multi-platform (Windows, Linux, and Mac OS X) Firefox[13] plugin that allows to gather, manage, and analyze bibliographic references. Being an extension of a web browser, Zotero is particularly useful for extracting bibliographic data from the Internet. The reference database can be exported as a `.pdf` file,[14] a text file, a BibTeX database, and several other formats.

[3] `http://www.latexsoft.com/bibtexmng.htm`

[4] `http://bibdesk.sourceforge.net/`

[5] `http://www.unix-ag.uni-kl.de/~fischer/kbibtex/`

[6] `http://www.pybliographer.org/Welcome/`

[7] `http://bibliographer.homelinux.net/`

[8] `http://wolfgang.plaschg.net/bibwiki/`

[9] `http://www.molspaces.com/d_cb2bib-overview.php`

[10] `http://www.zotero.org/`

[11] `http://jabref.sourceforge.net/`

[12] cb2bib implements patterns to extract data from the following formats: PNAS Table of Contents Alert (`http://www.pnas.org/`), ISI Web of Knowledge Table of Contents Alert (`http://isiknowledge.com/`), Wiley InterScience Journal Abstracts (`http://www.interscience.wiley.com/`), American Chemical Society Publications (`http://pubs.acs.org/`), ScienceDirect (`http://www.sciencedirect.com/`), Digital Bibliography & Library Project (`http://dblp.uni-trier.de/`), Nature from First Paragraph (`http://www.nature.com/nature/`), IOP Electronic Journals (`http://www.iop.org/EJ/`), JSTOR (`http://www.jstor.org/`), ISI Reference Format (`http://scientific.thomson.com/isi/`), RIS Reference Format.

[13] `http://www.mozilla.com/firefox/`

[14] The list in the `.pdf` file is formatted according to predefined styles that can be selected by the user.

Zotero can be used together with WYSIWYG[15] software such as Microsoft Word[16] and OpenOffice.[17] Zotero can also perform advanced searches in its libraries and import entries from several formats. A unique feature of Zotero is the ability to create online libraries that can be used from different computers over the Internet.

JabRef. JabRef is an open-source multi-platform (Windows, Linux, and Mac OS X) software written in Java for creating and managing bibliographic databases in the BibTeX format. Entries can be created by editor panels whose fields depend on the type of entry (book, article, proceedings, etc.); these panels can also be customized by adding or removing fields.

JabRef can be used to search articles on Medline[18] and Citeseer[19] and to import bibliographic entries from several formats such as BibTeXML, CSA, Refer/Endnote, ISI Web of Science, SilverPlatter, Medline/Pubmed (XML), Scifinder, OVID, INSPEC, Biblioscape, Sixpack, JSTOR and RIS. JabRef offers advanced search and management based on keywords. Databases can be printed or exported in HTML, Refer/Endnote, Docbook, BibTeXML, MODS, RTF, and OpenOffice.

JabRef can automatically create the BibTeX keys (for example by taking the first author's surname and the publication year) and insert the citations into several text editors such as LyX, Kile, and WinEdt. JabRef can associate a `.pdf` file to every entry and open it with external software. It can also associate a `url` or a DOI;[20] in both cases, JabRef can open a web browser on the corresponding page.

Bibliography resources on the Internet. Several web sites, both of journals and of bibliographic databases, can be used to export entries directly in the BibTeX format. Some of the journals are ACM,[21] Science,[22] PNAS,[23] and The Journal of Chemical Physics.[24] Some of the databases are Lead2Amazon[25] (a web site that uses Amazon.com, .ca, .co.uk, .de, .fr, .co.jp to automatically generate BibTeX entries), Google Scholar[26] (Google search engine dedicated to scientific publications; select "BibTeX" in "Scholar Preferences"), BibSonomy[27] (web site to share links to publications), CiteSeer[28] (search engine and digital library for scientific articles), CiteULike[29] (web site to share links to publications), The Collection of Computer Science Bibliographies[30] (computer science bibliographic database with more than two million entries), HubMed[31] (alternative interface to PubMed[32] that can export entries in BibTeX format), TeXMed[33] (another alternative interface to PubMed that can export entries in BibTeX).

[15] Acronym for "What You See Is What You Get".
[16] `http://office.microsoft.com/`
[17] `http://www.openoffice.org/`
[18] `http://www.ncbi.nlm.nih.gov/sites/entrez`
[19] `http://citeseer.ist.psu.edu/`
[20] DOI is the acronym for Digital Object Identifier and represents the future for references to electronic publications. Unlike an `url`, a DOI is associated with an object (scientific article) and not to the place where it is stored (web site). This guarantees a longer longevity of the link. More information can be found at `http://www.doi.org/`.
[21] `http://portal.acm.org/`
[22] `http://www.sciencemag.org/`
[23] `http://www.pnas.org/`
[24] `http://jcp.aip.org/`
[25] `http://keijisaito.info/lead2amazon/e/`
[26] `http://scholar.google.com/`
[27] `http://www.bibsonomy.org/`
[28] `http://citeseer.ist.psu.edu/`
[29] `http://www.citeulike.org/`
[30] `http://liinwww.ira.uka.de/bibliography/`
[31] `http://www.hubmed.org/`
[32] `http://pubmed.gov/`
[33] `http://www.sbg.bio.ic.ac.uk/~mueller/TeXMed/`

2.3 Manual composition: `thebibliography`

The `thebibliography` environment must be placed in the source document where the author wants the list of references to appear, typically just before `\end{document}`.

```
\documentclass{...}
\begin{document}
...
\begin{thebibliography}{argument}
...
\end{thebibliography}
\end{document}
```

The argument of `thebibliography` defines the maximum length of the labels. If the labels are automatically generated by LaTeX, usually the argument is chosen as "9", if the list contains less than ten entries, "99", if less than one hundred, etc. When using customized labels (e.g.: [Mori 06]) the argument must be the widest label.

The `thebibliography` environment works in a very similar way to the `itemize` environment. Each entry of the list begins with the `\bibitem` command and its argument, that works as reference identifier (similarly to `\label`), followed by the information about the entry (e.g. author, title, editor, year of publication), with explicit formatting and punctuation. The following example

```
\begin{thebibliography}{9}
\bibitem{Kopka1995} \textsc{Kopka}, H. and \textsc{
    Daly}, P.~W. (1995). \emph{A Guide to {\LaTeX
    }---Document Preparation for Beginners and
    Advanced Users}. Addison-Wesley.
\end{thebibliography}
```

produces Fig. 2.

```
[1] Kopka, H. and Daly, P. W. (2003). A Guide to LaTeX—Document Preparation for Beginners and Advanced Users, Fourth Edition. Addison-Wesley.
```

Figure 2: List of references obtained with a manual `thebibliography` environment.

2.4 From BibTeX to thebibliography

Some journals require an explicit `thebibliography`. Even in these cases it is possible to manage the bibliography with an external `.bib` database and BibTeX. As a matter of fact, BibTeX just extracts the useful information from the `.bib` database, formats it according to the `.bst` style, and write the output with the `thebibliography` syntax into the `.bbl` file. Hence, at the end of the work, it is possible to copy the content of the `.bbl` file into the `.tex` file.

2.5 From thebibliography to BibTeX

There is no automatic method to convert the content of a `thebibliography` environment into BibTeX format. Often it may be convenient to import entries from online databases. Otherwise cb2Bib can be used to try to import the content of a `thebibliography` environment. For both methods refer to section 2.2.4.

2.6 What method to use

BibTeX makes the bibliography management automatic but also has some disadvantages:

- it increases the complexity of the TeX environment (adding one or more external programs);
- although it is flexible, BibTeX does not allow completely free composition of bibliography entries.

However, the advantages are greater than these disadvantages:

- it automates tedious operations, especially ordering the bibliography entries;
- it automates bibliography formatting;
- several programs are available for creating and managing BibTeX databases.

2.7 Specialties

2.7.1 Renaming the references section

The name of the bibliography section depends on the class and language selected. The report and book classes use the variable `\bibname` to specify the name; the babel package defines this variable as "Bibliography" in English, "Bibliografia" in Italian, and so on. The article class uses the variable `\refname`, which becomes "References" in English, "Riferimenti bibliografici" in Italian, and so on.

The `\renewcommand` command can be used to change the names that are assigned by default to the bibliography, as in the following example:

```
\renewcommand\bibname{Useful references}
```

2.7.2 Multiple bibliographies

The chapterbib[34] package can be used to create separate bibliographies for each file added to the main document with the `\include` command, and not only for each chapter as suggested by its name. The package is very easy to use since the different bibliographies are created automatically by the `\include` commands and do not require special commands for the references in the text.

The bibunits[35] package can be used to create separate bibliographies for different parts of the document such as chapters, sections, etc. In addition, a global bibliography can be added at the end of the document. This package does not require *ad hoc* commands for the citations in the text, only the `\bibliographyunit{unit}` command in the preamble (where `unit` corresponds to the document structure to be used for the bibliographies, such as `chapter`, `section`, etc.).

The multibib[36] package can be used to create multiple bibliographies; unlike the bibunits and chapterbib packages, the bibliographies can be placed anywhere in the document, not only at the end of certain parts. Each bibliography can have a different `.bst` style and `.bib` database. This package, however, requires special commands for the citations in the text: for each bibliography a different type of `cite` command has to be used.

The multibbl[37] package works similarly to multibib but does not allow the creation of a general bibliography at the end of the document.

The bibtopic and splitbib give results similar to multibib and multibbl but follow a different strategy: instead of using special commands for the citations in the text, they require a distinction between the entries of each bibliography. The bibtopic[38] package requires that each bibliography has a different

[34] The chapterbib package was written by Donald Arseneau and the last version was released in 2008.

[35] The bibunits package was written by Thorsten Hansen and the last version was released in 2004.

[36] The multibib package was written by Thorsten Hansen and the last version was released in 2004.

[37] The multibbl package was written by Apostolos Syropoulos and the last version was released in 2004.

[38] The bibtopic package was written by Pierre Basso and Stefan Ulrich and the last version was released in 2006.

`.bib` database. This is particularly convenient when working with database management software such as JabRef that makes very easy to create several small databases from a global one. The splitbib[39] package, on the contrary, requires that the different categories and the respective entries are defined in the preamble.

2.7.3 Multilingual bibliographies

The babelbib[40] package, together with babel, can be used to create multilingual bibliographies in which:

- each entry is in a specific language, or
- all the entries are in the same language.[41]

The second case (all the entries are in the same foreign language) can also be handled by creating a style `.bst` with `makebst`, as described in section 3.2.2. For the first case (entries in different languages), babelbib is very convenient: the `language` field can be used to specify the language for each entry as in the following example:

```
@BOOK{Lucchesi1989,
  title = {La cucina di Lucchesia e Versilia},
  publisher = {Franco Muzzio Editore},
  year = {1989},
  author = {E. Lucchesi},
  language = {italian},
}
```

3 Bibliography styles

'Bibliographic style' can have two meanings:

- the style of the bibliographic entries (usually at the end of the document),
- the style of the references in the text.

The three main styles for the references in the text (numbered, author-year, footnote) are discussed in section 3.1. Even though the style for the references in the text influences the style of the bibliography entries, LaTeX separates these two aspects: the style of the bibliography can be controlled with the `.bst` file, as discussed in section 3.2.

3.1 References in the text

There are three main style families for the references in the text: numbered, author-year, and footnote. Each discipline has its own standards that depend on how the bibliography is used (Garcia, 2007).

[39] The splitbib package was written by Nicolas Markey and the last version was released in 2007.

[40] The babelbib package was written by Harald Harders and the last version was released in 2006.

[41] At present the package only supports Afrikaans, Danish, Dutch, English, Esperanto, Finnish, French, German, Italian, Norwegian, Portuguese, Spanish, and Swedish.

3.1.1 Numbered

Numbered references usually appear in square brackets and use arabic numerals (e.g.: [1]). The main advantage of this type of references is that they can be used both for direct references (e.g.: 'see [1] as a reference for the theory') and for indirect ones (e.g.: 'this has already been shown [1]'). Another advantage is that these references can appear close to parentheses in sentences such as '(further details can be found in [1].)'. In general, this form of the reference in the text, as output by `\cite` in LaTeX, is independent from the type of sentence in which it appears and this has made this style quite popular.

3.1.2 Author-year

When an article is cited for referring to a theorem or a theory, it is not necessary that the reader knows who wrote it and when. The interested reader can get this information from the reference list at the end of the document. For this reason, the numbered reference style is the most popular for sciences. In the humanities, however, referring to one author rather than another or to one historic period rather than another has significance in and of itself, and it is important that the reader gets this information directly from the text. For this reason, in the humanities the so-called author-year style is preferred. This style summarizes all the relevant information in the reference; usually it reports the first author's surname and the year of publication, e.g. (Mori et al., 2006).

Since parentheses have already a meaning, this style may lead to misunderstandings. For example it is possible to say '... this has already be proven (Mori, 2006)', but not '(Mori, 2006) proved that ...'. For these reasons, the author-year styles provide many variants for the references to solve grammatical or aesthetic problems. Some examples are:

- 'This has already been proven (Mori, 2006).'
- 'Further details can be found in Mori (2006).'
- '(see [Mori, 2006])'

The author can choose among these variants by using different commands instead of the usual `\cite`.

Packages The most popular packages for author-year citations are harvard, achicago, and natbib. They offer more or less the same features, natbib being the most versatile, but their commands follow different logics. The harvard[42] commands are based on the logical function of the reference in the sentence. For

[42] The harvard package was written by Peter Williams and Thorsten Schnier and the last version was released in 1996.

example \citenoun has to be used when the reference is a noun. With achicago[43] the names depend on the form of the reference. For example \citeA can be used to have references that contain only the author name ("A" stands for "author"). natbib[44] is based on the same logic of harvard and is the most flexible package for managing author-year citations. It will be discussed in section 4.

3.1.3 Footnotes

Some disciplines, mostly in the humanities, use footnote references. This style is especially common in journals that do not have a reference list at the end of each article.

Packages. The footbib[45] package defines the command \footcite which formats all references as superscript numbers in square brackets (e.g. [1]). The information of each entry is at the bottom of the page and their number does not follow that of the footnotes. The \footcite command does not interfere with \cite, it is possible to add a list of references at the end of the document.

The opcit[46] package creates entries that are true footnotes and follow their number (e.g. [1]). The package considers which references have already been cited in order to avoid repetitions by automatically using conventional forms such as "Idem" and "op. cit.". Besides the manual, the interested reader should also read Garcia (2007).

The jurabib[47] package, originally written for German law documents, offers many tools to manage footnote references. Similarly to opcit, it formats the bibliographic references as regular footnotes.

The natbib package provides the super option that, similarly to footbib, creates references that do not follow the number of footnotes and are not enclosed in brackets (e.g. [1]). The list of the references appears at the end of the document and not at the bottom of the pages. Although natbib offers fewer options for footnote references than the other packages, it uses the same syntax for footnote, author-year, and numbered styles. This allows switching from one style to another by merely changing the package options in the preamble, without changing the references in the text of the document.

The inlinebib[48] package can be used for footnote references but is not recommended since it is rather old, does not offer many options, and can be used only with the indexing.bst style.

3.1.4 Hybrid approaches

Between author-year and numbered The style alpha.bst is halfway between the author-year style, such as '(Mori, 2006)', and the numbered style, such as '[1]'. It produces references such as '[Mor06]'. Since this style is not as concise as the numbered and not as informative as the author-year, it is not recommended. It is supported by natbib.

Between author-year and footnote Some humanities journals report the bibliographic entries directly in the text. The bibentry[49] package can be used for this purpose. It requires only a few modifications to the .bst style, hence, it can be used almost with every style.

The inlinebib package, created for footnote references, can also be used for in-text-references although it is not recommended, as noted above.

The jurabib package, also created for footnote references, can be used for in-text-references with a small number of styles (jurabib.bst, jhuman.bst, and two styles of the 'Chicago' family).

3.2 Style of the reference list

3.2.1 Existing styles

Almost every journal and publisher have their own rules for formatting the bibliography (use of boldface or italic for the issue or volume number, use of parentheses and of punctuation, etc.). Many journals provide the BibTeX style, hence TeX distributions usually come with a lot of bibliographic styles; the Comprehensive TeX Archive Network (CTAN)[50] provides even more styles.

BibTeX comes with four styles (plain, unsrt, abbrv and alpha) that were created by the author of the program, Oren Patashnik. These styles, however, do not support the author-year approach (section 3.1.2).

Ken Turner's web site[51] provides examples of the most popular .bst styles. Another excellent review of the available styles is available on the Reed College web site.[52]

[43] The achicago package was written by Matt Swift and the last version was released in 2001.

[44] The natbib package was written by Patrick Daly and the last version was released in 2009.

[45] The footbib package was written by Eric Domenjoud and the last version was released in 2004.

[46] The opcit package was written by Federico Garcia and the last version was released in 2007.

[47] The jurabib package was written by Jens Berger and the last version was released in 2004.

[48] The inlinebib package was written by René Seindal and the last version was released in 1995.

[49] The bibentry package was written by Patrick Daly, also the author of natbib, and the last version was released in 2000.

[50] http://www.ctan.org/

[51] http://www.cs.stir.ac.uk/~kjt/software/latex/showbst.html

[52] http://web.reed.edu/cis/help/LaTeX/bibtexstyles.html

The best way to test a `.bst` style is by using the `xampl.bib` database that comes with the BibTeX documentation. If for example we want to test the `example.bst` style, we can use the following:

```
\documentclass{article}
\begin{document}
\bibliographystyle{example}
\nocite{*}
\bibliography{xampl}
\end{document}
```

3.2.2 Customizing the style with makebst

There are two main reasons to create custom `.bst` styles:

1. none of the available `.bst` styles satisfy the author (or publisher),
2. the document is written in a language different from English.[53]

Writing a `.bst` style may prove to be difficult since BibTeX uses a rather non-intuitive programming language. Luckily, Patrick Daly, who is also the author of natbib and coauthor of the excellent book about LaTeX (Kopka and Daly, 2003), wrote a program called makebst that can be used to create interactively a customized `.bst` style for BibTeX (and fully compatible with natbib). The program is usually distributed as the custom-bib package.

The generic style `merlin.mbs` is the heart of the program: it contains alternative code for all aspects of a bibliographic style and is analyzed in detail in Daly (2007b). This file has to be compiled by the docstrip[54] program in order to produce the respective `.bst` style. Since the number of options is very high (about one hundred), the program provides a graphic interface by means of the `makebst.tex` file. The first step consists in compiling `makebst.tex` with either TeX or LaTeX: at this point the user has to answer interactively to the questions that appear on the screen. At the very beginning the user has to select an `.mbs` file, and, depending on the choice, a docstrip batch file is created. This batch file can be used to create a bibliographic style with the characteristics of the `.mbs` file: the options that the user can select interactively depend on the chosen `.mbs` file. `merlin.mbs` is a third generation bibliographic style that has replaced `genbst.mbs` (released in November 1993) and its multilingual version `babel.mbs`. Unlike its predecessors, in `merlin.mbs` all the words such as "editor" are represented by variables (in this case `bbl.editor`) that assume different values depending on the language in use (`bbl.editor` becomes "curatore" in Italian, "editor" in English, "Redakteur" in German, "redacteur" in French, etc.). `merlin.mbs` supports only the options `English` and `babel`; the definitions for all the other languages are provided by separated `.mbs` files (e.g. `italian.mbs`). The language must be chosen at very beginning: when asked "Enter the name of the MASTER file" select `merlin.mbs`, when asked "Name of language definition file" select the `.mbs` file corresponding to the desired language. If the `.mbs` file is not available for the desired language,[55] the user can select `babel` that, instead of substituting the variables with their translation, substitutes them with commands (in this case `\bbleditor{}`) whose definition must be written in the `babelbst.tex` file.

Except from the language, some of the customizations offered by `merlin.mbs` are:

- author-year or numbered citations;
- criteria for ordering the entries: citation order, year ordered and then by authors, reverse year ordered and then by authors, etc.;
- format for author names: full with surname last, initials and surname, surname and initials, etc.;
- the number of names to report before substituting them with "et al.";
- formatting for the author names;
- position of the date;
- format for volume, issue, and page number;
- punctuation.

At the end, the procedure creates a `.dbj` file that has to be compiled with LaTeX in order to obtain the corresponding `.bst` style. If you want to modify a style created with this procedure, it is very convenient to open the `.dbj` file and modify it, instead of answering again to the interactive questions. Further details on makebst can be found in Daly (2007a,b).

4 The natbib package

The natbib package is highly recommended for bibliography customization. The most common options will be described in the following paragraphs and the other details can be found in Daly (2009).

4.1 Compatible styles

natbib only works with styles that support its options; the three that come with the package (`plainnat.bst`, `abbrvnat.bst`, and `unsrtnat.bst`) can replace the corresponding BibTeX standard styles (`plain.bst`, `abbrv.bst`, and `unsrt.bst`) with the advantage that

[53] Almost all the available styles are in English.

[54] docstrip, written by Frank Mittelbach, is a standard part of LaTeX distributions.

[55] At the moment `.mbs` files are available for Catalan, Danish, Dutch, Esperanto, Finnish, French, German, Italian, Norwegian, Polish, Portuguese, Slovene, and Spanish.

Table 1: List of the commands for references in the text and their effect with an author-year style (option authoryear).

Reference in the text		
`\citet{mori06}`	⇒	Mori et al. (2006)
`\cite{mori06}`	⇒	Mori et al. (2006)
`\citet[chap.~2]{mori06}`	⇒	Mori et al. (2006, chap. 2)
References with parentheses		
`\citep{mori06}`	⇒	(Mori et al., 2006)
`\citep[chap.~2]{mori06}`	⇒	(Mori et al., 2006, chap. 2)
`\citep[see][]{mori06}`	⇒	(see Mori et al., 2006)
`\citep[see][chap.~2]{mori06}`	⇒	(see Mori et al., 2006, chap. 2)
References with complete author list		
`\citet*{mori06}`	⇒	Mori, Lee, and Krishnan (2006)
`\citep*{mori06}`	⇒	(Mori, Lee, and Krishnan, 2006)
Multiple references		
`\citet{mori06,rossi07}`	⇒	Mori et al. (2006); Rossi et al. (2007)
`\citep{mori06,rossi07}`	⇒	(Mori et al., 2006; Rossi et al. 2007)
`\citep{mori06,mori07}`	⇒	(Mori et al., 2006, 2007)
`\citep{mori06a,mori06b}`	⇒	(Mori et al., 2006a,b)
References without parentheses		
`\citealt{mori06}`	⇒	Mori et al. 2006
`\citealt*{mori06}`	⇒	Mori, Lee e Krishnan 2006
`\citealp{mori06}`	⇒	Mori et al., 2006
`\citealp*{mori06}`	⇒	Mori, Lee e Krishnan, 2006
`\citealp{mori06,rossi07}`	⇒	Mori et al., 2006; Rossi et al., 2007
`\citealp[p.~32]{mori06}`	⇒	Mori et al., 2006, p. 32

Table 2: List of the commands for references in the text and their effect with a numbered style (option numbered).

References in the text		
`\citet{mori06}`	⇒	Mori et al. [11]
`\citet[chap.~2]{mori06}`	⇒	Mori et al. [11, chap. 2]
References with parentheses		
`\citep{mori06}`	⇒	[11]
`\cite{mori06}`	⇒	[11]
`\citep[chap.~2]{mori06}`	⇒	[11, chap. 2]
`\citep[see][]{mori06}`	⇒	[see 11]
`\citep[see][chap.~2]{mori06}`	⇒	[see 11, chap. 2]
Multiple references		
`\citep{mori06a,mori06b}`	⇒	[11, 18]

they can be used for both numbered (the only option available for the three original styles) and author-year references. Several other styles that support natbib are available on the Internet. This format is also supported by makebst (section 3.2.2).

4.2 Commands for in-text references

natbib provides two main commands that substitute the regular `\cite` for the citations: `\citet` for the citations in the text and `\citep` for the citations in parentheses. Both of them have a starred version (`\citet*` and `\citep*`) that produce the complete list of authors rather than the abbreviated one. All commands have two optional arguments to add text before and after the reference. Like `\cite`, these commands can be used for multiple citations. The package also provides commands that remove the parentheses from the citations: `\citealt` instead of `\citet` and `\citealp` instead of `\citep`. Examples of these commands are shown in Tab. 1 for the author-year style (option authoryear) and in Tab. 2 for the numbered style (option number).

The standard `\cite` command can still be used with natbib and it is interpreted as `\citet` for author-year bibliographies (option authoryear) and as `\citep` for numbered bibliographies (option number).

4.3 Package options

4.3.1 Type of parentheses

References can be enclosed in different types of brackets by using the following options:

- round (default): round brackets as in "see Mori (2006)" or "see (2)";
- square: square brackets as in "see Mori [2006]" or "see [2]";
- curly: curly brackets as in "see Mori {2006}" or "see {2}".

The `\bibpunct` command can also be used to select the bracket type (Daly, 2007b).

4.3.2 Punctuation

The punctuation used to separate multiple references can be selected with the following options:

- colon (default): the colon as in "(Mori et al., 2006; Rossi et al., 2007)" or "Mori et al. (2006); Rossi et al. (2007)";
- comma: the comma as in "(Mori et al., 2006, Rossi et al., 2007)" or "Mori et al. (2006), Rossi et al. (2007)";

`\bibpunct` can also modify the punctuation (Daly, 2007b).

4.3.3 Bibliography style

The bibliographic style can be set by invoking the respective option when loading natbib: authoryear (default) loads the author-year style, numbers the numbered style, and super the footnote style. The selected `.bst` style must support the reference type chosen.

4.3.4 Ordering and compressing multiple references

Multiple references such as `\cite{a,b,c,d}` by default produce bad results such as "[2,6,4,3]". Ordering the references by hand (i.e. `\cite{b,c,d,a}`) would give "[2,3,4,6]" but such manual operations are undesirable. natbib, when used with the numbers option, provides the sort&compress option that automatically orders and compresses multiple references. For example `\cite{a,b,c,d}` would produce "[2–4,6]".

4.3.5 References from the bibliography to the text

In the bibliography of a long document, it can be useful to provide the page on which each reference appears. Both the backref and the citeref packages can be used for this purpose, although the former is more modern and can work together with hyperref; namely, it can create hyperlinks when used together with hypernat. Neither of the two packages can compress the page list ("5, 6, 7" cannot be automatically converted into "5–7") but they do not repeat a page number if the same reference appears several times in it.

4.3.6 Reducing the space between the references

The bibliography is composed as a list (very similar to `itemize`, `enumerate`, and `description`) and the space between the items can be controlled with the `\itemsep` parameter (UK TUG, 2009):

```
\let\oldbibliography\thebibliography
\renewcommand{\thebibliography}[1]{%
    \oldbibliography{#1}%
    \setlength{\itemsep}{0pt}%
    }
```

The natbib package offers an even better solution by providing the `\bibsep` parameter that can be used as in the following example:

```
\setlength{\bibsep}{0pt}
```

4.3.7 Style of the numbers in the bibliography

By default, LaTeX formats the entry numbers in the following way:

> [1] Garcia, F. (2007). LaTeX and the different bibliography styles. *TUGboat*, 28(2).
>
> [2] Goossens, M., Mittelbach, F. and Samarin, A. (1995). *The LaTeX Companion*. Addison-Wesley.

This behavior can be customized with commands such as (UK TUG, 2009):

```
\makeatletter
\renewcommand*{\@biblabel}[1]{\hfill#1.}
\makeatother
```

or, if using natbib,

```
\renewcommand{\bibnumfmt}[1]{#1.}
```

Both of them give:

> 1. Garcia, F. (2007). LaTeX and the different bibliography styles. *TUGboat*, 28(2).
> 2. Goossens, M., Mittelbach, F. e Samarin, A. (1995). *The LaTeX Companion*. Addison-Wesley.

5 BibLaTeX

The BibLaTeX package offers a general solution for customizing bibliographic entries and references. Besides offering features similar to those of the packages analyzed so far, it can be used to modify a bibliographic style by using only LaTeX commands.

This package, written by Philipp Lehman, is still under development and, although (at the author's

request) it is not included in many LaTeX distributions, it is available on CTAN.[56] BibLaTeX requires ε-TeX, the etoolbox package, also under development, and the packages keyval, ifthen, and calc. In addition, babel and csquote, while not required, are recommended in order to use the full potential of BibLaTeX. A detailed analysis of BibLaTeX would require a separate article; the following sections only describe its main characteristics. Further details can be found in the package documentation (Lehmann, 2009).

5.1 BibLaTeX styles

The main limitation of BibTeX is that, in order to have full control of the bibliographic style, the user has to learn a new language that is completely different from that of (La)TeX. The custom-bib package, as explained in section 3.2.2, is very helpful but is not always enough to obtain the desired result.

With BibLaTeX the bibliographic references and citations can be fully controlled with LaTeX commands. The style is contained, not in a `.bst` file, but in a `.bbx` (bibliographic style) or `.cbx` (citation style) file. The `.bbl` file that is created when compiling does not contain a `thebibliography` environment, but rather a series of macros that contain the bibliographic data.

The style can be specified by declaring a package option:

```
\usepackage[style=numeric]{biblatex}
```

or:

```
\usepackage[bibstyle=authortitle,%
  citestyle=verbose-trad1]{biblatex}
```

In the first case the value numeric is assigned both to bibstyle and to citestyle.

The package comes with some styles. The bibliographic styles cover the four traditional categories: numeric, alphabetic, authoryear, and authortitle. A citation style can be associated with each of them to control the references in text (numbered, author-year, footnote). The verbose style uses the complete citation the first time and an abbreviated form, such as *idem*, *ibidem*, *op. cit*, and *loc. cit* subsequently.

The user can modify the standard styles inside the document or create new ones. For example, to make the article titles in italic and the journal name enclosed in quotation marks, the following commands can be given in the preamble:

```
\DeclareFieldFormat[article]{title}%
  {\mkbibemph{#1\isdot}}
\DeclareFieldFormat{journaltitle}%
  {\mkbibquote{#1}}
```

To modify the author-year style in order to use it again with other documents, a user can write the following in the `myauthoryear.bbx` file:

```
\RequireBibliographyStyle{authoryear}
\DeclareFieldFormat[article]{title}%
  {\mkbibemph{#1\isdot}}
\DeclareFieldFormat{journaltitle}%
  {\mkbibquote{#1}}
\endinput
```

and specify myauthoryear as the value for bibstyle.

5.2 BibLaTeX commands for references

BibLaTeX, besides the standard `\cite` and `\nocite` commands, provides citation commands for different contexts: `\parencite` encloses the reference in round brackets; `\footcite` inserts the reference in a footnote; `\textcite` for when the citation is part of a sentence; `\supercite` (only for numbered styles) for superscript citations, and `\fullcite` insert the full bibliographic entry. Finally, `\autocite` automatically invokes the suitable command of these depending on the context. Examples of these commands are reported in Tab. 3 and 4. Commands to cite parts of the entry are also available: `\citeauthor`, `\citetitle`, `\citeyear`, and `\citeurl`.

5.3 BibLaTeX commands for bibliographies

The bibliographic list of references in BibLaTeX is inserted differently than with BibTeX. As discussed earlier, the bibliography style is specified as an option to the package instead of with `\bibliographystyle`. The `\bibliography` command only specifies which databases to use, and does not create any list. The entry list can be generated with `\printbibliography`, which accepts an optional argument. This argument can be used to filter the entries:

- by special fields (`type` or `keyword`),
- via the definition of categories in the preamble (using `\DeclareBibliographyCategory`), then assigning each entry to one of these categories (using `\addtocategory`),
- depending on the position of the citation inside the document, by using the `refsection` or `refsegment` options.[57]

This allows easily dividing the bibliography by chapters or topics by using `\printbibliography` several times with different filters. When making multiple

[56] http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/

[57] The sections can also be defined manually enclosing portions of the document between `\begin{refsection}` and `\end{refsection}`.

Table 3: List of the BibLaTeX commands for the references in the text and their effect in the compact author-year style (option authoryear-comp).

References in the text		
`\textcite{mori06}`	⇒	Mori et al. (2006)
`\textcite[chap.~4]{mori06}`	⇒	Mori et al. (2006, chap. 4)
References with parentheses		
`\autocite{mori06}`	⇒	(Mori et al. 2006)
`\parencite{mori06}`	⇒	(Mori et al. 2006)
`\parencite[chap.~4]{mori06}`	⇒	(Mori et al. 2006, chap. 4)
`\parencite[see][]{mori06}`	⇒	(see Mori et al. 2006)
`\parencite[see][chap.~4]{mori06}`	⇒	(see Mori et al. 2006, chap. 4)
Multiple references		
`\cite{mori06,rossi07}`	⇒	Mori et al. 2008; Rossi et al. 2007
`\textcite{mori06,rossi07}`	⇒	Mori et al. (2008); Rossi et al. (2007)
`\parencite{mori06,rossi07}`	⇒	(Mori et al. 2008; Rossi et al. 2007)
`\cite{mori06,mori08}`	⇒	Mori et al. 2006, 2008
`\cite{mori06a,mori06b}`	⇒	Mori et al. 2006a,b
References without parentheses		
`\cite{mori06}`	⇒	Mori et al. 2006
`\cite[chap.~4]{mori06}`	⇒	Mori et al. 2006, chap. 4

Table 4: List of the BibLaTeX commands for the references in the text and their effect in the numbered style (option numeric-comp).

References in the text		
`\textcite{mori06}`	⇒	Mori et al. [6]
`\textcite[chap.~4]{mori06}`	⇒	Mori et al. [6, chap. 4]
References with parentheses		
`\cite{mori06}`	⇒	[6]
`\parencite{mori06}`	⇒	[6]
`\autocite{mori06}`	⇒	[6]
`\cite[chap.~4]{mori06}`	⇒	[6, chap. 4]
`\cite[see][]{mori06}`	⇒	see [6]
`\cite[see][chap.~4]{mori06}`	⇒	see [6, chap. 4]
Multiple references		
`\cite{mori06,mori08}`	⇒	[3, 6]

bibliographies, a complete entry list can still be generated with the `\bibbysection`, `\bibbysegment` or `\bibbycategory` commands.

5.4 Commands for indexing

Automatic indexing of bibliographic entries is a very interesting feature of BibLaTeX. An index and an author index are very useful for any kind of book and thesis. BibLaTeX can automatically add the authors cited in the text into an author index (or another type of index) by using the indexing option. BibLaTeX uses external packages to create indexes: makeidx for basics operations and index for advanced features such as multiple indexes.

This section will show an example of automatic indexing; further details can be found in Lehmann (2009, in particular section 3.1.2 and the templates that come with the package documentation).

Assuming that our bibliographic database is called `database.bib`, the following code can be used to create an index with BibLaTeX:

```
\documentclass{...}

\usepackage[indexing]{biblatex}
\bibliography{database}

\usepackage{makeidx}
\makeindex

\begin{document}

As reported by \textcite{Kopka1995}...
\clearpage

As discussed by \citeauthor{Goossens1995} in...
\clearpage

\nocite{*}
\printbibliography
\printindex
\end{document}
```

If our document is called `document.tex`, we need to compile it in this order:

```
latex document
bibtex document
latex document
makeindex document
latex document
```

This produces a document with a bibliography followed by an index with the name of all the cited authors.

5.5 Multilingual bibliographies

BibLaTeX also includes some of babelbib's features. When the babel option is used with one of the values hyphen or other, the package checks the `hyphenation` field for each bibliography entry. If a language is specified through this field, BibLaTeX uses the appropriate hyphenation rules and the translation for words such as "editor", "volume", etc. The translations used are stored in `.lbx` files that come with the package.

6 Acknowledgments

I would like to thank Massimiliano Dominici for writing section 5 about BibLaTeX and Gustavo Cevolani for writing section 5.4 about indexes with BibLaTeX. I also would like to thank Valeria Angeli, Claudio Beccari, Caterina Mori, and Gianluca Pignalberi for their suggestions during both the writing and the reviewing process of this article.

References

Daly, P.W. "Customizing Bibliographic Style Files", 2007a. `http://mirror.ctan.org/macros/latex/contrib/custom-bib/makebst.pdf`.

Daly, P.W. "A Master Bibliographic Style File for numerical, author-year, multilingual applications", 2007b. `http://mirror.ctan.org/macros/latex/contrib/custom-bib/merlin.pdf`.

Daly, P.W. "Natural Sciences Citations and References (Author-Year and Numerical Schemes)", 2009. `http://mirror.ctan.org/macros/latex/contrib/natbib`.

Garcia, Federico. "LaTeX and the different bibliography styles". *TUGboat* **28**(2), 2007. `http://tug.org/TUGboat/Articles/tb28-2/tb89garcia.pdf`.

Kopka, H., and P. Daly. *A Guide to LaTeX — Document Preparation for Beginners and Advanced Users*. Addison-Wesley, fourth edition, 2003.

Lehmann, Philipp. "The biblatex package". 2009. `http://mirror.ctan.org/macros/latex/exptl/biblatex/doc/biblatex.pdf`.

Patashnik, Oren. "BibTeXing", 1998. `http://mirror.ctan.org/biblio/bibtex/contrib/doc/btxdoc.pdf`.

UK TUG. *The UK TUG FAQ*, 2009. `http://www.tex.ac.uk/faq`.

◇ Lapo F. Mori
Dipartimento di Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
Pisa, Italy
`lapo dot mori (at) ing dot unipi dot it`

Managing languages within MlBibTeX

Jean-Michel Hufflen

Abstract

We explain how the information about natural languages used throughout documents is managed in MlBibTeX, our multilingual reimplementation of BibTeX. That allows us to show how the interface between MlBibTeX and LaTeX or ConTeXt's tools for multilinguism — e.g., the babel package — is organised, by means of a powerful data structure. We also show how the generated texts for LaTeX are built. In fact, they take as much advantage as possible of the multilingual packages of LaTeX's recent versions.

Keywords MlBibTeX, multilingual features, multilingual LaTeX packages, ConTeXt, tries, multilingual method, Scheme.

1 Introduction

The bibliography of a printed document, that is, the list of its bibliographical *references*, can be prepared manually, in which case its items may not be directly reusable elsewhere. The layout of bibliographies is ruled by *styles* that are influenced by cultural background. As a consequence, it can vary from a document to another: for example, the bibliography of some documents[1] use *plain* styles where items are labelled with numbers, some use *alpha* styles based on keys built from authors' last names and publication's years, e.g., '[Robeson 1965]' or '[Rob65]' — see [34, § 13.5.1] for a survey of available styles and corresponding layouts. In addition, some information may depend on the printed document's language: let us consider the date of a publication; a publisher may require that month names are printed in English for the bibliography of a document written in English, in French (resp. German, ...) for a document written in French (resp. German, ...). So managing bibliographical references already typeset for a particular document is tedious, and it is better for such references to be automatically generated from a database containing bibliographical *entries*.[2] In particular, this allows us to put as much information as we want within entries, even if some parts of information do not appear within generated texts.

As an accurate example, the bibliography program BibTeX [36] is often used to build 'References' sections for documents suitable for LaTeX [34, § 12.1.3]. BibTeX searches bibliography (`.bib`) files for keys cited throughout a document: to do that, it uses information put in *auxiliary* (`.aux`) files produced by LaTeX [34, Fig. 12.1]. BibTeX's bibliography styles[3] are programmed using a stack-based language [34, § 13.6]. By means of such a bibliography program, we should be able to fill in all the fields of a bibliographical entry once, and derive as many references as we want, according to layouts expressed by bibliography styles. This is true in most cases, but not always, depending on the expressive power of bibliography styles. For example, let us consider *annotated* bibliographies: the annotations should be expressed in the document's language. If we wish to avoid the duplication of bibliographical entries according to the language of an added annotation, such annotations can be given different field names:

```
english-ANNOTE = ..., french-ANNOTE = ..., ...
```

but in this case, we have to generate several bibliography styles differing only by the name of the chosen annotation. This example shows that BibTeX was not ideally designed for multilingual applications. There have been some attempts to insert multilingual features into texts generated by BibTeX — e.g., in the jurabib package [34, pp. 733–735] and the `custom-bib` tool (usable by applying LaTeX to the `makebst.tex` program) [34, § 13.5.2] — but BibTeX itself does not take enough advantage of multilingual features of LaTeX's recent versions. In addition, we think that the language BibTeX uses for bibliography styles leads to non-modular programs, which are monolithic and hard to maintain, as we explained in [17].

MlBibTeX aims to ease the development of multilingual bibliographies, without giving any privilege to a particular language, as the babel package does for documents written with LaTeX's modern versions [34, Ch. 9]. MlBibTeX's current version (1.3), described in [18] and developed in Scheme [25], is usable to generate bibliographies for LaTeX documents. This bibliography processor also opens a window towards the world of XML,[4] which has become a central formalism for document interchange. Since parsing a `.bib` file results in a tree that can be viewed as an XML tree, this choice more easily allows us to build other output files than `thebibliography` environments for LaTeX [34, § 12.1.2]. In particular, we can generate (X)HTML[5] pages for bibliographies to be

[1] This article, for example.

[2] Within MlBibTeX ('**Multi**ling**u**al BibTeX'), we use precise terminology: bibliographical *entries* are specified in bibliography (`.bib`) files, and bibliographical *references* — in `.bbl` files for use with LaTeX — are what a word processor typesets.

[3] A representative selection of bibliography styles usable with BibTeX is given in [34, Table 13.4].

[4] E**X**tensible **M**arkup **L**anguage. Readers interested in an introduction to this metalanguage can refer to [39].

[5] (E**X**tensible) **H**yper**T**ext **M**arkup **L**anguage. XHTML is a reformulation of HTML using XML conventions. [35] is a good introduction to these languages.

```
@BOOK{robeson1965,
        AUTHOR = {first => Kenneth, last => Robeson},
        TITLE = {The Polar Treasure},
        PUBLISHER = {Bantam},
        SERIES = {Doc Savage},
        NUMBER = 4,
        NOTE = {[Titre de la traduction française : ‘‘Le trésor polaire’’] ! french
                [Titel der deutschen \"{U}bersetzung: ‘‘Das Wrack im Eis’’] ! german
                [T\’{i}tulo de la traducci\’{o}n al Espa\~{n}ol: ‘‘El tesoro del Polo’’] ! spanish}
        YEAR = 1965,
        MONTH = apr,
        LANGUAGE = english}
```

Figure 1: Example of an entry using MlBibTeX's features.

displayed on the *Web*, or XML files written according to the rules of XSL-FO[6] [43]. In fact, bibliography styles are now programmed using a new language, called nbst,[7] close to XSLT[8] [41], the language of transformations for XML documents.

We have already written some documents about MlBibTeX's implementation. In [19], we explain why we have started a new implementation using Scheme [25], after a first project in C [26]. We have also begun to describe the broad outlines of this implementation using Scheme in [22]. Here[9] we explain how the information about the natural languages used throughout bibliographies — and LaTeX documents — is organised. In the next section, we show the drawbacks of deferring the generation of multilingual bibliographies to LaTeX. Then Section 3 exposes the notion of *language identifiers*, introduced in MlBibTeX. Section 4 explains how our data structure for handling language identifiers is built and how it allows us to generate multilingual bibliographies. We do not describe this data structure in Scheme directly, but using an abstract way, so that we can see that it could be implemented in any programming language.[10] Finally, we show that this data structure should be able to evolve for MlBibTeX's future versions.

We assume that readers are familiar with the multilingual babel package of LaTeX 2ε, developed by Johannes Braams and described in [34, Ch. 9]. We also assume that readers can understand some simple macros, expressed using TeX's language [28, Ch. 20]. About BibTeX, XML, and Scheme, basic knowledge is sufficient to read this article, as well as basic notions about the use of trees in programming. Some notions related to specialised structures for searching strings are recalled in footnotes.

2 Difficulty related to languages

2.1 Accents and other diacritical signs

Let us consider the `robeson1965` entry given in Figure 1. It looks like a BibTeX entry, but some syntactic features indisputably belonging to MlBibTeX can be noticed: more user-friendly syntax for person names (`AUTHOR` and `EDITOR` fields), the use of multilingual switches (‘`[...] ! ...`’) within the value of the `NOTE` field. These notations are detailed in [18].

As mentioned in the introduction, such an entry is viewed as an XML tree in the sense that we can address its parts by using the XPath language [42]. As an example, Figure 2 gives the representation of the value associated with the `NOTE` field.[11] We can remark that quotations are uniformly expressed by using the American quotation marks (‘‘`...`’’) within a `.bib` file (see Figure 1), but each quotation is transformed into an XML element — an occurrence of the `emph` element with accurate attributes[12] — so putting quotation marks belonging to other languages is eased: ‘ « ... » ’ in French, ‘„...“ ’ in German, etc. More exactly, bibliography stylesheets are in charge of this. Likewise, we can remark that some accented letters can be typed directly by end-users (see the group expressed in the French language in Figure 1) or by using TeX commands (see the group written

[6] E**X**tensible **S**tylesheet **L**anguage — **F**ormatting **O**bjects: this language aims to describe high-quality print outputs. An introductory reference to XSL-FO suitable for LaTeX users is [24], a more general one is [37].

[7] **N**ew **B**ibliography **ST**yles.

[8] E**X**tensible **S**tylesheet **L**anguage **T**ransformations.

[9] The present article is a renewed and updated version of previous material. A first version was initially designed for the PracTeX 2005 conference. Later, this presentation was given at a conference of the German-speaking TeX user group — at Berlin, in 2006 — and was entitled *Sprachen in MlBibTeX*.

[10] Besides, the implementation used within our preliminary project in C [19] was quite close to the current one.

[11] In fact, MlBibTeX internally uses the conventions of SXML (Scheme implementation of XML) [27]. See [22] for more details.

[12] Readers interested in a description of elements and attributes used throughout the XML versions of `.bib` files can refer to [16]: that is an earlier version, but changes are slight.

```
<note>
  <group language="french">
    Titre de la traduction française :
    <emph emf="no" quotedbf="yes">
      Le trésor polaire
    </emph>
  </group>
  <group language="german">
    Titel der deutschen Übersetzung:
    <emph emf="no" quotedbf="yes">
      Das Wrack im Eis
    </emph>
  </group>
  <group language="spanish">
    Título de la traducción al Español:
    <emph emf="no" quotedbf="yes">
      El tesoro del Polo
    </emph>
  </group>
</note>
```

Figure 2: Multilingual note as an XML tree.

in Spanish), but as shown in Figure 2, the characters resulting from these commands are directly put within text nodes.[13]

Letters with accents and other diacritical signs illustrate some deficiency of the information put into `.aux` files. As a consequence, we have to parse the preamble of `.tex` source files[14] to get this information. When we generate texts, we cannot know if we can insert such letters directly, or if we have to write TEX commands to produce them. This last solution works in any case, provided that such commands belong to LATEX's basic set. That is true for commands that produce most accents (acute, grave, circumflex, ...) but there is a simple counter-example: the French *guillemets*.

> Concerto comique n° 25 en sol mineur « Les Sauvages et la Furstenberg ».

Such a French title may be mentioned within a book written in any language. In other words, we may have to write French guillemets even within a document written in English. The LATEX commands to produce them depend on the package used to write French fragments: either '`\og`' and '`\fg`' for the `frenchb` option[15] [5] of the babel package, or '`\guillemets`' and '`\endguillemets`' for the frenchpro package[16] [11]. There are other commands to get these guillemets, but they depend on other packages: if the fontenc package [34, § 7.5.3] has been loaded with the `OT1` option,[17] the commands `\guillemotleft` and `\guillemotright` produce opening and closing guillemets as single characters. If LATEX uses its default encoding (`OT1`), these commands are not provided and using them causes errors. However, getting French guillemets by combination of several characters is possible as shown in [21, Fig. 3]. Besides, a more immediate way exists if your keyboard allows you to type these guillemets, in which case using the inputenc package [34, § 7.5.2] with the `latin1` option allows LATEX to process them directly. If these characters are unavailable on your keyboard, you can use the sequences '`<<`' and '`>>`' with the `frenchb` option [5] of the babel package.[18]

These details may seem to be anecdotal, nevertheless they show how difficult the automatic generation of such texts is, especially if we wish to generate 'nice' texts, that is, readable by human agents. An end-user can solve this problem by typing LATEX commands directly within `.bib` files, but as a consequence, such bibliography files become difficult to be shared among several users, unless they make sure that the same packages with compatible options will be loaded when texts are processed. In addition, some files can be unusable for building other output files than those suitable for LATEX:[19] in particular, this point can obstruct the generation of bibliographies suitable for ConTEXt, another format built on TEX, created by Hans Hagen [14].

2.2 Using advanced LATEX commands

Let us consider again the value of the `NOTE` field within the `robeson1965` entry of Figure 1. This multilingual text could be transformed for use with LATEX by means of the `\iflanguage` command of the babel package [34, § 9.2.1] as follows:

```
NOTE ↦ \iflanguage{frenchb}{...}{%
         \iflanguage{germanb}{...}{%
          \iflanguage{spanish}{...}{}}}
```

[13] Although this behaviour only holds about the Latin 1 encoding (ISO-8859-1) presently. Future versions will probably extend it to the other encodings summarised in [13, Table C.4].

[14] That is, the commands located before the document itself, introduced by '`\begin{document}`'.

[15] This option has two aliases: `french` and `francais`. We will come back to this point later.

[16] This is a successor of the french package described in [7]. For reasons explained in [8, 9, 10], it has been replaced by a *freeware* version frenchle — 'french *al**LÉ**gé*' (for 'lightened') — [12] and a *shareware* version frenchpro — 'french **PRO**fessional' — [11]. The development of the freeware version seems to have stopped since B. Gaulle's death, in August 2007. Coming back to the French guillemets, the frenchle package provides only some compatibility with the commands '`\og`' and '`\fg`' of the babel package's `frenchb` option [12, § 7].

[17] That is, if the Cork encoding is used, with a range of 256 characters.

[18] ... or the frenchpro package [11].

[19] ... although MlBibTEX's next version will probably be able to solve this problem: cf. [20].

provided that this entry is cited only by documents written with the babel package, loaded with at least the options `frenchb`, `germanb`,[20] and `spanish`. However, if a user writes only in French by means of the frenchle package, this source text is unusable.[21] In addition, let us consider that we are building a bibliography for a document whose main language is French. Therefore, the bibliographical reference for `robeson1965` is surrounded as follows:

```
\bibitem[...]{robeson1965}
\begin{otherlanguage*}{english}Robeson
  (Kenneth)...
\end{otherlanguage*}
```

[34, § 9.2.1]. Besides, MlBibTeX offers a choice [18] between two kinds of bibliography styles:

- a *language-dependent* style, that is, each bibliographical item is expressed only in the entry's language,
- a *document-dependent* style, that is, each bibliographical item is expressed in the document's language, as far as possible.

In the first case, nothing need be done because the `robeson1965` entry characterises a book in English. In the second case, the French version of the `NOTE` field should be put and the previous text of this note should be surrounded as follows:

```
NOTE ↦ \begin{otherlanguage*}{frenchb}%
       \iflanguage{frenchb}{...}{%
        \iflanguage{germanb}{...}{%
         \iflanguage{spanish}{...}{}}}%
       \end{otherlanguage*}
```

Even if such texts are generated automatically, we can see that they are quite complicated.

Now let us consider the entry given in Figure 3: it concerns the English translation of a French book, so most information is given in English, except for the author's name and the original title, given in French. We wish these French fragments to be hyphenated correctly if need be, but if there is no way to typeset French fragments, we accept them to be typeset according to the rules of the language in use at this point. So we can write a robust version of the `\foreignlanguage` command provided by the babel package [34, § 9.1.2], here called `\putwrtlanguage`:[22]

```
@ARTICLE{ayerdhal2001,
       AUTHOR = {[last => Ayerdhal] : french}
       TITLE = {Flickering},
       JOURNAL = {Interzone},
       NUMBER = 167,
       PAGES = {6--13},
       NOTE = {English translation of
              ``[Scintillements] : french'', by
              Sheryl Curtis},
       YEAR = 2001,
       MONTH = may,
       LANGUAGE = english}
```

Figure 3: English translation of a French writer's book.

```
\def\putwrtlanguage#1#2{%
 \expandafter%
 \ifx\csname l@#1\endcsname\relax%
  \typeout{Language #1 unusable.}#2\else%
  \ifnum\csname l@#1\endcsname=\language%
   #2\else%
   \foreignlanguage{#1}{#2}%
  \fi%
 \fi}
```

This command could be used to process the two French fragments of the bibliographical reference for `ayerdhal2001`:

```
 \putwrtlanguage{frenchb}{Ayerdhal}
 \putwrtlanguage{frenchb}{Scintillements}
```

So this command is used twice when this bibliographical reference is processed. That is, checking whether the `frenchb` language is known is performed twice, although the answer is always the same. Either the `\l@frenchb` command is available for the whole of the document, or it is not at all. However, this replication does not result in great loss of efficiency: we can imagine that TeX can check a command's existence quickly. But in this first version, we assumed that the multilingual tool used was the babel package. If we take LaTeX's other multilingual packages into account — frenchle, german [38], ngerman and polski [4, § F.7] — our command looks like:[23]

```
\def\putwrtlanguage#1#2{%
 \@ifpackageloaded{babel}{\expandafter...
      ...          % (As previously.)
  }{\@ifpackageloaded{frenchle}{%
     \ifthenelse{\equal{#1}{french}}{%
      \french#2}{\english#2}}{%
```

[20] Like the babel package's option for French (cf. footnote 15), '`german`' is an alias and the option's actual name is `germanb`.

[21] In fact, frenchle may be used as an option of babel [12, § 6.5]. However, frenchle has been developed to be loaded as a package, in which case babel's commands are unknown.

[22] Let us recall that the commands giving access to languages are defined by natural numbers, thus we can use `\ifnum` to compare them.

[23] The frenchle package is not wholly multilingual in the sense that it deals with the French language, and can revert to LaTeX's original configuration — by means of the `\english` command [12, § 6.5] — in which case texts are supposed to be in English, as we do in the second version of the `\putwrtlanguage` command.

```
\@ifpackageloaded{german}{...}{%
 \@ifpackageloaded{ngerman}{...}{%
  \@ifpackageloaded{polski}{...}{%
   #2}}}}}
```

The waterfall of tests makes the command slower, and as many times as it is called, the corresponding results will be retrieved more and more slowly.[24]

These two examples show that implementing multilingual bibliographies by means of LaTeX commands only results in complicated texts. In addition, these texts are suitable for LaTeX only. If we wish to derive bibliographies for another word processor — e.g., ConTeXt — we have to put the same basic algorithms into action, but with the library of another language. So it seems to be better for such algorithms to be put into action by the bibliography processor itself.

3 Language identifiers

If we consider the results of working groups related to XML, natural languages throughout bibliographical data bases should be specified using the two-letter language, optionally followed by a two-letter country code,[25] described in [1] and [13, § C.1]. This convention allows the general reference to a language as well as a more precise reference to a local variant of it. For example, '`en`' is for the English language in general, whereas '`en-UK`' (resp. '`en-US`') is for British (resp. American) English only. In particular, using this convention would simplify an interface with the ConTeXt format, which also uses these codes. For example, ConTeXt uses the statement '`\language[fr]`' to change the document's current language into French [14, Ch. 7]. When MlBibTeX's first version was designed [15], it aimed at being a 'better BibTeX', mainly usable in cooperation with LaTeX; we did not relate this to XML features. Besides, we knew that many users of BibTeX put LaTeX commands within values of BibTeX fields. For example, it seemed to be interesting to process differently the texts written in French by using the successors of the frenchle package and those using the `frenchb` option of the babel package. The compromise we have settled is:

- a *language identifier* of MlBibTeX is a non-ambiguous prefix of:
  - either an option of the babel package,
  - or a multilingual *ad hoc* package;

  the multilingual *ad hoc* packages we recognise are frenchle, german, ngerman, and polski;
- by 'non-ambiguous', we mean that a language identifier can denote several ways to get access to the *same* language.

  As examples:[26]
- '`po`' is ambiguous because that it may start 'Polish' or 'Portuguese', two different languages;
- '`frenchb`' is a language identifier that gets access to only the `frenchb` option of the babel package;
- '`fr`' and '`fre`' are not ambiguous and get access to either babel's option or the frenchle package. The '`french`' identifier has the same property. Since it can get access to the `frenchb` option of babel, do not confuse this feature with aliases handled by the babel package. The language definition file for French is `frenchb.ldf`,[27] but this option may be loaded by '`frenchb`', '`french`' or '`francais`' (see footnote 15). This last identifier is unusable with MlBibTeX, because it only recognises the names of the `.ldf` files located at babel's directory.[28]

4 Implementation issues

4.1 Implementing language identifiers

The language identifiers handled by MlBibTeX obviously form a dictionary. As we show in the previous section, we have to look into this dictionary not only for complete language identifiers but also for non-ambiguous prefixes. So this dictionary's implementation must be efficient. *Tries*[29] are the best implementation to put into action such information retrieval. Such a trie implementing our dictionary is pictured in Figure 4. The root is an array indexed by all the letters of the alphabet. Each component is either a null pointer, in which case the word does not exist within the dictionary, or an access to another letter-indexed array if there are words beginning with the recognised prefix, or a pointer to a resource if a

[24] Also, any TeXnician will have noticed that this new version requires the ifthen package [34, § A.3.2].

[25] For a fragment of a document, such codes are used by the predefined `xml:lang` attribute [39, p. 276]. Within DocBook documents, this attribute is named `lang` [45, p. 81].

[26] In the following we assume that the available packages and options are those of TeX Live 2008.

[27] '`.ldf`' is for '**L**anguage **D**efinition **F**ile', see [34, § 9.5.3].

[28] In the directory `.../texmf-dist/tex/generic/babel` if we consider the TeX Live implementation.

[29] Here is some terminology about trees implementing dictionaries:

- a **digital tree** is a tree for storing strings in which nodes are organised by substrings common to two or more strings;
- a **trie** is a particular case of a digital tree: there is only one node for every common prefix.

The name 'trie' originates from the central letters of the word 're**TRIE**val' [6]. A good but old-fashioned description of this structure has been given by Donald E. Knuth: cf. [31, § 6.3] & [30, Ch. 6, §§ 17–31]. Tries are used within TeX's program [29, §§ 920–924].

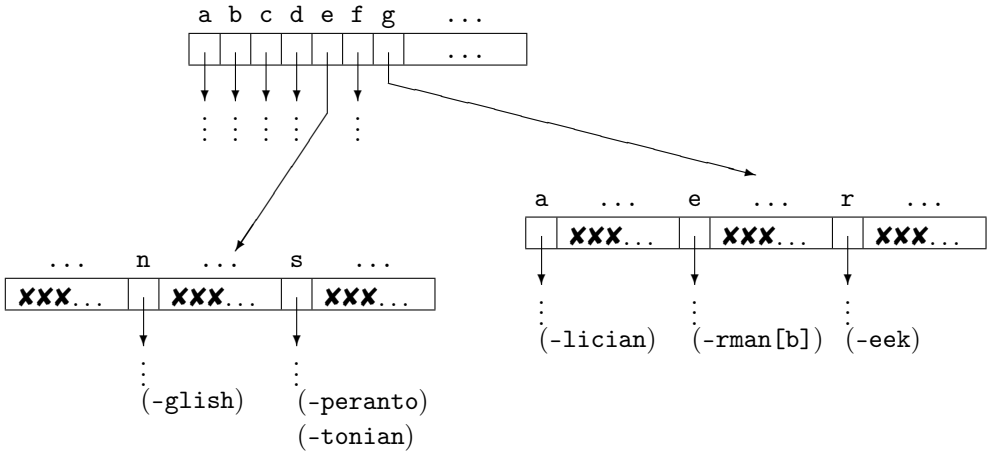


Figure 4: Searching for language identifiers by means of a trie.

word's end has been reached. In the trie in Figure 4, we see that the only language identifiers beginning with 'e' are those whose second letter is 'n' (for 'english') or 's' (for 'estonian'). Likewise, we can see that the language identifiers beginning with 'g' are the non-ambiguous prefixes of `galician`, `german[b]` and `greek`.

As the authors of [2] noticed, such an implementation by means of an array can be very space-consuming since there is many empty locations in the arrays of a trie. That is particularly true in our case, since there is only a few words denoting natural languages' names, in comparison with the whole of a dictionary for a complete language. So we decided to implement such tries by **ternary search trees**,[30] as shown in Figure 5, where the trie of MlBibTeX's language identifiers is sketched. Such a ternary search tree either is a leaf, or has three branches. Left and right branches — pictured in Figure 5 by a double-headed arrow — give access to letters less and greater than the current one. A middle branch gets access to the following letter of a word. A boxed character[31] means that this character comes last in the shortest non-ambiguous prefix of a language identifier.

At MlBibTeX's installation, we consider the *ad hoc* packages' names and the `.ldf` files located in the babel's package directory. These names are used to build a *height-balanced* ternary search tree.[32] In our case, this property means that if we are located at any node within our ternary search tree, the numbers of letters to the left and right of the current one differ by one at most.

4.2 Multilingual method information

When MlBibTeX searches the language identifier trie for a non-existing or ambiguous identifier, the result is `#f`, the 'false' value in Scheme [25, § 6.3.1]. Otherwise, the result is a linear list whose elements — called **multilingual methods** w.r.t. MlBibTeX's terminology — are organised this way:

(⟨*marker*⟩ ⟨*opening*⟩ . ⟨*closing*⟩)

where:

⟨*marker*⟩ specifies a method used to switch to the language denoted by the identifier, e.g., an option of the babel package or an *ad-hoc* package;

⟨*opening*⟩ is a thunk[33] that results in a string put before a fragment written in the corresponding natural language;

⟨*closing*⟩ the same, but the string result is put after a fragment in the corresponding language.

Figure 6 shows how the language identifiers for French allow us to get access to the different ways to surround a fragment written in French. Given a character within our trie, a dashed arrow points to the result of the function searching for a string ending with this character.[34] There exist *default* multilingual methods, e.g.:

[30] Another solution could have used a compact implementation of *maps*, as provided by Python [32, pp. 49–51]. Often *hash tables* — associating *keys* with *values* by means of a *hash code* — are used for such search. They are directly provided by Common Lisp [40], Perl [44, Ch. 2], and Ruby [33], but are not as efficient as ternary search trees, as shown in [3].

[31] The '`#\...`' notation for a character comes from Scheme [25, § 6.3.4].

[32] The **height** of a tree is the maximum distance of any leaf from the root of a tree. In a **height-balanced** ternary tree, left and right branches differ in height by no more than one and each of these two branches is recursively height-balanced, too. Searching balanced trees is more efficient on average. See [31] for more details about this notion.

[33] In functional programming, this word denotes a zero-argument function.

[34] In fact, the actual implementation — more efficient — is slightly different, due to some advanced features of Scheme. But our functions behave exactly as shown in Figure 6.

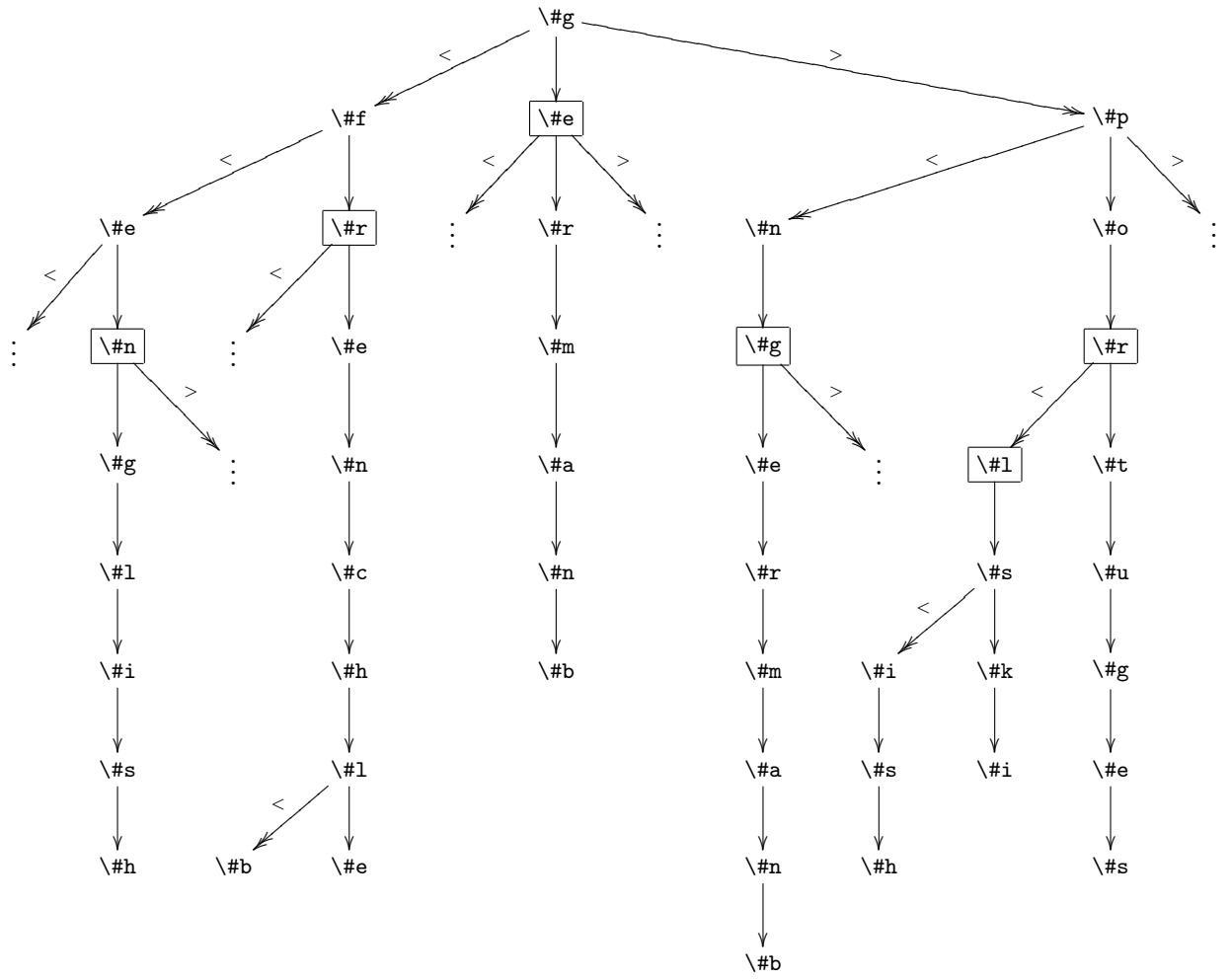


Figure 5: Implementing a trie by means of a ternary search tree.

```
((*frenchle*) (lambda () "{\english") .
              (lambda () "}"))
```

being used for natural languages other than French if the frenchle package has been loaded when the document is processed.[35]

In Figure 6, we can remark that the approach used in ConTEXt is included in such lists, except if the language identifier gets access to one method suitable for LATEX even though other methods for the same language exist. So the identifiers `fr`, `fre`, ... `french` can be used when a bibliography for ConTEXt is derived, but neither `frenchb` nor `frenchle`.

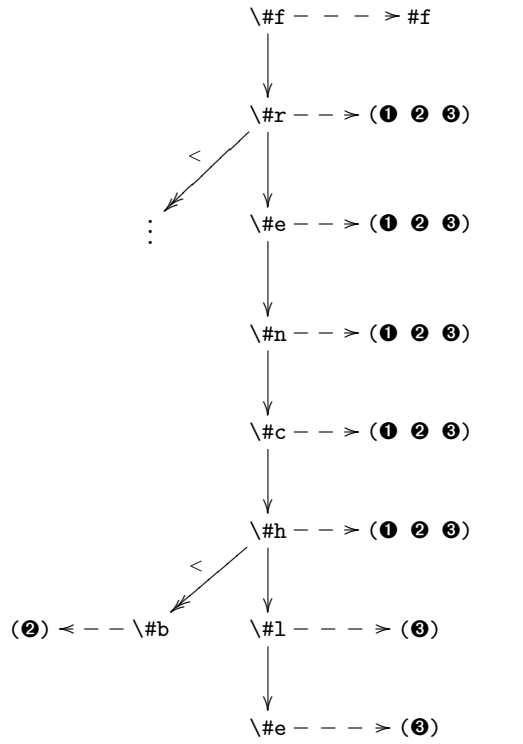
5 Conclusion

This article is an introduction to MlBibTEX's implementation core. We have tried to be precise as far as possible and avoid low-level details. Our goal was to show our realisation as a compromise between user-friendliness and a high-performing implementation. At the time of writing, the available backends are LATEX and ConTEXt. As we wrote in [23], 'when we began [our adaptation of MlBibTEX to ConTEXt] (...), we were afraid we would have to reprogram some important parts of MlBibTEX'. As shown by our examples, the management of language identifiers did not need a major revision when we integrated a backend for ConTEXt. So we think that our implementation is robust. Other adaptations to other backends — e.g., for (X)HTML — should confirm that. We are confident.

6 Acknowledgements

After discussion with some people, I realised that tries were not very well-known. Implementing them is not a small exercise, but is very worthwhile... and actually useful within natural language processing. I

[35] But the language identifier *must* be recognised. If not, any multilingual method, even a default one, cannot be used.



where:

```
❶ ≡ ((*context*) (lambda () "{\language[fr]") .
                 (lambda () "}"))
❷ ≡ ((*babel-option*)
     (lambda ()
       (if ((l-available-languages 'check?) 'frenchb)
           "\begin{otherlanguage*}{frenchb}"
           "")) .
     (lambda ()
       (if ((l-available-languages 'check?) 'frenchb)
           "\end{otherlanguage*}"
           "")))
❸ ≡ ((*frenchle*) (lambda () "{\french ") .
                  (lambda () "}"))
```

Figure 6: Multilingual methods associated with a language identifier.

hope that this article will contribute to demystifying this structure. Thanks to Karl Berry and Barbara Beeton, who kindly proofread this article.

References

[1] Harald Tveit Alvestrand: *Request for Comments: 1766. Tags for the Identification of Languages*. UNINETT, Network Working Group. March 1995. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1766.html.

[2] Jun-Ichi Aoe, Katsuhi Morimoto and Takashi Sato: "An Efficient Implementation of Trie Structures". *Software — Practice and Experience*, Vol. 22, no. 9, pp. 695–721. September 1992.

[3] Jon L. Bentley and Robert Sedgewick: "Algorithms for Sorting and Searching Strings". In: *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 360–369. January 1997.

[4] Antoni Diller: *LaTeX wiersz po wierszu*. Wydawnictwo Helio, Gliwice. Polish translation of *LaTeX Line by Line* with an additional annex by Jan Jelowicki. 2001.

[5] Daniel Flipo: *Documentation sur le module frenchb de babel*. Version 2.3c. February 2009. http://daniel.flipo.free.fr/frenchb/frenchb2-doc.pdf.

[6] Edward Fredkin: "Trie Memory". *Communications of the ACM*, Vol. 3, no. 9, pp. 490–499. September 1960.

[7] Bernard Gaulle : « Comment peut-on personnaliser l'extension french de LaTeX ? ». *Cahiers GUTenberg*, Vol. 28–29, p. 143–157. Actes de la x^e conférence TeX européenne. Saint-Malo, France. Mars 1998.

[8] Bernard Gaulle : « À propos de french ». *La lettre GUTenberg*, Vol. 15, p. 16–17. Juillet 1999.

[9] Bernard Gaulle : « Nouvelles french ». *La lettre GUTenberg*, Vol. 16, p. 11–12. Décembre 1999.

[10] Bernard Gaulle : « À propos de french ». *La lettre GUTenberg*, Vol. 20, p. 5–7. Octobre 2001.

[11] Bernard Gaulle : *Notice d'utilisation de l'extension frenchpro pour LaTeX. Version V5,995*. Avril 2005. http://www.frenchpro6.com/frenchpro/french/ALIRE.pdf.

[12] Bernard Gaulle : *L'extension frenchle pour LaTeX. Notice d'utilisation. Version V5,9993*. Février 2007. http://www.tug.org/texlive/Contents/live/texmf-dist/doc/latex/frenchle/frenchle.pdf.

[13] Michel Goossens and Sebastian Rahtz, with Eitan M. Gurari, Ross Moore and Robert S. Sutor: *The LaTeX Web Companion*. Addison-Wesley Longmann, Inc., Reading, Massachusetts. May 1999.

[14] Hans Hagen: *ConTeXt, the Manual*. November 2001. http://www.pragma-ade.com/general/manuals/cont-enp.pdf.

[15] Jean-Michel Hufflen: "MlBibTeX: a New Implementation of BibTeX". In: Simon Pepping, ed., *EuroTeX 2001*, pp. 74–94. Kerkrade, The Netherlands. September 2001.

[16] Jean-Michel Hufflen: "Multilingual Features for Bibliography Programs: From XML to MlBibTeX". In: *EuroTeX 2002*, pp. 46–59. Bachotek, Poland. April 2002.

[17] Jean-Michel Hufflen: "European Bibliography Styles and MlBibTeX". *TUGboat*, Vol. 24, no. 3, pp. 489–498. EuroTeX 2003, Brest, France. June 2003.

[18] Jean-Michel Hufflen: "MlBibTeX's Version 1.3". *TUGboat*, Vol. 24, no. 2, pp. 249–262. July 2003.

[19] Jean-Michel Hufflen: "A Tour around MlBibTeX and Its Implementation(s)". *Biuletyn GUST*, Vol. 20, pp. 21–28. In *BachoTeX 2004 conference*. April 2004.

[20] Jean-Michel Hufflen: "MlBibTeX: beyond LaTeX". In: Apostolos Syropoulos, Karl Berry, Yannis Haralambous, Baden Hugues, Steven Peter and John Plaice, eds., *International Conference on TeX, XML, and Digital Typography*, Vol. 3130 of *LNCS*, pp. 203–215. Springer, Xanthi, Greece. August 2004.

[21] Jean-Michel Hufflen: "Making MlBibTeX Fit for a Particular Language. Example of the Polish Language". *Biuletyn GUST*, Vol. 21, pp. 14–26. 2004.

[22] Jean-Michel Hufflen: "MlBibTeX in Scheme (*First Part*)". *Biuletyn GUST*, Vol. 22, pp. 17–22. In *BachoTeX 2005 conference*. April 2005.

[23] Jean-Michel Hufflen: "MlBibTeX Meets ConTeXt". *TUGboat*, Vol. 27, no. 1, pp. 76–82. EuroTeX 2006 proceedings, Debrecen, Hungary. July 2006.

[24] Jean-Michel Hufflen: "Introducing LaTeX users to XSL-FO". *TUGboat*, Vol. 29, no. 1, pp. 118–124. EuroBachoTeX 2007 proceedings. 2007.

[25] Richard Kelsey, William D. Clinger, Jonathan A. Rees, Harold Abelson, Norman I. Adams iv, David H. Bartley, Gary Brooks, R. Kent Dybvig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Edmund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman and Mitchell Wand: "Revised^5 Report on the Algorithmic Language Scheme". *HOSC*, Vol. 11, no. 1, pp. 7–105. August 1998.

[26] Brian W. Kernighan and Dennis M. Ritchie: *The C Programming Language*. 2nd edition. Prentice Hall. 1988.

[27] Oleg E. Kiselyov and Kirill Lisovsky: "XML, XPath, XSLT Implementations as SXML, SXPath, and SXSLT". In: *International Lisp Conference 2002*. San Francisco, California. October 2002.

[28] Donald Ervin Knuth: *Computers & Typesetting. Vol. A: The TeXbook*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[29] Donald Ervin Knuth: *Computers & Typesetting. Vol. B: TeX: The Program*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1986.

[30] Donald Ervin Knuth: *Literate Programming*. No. 27 in Lecture Notes. Center for the Study of Language of Information. 1992.

[31] Donald Ervin Knuth: *Sorting and Searching*, Vol. 3 of *The Art of Comuter Programming*. 2nd edition. Addison-Wesley, Reading, Massachusetts. 1998.

[32] Alex Martelli: *Python in a Nutshell*. O'Reilly. March 2003.

[33] Yukihiro Matsumoto: *Ruby in a Nutshell*. O'Reilly. English translation by David L. Reynolds, Jr. November 2001.

[34] Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LaTeX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[35] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide*. 5th edition. O'Reilly & Associates, Inc. August 2002.

[36] Oren Patashnik: *BibTeXing*. February 1988. Part of the BibTeX distribution.

[37] Dave Pawson: *XSL-FO*. O'Reilly & Associates, Inc. August 2002.

[38] Bernd Raichle: *Die Makropakete „german" und „ngerman" für LaTeX 2ε, LaTeX 2.09, Plain-TeX and andere darauf Basierende Formate*. Version 2.5. Juli 1998. Im Software LaTeX.

[39] Erik T. Ray: *Learning XML*. O'Reilly & Associates, Inc. January 2001.

[40] Guy Lewis Steele, Jr., Scott E. Fahlman, Richard P. Gabriel, David A. Moon, Daniel L. Weinreb, Daniel Gureasko Bobrow, Linda G. DeMichiel, Sonya E. Keene, Gregor Kiczales, Crispin Perdue, Kent M. Pitman, Richard Waters and Jon L White: *Common Lisp. The Language. Second Edition*. Digital Press. 1990.

[41] W3C: *XML Names*. W3C document. Edited by Tim Bray, Dave Hollander, and Andrew Layman. January 1999. `http://www.w3.org/TR/1999/REC-xml-names-19990114/`.

[42] W3C: *XSL Transformations (XSLT). Version 1.0*. W3C Recommendation. Edited by James Clark. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116/`.

[43] W3C: *Extensible Stylesheet Language (XSL). Version 1.0*. W3C Recommendation. Edited by James Clark. October 2001. `http://www.w3.org/TR/2001/REC-xsl-20011015/`.

[44] Larry Wall, Tom Christiansen and Jon Orwant: *Programming Perl*. 3rd edition. O'Reilly & Associates, Inc. July 2000.

[45] Norman Walsh and Leonard Muellner: *DocBook: The Definitive Guide*. O'Reilly & Associates, Inc. October 1999.

Graphics

Asymptote: Lifting TEX to three dimensions

John C. Bowman and Orest Shardt

Abstract

Asymptote, a modern successor to the METAPOST vector graphics language that features robust floating-point numerics, high-order functions, and deferred drawing, has recently been enhanced to generate fully interactive three-dimensional output. This data can either be viewed with Asymptote's native OpenGL-based renderer or internally converted to Adobe's highly compressed PRC format for embedding within a PDF file. Asymptote thus provides the scientific community with a self-contained and powerful TEX-aware facility for generating portable interactive three-dimensional PDF files.

1 Introduction

The descriptive vector graphics language Asymptote[1] was developed to provide a standard for drawing mathematical figures, just as TEX and LATEX have become the standard for typesetting equations in the mathematics, physics, and computer science communities [1]. For professional quality and portability, Asymptote natively generates PostScript, PDF, and PRC vector graphics output. The latter is a highly compressed 3D format that is typically embedded within a PDF file and viewed with Adobe Reader.

In both two and three dimensions, consistent fonts and equations should be used in the graphics and text portions of a document. This implies that labels must be typeset directly by TEX. This article provides an overview of the major advances in the current version (1.82) of Asymptote that allow it to extract and lift Bézier font descriptions generated by TEX and Dvips into 3D, using efficient algorithms for partitioning planar regions into nondegenerate Coons patches [3]. Together with 3D generalizations of the METAFONT path operators and a method for computing twist-free tubes and arrowheads, these algorithms provide the 3D foundation of Asymptote.

2 Bézier surfaces

A major recent advance in Asymptote is the ability to embed Bézier surfaces as interactive PRC content within a PDF file, as illustrated in Fig. 1.[2] In contrast, the version of U3D supported by Adobe can only render surfaces described by polygons and hence is not a suitable vector graphics format.

Figure 1: An interactive 3D PDF of a Bézier surface representation of the Utah Teapot.

3 Three-dimensional TEX

TEX produces output in a special device independent format (DVI). While this output can be easily turned into PostScript, one needs a way of extracting Bézier curves that describe properly kerned font characters. Asymptote does this by overloading the PostScript `/show` operator, as described in Appendix A. Special care was required to handle the filled rectangles that TEX uses to draw square root symbols and fraction bars. The resulting exact 2D vector representation of the original TEX input is treated by Asymptote as an array of paths to be filled with the PostScript nonzero winding number fill rule.

The routine `bezulate` described in Figs. 2 and 3, along with the nondegenerate patch splitting algorithms described in [3], is used to convert the resulting Bézier paths to Bézier surfaces. These surfaces are then output in the PRC format, along with a rendered preview image for noninteractive viewing and printing. Using these techniques, Asymptote is then able to typeset the Gaussian integral in Fig. 4 as an interactive 3D diagram.

4 Thick lines in 3D

Figure 5 depicts capped thick lines and Asymptote's five (METAPOST-inspired) path connectors [2]:

```
-- .. & --- ::
```

for the following path, when lifted to the x–y plane:

[1] Andy Hammerlindl, John Bowman, and Tom Prince, available under the GNU Lesser General Public License from `http://asymptote.sourceforge.net/`.

[2] An interactive PDF version of this article may be found at `http://asymptote.sourceforge.net/articles/`.

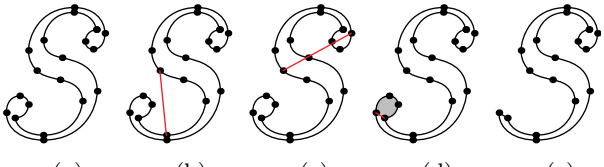


Figure 2: The `bezulate` algorithm. Starting with the original curve (a), several possible connections between nodes separated by 3 or 2 segments are tested. Connections are rejected if they do not lie entirely inside the original curve. This occurs when the midpoint is not inside the curve (b), or when the connecting line segment intersects the curve more than twice (c). If a connecting line passes both tests, the shaded section is separated (d) and the algorithm continues with the remaining path (e).

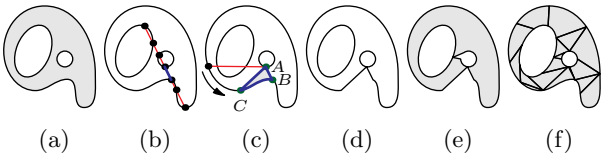


Figure 3: Splitting of non-simply connected regions into simply connected regions. Starting with a non-simply connected region (a), the intersections between each curve and an arbitrary line segment from a point on an inner curve to the outer curve are found (b). Consecutive intersections of this line segment, at points A and B, on the inner and outer curves, respectively, identify a bounded region. Such a region can be found by searching along the outer curve for a point C such that the line segment AC intersects the outer curve no more than once, intersects an inner curve only at A, and determines a region ABC between the inner and outer curves that does not contain an inner curve. Once such a region is found (c), it is extracted (d). This extraction merges the inner curve with the outer curve. The process is repeated until all inner curves have been merged with the outer curve, leaving a simply connected region (e) that can be split into Bézier surface patches. The resulting patches and extracted regions are shaded in (f).

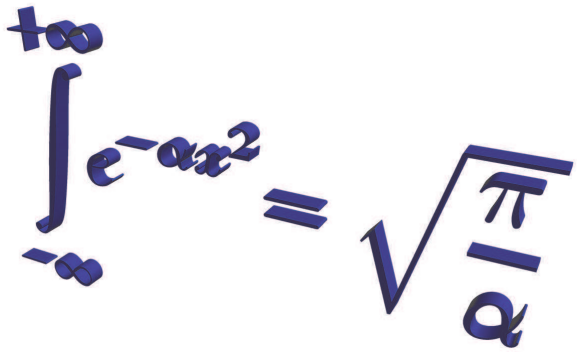


Figure 4: The Gaussian integral lifted to 3D.

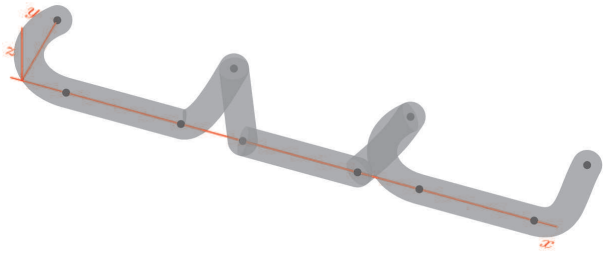


Figure 5: Interactive 3D diagram illustrating thick capped lines, opacity, and the five Asymptote path connectors.

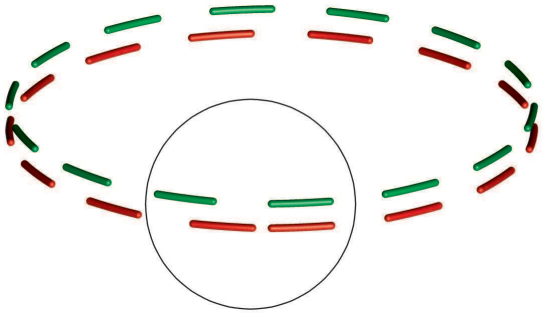


Figure 6: Comparison of arc length adjusted (green) and unadjusted (red) 3D dashed lines.

```
(0,10)..(5,0)---(18,0)::{(0,1)}(20,10)
&(20,10)..(25,0)--(38,0)::{(0,1)}(40,10)
&(40,10)::(45,0)---(58,0)..{(0,1)}(60,10).
```

Hemispheres are aligned at discontinuous junctions of Bézier segments. Disks, hemispheres, or closed cylinders can be used to cap the ends of a Bézier curve, according to the specified PostScript line cap.

Just as in 2D, the on-off duty cycle pattern for generating dashed lines can be automatically adjusted slightly to fit the path arc length evenly, as illustrated in Fig. 6.

A modification of Asymptote's adaptive thick line routine, contributed by Philippe Ivaldi and based on the rotation minimizing frame algorithm described by Wang [4], can be used to construct a tube of arbitrary (noncircular) cross section. For example, Fig. 7 was created by rotating the Greek letter π along a curve describing a trefoil knot.

Jens Schwaiger used similar methods to design a 3D version of Asymptote's `labelpath` function for typesetting text along curves and surfaces, as illustrated in Fig. 8.

5 Arrowheads in 3D

Arrows are frequently used in illustrations to draw attention to important features. We designed curved

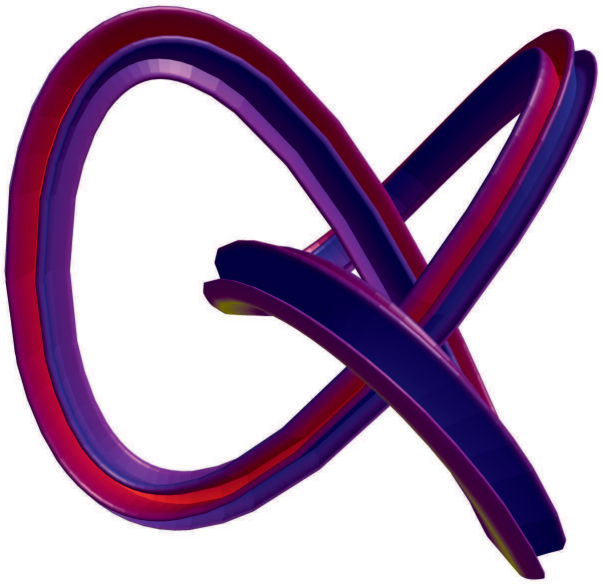


Figure 7: A trefoil knot drawn with Asymptote's arbitrary cross section `tube` module.

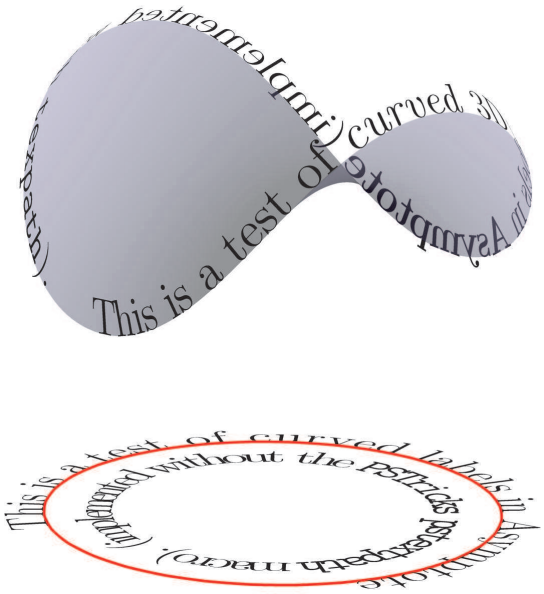


Figure 8: Illustration of curved labels drawn with the `labelpath3` module.

3D arrowheads that can be viewed from a wide range of angles. For example, the default 3D arrowhead was formed by bending a cone around the tip of a Bézier curve using the same algorithm as is used for constructing thick lines. Planar arrowheads derived from 2D arrowhead styles are also implemented; they are oriented by default on a plane perpendicular to the initial viewing direction. Examples of these arrows are displayed in Figs. 9 and 10. An engineering drawing that uses planar arrows is displayed in Fig. 11.

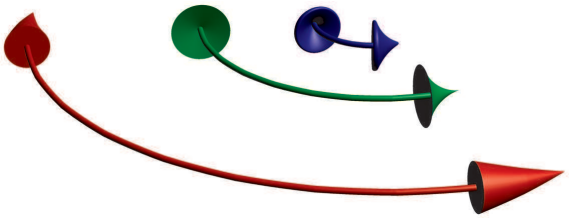


Figure 9: Predefined 3D revolved arrowheads: (blue) `TeXHead3`; (green) `HookHead3`; (red) `DefaultHead3`.

6 Double deferred drawing

Journal size constraints typically dictate the final width and height, in PostScript coordinates, of a 2D or projected 3D figure. However, it is often convenient for users to work in more physically meaningful coordinates. This requires *deferred drawing:* a graphical object cannot be drawn until the actual scaling of the user coordinates (in terms of PostScript coordinates) is known [1]. One queues a function to do the drawing only once the overall scaling is known. Asymptote's high-order functions provide a flexible automatic sizing mechanism: either or both of the 3D model dimensions and the final projected 2D size may be specified. This requires two levels of deferred drawing, a first pass to size the 3D model and a second pass to scale the resulting picture to fit the 2D size specification.

Deferred drawing allows one to draw a fixed-sized object at a scaled coordinate. The following code shows how to draw circles with 5mm radii at each vertex of a unit cube, independent of the overall picture scaling (cf. Fig. 12):

```
import three;
size(4cm);
currentprojection=orthographic(5,4,2);

void Circle(triple c, pen p) {
  picture pic;
  draw(pic,scale3(5mm)*unitcircle3,p);
  add(pic,c);
}

path3[] g=unitbox;
draw(g);

for(path3 p : g)
  for(int i=0; i < length(p); ++i)
    Circle(point(p,i),red);
```

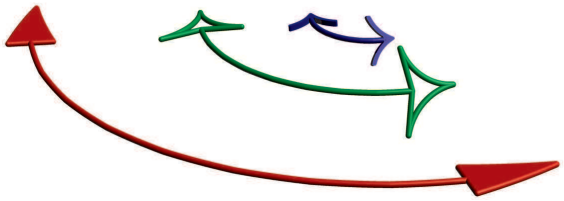


Figure 10: Predefined planar curved arrowheads: (blue) `TeXHead2`; (green) `HookHead2`; (red) `DefaultHead2`.

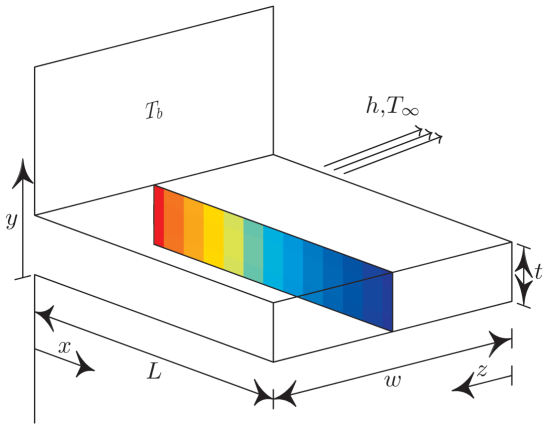


Figure 11: Temperature distribution in a cross section of a heat fin.

7 Interactive 3D Graphs

An important application of 3D TEX is in scientific graphing. The following code generates the interactive 3D surface in Fig. 13.

```
import graph3;
import grid3;
import palette;

currentprojection=orthographic(0.8,0.7,1.5);
size(225pt,0,IgnoreAspect);

real f(pair z) {
  return cos(2pi*z.x)*sin(2pi*z.y);
}
surface s=surface(f,(-1/2,-1/2),(1/2,1/2),20,
                  Spline);
draw(s,mean(palette(s.map(zpart),Rainbow())),
     black);
xaxis3(Label("$x$",0.5),Bounds,InTicks);
yaxis3(Label("$y$",0.5),Bounds,InTicks);
zaxis3(Label("$z$",0.5),Bounds,-1,1,
       InTicks(trailingzero));
grid3(XYZgrid);
```

In Fig. 14, a 3D interactive plot of the surface of the function $\Gamma(z) = \int_{0^+}^{\infty} e^{-t}t^{z-1}\,dt$, extended analytically to the complex plane, emphasizes its poles

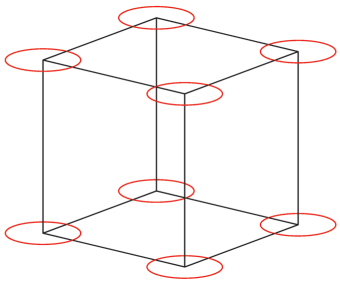


Figure 12: Example of double deferred drawing.

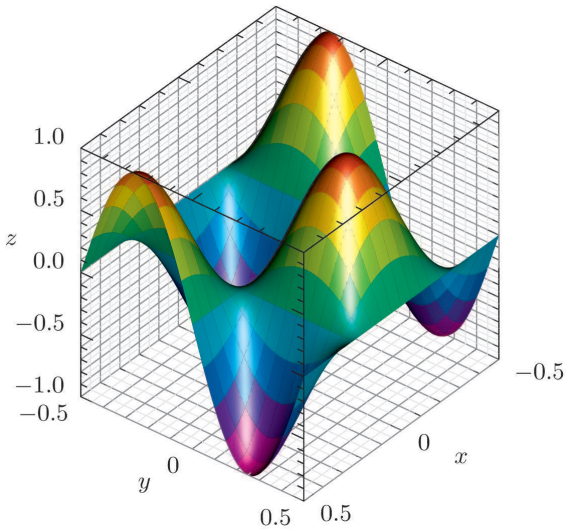


Figure 13: An interactive surface plot with elevation coloring.

at the origin and at negative integers. This was produced with the Asymptote code:

```
import graph3;
import palette;
size(225pt,0,IgnoreAspect);
currentprojection=orthographic(1,-1.8,1);

real X=4.5; real M=abs(gamma((X,0)));
pair Gamma(pair z) {
  return (z.x > 0 || z != floor(z.x)) ?
    gamma(z) : M;
}
real f(pair z) {return min(abs(Gamma(z)),M);}

surface s=surface(f,(-2.1,-2),(X,2),60,Spline);

real Arg(triple v) {
  return degrees(Gamma((v.x,v.y)),warn=false);
}

s.colors(palette(s.map(Arg),Wheel()));
draw(s);
```

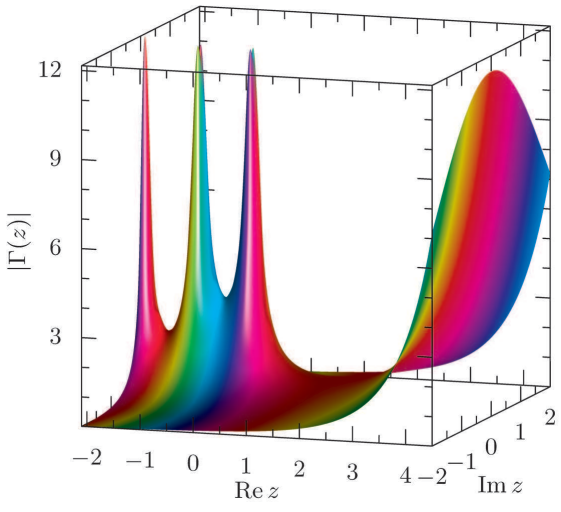
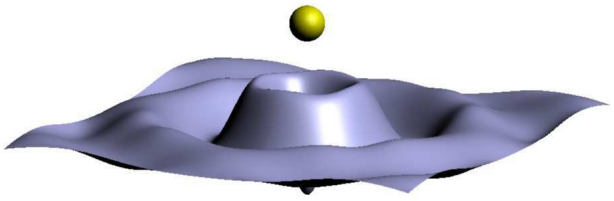


Figure 14: Surface plot of $\Gamma(z)$ in the complex plane, using an RGB color wheel to represent the phase. Red indicates real positive values.

```
xaxis3("$\mathop{\rm Re} z$",Bounds,InTicks);
yaxis3(Label("$\mathop{\rm Im} z$",align=Y-3Z),
       Bounds,InTicks);
zaxis3(rotate(90)*"$|\Gamma(z)|$",
       Bounds,InTicks);
```

8 Inline 3D PDF animations

Inline 3D PDF movies like the one below can be embedded with the help of the LaTeX `animate.sty` package. Unlike 2D inline PDF movies, each frame of a 3D movie is currently pre-rendered by Asymptote to a specified resolution, in order to resolve hidden surfaces correctly.



9 Future directions

There are still a number of applications (including the above animation) where vector PostScript or non-interactive PDF output of 3D scenes would be desirable. For example, Adobe Reader currently cannot generate and print high-resolution renderings of 3D objects.

PostScript is a 2D language that supports only Bézier splines and surfaces, which are shape invariant under affine (orthographic) projection but not perspective projection. In contrast, nonuniform rational B-splines are invariant even in the presence of perspective distortion since they are Bézier curves in a projective space described by homogeneous coordinates. Although PostScript is only a 2D language, vector graphics projections of Bézier surfaces are nevertheless possible using tensor product patch shading and hidden-surface splitting along approximations to the visible surface horizon.

We plan to implement partial prerendering of 3D manifolds to allow 3D scenes to be described within a 2D language like PostScript, without giving up on a vector (scalable) description. The idea is to extend Asymptote's 3D picture structure to segment and sort Bézier surfaces to resolve hidden surfaces correctly in the projected PostScript output. This will require the development of new algorithms for approximating intersections of Bézier surfaces and curves with each other. In collaboration with Troy Henderson and L. G. Nobre, we also plan to investigate techniques for optimally approximating nonuniform rational B-splines by Bézier curves through the addition of new control points. This will allow 2D projections of Bézier curves and surfaces to be well described as vector graphics objects in PostScript.

In the near future, we plan to provide JavaScript support for stationary billboards that always face the camera, as well as PRC animations.

As an aside, let us return to the issue regarding implicit equation solving raised in [1]. Unlike METAFONT and METAPOST, Asymptote does not currently have the notion of a `whatever` unknown. It was pointed out in [1] that the most common uses of `whatever` in METAPOST are probably more clearly written using explicit functions like `extension`. One METAPOST user recently asked us whether there is an elegant way to construct the circumscribed circle of a triangle, centered at the intersection point of two perpendicular bisectors. Indeed, the METAPOST code:

```
beginfig(1)
  path tri;
  u := 1in;
  tri := (origin--(1,0)--(2,1)--cycle) scaled u;
  z0 = (point 0.5 of tri) + whatever *
         (direction 0.5 of tri rotated 90);
  z0 = (point 1.5 of tri) + whatever *
         (direction 1.5 of tri rotated 90);
  dotlabel(btex etex, z0);
  draw fullcircle scaled
    (2*abs(z0-point 0 of tri)) shifted z0;
  draw tri withcolor red;
endfig;
end
```

can be written elegantly in Asymptote:

```
unitsize(1inch);
path tri=(0,0)--(1,0)--(2,1)--cycle;
```

```
pair z1=point(tri,0.5);
pair z2=point(tri,1.5);
pair z0=extension(z1,z1+I*dir(tri,0.5),
                  z2,z2+I*dir(tri,1.5));
dot(z0);
draw(circle(z0,abs(z0-point(tri,0))));
draw(tri,red);
```

Perhaps this example will help motivate hesitant METAPOST users to migrate to Asymptote, allowing them to take full advantage of the powerful interactive 3D functionality described in this article.

10 Conclusions

We believe that Asymptote is the first software package to lift TEX into 3D. It also provides a self-contained open source tool for producing portable 3D PDF files that support Bézier surfaces. As illustrated in the examples we have provided, these are important features for publication-quality scientific graphing. Interactivity is critical for visualization and mental reconstruction of 3D data, as it helps the human brain resolve the degeneracy inherent in 2D projection.

11 Credits

We thank Philippe Ivaldi, Radoslav Marinov, Malcolm Roberts, Jens Schwaiger, and Olivier Guibé for discussions related to this work. Special thanks goes to Andy Hammerlindl, who designed much of the underlying Asymptote language. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada.

A Extracting Bézier curves from TEX

We now describe the PostScript code used to extract smooth font descriptions from Dvips output. First, a PostScript procedure is defined to output a coordinate:

```
/ASYo {( ) print 12 string cvs print} bind def
```

The PostScript `/show` operator can then be overloaded, using the `pathforall` operator to obtain the coordinates of the Bézier control points:

```
/show {currentpoint newpath moveto false charpath
{( moveto) print ASYo ASYo}
{( lineto) print ASYo ASYo}
{( curveto) print ASYo ASYo ASYo ASYo ASYo ASYo}
{( closepath) print}
pathforall} bind def
```

The filled rectangles that TEX and Dvips use to draw square root symbols and fraction bars are extracted by overloading the `/v` procedure:

```
/v {neg exch 4 copy 4 2 roll 2 copy 6 2 roll
    2 copy
( moveto) print ASYo ASYo
( lineto) print ASYo add ASYo
( lineto) print add ASYo add ASYo
( lineto) print add ASYo ASYo
( closepath) print} bind def
```

This technique was used to form the TEX characters in the 3D Asymptote logo in Fig. 15.

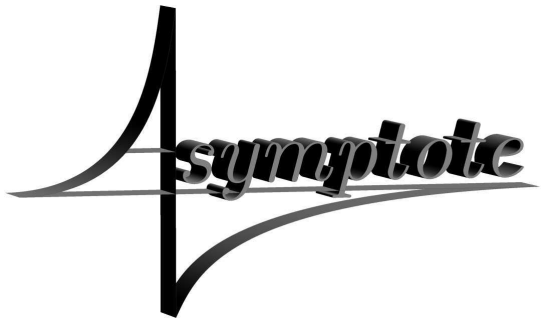


Figure 15: The Asymptote logo in three dimensions.

References

[1] John C. Bowman and Andy Hammerlindl. Asymptote: A vector graphics language. *TUGboat: The Communications of the TEX Users Group*, 29(2):288–294, 2008.

[2] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[3] Orest Shardt and John C. Bowman. Three-dimensional vector representations of nonsimply connected planar surfaces. *Submitted to ACM Trans. Graph.*, 2009.

[4] Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. Computation of rotation minimizing frames. *ACM Trans. Graph.*, 27(1):1–18, 2008.

◇ John C. Bowman
Dept. of Mathematical and Statistical Sciences
University of Alberta
Edmonton, Alberta
Canada T6G 2G1
bowman (at) math dot ualberta dot ca
http://www.math.ualberta.ca/~bowman/

◇ Orest Shardt
Dept. of Chemical and Materials Engineering
University of Alberta
Edmonton, Alberta
Canada T6G 2V4
shardt (at) ualberta dot ca

Supporting layout routines in MetaPost

Wentao Zheng

Abstract

METAPOST is known as a powerful graphics drawing language. However, METAPOST dose not provide any mechanisms to automatically lay out graphic objects. In this article, we present two approaches to help METAPOST users to automatically or semi-automatically lay out objects that they are drawing.

1 Introduction

METAPOST is widely adopted by LaTeXers to generate high quality graphics in their documents. It is well known for its precisely controlled geometric restriction, textual label integration with TeX, and extendable macros. A variety of packages/macros, like MetaUML, m3D, have been created that allow LaTeX users to draw high quality graphics easily and professionally. Although these packages provide us with the functionality to draw objects and links, it is often noticed that we still need to spend a lot of effort laying out the objects we are drawing.

Let's take a look at an example diagram (Figure 1) from John Hobby's METAPOST manual [1] on page 63 (it may appear on different pages in different versions of the METAPOST manual). It is a simple finite state diagram that has five states, ten arrow links and corresponding labels. There are several routines in the source code that have something to do with the diagram's layout, i.e., state (node) positioning, arrow (link) direction tuning and label positioning. And these routines take at least half of the total source code. The problem is clear now: can we develop a METAPOST package that provides users with automatic or semi-automatic layout routines?

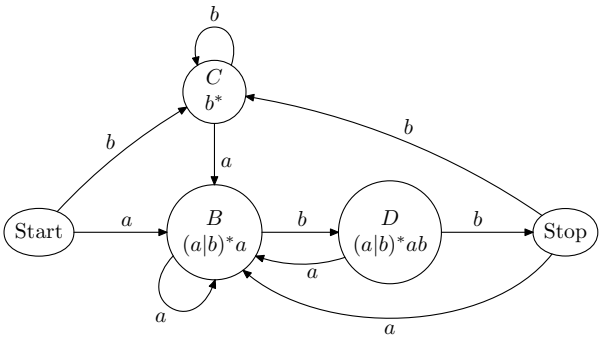


Figure 1: An example diagram from the METAPOST manual

The problem of automatically laying out general graphs is not new. A number of academic activities, such as the International Symposium on Graph Drawing, have been existed for decades on the research of algorithms and methods for graph visualization. Aesthetics and computational complexity are the major concerns in the research. However, algorithms/methods for general graph drawing problem with both efficiency and aesthetics have not been presented. Most current research is focused on special graph drawing problems or approximated solutions.

Therefore, the question that whether we can develop a (semi-)automatic layout package for METAPOST cannot be answered simply by a 'Yes' or 'No'. In this article, we first address some issues and challenges in developing a layout package for METAPOST (Section 2), then propose several possible approaches (Section 3). Some future work is also discussed at the end (Section 4).

2 Challenges

Although we will not develop a general method to lay out graphs, even (semi-)automatic layout is difficult.

First of all, METAPOST is not an object-oriented programming language. Although it has facilities to simulate some aspects of OO programming, there is no "base object" in METAPOST. Therefore it is not easy to write a layout routine that can be applied on different graphic objects. For example, METAPOST's `boxes` macro introduces a kind of object (box and circle) with properties

```
c n e s w ne nw se sw
```

so we can manage an object's positions by manipulating geometric relations on those properties. But what if another user wants to use the routine to lay out objects without the properties mentioned above? A practical, though not friendly, solution is to specify rules (properties and methods) to objects that need to use the layout routine.

Secondly, developing a purely automatic layout routine is difficult, even impossible. In the research of graph drawing, practical algorithms exist only for special graphs, such as trees, DAGs (direct acyclic graphs). There exists no general layout method for an arbitrary graph with satisfactory aesthetics and acceptable running time.

The first solution to this problem is designing on demand. That is to say, to develop layout routines for specific graphs. Graphviz is a graph drawing program that takes this approach, providing several practical routines to draw graphs.

Another solution is using the KISS (keep it simple, stupid) principle. That is to keep layout routines small, easy to understand and practical to use. However, by taking this approach, another problem arises: how to design those routines? That is, what to provide, and what to omit? It has been seen in some

diagramming tools that small layout routines are very useful and easy to use. For example, "horizontal or vertical alignment", "equal height or width" are good layout routines. But these examples are not enough; we need to design more routines and expect some combinations of them will generate very useful and sophisticated results.

3 Possible approaches

In this section, we will present several approaches to developing layout routines in METAPOST.

3.1 Reusing Graphviz

Actually, Graphviz is a set of programs for automatically specifying graph layout:

dot makes hierarchical or layered drawings of directed graphs.

neato and **fdp** make spring model layout.

twopi makes radial layout.

crico makes circular layout.

For more information, please take a look at their web site: `http://www.graphviz.org`.

There is a LaTeX package called `dot2tex` that makes use of Graphviz to generate PSTricks and PGF/TikZ commands in LaTeX documents. For detailed information, please take a look at their web site: `http://www.fauskes.net/code/dot2tex`. It is obvious that we can take a similar approach to adopt Graphviz in METAPOST.

In the rest of this section, we are going to use **dot** as an example to show how to use Graphviz to generate automatic layout routines for METAPOST.

Let's first take a look at how to represent the simple graph in Figure 2 in the **dot** language:

```
digraph G {
    A -> B [label = "x"];
    A -> C [label = "y"];
    C -> B [label = "z"];
}
```

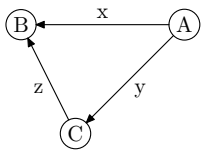


Figure 2: A simple graph

By using the **dot** program, a diagram with automatic layout is generated, as shown in Figure 3. We can see that nodes are separated with proper distances, links are placed with appropriate angular resolutions, and labels are displayed at the right

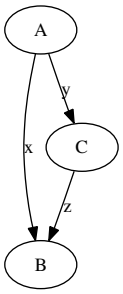


Figure 3: Diagram generated by **dot**

places. Although the diagram is not as "pretty" as the one in Figure 2, the layout is at least readable.

In **dot**, some mechanisms are provided to tune the graph layout with manual control of nodes, links, and labels. We won't introduce them here because we want to keep our focus on the adoption of **dot** in METAPOST.

dot supports several kinds of output format, such as plain text, PostScript, SVG, and binary images. Among those, plain text is the easiest to reuse in METAPOST. The following text is the compiled output of the aforementioned **dot** source code.

```
digraph G {
  node [label="\N"];
  graph [bb="0,0,85,212"];
  A [pos="27,194",
     width="0.75",
     height="0.50"];
  B [pos="27,18",
     width="0.75",
     height="0.50"];
  C [pos="58,106",
     width="0.75",
     height="0.50"];
  A -> B [label=x, pos="e,23,36
                        23,176
                        ...
                        21,46",
                  lp="18,106"];
  A -> C [label=y, pos="...",
                  lp="46,150"];
  C -> B [label=z, pos="...",
                  lp="47,62"];
}
```

We can see that layout information can be extracted from the output. Graph nodes, such as `A`, are indicated by `pos` (position), `width`, and `height`, while links, such as `A -> B`, are indicated by `label`, `pos` (path points), and `lp` (label position). It is easy to automatically extract the layout information from the output and then use it in METAPOST.

In order to use **dot** in METAPOST, we should firstly write METAPOST/**dot** hybrid code (named as an **MPdot** file) as follows:

```
input boxes;
beginfig(1);
  circleit.a(btex A etex);
  circleit.b(btex B etex);
  circleit.c(btex C etex);

  begindot % begin of dot code
    digraph G {
      a -> b [label = "x"];
      a -> c [label = "y"];
      c -> b [label = "z"];
    }
  enddot % end of dot code
endfig;
```

It is noticed that the content between `begindot` and `enddot` is written in **dot** language. We are using METAPOST suffixes, such as `a`, instead of their labels, such as `"A"`, to represent nodes in **dot**. This is because we want to connect the **dot** with the METAPOST code. We show the importance and desirability of doing this below.

In the next step, we use a program to parse **dot** code from the **MPdot** file and rewrite it into another intermediate **dot** file (named an **IMdot** file) for compilation. For those nodes represented by METAPOST suffixes, such as `a`, their respective definitions, like `circleit.a(...)`, will be used to determine their dimensions (width and height). The following code shows what the generated **IMdot** file looks like.

```
digraph G {
  a [ height = 0.19595, width = 0.19595,
      label = "" ];
  b [ height = 0.19174, width = 0.19174,
      label = "" ];
  c [ height = 0.19313, width = 0.19313,
      label = "" ];
  a -> b [ label = "x" ];
  a -> c [ label = "y" ];
  c -> b [ label = "z" ];
}
```

We can see that nodes are defined with `height`, `width`, and `label` properties. The height and width of a node are calculated based on the corresponding suffix defined in METAPOST code. This is the reason why we use METAPOST suffixes to represent **dot** nodes in the **MPdot** file.

The **IMdot** is then sent to the **dot** program for compilation, and layout information is generated. We can extract the layout information from the output and generate METAPOST code to replace the **dot** code in **MPdot** file, resulting in the final METAPOST file. The following is the final METAPOST file, in which the **dot** code is replaced by generated METAPOST code. After compilation, it outputs a graph shown in Figure 4.

```
input boxes;
beginfig(1);
  circleit.a(btex A etex);
  circleit.b(btex B etex);
  circleit.c(btex C etex);

  % the following code is auto generated
  a.c = (7pt,141pt);
  b.c = (7pt,7pt);
  c.c = (24pt,74pt);
  drawunboxed(a,b,c);
  draw fullcircle scaled 0.19in
       shifted a.c;
  draw fullcircle scaled 0.19in
       shifted b.c;
  draw fullcircle scaled 0.19in
       shifted c.c;
  label(btex $x$ etex, (4pt,74pt));
  label(btex $y$ etex, (19pt,108pt));
  label(btex $z$ etex, (19pt,40pt));
  drawarrow
  (6pt,134pt)..(0,62pt)..(6pt,14pt);
  drawarrow
  (9pt,134pt)..(16pt,105pt)..(22pt,81pt);
  drawarrow
  (22pt,67pt)..(15pt,38pt)..(9pt,14pt);
endfig;
```

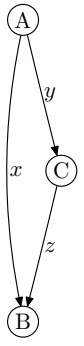


Figure 4: Graph generated by METAPOST with **dot** layout information

With TEX labels integrated and METAPOST's curve path tuning, the graph shown in Figure 4 looks better than that in Figure 3.

The approach of reusing Graphviz that we just explained can be summarized in Figure 5. At first, a user writes a **MPdot** file, in which the **dot** code is translated into a **IMdot** file. The **IMdot** file is then

sent to **dot** for compilation, and layout information is returned. The information is extracted and translated into METAPOST statements to replace the **dot** code in **MPdot** file, resulting a pure METAPOST file, based on which the final graphics is generated.

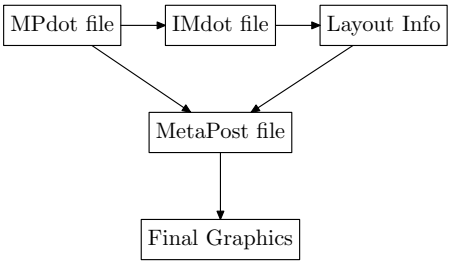


Figure 5: Approach of reusing Graphviz (**dot**)

3.2 Small, stupid routines

As we can see in the previous section, Graphviz is not a perfect layout tool. Users with strong sense of aesthetics may not be satisfied with Graphviz's result. This is why we propose another approach: designing small and stupid routines.

Trivial layout routines have long existed in various diagramming software and user interface designing tools. For example, you can select a number of graphic objects, make them align horizontal, from left to right, and have same width and height. These routines, including alignment, order, and dimension specification are very useful when we are drawing diagrams. So we are going to extend them and make them available in METAPOST.

Generally, there are three types of graphic objects in diagrams, i.e., shapes, links, and labels. A shape is an object with a surrounding path (usually closed), such as a rectangle, ellipse, etc. A link is a path connecting two shapes, usually parameterized with a start shape and end shape, such as arrow link, line link, etc. A label is a textual container containing formatted text, and a transparent surrounding path. Figure 6 shows two shapes (rectangles) connected by an arrow link labeled by "Label".

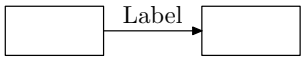


Figure 6: Three types of graphic objects

For a shape, its internal properties should be set by end users or calculated based on its inner label. For a link, the objects it connects to should be set by end users. For a label, only the textual content should be set by end users. Therefore, layout routines should care about where a shape is located, what path points a link should go through, and where a label is placed.

As we mentioned in Section 2, METAPOST is not an object-oriented language. So it is difficult to design routines for different graphic objects. For simplicity, let us focus on laying out graphic objects defined by the `boxes` package, i.e., `box` and `circle`. The common attributes they share are (as shown in Figure 7):

`c` center point of a shape
`n` north point of a shape
`s` south point of a shape
`e` east point of a shape
`w` west point of a shape

Another very important attribute is `bpath`, which is the surrounding path of the shape. We can use this path to determine a shape's bounding area, and ensure that a link's end points are tightly connected on the path.

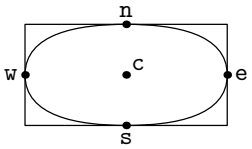


Figure 7: Attributes of a `box` or `circle`

Let's focus on shape layout routines first. The simplest and most frequently used is linear alignment. That is to say, align a number of objects through a line. Consider the following macro

```
line_align <dir>,<gap>,<objects>
```

It uses `dir` (the direction of the line), `gap` (distance between consecutive objects), and `objects` (objects to be aligned) as parameters.

Sometimes, line alignment is not sufficient, so we present another way to align objects: general path alignment. It looks like

```
path_align <path>, <objects>
```

The parameter `path` specifies a path (line or curve) along which the `<objects>` are placed and separated evenly.

Being different from shapes, there is no need to specify the location of a link, because it is used to connect two shapes (in most cases). After the laying out of shapes, the question of where links start and end is quite easy to answer. So link layout should be focused on how we link two shapes: on a straight line, curve or orthogonal polyline. The following macros are used to specify how to layout links:

```
line_link <start_shape>, <end_shape>
curve_link <start_shape>, <start_dir>,
```

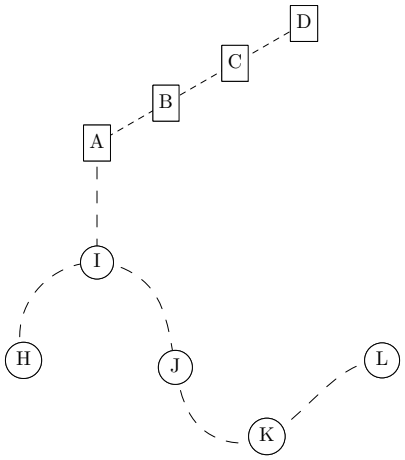


Figure 8: Alignment of shapes

```
          <end_shape>, <end_dir>
orth_link <start_shape>, <start_side>,
          <end_shape>, <end_side>
```

The `line_link` is used to connect `start_shape` and `end_shape`. The `curve_link` takes two other parameters, i.e., `start_dir` (the direction of link path at the start point) and `end_dir` (the direction of link path at the end point). Similarly, the `orth_link` takes parameters `start_side` (north, east, south, or west) and `end_side`. For the first and second link macros, it's easy to implement. But for the last one, more effort is required, and we are not going to solve it in this article. Figure 9 shows three types of link layout (the orthogonal one is drawn manually, just to show what it looks like).

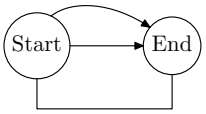


Figure 9: Layout of links

A label's layout is a little complicated. First of all, labels can be treated as a special kind of shape without a surrounding path. It is natural that we let shape layout routines, such as linear alignment, be applicable for labels. Besides, labels have other means for layout. An example is creating a label for a link or a shape. We name this kind of label an association label.

Let's start with association labels for links. Because a label is usually placed somewhere along the path of a link, we can use the following macro to lay out the label.

```
link_label <label> <link> <portion>
```

`label` defines what textual content to be displayed, `link` is a link object suffix, and `portion` is a number between 0 and 1 that denotes where the label is placed along the link path.

Association labels for shapes are easier to handle. In most cases, a shape's properties like `n` and `c` is sufficient for manipulating the positions of labels. Figure 10 shows a number of labels for links and shapes.

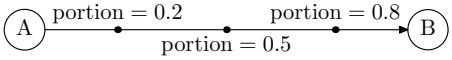


Figure 10: Layout of association labels

After introducing some small and stupid layout routines, we suggest users use them in the following order:

1. Declaring shapes with macros like `boxit` and `circleit`
2. Laying out shapes by using the aforementioned routines
3. Declaring and laying out links
4. Declaring and laying out labels

The reason for this order is that label positions rely on links and shapes, and link paths relies on shapes. So it is necessary to first lay out shapes, then links, and do labels last.

4 Future work

In this article, we propose two approaches of supporting layout routines in METAPOST, to make the drawing of diagrams convenient and aesthetic. We introduce them separately with detailed explanation and some examples. However, the methods presented in this article are at a very early stage; refinement and extension must be done to make them more practical. This is planned for the near future.

References

[1] John Hobby, "METAPOST: A User's Manual", `ctan:graphics/metapost/base/manual/mpman.pdf`.

◇ Wentao Zheng
IBM China Research Laboratory
`zhengwt (at) cn dot ibm dot com`

Glisterings

Peter Wilson

> Calm was the day and through the trembling air
> Sweet-breathing Zephyrus did softly play—
> A gentle spirit, that lightly did delay
> Hot Titan's beams, which then did glister fair.
>
> *Prothalamion*, Edmund Spenser

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

Corrections, suggestions, and contributions will always be welcome.

This installment is not really about (LA)TEX, except peripherally.

> Nothing in India is identifiable, the mere asking of a question causes it to disappear or to merge into something else.
>
> *A Passage to India*, E. M. Forster

1 Reprise

Following the last column [4], Prof. Klaus Lagally wrote to me with another way of discarding an unwanted character at the end of a command. I had shown some code that acted in a similar manner to LaTeX starred macros, except with a '?' instead of a '*'. The problem was to recognise the presence or absence of the character '?' and take different actions according to whether it was there or not, and to also discard the '?' if it was present. More precisely I presented

```
\makeatletter
\def\maybeQ{%
  \@ifnextchar ?{\@maybeQ}{\@maybe}}
\def\@maybeQ#1#2#3{Query (#2) and (#3).}
\def\@maybe#1#2{(#1) and (#2).}
\makeatother
```

Prof. Lagally instead suggested that `\@maybeQ` could be more simply defined as:

```
\makeatletter
\def\@maybeQ ?#1#2{Query (#1) and (#2).}
\makeatother
```

as a means of disposing of the '?'. In either version here are a couple of example results:

`\maybeQ{1st}{2nd}` -> (1st) and (2nd).
`\maybeQ?{1st}{2nd}` -> Query (1st) and (2nd).

> Child! do not throw this book about!
> Refrain from the unholy pleasure
> Of cutting all the pictures out!
> Preserve it as your chiefest treasure!
>
> *A Bad Child's Book of Beasts*,
> Hillaire Belloc

2 MetaPost and pdfLaTeX

The MetaPost program generates PostScript illustrations. These can easily be inserted into a document to be processed by (LA)TEX to produce a `dvi` file. Generally speaking, though, pdfLaTeX cannot handle PostScript files. Fortunately it can handle the limited form of PostScript that MetaPost generates, and so MetaPost illustrations can be directly embedded into a pdfLaTeX document. This, though, is not quite as straightforward as it might be.

Given a file called, say, `figs.mp`, which contains perhaps three pictures, MetaPost will generate 3 files, `figs.1`, `figs.2` and `figs.3`, one for each picture. On the other hand, pdfLaTeX expects MetaPost generated PostScript files to have an `.mps` extension. If you use the graphicx package you can get it to accept files with numeric extensions as though they had an `mps` extension by specifying:

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

which tells `\includegraphics` to treat any extension it does not recognise as though it were `mps`.

LaTeX, or at least programs like `dvips` or `xdvi`, can handle Encapsulated PostScript (`eps`) files, and you can perform similar magic for the graphicx package:

```
\usepackage{ifpdf}
\ifpdf
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{mps}{*}{}
\else
  \usepackage{graphicx}
  \DeclareGraphicsRule{*}{eps}{*}{}
\fi
```

If a MetaPost illustration might be used in a LaTeX (as opposed to pdfLaTeX) document, then put
`prologues := 1;`
at the start of the MetaPost file, which tells MetaPost to generate Encapsulated PostScript files. It seems to do no harm to use the same `prologues` specification for pdfLaTeX.

> A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas.
>
> *A Mathematician's Apology*, G. H. Hardy

3 Spidrons

The other week I was idly glancing through *Science News* when I came across a short article about spidrons [3]; try googling for 'spidron' to get more on the subject. Spidrons, which were discovered and named by the Hungarian designer and graphic artist Dániel Erdély while doodling with hexagons, are made up of ever smaller connected triangles alternating between isosceles and equilateral in form.

It occurred to me that MetaPost could be used to draw these and after a little trial and error I came up with the following MetaPost program to support drawing spidrons.

```
%% semispid.mp  MP macro to draw a semi-spidron
% semispid(center, vertex, iterations,
%          color1, color2, clockwise)
def semispid(suffix $$, $)%
            (expr iter, shadea, shadeb, clock) =
if clock: hxa := -60; else: hxa := 60; fi
pair v[];
path phex[];
v0 := z$$;
v1 := z$;
% enclosing hexagon
for i := 2 upto 6:
  v[i] := v1 rotatedaround(v0,(i-1)*hxa);
endfor
z$a = v1; z$b = v2; z$c = v3;
z$d = v4; z$e = v5; z$f = v6;
phex0 := v1--v2--v3--v4--v5--v6--cycle;
if showverts:
  dotlabels.lft($a,$b,$c,$d,$e,$f);
fi
if showlines:
  draw v1--v3--v5--cycle;
  draw v2--v4--v6--cycle;
fi
% construct triangles
for n:= 1 upto iter:
  k := 10(n-1);
  j := 10n;
  v[1+j] := (v[1+k]--v[3+k])
            intersectionpoint
            (v[2+k]--v[6+k]);
  for i := (2+j) upto (6+j):
    v[i] := v[1+j]
            rotatedaround
           (v0, (i-1-j)*hxa);
  endfor
  if showlines:
    draw v[1+j]--v[3+j]--v[5+j]--cycle;
    draw v[2+j]--v[4+j]--v[6+j]--cycle;
  fi
  phex[n] := v[1+j]--v[1+k]--v[2+k]--cycle;
  phex[n+1] := v[1+j]--v[2+j]--v[2+k]--cycle;
  fill phex[n] withcolor shadea;
  fill phex[n+1] withcolor shadeb;
  if showcells:
    draw phex[n]; draw phex[n+1];
  fi
  if showedges:
    draw v[1+k]--v[1+j];
    draw v[2+k]--v[2+j];
  fi
endfor
if showedges: draw v[1+j]--v[2+j]; fi
if showhex: draw phex0; fi
enddef;
```

As its name implies, the routine `semispid` generates and draws half of a spidron, which Erdély called a semi-spidron, and this is contained within a hexagon. The location arguments are the center point of the enclosing hexagon and the location of one of the vertices. The other arguments control the number of triangles and two colors for coloring alternate triangles. The routine uses booleans, specified elsewhere, to control the display of various aspects of the construction method.

I used the next MetaPost program to create the spidron shown in Figure 1.

```
% glstr9.mp  MP spidron figures
prologues := 1;
input semispid
%%% define the boolean flags and defaults
% show the initial hexagon
boolean showhex; showhex := false;
% label vertices
boolean showverts; showverts := false;
% draw construction lines
boolean showlines; showlines := false;
% draw triangle cell boundaries
boolean showcells; showcells := false;
% work clockwise (yes = true)
boolean rh; rh := false;
% draw sem-spidron outline
boolean showedges; showedges := false;
% shading
color light,dark;
light := 0.1[white,black];
dark := 0.2[white,black];

beginfig(1); % a spidron
  u := 1in;  % units
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;
% center & initial vertex
z0 = (0,0);
z1 = (x0-2u,y0) rotatedaround(z0,60);
semispid(0, 1, 9, dark, light, rh);
y0-y1a = y1a-y10; x10=x0;
```

Figure 1: A spidron

```
z11 = z1b;
semispid(10, 11, 9, light, dark, rh);
endfig;
%% more pictures here
end
```

The construction details of a semi-spidron are illustrated in Figure 2. The `semispid` routine generates the vertices of a hexagon, labelling the given one as 'a', then the others in turn as 'b', 'c', etc. The hexagon is repeatedly partitioned by joining alternate vertices, which creates a smaller interior hexagon, which is then partitioned into a smaller one again, and so on until it all gets 'too small'. The shaded triangles form a semi-spidron, starting on the 'a-b' side of the hexagon, and finishing close to the center. The second half of the complete spidron is a rotation of the first semi-spidron about the midpoint of the 'a-b' edge of the hexagon, with the colors reversed.

Spidrons are space-filling; that is, they can be assembled to completely cover, or tile, a plane surface. You can get a hint about this from Figure 3 which shows three semi-spidrons constructed in a single hexagon. The empty spaces can be exactly filled by three more semi-spidrons. A plane can be completely tiled using hexagons; in this particular case it happens that it can also be completely tiled

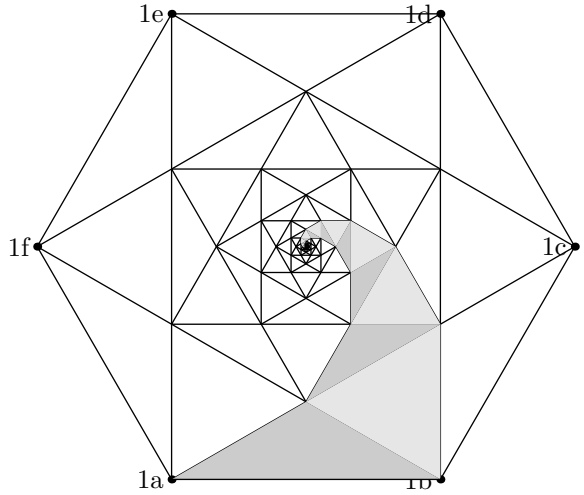


Figure 2: Construction details of a spidron

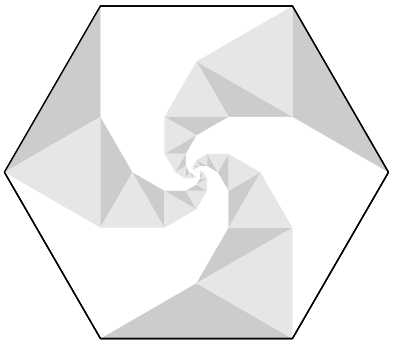


Figure 3: Three semi-spidrons in a hexagon

by spidrons. Interesting effects can be achieved by changing the coloring of the spidrons. An example is shown in Figure 4. For much, much, more on tilings see *Tilings and Patterns* [2], although it doesn't include spidrons as they hadn't been discovered when the book was published.

There is an associated figure that can also be made out of two semi-spidrons. In a spidron the two semi-spidrons are rotations of each other. In the shape that Erdély calls a *hornflake*, shown in Figure 5, the two halves are mirror images of each other. Unlike spidrons, hornflakes are not space-filling but can be used for tiling if they are suitably combined with spidrons, as can be seen in Figure 4.

In his article, Peterson says that

> [Erdély's] insight was to start with an array of hexagons drawn on a sheet of paper and laid as if they were bathroom tiles. By creasing the pattern in the right combinations of mountains and valleys at the lines within each spidron arm and leaving a small

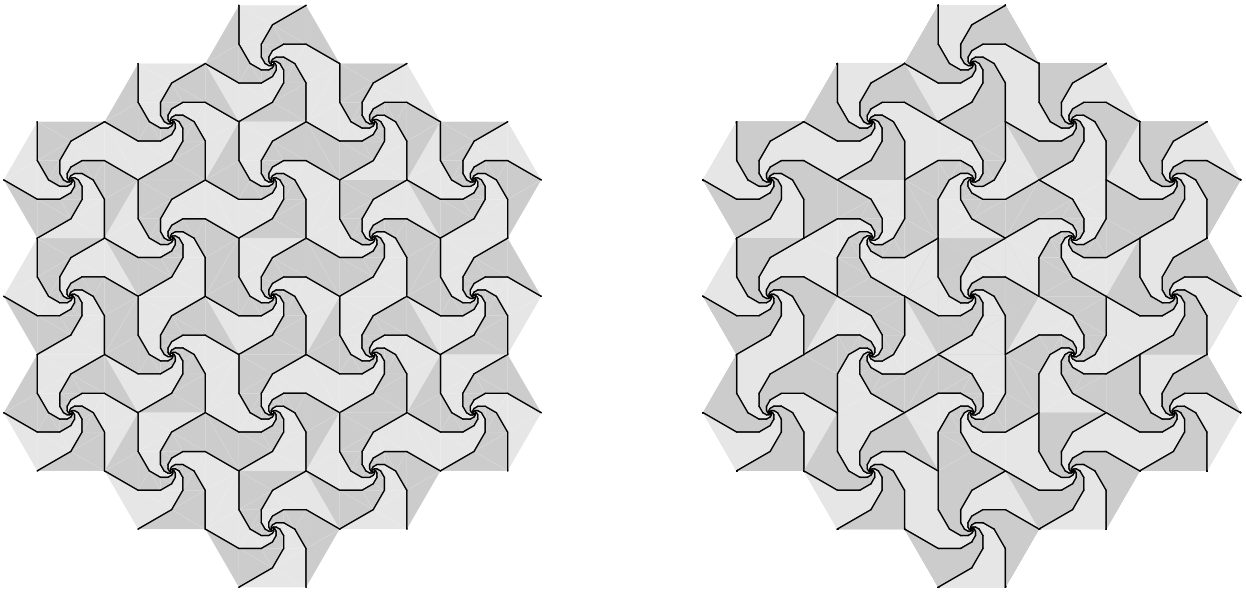


Figure 4: Tilings: (left) Spidrons can do it alone (right) Hornflakes need spidrons

Figure 5: A hornflake

> hole at the center of each hexagon, he crinkled the whole aray into a dramatic three-dimensional relief.

It turns out that spidron patterns can also be assembled into novel three-dimensional crystal-like froms with spiral polygonal faces.

What is missing from the article is any hint as to what the 'right combinations' of folds might be to create these effects. After some searching on the web I found the following remarks by Erdély [1].

> I folded every second edge, reaching to the centre of the created hexagon in the given Spidron system, as a spine and folded every first edge as a groove. The resulting relief-like surface, under the impact of an external deforming force, does not show simple linear displacements, such as those produced with an accordion; instead, the edges between the vertices and the centres of the original hexagonal system move in a vortex within each hexagon.

After a lot of cogitation and physical experimentation I came to believe that among the 'right combinations' are the ones shown in Figure 6, which shows half a hexagon with three semi-spidrons. The dotted lines indicate 'valley' folds (paper on either side of the fold, or crease, is bent upwards) and the full lines indicate 'mountain' folds (paper on either side of the crease is bent downwards).

If you want to create a large construct for folding, here is the code for generating the spidron tiling

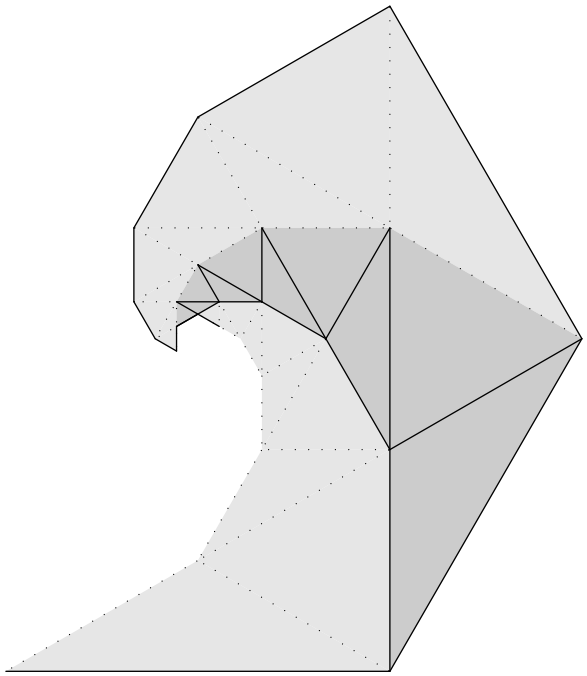


Figure 6: Folding

shown in Figure 4. You can, of course, modify this to meet your needs.

```
% glstr9.mp  MP spidron figures
% earlier pictures
beginfig(5); % spidron tiling
  u := 0.175in;
showhex := false;
showverts := false;
showlines := false;
showcells := false;
rh := false;
showedges := false;  showedges := true;
color cola, colb;
cola := light; colb := dark;
depth := 7;
rad := 2u;
z0 = (0,0);
% fill initial hexagon
for kn := 1 upto 6:
z[kn] = (x0-2u,y0) rotatedaround(z0,60*(kn-1));
if odd kn:
  cola := light;
else:
  cola := dark;
fi
 colb := cola;
semispid(0, [kn], depth, cola, colb, rh);
endfor
% copy (in circles) the filled hexagon
% to make the tiling
shd := (sqrt 3)/2*rad; % shift up/down
shr := 3rad;           % shift left/right
picture pic[];
pic100 := currentpicture;
pic0 := pic100 shifted (0,2shd);
for kn := 1 upto 6:
  pic[kn] := pic0 rotatedaround(z0,60kn);
  draw pic[kn];
endfor
pic10 = pic100 shifted (0,4shd);
for kn := 1 upto 6:
  pic[10+kn] := pic10 rotatedaround(z0,60kn);
  draw pic[10+kn];
endfor
pic20 = pic100 shifted (3rad, 0);
for kn := 1 upto 6:
  pic[20+kn] := pic20 rotatedaround(z0,60kn);
  draw pic[20+kn];
endfor
endfig;
% more pictures
end
```

However, I found that it was difficult enough to properly fold even a single large filled hexagon e.g., one that just fitted onto a typical sheet of paper, such as letter paper or A4. I decided that the best way was to use single spidrons, fold them appropriately, and then hinge them together with sticky tape. I then concluded that it was much more pleasurable to look at pictures of what others had accomplished (most of which, I suspect, were done using computer graphics instead of using physical methods and photographing the results).

References

[1] Dániel Erdély. Spidron system: A flexible space-filling structure. Idea 1979, first presented on the Twelfth International Conference on Crystal Growth in 1998, 2002. Possibly available at `http://www.szinhaz.hu/spidron`.

[2] Branko Grunbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman, 1987.

[3] Ivars Peterson. Swirling seas, crystal balls. *Science News*, 170(17):266–268, 21 October 2006.

[4] Peter Wilson. Glisterings. *TUGboat*, 29(2):324–327, 2008.

◇ Peter Wilson
18912 8th Ave. SW
Normandy Park, WA 98166
USA
herries dot press (at)
earthlink dot net

METAPOST macros for drawing Chinese and Japanese abaci

Denis Roegel

for 荷花

Abstract

This article shows how Chinese (算盘, suànpan) and Japanese abaci (算盤, soroban) can be drawn with METAPOST, and is illustrated with the details of a simple algorithm.

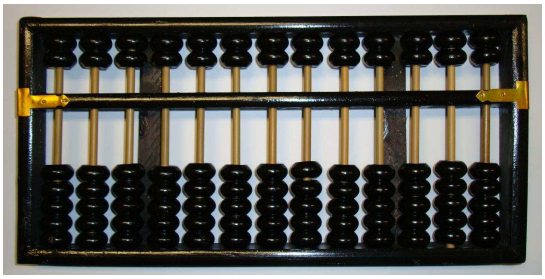


Figure 1: A traditional Chinese abacus (算盘, suànpan) with all its beads set to 0. (Photograph: author's collection)

1 Introduction

One of the oldest calculating tools still in use today is the abacus (Knott, 1886; Smith and Mikami, 1914; Li Shu-T'ien, 1959; Needham and Wang Ling, 1959; Moon, 1971; Ifrah, 2000; Martzloff, 2006). It is now mainly used in Asia for performing arithmetical calculations. Until recently, the use of the abacus was still taught in Chinese schools and there were abacus proficiency tests for applying for certain occupations. In Japan, the first such proficiency test was held in Tokyo in 1928.

An experienced abacist can be very fast, faster than a person using a handheld calculator, at least for small values and basic processes, such as addition or multiplication. Abaci can be used for more advanced tasks, such as extracting a square or cube root, but these tasks may require a non-standard abacus, large enough to store all values.

An abacus is basically a tool to store numerical values by the position of beads on rods. The values which are stored can be changed following an algorithm and an abacist can operate very quickly using automatic patterns which are applied in sequence.

Abaci have a long history and there have been many variants which will not be considered here. The Chinese abacus probably goes back 1000 years or more. Other civilizations, such as Rome and Greece, have used related tools where the stored values were marked by pebbles, or special tokens.

In this article, we describe METAPOST macros to draw the common Asian abaci as well as the operations which are performed on them.

2 Types of abaci

We will consider only two types of abaci, namely the typical current Chinese and Japanese abaci which are still in use.

2.1 The *suànpan*

The Chinese abacus is called 算盘 (*suànpan*). The Chinese word 算 (suàn) means "to calculate" and 盘 (pan) is the word for a "tray". A *suànpan* can come in various widths. The standard *suànpan* has 13 rods with five beads in the lower deck and two beads in the upper one (figure 1). Each bead in the upper deck is worth five beads in the lower one. Four of the lower and one of the upper beads are normally enough for decimal computation, but it seems that the extra beads were originally used to represent an hexadecimal digit, which was useful for the traditional weighing system where one jīn (斤) is equal to sixteen liǎng (两) (about 50 grams). However, these extra beads were also useful to simplify (and accelerate) some computations (Moon, 1971, p. 85).

2.2 The *soroban*

The 算盤 (そろばん, *soroban*) is the Japanese form of the Chinese *suànpan*, and was derived from it. "盤" is the traditional character for "tray", still used in Japan. The basic *soroban* usually also has 13 rods, but there are only four beads in the lower deck and one in the upper deck. In some cases, there are five beads in the lower deck, but only one in the upper deck. A *soroban* has an additional feature which distinguishes it from the *suànpan*, namely that every third rod is marked by a dot. These are the *unit rods*. This makes it easier for calculations and for setting values on the *soroban*.

3 The suanpan METAPOST package

In order to show how to operate an abacus, we have written a METAPOST package to produce simple — but flexible — abaci representations. METAPOST is a powerful graphical tool, well suited for technical or geometrical drawings (Goossens, Mittelbach, Rahtz, Roegel, and Voss, 2008; Hobby, 2008). All the figures in this article were produced with the suanpan METAPOST package, available on CTAN. This package should however be seen only as a basis and it can easily be extended, for instance to vary the shape

of the beads, or to automatically demonstrate more complex algorithms than what we show here.

There are currently two other packages by Alain Delmotte for drawing a *soroban* with PSTricks or PGF, but these packages do not (yet) implement calculation algorithms (Delmotte, 2007a; Delmotte, 2007b).

4 Algorithms on abaci

Calculating on an abacus amounts to resetting the abacus to a standard position, then setting (storing) a value, and then performing some operation, following a known algorithm. The result is then read off the abacus.

4.1 Initial position

Figure 1 shows the initial position of a Chinese *suànpan* and figure 2 compares the Chinese and Japanese abaci. The two decks are divided by a bar known as the reckoning bar. In the standard position, all the beads are moved away from the reckoning bar, and this represents the value 0. Each rod represents one decimal (or sometimes hexadecimal) place, the units being normally at the right. The rods are usually numbered, but this feature can be deactivated using the boolean `rod_numbers` as shown below.

Using the `suanpan` macros, the initial position of a *suànpan* is obtained as follows:

```
input suanpan
setup_abacus(N=13,NBL=5,NBU=2,
             bead="suanpan",units=0);
beginfig(1);
  rod_numbers:=false;
  reset_abacus;draw_abacus;
endfig
end
```

The `setup_abacus` macro sets the number of rods (`N`), as well as the number of beads in each deck (`NBL` and `NBU`), the type of bead (`bead`) and the unit rods (`units`). The arguments are given as *key=value* pairs. Currently two bead types are possible, corresponding to the strings `"suanpan"` (almost round beads) and `"soroban"` (biconal beads).

4.2 Setting a value

Setting a value on an abacus is equivalent to moving some of the beads towards the reckoning bar. A bead from the lower deck represents one unit of the corresponding place, and a bead from the upper deck represents five units. Using four of the lower beads and one upper bead, one can therefore set values up to $5 + 4 = 9$. If all the beads of a Chinese abacus are used, and the upper beads are still weighing 5 lower beads, then the maximum value on a rod is $5 + 5 + 5 = 15$. All values between 0 and 15 can be expressed that way.

In the `suanpan` macros, the number of beads set in each deck is stored in two arrays, and all values of this array can be set by hand as follows (figure 3, left):

```
beginfig(3);
  reset_abacus;
  valL[1]:=2;valL[3]:=5;
  valU[2]:=1;valU[4]:=2;
  draw_abacus;
endfig;
```

Proceeding this way can be useful when the abacus needs to be set in a non-standard decimal position. This is the case above, with one of the rods having 5 beads set in the lower deck. The `suanpan` macros do currently not support hexadecimal computations, but they could easily be handled, based on the implementation for decimal numbers.

In the usual case, at most four beads are set in the lower deck. There is a macro `set_abacus_val` which automates the setting of an initial value (figure 3, right):

```
beginfig(4);
  reset_abacus;
  set_abacus_val("651324");
  draw_abacus;
endfig;
```

If n is the number of rods, only the rightmost n digits of the initial value are taken into account.

4.3 Adding a value

Once a value is stored in the abacus, we can apply simple algorithms to change this value. In this article, we will consider only addition. Even for addition, one can contemplate different methods, and one typical algorithm performs the addition not from right to left, but from left to right. In order to demonstrate the process, the `suanpan` package provides a macro `add_val` which decomposes the addition in a number of steps. This command *should not* be used inside a `beginfig`/`endfig` pair, as it generates a number of such environments. We demonstrate the calculation on a *soroban*, using the initial value 651324 of the above example (figure 4).

```
setup_abacus(N=13,NBL=4,NBU=1,
             bead="soroban",units=1);
set_abacus_val("651324");
add_val(v="82363456",iv=100,fig=true);
```

If the main file is `abacus.mp`, the above command produces files `abacus.100`, `abacus.101`, ..., `abacus.108`, which can then be included in a LaTeX file.

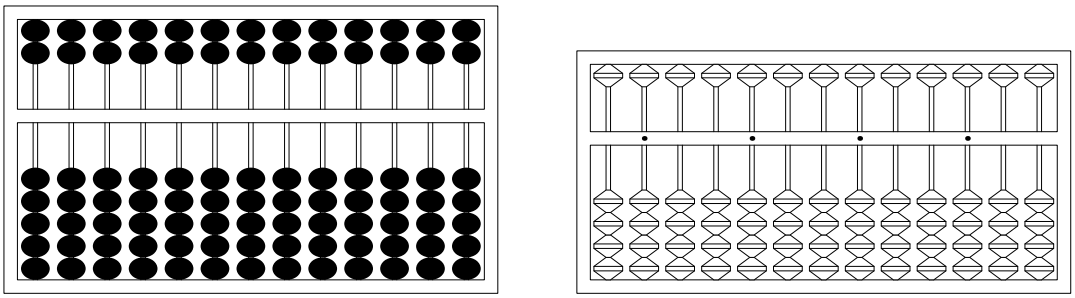


Figure 2: Initial setting of a 算盘 (*suànpan*, left) and of a 算盤 (*soroban*, right).

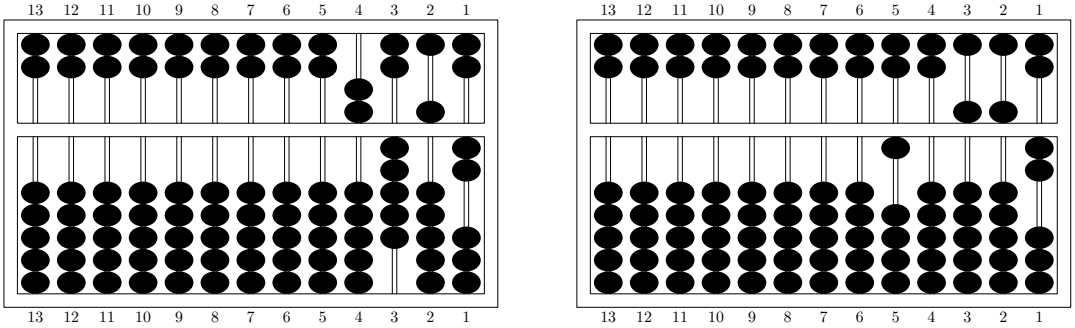


Figure 3: The decimal value 10552 represented in a non-standard way (left) and a standard one (right).

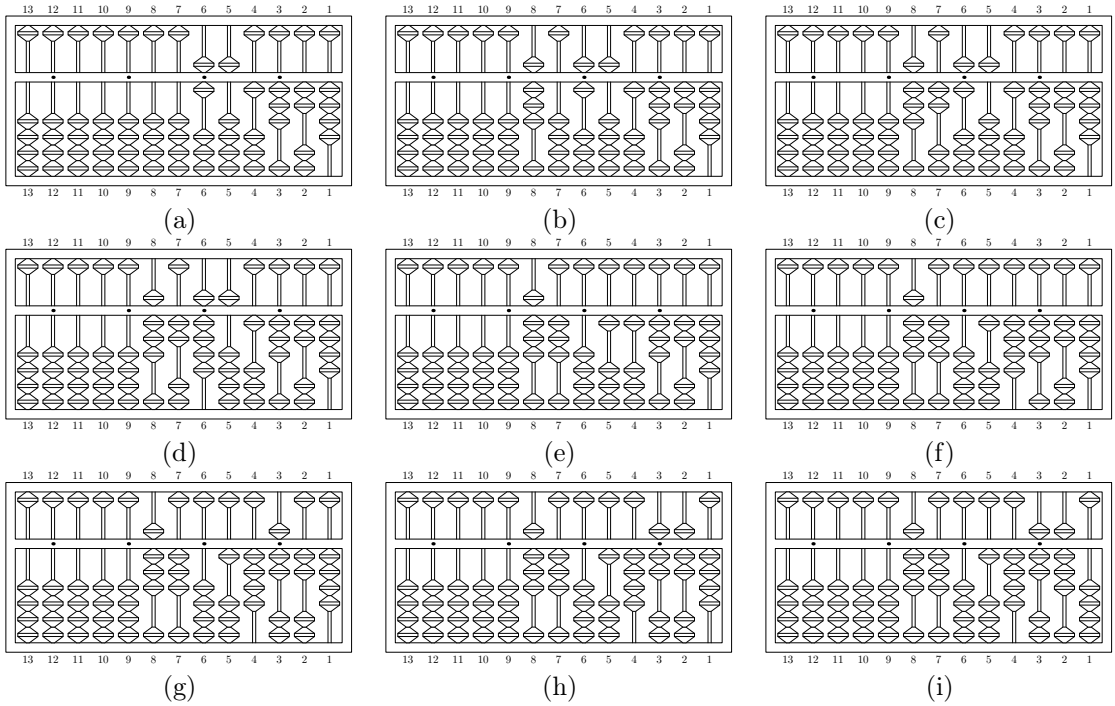


Figure 4: The decomposition of an addition in nine steps on a *soroban*.
(a) represents the initial value 651324, and we add 82363456 in eight steps, one for each digit.

Figure 4 (a) shows the initial state of the abacus, with the value 651324. In a first step (b), we add 8 to rod 8, hence moving three beads from the lower deck and one bead from the upper deck. The other beads are not moved. Then, step (c) adds 2 to rod 7. So far, the changes were straightforward, since these two rods were initially set to 0. In step (d), 3 is added to rod 6, which now contains the value 9. Step (e) adds 6 to rod 5 which contained 5, and this leads to the value 11, hence only 1, and a carry of 1. So, this configuration shows one bead set in the lower deck of rod 5, no bead set in the upper deck, and an additional bead carried to rod 6. However, rod 6 already contained the value 9, and this leads itself to another carry. Finally, it is rod 7 which has an additional bead set in its lower deck. This process goes on digit by digit, until the units have been added. Every single digit addition is therefore sometimes decomposed in multiple steps which are not detailed here. They could be made explicit by other macros.

The `add_val` macro can also be used without generating new drawings, by giving `false` for its `fig` argument. It then only applies the standard addition algorithm and produces the result in the stored arrays.

Of course, additions can also be simulated by doing the computation externally and setting new values for each step. As such, the `suanpan` macros could be used as a back-end for other tools.

Here is for instance an addition not producing any intermediate figures:

```
beginfig(200);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("82951324");
  draw_abacus;
endfig;

beginfig(201);
  set_abacus_val("82951324");
  add_val(v="60000",iv=100,fig=false);
  draw_abacus;
endfig;
```

An addition can produce an overflow, and this sets the `overflow` boolean to `true`. The `add_val` macro resets this value to `false` before performing the addition.

4.4 Shortcuts for fast computation

In order to become proficient with the abacus, it is useful to memorize a number of patterns which recur very frequently and which enable automating much of the computation. A simple example will show what is meant.

If one of the rods of the abacus has three beads set in the lower deck, and one more bead has to be set, then this additional bead can merely be moved towards the reckoning bar. However, if three units had to be added instead of one, then a novice user of the abacus would probably mentally compute $3 + 3 = 6$, then remove 5 and set only one bead in the lower deck, while adding one too in the upper deck. However, this is inefficient, because the burden of the computation is on the user. Instead, if three beads cannot be moved, one should consider $3 = 5 - 2$ and therefore perform two operations: *adding* one (5) to the upper deck, and *removing* 2 from the lower deck. This is a typical shortcut, which does not require the calculation of $3 + 3 = 6$, and only requires to notice that three more beads cannot be set in the lower deck.

Some of the operations in the upper deck may also be impossible, and may require similar rewritings. If one bead (5) cannot be added in the upper deck part, we can instead write $5 = 10 - 5$ and add one (10) to the units of the next rod, and remove one bead from the upper deck of the current rod. This process then repeats until the computations have been entirely performed.

If five or more beads have to be added in the lower deck, the number of beads to be added can be written either as $5 + a$ or as $10 - b$, and whichever is possible must then be applied. For instance, if we still have three beads in the lower deck, and if we have to add six beads, we can write either $6 = 5+1$ or $6 = 10-4$. The second decomposition cannot be performed, because it amounts to removing four beads from the lower deck. But the first decomposition is possible, and so we set one more bead in the lower part, as well as one more bead in the upper deck. If the latter is not possible, we again decompose the calculation.

More complex operations, such as multiplications, divisions, square roots, etc., can be performed efficiently using tables that the abacist has to memorize. Examples of such tables for the *soroban* are given by Knott (Knott, 1886).

5 Special abaci macros

In order to explain how to operate an abacus, it is sometimes useful to mark some of the beads. Two possibilities are provided by the `suanpan` package: some of the beads can be shown in gray, or they can be marked with a label.

Using the macro `set_abacus_gray`, it is easy to put some of the beads in gray. This macro takes three arguments, given as *key=value* pairs. The `deck` key identifies the deck (lower or upper), and the other

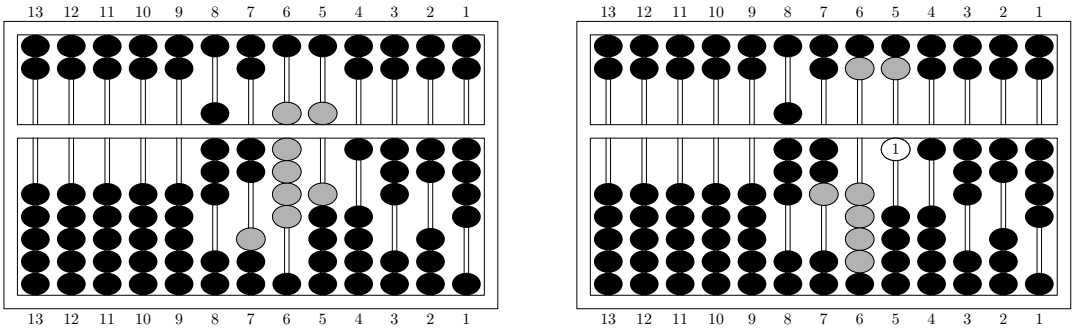


Figure 5: Highlighting one of the steps of the addition, with special marks. All the beads which have been moved have been drawn in gray, and, in addition, the sole bead moved from the lower deck of the fifth rod has been marked with '1'.

two are strings with one digit for consecutive rods starting from the right. The key `below` corresponds to the beads which are in the lowest positions in a deck. If the value is 2, for instance, it means that the two *top* beads in the lower part of the deck (upper or lower) will be grayed. These are the first beads that would be moved if two beads had to be set (in the lower deck) or reset (in the upper deck).

There is currently no automated way to produce these special marks, but their automation is of course possible. There are however so many different imaginable schemes, that we have decided not to implement them for the moment. Only low-level commands are currently supported.

We illustrate these commands by considering again the addition seen previously, but this time marking all the changes. The resulting configurations are shown in figure 5 and the code which produces them is the following:

```
beginfig(202);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("82951324");
  set_abacus_gray(deck="lower",
     below="1010000",above="0400000");
  set_abacus_gray(deck="upper",
     below="0110000",above="0000000");
  draw_abacus;
endfig;

beginfig(203);
  reset_abacus_gray;
  add_val(v="60000",iv=100,fig=false);
  set_abacus_gray(deck="lower",
     below="0400000",above="1010000");
  set_abacus_gray(deck="upper",
     below="0000000",above="0110000");
  draw_abacus;
  mark_abacus(5,5)(btex 1 etex);
endfig;
```

In these examples, `reset_abacus_gray` merely resets all the grayed beads. It will be easy to see how the gray encoding translates to the figures.

The other macro to mark beads is `mark_abacus`. This macro overwrites a bead with a (short) label. `mark_abacus(3,5)(btex 1 etex)` writes '1' over the fifth bead (from the bottom) in the third rod from the right. One of the advantages of this encoding is that even if a bead is moved, the mark will still remain on it and the command will not need to be altered.

6 Abaci in other bases

As was explained before, the Chinese abacus can be used both for computing with decimal values and with hexadecimal values, depending on the use of the two extra beads in the lower and upper decks. We can imagine other abaci, adapted to other bases.

Figure 6, for instance, shows an addition with a base-8 abacus. Each rod has three beads in the lower deck and one bead in the upper deck, and a rod can hold values from 0 to 7. The figure on the left represents the value 3401256_8. Adding 1234_8, we get 3402512_8.

Producing these drawings is straightforward: in addition to the number of beads in each deck, there is a variable `vbu` representing the value of one bead in the upper deck. In a natural base-8 abacus, the upper beads would be worth 4 lower beads, and so, we can just write the following to produce the two configurations, before and after the addition. Similar constructions are possible in other bases, but experienced users would have to adapt all their mnemonic rules to fit these new configurations.

```
vbu:=4; % upper deck value of a bead
setup_abacus(N=13,NBL=3,NBU=1,
             bead="suanpan",units=0);
```

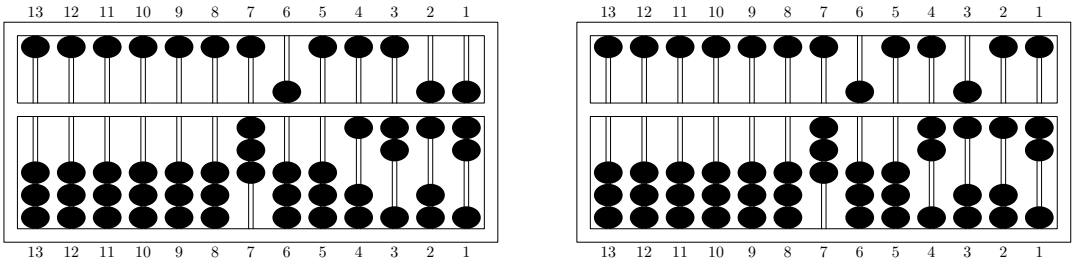


Figure 6: Base-8 abacus addition. The value on the left is 3401256_8 and we add 1234_8, which produces 3402512_8 on the right.

```
beginfig(300);
  reset_abacus;reset_abacus_gray;
  set_abacus_val("3401256");
  draw_abacus;
endfig;

beginfig(301);
  add_val(v="1234",iv=100,fig=false);
  draw_abacus;
endfig;
```

7 Conclusion and future extensions

This article is meant as an illustration of some simple METAPOST macros for drawing Chinese or Japanese abaci, and we have only strived to provide a good foundation. Many improvements are possible, both graphically and algorithmically. The abaci can be made more realistic, and in particular other bead shapes could be supported. The main possible improvements, however, concern the implementation of new algorithms. So far, we have only concentrated on the implementation of one addition algorithm, but other algorithms are possible, for instance one where the additions are performed from the right-most rod to the left-most one. More complex operations, such as multiplication, division, the calculation of square or cube roots, etc., could also be supported (Knott, 1886; Kojima, 1963; Moon, 1971; Heffelfinger and Flom, 2007). For each of these cases, it would be desirable to provide automatic output detailing each algorithm. This could easily be built upon the existing macros.

References

Delmotte, Alain. "Soroban abacus: package `pgf-soroban`". 2007a. Available on CTAN.

Delmotte, Alain. "Soroban abacus: package `pst-soroban`". 2007b. Available on CTAN.

Goossens, Michel, F. Mittelbach, S. Rahtz, D. Roegel, and H. Voss. *The LaTeX Graphics Companion, Second Edition*. Boston: Addison-Wesley, 2008.

Heffelfinger, Totton, and G. Flom. "算盤 Abacus: Mystery of the Bead". 2007. `http://webhome.idirect.com/\~{}totton/abacus`.

Hobby, John. "METAPOST: A User's Manual". 2008. Updated version of the original manual; available at `http://tug.org/docs/metapost/mpman.pdf`.

Ifrah, Georges. *The Universal History of Computing: From the Abacus to the Quantum Computer*. New York: John Wiley, 2000.

Knott, Cargill Gilston. "The Abacus in its Historic and Scientific Aspects". *Transactions of the Asiatic Society of Japan* **14**, 18–71, 1886.

Kojima, Takashi. *Advanced Abacus: Japanese Theory & Practice*. Tokyo: Charles E. Tuttle & Company, 1963.

Li Shu-T'ien. "Origin and Development of the Chinese Abacus". *Journal of the ACM* **6**(1), 102–110, 1959.

Martzloff, Jean-Claude. *A history of Chinese Mathematics*. New York: Springer, 2006.

Moon, Parry. *The Abacus: Its history; its design; its possibilities in the modern world*. New York: Gordon and Breach Science Publishers, 1971.

Needham, Joseph, and Wang Ling. *Science and Civilisation in China, volume 3: Mathematics and the Sciences of the Heavens and Earth*. Cambridge: Cambridge University Press, 1959.

Smith, David Eugene, and Y. Mikami. *A history of Japanese mathematics*. Chicago: The Open Court Publishing Company, 1914.

◇ Denis Roegel
LORIA — BP 239
54506 Vandœuvre-lès-Nancy cedex
France
`roegel (at) loria dot fr`
`http://www.loria.fr/~roegel`

Spheres, great circles and parallels*

Denis Roegel

Abstract

Each domain has its graphical archetypes. In particular, spheres are unavoidable components of domains such as geography or astronomy. However, when perusing a number of publications, we noticed that spheres were often incorrectly drawn with respect to their features such as great circles and parallels. This article examines several simple METAPOST techniques that remedy these problems.

1 Introduction

The spheres and their components (great circles, meridians, parallels) make up the typical illustrations in certain fields such as geography or astronomy. For instance, the motion of the Sun in the sky will often be represented as a sphere with the celestial equator, the ecliptic and the apparent path of the Sun on this sphere. In certain fields, spheres illustrate projections, be it in cartography, gnomonics, or elsewhere. The representations of spheres in publications are themselves projections.

Here we examine the simplest case: spheres represented in parallel projection on a plane. In that case, the projection is done along parallel lines. We will also assume, for simplification, that the projection plane is orthogonal to the projection direction, although part of our conclusions are independent of this assumption.

More precisely, the problem we consider is that of drawing a sphere, with an equator, meridians, other great circles, parallels, all of them with correct dashed lines.

In order to get a good understanding of the possible difficulties of this task, it is useful to review the general principles of the projections which are commonly used.

2 Projections

The main projections are illustrated in figure 1. We have represented the projections of the equator, of the North pole and of one of the points whose projection follows a line which is tangent to the sphere.

3 How the problem is handled in the literature

A perusal of the literature, be it on paper or the Internet, is a source of surprises. Assuming that the projections are done on a plane and either along parallels or in a perspective manner, two totally natural assumptions, it appears that the majority of the books consulted represent the spheres in a contradictory way.

The problems are all confined to figures which have not been drawn by projection. For instance, aside from the fact that many of these figures do not represent the projected circles as ellipses, the problems displayed in most of the printed figures concern the position of certain points, in particular the poles. For instance, in the case of the projections of figure 1, when the equator is transformed in an ellipse, the poles should not be positioned at the periphery of the projected sphere, but this is unfortunately often the case on the printed representations.

To support our claim, we give a list of a few books where the spheres are problematic, with a page example, which will allow the interested reader to locate them:

- W. M. Smart: *Celestial Mechanics*, New York: Longmans, 1953, p. 24.
- Derek J. Price: *The equatorie of the planetis*, Cambridge: the University press, 1955, p. 96.
- John D. North: *Richard of Wallingford*, Oxford: Clarendon Press, 1976, vol. 3, p. 152.
- René R. J. Rohr: *Sundials: History, Theory, and Practice*, New York: Dover Publications, 1996, p. 25.
- Gianni Pascoli: *Éléments de mécanique céleste*, Paris: Masson, 1997, p. 12.
- Raymond d'Hollander: *L'astrolabe : histoire, théorie et pratique*, Paris: Institut océanographique, 1999, p. 26.
- Denis Savoie: *La gnomonique*, Paris: Les Belles Lettres, 2001, p. 44.
- Denis Savoie: *Cosmographie*, Paris: Belin-Pour la science, 2006, p. 17.

That said, some books take care not to put the poles on the limit circle, and this is in particular the case in Otto Neugebauer's classic *A History of Ancient Mathematical Astronomy*, New York: Springer, 1975, p. 1408.

A number of web sites are also faulty, for instance those of the Paris-Meudon observatory or of the *Institut de Mécanique Céleste et de Calcul des Éphémérides* (`http://www.imcce.fr`) which display objectionable representations.

The reasons for perpetuating these errors are not totally clear; it seems that it is a certain habit, perhaps a kind of laziness, and — in some cases — the result of the subcontracting of figures by the authors.

* Translation of "Sphères, grands cercles et parallèles," *Les Cahiers GUTenberg*, number 48, April 2007, pages 7–22. Reprinted with permission.

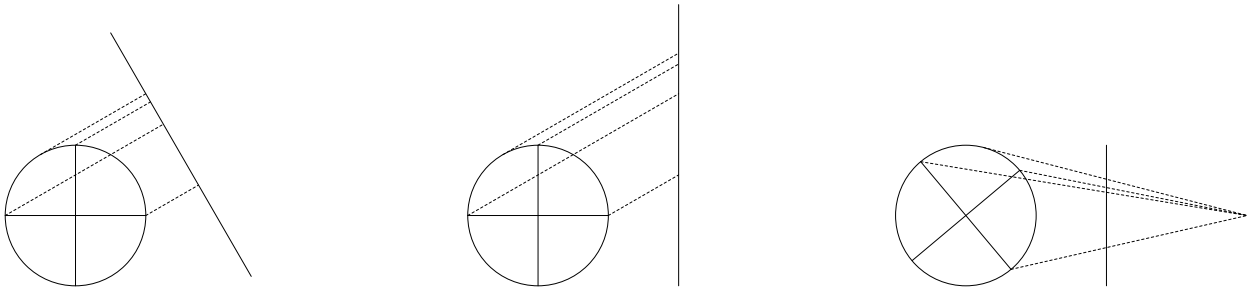


Figure 1: Orthogonal (left), oblique (center) and perspective (right) projections on a vertical plane.

4 A METAPOST approach

Although our application is very simple, it doesn't seem to have been handled with the METAPOST software, or with other graphical TEX tools such as `PSTricks`. The extensions of the latter system already provide a number of facilities for the representation of 3-dimensional objects, but the representation of objects in space obscures the hidden parts by overlaying them and therefore doesn't involve the computation of boundaries between visible and invisible parts.

One of the difficulties of the representation of spheres is related to dashed lines. Dashed lines are traditionally used for representing the hidden parts. It is therefore necessary to ensure that these lines start and end at the right places, and this task usually requires the computation of intersections.

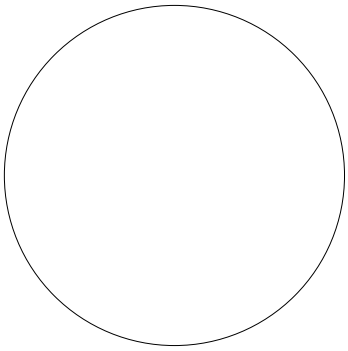
It is when designing a figure for a lecture in astronomy that we have, in the first place, made the same error as that of our predecessors; the reflex of "poles on the circle" was rooted in our habits. Figure 2 represents these first attempts, typical of the figures which are found almost everywhere. Figure 3 illustrates how the spheres should have been represented. The positions of the poles are here computed in an exact way, for the poles of the equator (N and S) as well as for those of the ecliptic (N^* and S^*). Moreover, the angle between the planes of the equator and the ecliptic is also correctly displayed ($23.5°$). In the case of the lunar orbit, however, we have intentionally increased the angle between that orbit and the plane of the ecliptic.

We will now describe how the correct figures were obtained, and we will restrict ourselves to the case of orthogonal projections. Our constructions will be in METAPOST, but nothing prevents the transposition of our techniques to other languages.[1]

4.1 The projection of the sphere

The orthogonal projection of the sphere is a circle whose diameter is that of the sphere. We will assume for simplification that the circle is centered at the origin.

```
r=5cm;draw fullcircle scaled 2r;
```



4.2 Definition of vectors

In order to precisely control the projection, we first define a vector type. METAPOST does not provide such a type, but it has a `color` type with three numerical components which we disguise as a vector. Accessing the components of the vectors is done with `Xp`, `Yp` and `Zp`. We then define a few elementary operations on these vectors, like the dot product (`dotproduct`), the vector product (`vecproduct`) and the construction of a unit vector.

```
let vector=color;
let Xp=redpart; let Yp=greenpart; let Zp=bluepart;

def dotproduct(expr Vi,Vj)=
  (Xp(Vi)*Xp(Vj)+Yp(Vi)*Yp(Vj)+Zp(Vi)*Zp(Vj))
enddef;

def vecproduct(expr Vi,Vj)=
  (Yp(Vi)*Zp(Vj)-Zp(Vi)*Yp(Vj),
   Zp(Vi)*Xp(Vj)-Xp(Vi)*Zp(Vj),
   Xp(Vi)*Yp(Vj)-Yp(Vi)*Xp(Vj))
```

[1] For an introduction to METAPOST, one can readily consult various tutorials on the web, the documentation available in most TEX distributions, or the second edition of the *LATEX Graphics Companion*.

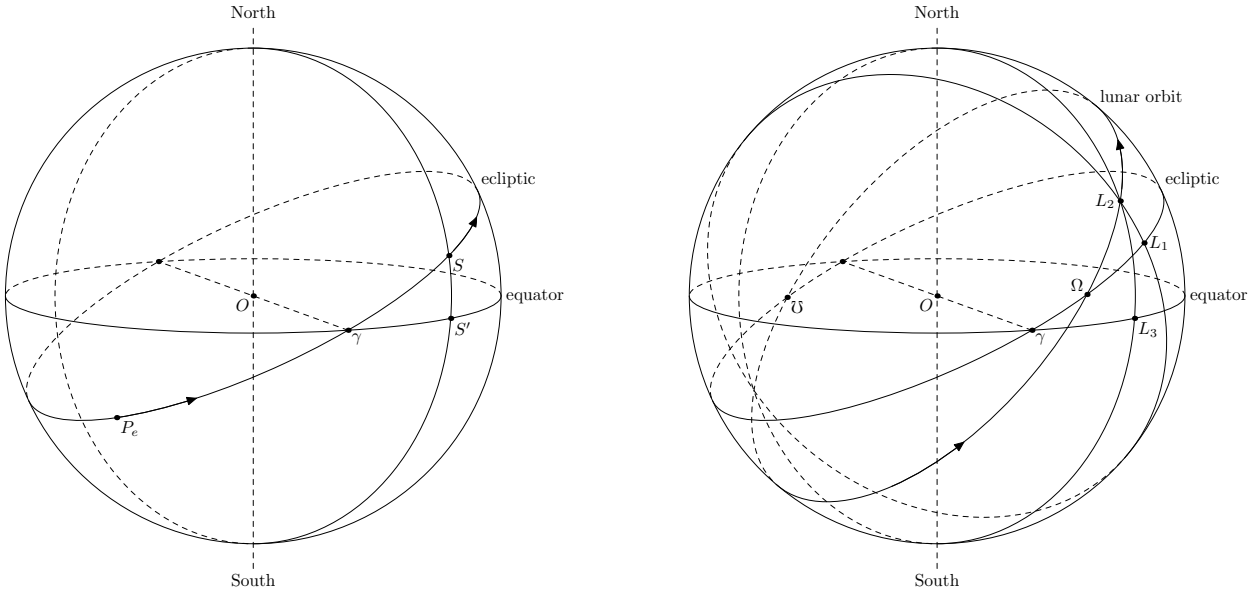


Figure 2: Two sphere drawings violating the properties of parallel projections on a plane. The poles are here put at the periphery of the spheres, although they should be located slightly inside of the spheres, given the angle under which the plane of the equator is seen.

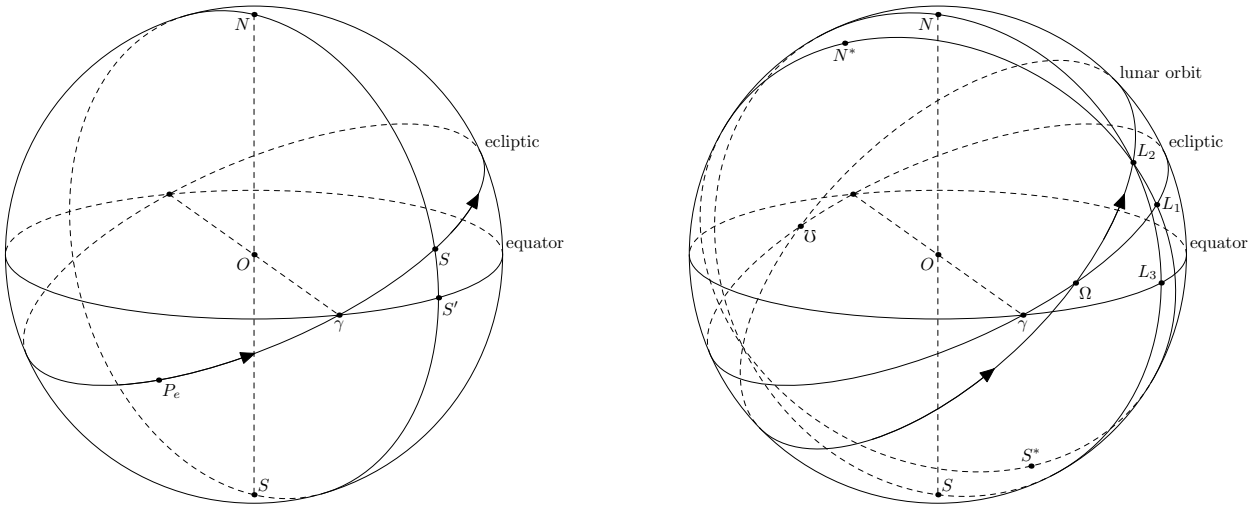


Figure 3: Two correct drawings of the planes of the equator and of the ecliptic, of the poles and of the meridians. The inclination of the lunar orbit has intentionally been magnified.

```
enddef;

def norm(expr V)= sqrt(dotproduct(V,V)) enddef;
def normed(expr V)= (V/norm(V)) enddef;
```

4.3 Orientation in space

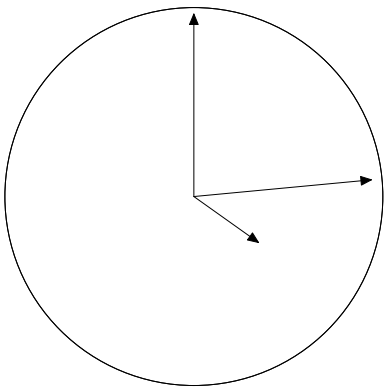
Before performing the projection, the sphere is oriented in space. More precisely, we construct three vectors $\overrightarrow{V_1}$, $\overrightarrow{V_2}$, $\overrightarrow{V_3}$ using the vectors of the orthonormal basis. We employ only two angles, and in that manner we maintain the vertical character of the projection of one of the vectors. θ is the angle by which $\vec{\imath}$ is rotated around $\vec{k}$, which produces $\overrightarrow{V_1}$. ϕ is the angle by which $\vec{k}$ is rotated around $\overrightarrow{V_1}$, which produces $\overrightarrow{V_2}$. $\overrightarrow{V_3}$ is the vector product of $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ and is oriented towards the observer. Finally, $\overrightarrow{V_1}$ represents the vector of the projection plane directed towards the right and $\overrightarrow{V_2}$ the one directed towards the top. The figures in the sequel were obtained with $\theta = 70$ and $\phi = -15$.

```
vector V[]; % vector array
theta=70;phi=-15;
V1=(cosd theta,sind theta,0);
V2=(sind(phi)*sind(theta),
    -sind(phi)*cosd(theta),cosd(phi));
V3=vecproduct(V1,V2);
```

4.4 The projection

The projection itself is very simple to achieve, as it is sufficient to determine the components of a vector in space in the $(\overrightarrow{V_1}, \overrightarrow{V_2}, \overrightarrow{V_3})$ base, something which is immediate with the dot product. Only the first two components are of interest to us, since $\overrightarrow{V_3}$ is parallel to the projection direction. A `project` function allows us to write this projection naturally, and this function therefore doesn't use the third vector:

```
def project(expr V,Va,Vb)=
  (dotproduct(V,Va),dotproduct(V,Vb))
enddef;
z0=(0,0);
z1=project((r,0,0),V1,V2);
z2=project((0,r,0),V1,V2);
z3=project((0,0,r),V1,V2);
drawarrow z0--z1;drawarrow z0--z2;
drawarrow z0--z3;
```

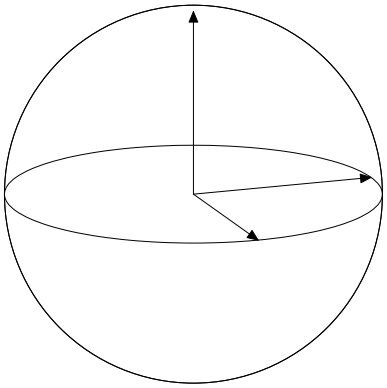


4.5 Construction of the equator

We can now draw a great circle, for instance the circle of the equator. Its equation is very simple: it is the set of points $(r\cos t, r\sin t, 0)$ for $0 \leq t < 360$, t being expressed in degrees. The `f_equ` macro corresponds to this expression and the projected curve is obtained by connecting the projections of points at regular intervals, here from 10 to 350 degrees.

```
def f_equ(expr r,t)=(r*cosd(t),r*sind(t),0) enddef;

path equator; equator=
  project(f_equ(r,0),V1,V2)
  for t=10 step 10 until 350:
   .. project(f_equ(r,t),V1,V2)
  endfor .. cycle;
draw equator withcolor blue;
```

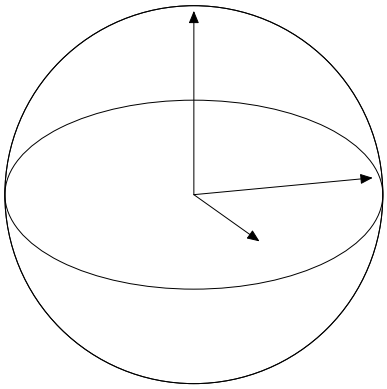


4.6 Simplification of the equator

The equator is now represented by a curve constructed from a large number of points. However, this curve should be an ellipse and we can obtain a very good approximation of it by constructing it using `fullcircle` instead. (It is only an approximation since `fullcircle` is not exactly a circle.)

The construction of an ellipse from a circle is done as follows, using the semi-major axis, the semi-minor axis and the angle of the ellipse. The correct drawing of the ellipse requires the knowledge of its two axes, which are not yet known in the above construction.

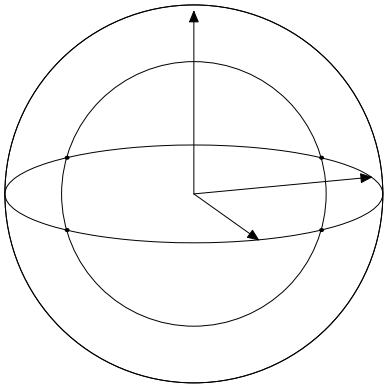
```
def ellipse(expr ra,rb,an)=
  (fullcircle xscaled 2ra yscaled 2rb rotated an)
enddef;
draw ellipse(r,.5r,0);
```



4.7 Determination of the elements of the ellipse

In order to obtain the elements of the ellipse (axes and orientation), the projection parameters can be used, or we can merely measure these elements on the ellipse as constructed pointwise. This can be done as follows:

- first, a circle is superimposed to the ellipse;
- the four intersections of this circle with the ellipse are determined (this may require the circle to be resized);
- the intersections easily provide the directions of the axes;
- these axes are then measured;
- finally, the ellipse is constructed in a more economical way.



We will now examine in more detail how this procedure is realized.

4.7.1 Orientation of the ellipse

In order to determine the orientation of the ellipse, we make use of the `ellipse_major_angle` macro below, which takes a path `p` representing an ellipse of semi-major axis `a` centered at the origin. A simple dichotomy looks for a half circle of radius `rc` with a non-void intersection with the ellipse. Then, two intersections (`pi1`, `pi2`) are obtained with the help of `intersectionpoint`, by carefully splitting the half-circle. By symmetry, these two intersections give two other intersections (`pi3`, `pi4`).

The orientation of the ellipse is obtained by locating two intersections, `pi5` and `pi6`. One of these intersections is with the major axis, the other with the minor axis.

```
vardef ellipse_major_angle(expr p,a)=
  save pa,pc,pi,ra,rb,rc,an;
  path pc[];pair pa,pi[];ra=.5a;rb=a;
  forever: %======== dichotomy ==========
    rc:=.5[ra,rb];
    pc0:=subpath(0,4) of fullcircle scaled 2rc;
    pa:=pc0 intersectiontimes p;
    exitif pa<>(-1,-1);ra:=rc;
  endfor;
  %=== computation of two intersections ===
  pi1=p intersectiontimes pc0;
  pc1=subpath(0,ypart(pi1)-0.01) of pc0;
  pc2=subpath(ypart(pi1)+0.01,length(pc0)) of pc0;
  pi1:=p intersectionpoint pc0;
  pi2:=p intersectiontimes pc1;
  if pi2=(-1,-1):
    pi2:=p intersectionpoint pc2;
  else:
    pi2:=p intersectionpoint pc1;
  fi;
  pi3=pi1 rotated 180;
  pi4=pi2 rotated 180; % other intersections
  %======= orientation ======
  pi5=p intersectionpoint
        (origin--(unitvector(pi2-pi1)*2a));
  pi6=p intersectionpoint
        (origin--(unitvector(pi1-pi4)*2a));
  if arclength(origin--pi5)>arclength(origin--pi6):
    an=angle(pi1-pi2);
  else:
    an=angle(pi1-pi4);
  fi;
  an % result of the macro
enddef;
```

4.7.2 The minor axis of the ellipse

The `ellipse_minor_axis` macro takes a path `p` representing an ellipse of semi-major axis `a` centered at the origin, and whose major axis is oriented according to the angle `an`. The macro merely determines the intersection of `p` and a line located at a right angle to the major axis and measures its distance from the center of the ellipse.

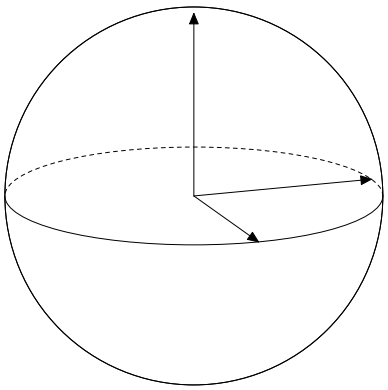
```
vardef ellipse_minor_axis(expr p,a,an)=
  save pa;pair pa;
  pa=p intersectionpoint (origin--(dir(an+90)*2a));
  arclength(origin--pa) % result
enddef;
```

These two macros make it therefore possible to determine all the parameters necessary to the economical drawing (that is, not pointwise) of an ellipse.

4.8 The dashes on the equator

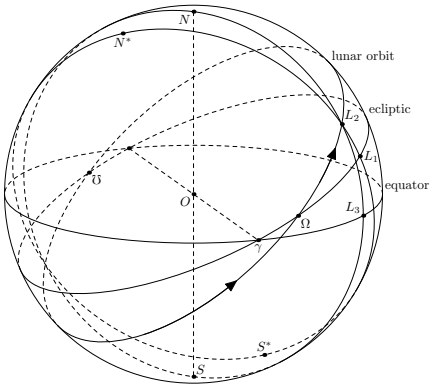
The dashes on the equator correspond to one half of the ellipse and the two halves are joined by the major axis. It is therefore sufficient to cut the ellipse in two parts and draw one in plain lines, the other in dashed lines. The ellipse returned by the `ellipse` macro is a parametric curve where the parameter goes from 0 to 8 (the base circle contains eight points), 0 being on the major axis, and the paths from 0 to 4 and from 4 to 8 are excerpted from it.

```
path pa,pb,pc;
pa=ellipse(r,rb,0);
pb=subpath(0,4) of pa;
pc=subpath(4,8) of pa;
draw pb dashed evenly; % hidden
draw pc; % visible
```



4.9 Great circles

The same principle is used for all the great circles. The only difficulty is the determination of an equation for these great circles. The macros used are parameterized in order to be able to choose which of the two parts is dashed.



Some of the circles are determined by certain constraints. For instance, in the above drawing, we were given point L_1 on the ecliptic, then the ecliptic meridian going through L_1 was constructed, leading to the determination of point L_2 on the lunar orbit. These intersections were obtained by the intersection of projections, but the intersection in space was then found again using the knowledge of the curves. Finally, the equatorial meridian going through L_2 was drawn, making it possible to obtain L_3.

4.9.1 Constraints

We can easily take such constraints into account by using the `rotatearound` macro which rotates a vector around another one.

```
% rotates Va around Vb by the angle 'a'
vardef rotatearound(expr Va,Vb,a)=
  save v;vector v[];
  v0=normed(Vb);v1=dotproduct(Va,v0)*v0;
  v2=Va-v1;v3=vecproduct(v0,v2);
  v4=v2*cosd(a)+v3*sind(a)+v1;
  v4 % result
enddef;
```

Therefore, for the case of the curve representing the ecliptic, whose equation is determined by the function

```
def f_ecliptic(expr t)=
  (a*(cosd(t),sind(t)*cosd(ec_angle),
      sind(t)*sind(ec_angle)))
enddef;
```

where `ec_angle` is the obliquity of the ecliptic plane (23.5°), we begin by determining the North pole (N^*) of the ecliptic, assuming $\gamma = (1, 0, 0)$:

```
vector North,North_Ec;North=a*(0,0,1);
North_Ec=rotatearound(North,(1,0,0),ec_angle);
```

Since point L_1 is chosen on the ecliptic, the meridian going through L_1 and N^* is determined by the two vectors $\overrightarrow{ON^*}$ and $\overrightarrow{OL_1}$, each point of the meridian being obtained by the rotation of $\overrightarrow{OL_1}$ around a vector orthogonal to $\overrightarrow{OL_1}$ and $\overrightarrow{ON^*}$. The following macro, parameterized by the point A (in space) on the ecliptic and an angle t, makes it possible to describe this meridian:

```
def f_ec_meridian(expr t,A)=
  (A*cosd(t)+North_Ec*sind(t))
enddef;
```

This function is then used to define the projected path `ec_meridian`, using `project`, as we did above when defining the `equator` path.

4.9.2 Inverse projection

The principle of the "inverse projection" is very simple and we will only sketch it. For instance, in order to determine L_2 from L_1 in the previous figure, we have on the one hand constructed the great circle going through L_1 and N^* as explained above (`ec_meridian`), and on the other hand the lunar orbit (`moon`) applying analogous principles. The intersection of these two projected curves was computed in the usual way:

```
Lp2=moon intersectionpoint ec_meridian;
```

Here, the `intersectionpoint` macro was assumed to return the correct intersection, which is not always the case.

Now, point L_2 in space is a linear combination determined by two vectors forming a basis of the plane of the lunar orbit. These two vectors can be determined by the equation of the lunar orbit and we call them `moon_x` and `moon_y`. We have therefore:

```
L2=m_x*moon_x+m_y*moon_y;
```

where `m_x` and `m_y` are scalar values. These unknowns can be determined by projecting the above equation, because a parallel projection is a linear transformation:

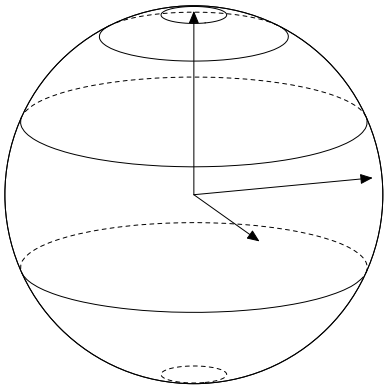
```
Lp2=m_x*project(moon_x,V1,V2)
   +m_y*project(moon_y,V1,V2);
```

The latter equation defines `m_x` and `m_y` from `Lp2` (point of the plane) and therefore defines at the same time `L2` (point in space).

Once L_2 is known, we are able to use it to obtain L_3 in an analogous way.

4.10 Parallels

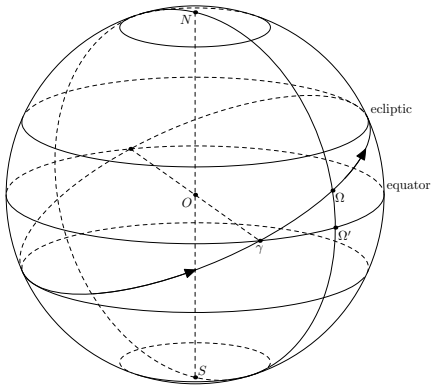
The case of the great circles was relatively simple, because these circles were always half visible and half invisible, the limit of visibility being on the major axis of the ellipse. This is not the case for the other circles of the spheres. We examine here only the case of the parallels to the equator.



Parallels have a number of distinctive features: they do not necessarily have as much visible as they have hidden; they can be totally visible or totally hidden; they have a visible/hidden limit which is not on the major axis.

In order to draw the parallels correctly, it is necessary to determine the limits between the visible part and the hidden part of a parallel.

The limits of visibility are determined by the intersection between the plane orthogonal to the viewing direction ($\overrightarrow{V_3}$), and the circle representing the parallel. This intersection can consist in zero points (the parallel is then totally visible, or totally hidden), two points (there is both a hidden and a visible part), or one point (this is the limit case between the two previous cases).



Once the intersections are obtained in space, they are converted into angles and the two arcs are drawn separately using these angles. The macro `draw_parallel` is defined in figure 4.

The equation of a parallel at latitude ϕ will be the following:

```
def f_parallel(expr r,theta,phi)=
  (r*cosd(phi)*cosd(theta),
   r*cosd(phi)*sind(theta),r*sind(phi))
enddef;
```

5 Conclusion

Having observed that many spheres were not correctly represented in the literature, we have analyzed the problem in detail and have written a few METAPOST commands to produce correct drawings. We now only hope that this work will contribute, even indirectly, to an improvement of the realism of the spheres in the way they are employed in cosmography and elsewhere.

Moreover, it seems interesting to extend other graphical packages with such functionalities, always for the purpose of fostering their use. The `PSTricks` package might benefit from such an extension, which would moreover allow for a comparison with our own implementation.

◇ Denis Roegel
LORIA — BP 239
54506 Vandœuvre-lès-Nancy cedex
France
roegel (at) loria dot fr
http://www.loria.fr/~roegel

```
% phi=latitude, col=color, side=1 or -1 depending on the dashes
vardef draw_parallel(expr phi,col,side)=
  save p;path p[];p0=project(f_parallel(a,0,phi),V1,V2)
  for t=0 step 10 until 360 :..project(f_parallel(a,t,phi),V1,V2) endfor;
  % we now search for the intersections of this parallel
  % with the projection plane:
  % plane:      V3x*x+V3y*y+V3z*z=0
  % parallel: x=r*cos(phi)*cos(theta), y=r*cos(phi)*sin(theta), z=r*sin(phi)
  % we search theta:
  save A,B,C,X,Y,ca,cb,cc,delta,nx,tha,thb;
  numeric X[],Y[];ca=Xp(V3);cb=Yp(V3);cc=Zp(V3);
  if cb=0:X1=-(cc/ca)*sind(phi)/cos(phi);nx=1;
  else:
    A=1+(ca/cb)**2;B=2*ca*cc*sind(phi)/(cb*cb);
    C=((cc/cb)*sind(phi))**2-cosd(phi)*cosd(phi);delta=B*B-4A*C;
    if delta<0:nx=0;% no intersection
    else:
      X1=((-B-sqrt(delta))/(2A))/cosd(phi); % = cos(theta)
      X2=((-B+sqrt(delta))/(2A))/cosd(phi); % = cos(theta)
      Y1=-((ca*X1+cc*sind(phi)/cosd(phi))/cb); % = sin(theta)
      Y2=-((ca*X2+cc*sind(phi)/cosd(phi))/cb); % = sin(theta)
      tha=angle(X1,Y1);thb=angle(X2,Y2);nx=2;
    fi;
  fi;
  if nx=0: % totally (in)visible parallel
    if side=1:draw p0 withcolor col;
    else:draw p0 withcolor col dashed evenly;fi;
    message "NO INTERSECTION";
  elseif nx=1:X10=angle(X1,1+-+X1);X11=360-X10;
  else: % general case
    if tha<thb:X10=tha;X11=thb;else:X10=thb;X11=tha;fi;
  fi;
  if nx>0: % determination of the two paths
    p1=project(f_parallel(a,X10,phi),V1,V2)
    for t=X10+1 step 10 until X11:..project(f_parallel(a,t,phi),V1,V2)
    endfor;
    p2=project(f_parallel(a,X11,phi),V1,V2)
    for t=X11+1 step 10 until X10+360:..project(f_parallel(a,t,phi),V1,V2)
    endfor;
    % drawing the two paths
    if side=1:draw p1 withcolor col;
    else:draw p1 withcolor col dashed evenly;fi;
    if side=1:draw p2 withcolor col dashed evenly;
    else:draw p2 withcolor col;fi;
  fi;
enddef;
```

Figure 4: Code for drawing a circle parallel to the equator.

An introduction to nomography: Garrigues' nomogram for the computation of Easter

Denis Roegel

Abstract

This article analyzes a calendrical nomogram for the determination of the date of (Julian or Gregorian) Easter, and shows how it can be reproduced with METAPOST.

1 Introduction

The field of nomography is ancient, and is related to slide rules. The object of nomography is to study the graphical representation of equations with n unknowns, in order to construct graphical tables representing mathematical laws of which these equations are the analytical expression. These tables are called "nomograms" and can be used to obtain one of the n values given the values of the $n-1$ other unknowns.

The art of nomography was developed extensively by Maurice d'Ocagne (1862–1938), from 1884 onwards. In his 1921 treatise on the subject, he mentioned a nomogram for the calendar, as well as the unpublished work of André Crépin on one for finding Easter (d'Ocagne, 1921, p. 468–470). Then, in 1939, Damien Garrigues published an article with a nomogram for finding the date of Easter in the Julian and Gregorian calendars (Garrigues, 1939). Garrigues did not refer to Crépin, and may have constructed his nomogram independently.

Our article explores this particular example and shows how this nomogram can be reproduced using METAPOST.[1] We will first analyze the structure of Garrigues' nomogram, and we will need to review some basic information on the calendar. Once we have a good grasp of the principles underlying Garrigues' nomogram, we will examine how to tackle its graphical challenges with METAPOST.

2 Easter in the Christian calendar

Easter is a Christian feast commemorating the resurrection of Christ and has been celebrated since the first centuries of our era. As time went by, it was decided to set the date of Easter on the Sunday immediately following the first full moon of Spring. For practical reasons, Spring is considered to begin on March 21st, and the full moons are based on simplified tables, not on astronomical observations. This rule applies both to the Julian calendar (before the Gregorian reform which took place in 1582) and to the Gregorian calendar. However, the tables for the paschal lunar phases were made more accurate in 1582, and the average year was made slightly shorter, so that this actually made the computation of Easter more complex.

We will not enter into the details of the history of the Christian calendar, nor of the many algorithms for the computation of Easter, but we will summarize — without proof — the basic procedure for the computation of Easter. More detailed information on calendrical calculations can be found in the standard (Dershowitz and Reingold, 2008), but information on Easter algorithms is scattered in multiple other sources. Besides (Gauss, 1973), one can consult (Knuth, 1997) for a simple (but exact) algorithm, and (Bien, 2004) for a comparison between a few Easter algorithms.

2.1 Julian calendar

In the Julian calendar, the date of Easter repeats itself after exactly 532 years. The computation is based on a lunar cycle of 19 years (the phase of the moon was supposed to be again the same after 19 years) and on a (solar) calendar cycle of 28 years (the years repeat after 28 years, since every fourth year is a leap year, since common years would repeat after seven years, and $28 = 4 \times 7$). We then merely have an Easter cycle of $532 = 19 \times 28$ years.

In this calendar, the position of a year in the 19 year lunar cycle is given by its Golden Number G:

$$Y \leftarrow \text{year} \tag{1}$$

$$G \leftarrow (Y \bmod 19) + 1 \tag{2}$$

Using the Golden Number, the (Julian) epact E_J of the year can be computed: the epact (in its modern sense) is the age of the moon on January 1st, minus one (Roegel, 2004). Since the moon phases shift by about 11 days every year, the epact consequently increases by about 11 units every year. It can be obtained from the Golden Number as follows:

$$E_J \leftarrow (11G - 3) \bmod 30 \tag{3}$$

And the value of the epact then determines the date of paschal full moon.

Sundays are determined by what is called the "dominical letter". All the days of a common year can be labeled by a letter from A to G, starting with A on January 1st, B on January 2nd, etc., G on January 7th, A again on January 8th, etc., reaching C on February 28, and D on March 1st (February 29 is considered to be without a letter). The "dominical letter" is then merely the letter associated to the Sundays of a year. When the year is a leap year, there are of course two dominical letters, one for January and February, and one for the other ten

[1] The complete METAPOST code is available on CTAN under the name `garrigues.mp`.

months, because the layout of the letters is defined for common years.

The date of Easter is obtained by combining the epact and the dominical letter.

2.2 Gregorian calendar

In the Gregorian calendar, the phases of the moon do no longer follow a 19 year cycle. The new cycle is more complex, as a consequence of a more accurate modeling of the mean motion of the moon, and because of the shorter mean solar year. The computation can still be based on the Golden Number and the (Julian) epact, but the epact is corrected as follows. We first define the secular part S of the year, then a correction M:

$$S \leftarrow \left\lfloor \frac{Y}{100} \right\rfloor \tag{4}$$

$$M \leftarrow \left(15 + S - \left\lfloor \frac{S}{4} \right\rfloor - \left\lfloor \frac{8S+13}{25} \right\rfloor\right) \bmod 30 \tag{5}$$

It was Gauss who introduced M in this form in 1816 (Gauss, 1800; Gauss, 1816).

What we call the "mean Gregorian epact" $\overline{E_G}$ is defined as follows:

$$\overline{E_G} \leftarrow (E_J - (M - 15)) \bmod 30 \tag{6}$$

The previous correction to E_J can also be used for the Julian calendar, by taking $M = 15$. In that case, $\overline{E_G} = E_J$.

The real (or corrected) Gregorian epact E_G, instead, is given by:

$$E_G \leftarrow \begin{cases} \overline{E_G} + 1 & \text{if } (\overline{E_G} = 25 \text{ and } G > 11) \\ & \quad \text{or } (\overline{E_G} = 24) \\ \overline{E_G} & \text{otherwise} \end{cases} \tag{7}$$

This value of the epact can be used to obtain a full moon in March. N_1 is the day in March for a full moon, but it may be another full moon than the paschal full moon (full moon on which the definition of Easter is based):

$$N_1 \leftarrow 44 - E_G \tag{8}$$

The real paschal full moon in March is:

$$N_2 \leftarrow \begin{cases} N_1 + 30 & \text{if } N_1 < 21 \\ N_1 & \text{otherwise} \end{cases} \tag{9}$$

Garrigues' nomogram computes the paschal full moon without the corrections for E_G, and obtains a date of Easter. Ignoring the corrections on the epact produces certain wrong epacts, but only some of these wrong epacts cause an incorrect date of Easter. The dates are incorrect in only rare circumstances, which are listed in the nomogram (1954, 2049, 2106, etc.) and which will be analyzed later in this article.

3 The structure of Garrigues' nomogram

3.1 An example

Garrigues' article shows the use of the nomogram for the year 1939, the year the article was published. Using the nomogram is straightforward. The year is first divided in its century number (called "partie séculaire", or secular part in French) and the last two digits of the year (merely called "Année", that is, year in French). Each of these parts is looked up in columns I and III (figure 1) and the centers of the two circles containing the values sought are connected by a dashed line. This line falls on a point in column II, and this point is in turn connected to the first point at the top of column IV. This is the Golden Number associated to 1939.

The secular part is reused in column VI, and joining it with the Golden Number just found, a new point is obtained in column V. This point is connected to the point labeled 10 in column VII, and this is the value of the (mean) Gregorian epact.

Now, using again the secular part in the right part of column VIII and the last two digits of the year in column X, we obtain a point labeled "A" in column IX. This point is connected to point "A" in column XI. Finally, the intersection of the lines connecting point 10 of column VII and point B on one hand, and point "A" of column XI and point C on the other hand, falls in the slot corresponding to April 9, which is the date of Easter in 1939.

Before attempting to reproduce the nomogram, we will first try to analyze its construction. This will provide us with enough insight and will lead seamlessly to the METAPOST code.

As we have just seen, Garrigues' nomogram is made of several parts, which are all fairly regular. The areas were numbered by Garrigues in Roman numerals I, II, III, ..., XVI, but in this article we will only consider the first eleven areas, the only ones which are concerned with the calculation of Easter. We will analyze each of these areas in sequence.

It is important to understand the geometry of the nomogram, because the geometry represents the relationship between the variables.

3.2 Basic features of the nomogram

The basic features of Garrigues' nomogram are the following:

- some lines or sequences of points are annotated using various functions: a set of points $1, 2, \ldots, i$ are distributed linearly and annotated with $f(i)$; examples are given in figure 2;
- additions are obtained by drawing a line: the addition is on the index values, that is, on po-

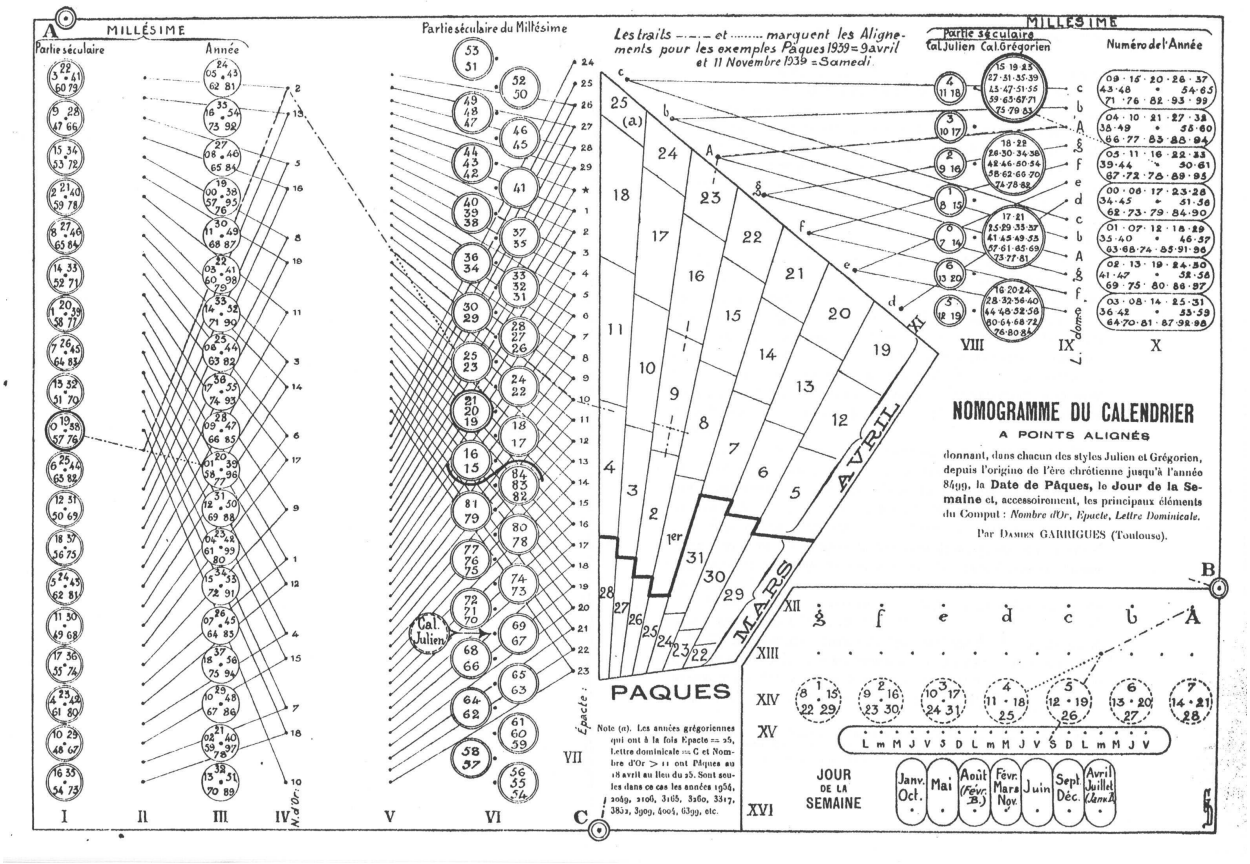


Figure 1: Garrigues' original nomogram (excerpt from (Garrigues, 1939)).

sitions; this scheme is used here three times; in each case, from two values among n values, we obtain $2n-1$ combined values.

- for columns I–III, $n = 19$;
- for columns IV–VI, $n = 30$;
- for columns VIII–X, $n = 7$.

We first consider the scheme represented on the left of figure 2. For $i = 0, 1, \ldots$, let $c(p_i) = (0, i)$, $c(q_i) = (2, i)$ and $c(r_i) = (1, i/2)$ be the coordinates of points p_i, q_i and r_i, and let $v(p_i)$, $v(q_j)$ and $v(r_k)$ be the values associated to p_i, q_j and r_k. We have of course $v(p_i) = i$, $v(q_j) = j$ and $v(r_k) = k$. Let $v'(p)$ be the value associated to the point at coordinates p, then $v'((0, i)) = i$, $v'((2, i)) = i$, and $v'((1, i/2)) = i$. Finally, $v'(c(p_i)) = i$, $v'(c(q_j)) = j$, and $v'\left(\frac{c(p_i)+c(q_j)}{2}\right) = v'((1, (i+j)/2)) = i+j$. The example shows how we obtain 5 by adding 2 and 3.

On the right of figure 2, instead, we do not add 2 and 3, but we obtain the position 5 from positions 2 and 3. 2, 3 and 5 are index values, not the values sought themselves. So, the scheme on the right can be used to compute z_{i+j} from x_i and y_j, but the value of z_{i+j} need not be the sum of x_i and y_j. The first case is of course a special case of the second one, where $x_i = i$, $y_j = j$ and $z_k = k$.

This scheme is used three times in Garrigues' nomogram, with x_0, y_0 and z_0 at the bottom in the three cases. In columns I–III (see figure 3), x_i is the sequence of Golden Numbers 4, 12, 1, 9, 17, we can take $y_i = x_i$ (or any other shifted sequence x_{i+s}), and z_i is the sequence 18, 7, 15, 4, 12, 1, 9, etc.

In columns IV–VI (figure 5), $x_i = E_J(i)$ (Julian epact), $y_i = 15 - M(i)$ and $z_i = \overline{E_G}(i)$ (mean Gregorian epact).

In columns VIII–X (figure 7), x_i, y_i and z_i are values associated to dominical letters.

- some values are rearranged: data can be transferred from one line to another, using a mapping; $f(i) = g(h(i))$ where $f(i)$ is the function on the first line, $g(j)$ is the function on the second line, and $j = h(i)$ is the mapping from one

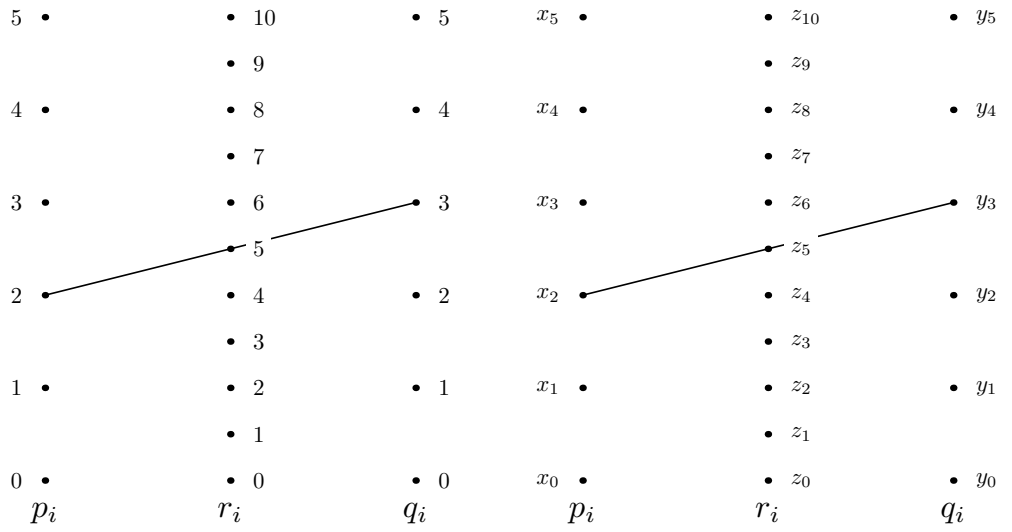


Figure 2: Basic addition on a nomogram: direct addition (left), and addition on indices (right). This scheme is used three times by Garrigues. The left part is a special case of the right one with $x_i = i$, $y_i = i$, and $z_i = i$.

line to the other; see for example figure 4.

- certain values are obtained as intersections in a 2-dimensional grid: from two lines indexed by i and j, a grid can be constructed from the values of $f(i, j)$. See for instance figure 10.

3.3 Description of the components of Garrigues' nomogram

The columns of the nomogram will be described in the following order, not strictly from left to right.

Columns I–III (figure 3) The purpose of the first three columns is to obtain the Golden Number G corresponding to a given year. The year is identified by its secular part S and by its last two digits A. The arrangement of columns I and III is a consequence of the arrangement of column II. We therefore first need to understand column II and then we can proceed with columns I and III.

Column II: The points in this column represent values of the Golden Number G, from top to bottom: 2, 13, 5, 16, 8, 19, 11, 3, 14, 6, 17, 9, 1, 12, 4, 15, 7, 18, 10, and then again 2, 13, ..., until 18 (the values are shown in column IV). This is the order of the Golden Numbers if they are rearranged by the corresponding values of the Julian epacts E_J which are 19, 20, (21), 22, 23, (24), 25, 26, (27), 28, (29), 0, 1, (2), 3, 4, (5), 6, (7), 8, 9, (10), 11, 12, (13), 14, 15, (16), 17, (18). (Values between parentheses do not occur as Julian epacts, hence the gaps in column IV.) So, $G = 2$ corresponds to Julian epact 19, $G = 13$ corresponds to Julian epact 20, and so on. Let $c_2[i]$ be the i-th Golden Number value (from the bottom) in column II: we have $c_2[1] = 18$, $c_2[2] = 7$, etc. It is easy to see that $c_2[i] = 1 + ((9 + 8i) \bmod 19)$. We can also write $c_2[20-i] = (5+11(i-1)) \bmod 19 = (13+11i) \bmod 19$, which shows that the Golden Numbers increase by 11 (mod 19) from top to bottom.

Column I: The first column is related to the secular parts S of the years, that is, the digits left when removing the last two digits of the year. 2008, for instance, has 20 for its secular part S. The secular parts are arranged by their remainder by 19 and there are therefore 19 circles with values inside. However, the circles are not ordered in the usual order, that is, remainder 1 does not follow remainder 0, remainder 2 does not follow remainder 1, etc. Instead, the secular parts are ordered according to their contribution to the Golden Number in column II. So, $S = 9$ follows $S = 3$ because $900 = 47 \times 19 + 7$, and $300 = 15 \times 19 + 15$, and 7 follows 15 in column II, and so on.

We refer to these circles by the smallest values found inside, namely 3, 9, 15, 2, 8, ..., 16. Two consecutive values differ by 6 (mod 19), because adding 6 to a secu-

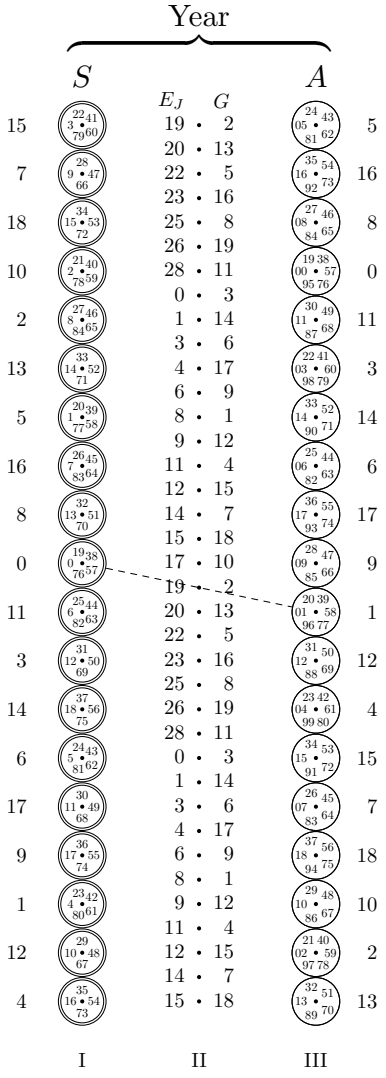


Figure 3: Finding the Golden Number in column II, using the components of the year. The Golden Numbers G are given in the order of the Julian epacts E_J, but without gaps. We have added the values of $(100S)$ mod 19 (left of column I) and A mod 19 (right of column III).

lar part is equivalent to adding 11 to the Golden Number ($6 \times 100 \equiv 11 \pmod{19}$), and we have seen that the Golden Numbers in column II increase by 11 (mod 19) from one point to the next point below.

Moreover, values such as 3, 22, 41, etc., are located in the same circles since they are equal modulo 19. If we call $c_1[i]$ the index value on the i-th point in column I (with point 1 at the bottom), we can see that $c_1[i] = (22 - 6i) \bmod 19$. For instance, $c_1[17] = (22 - 6 \times 17) \bmod 19 = 15$. When the secular part is S, it goes on the point $point_S(S) = 1 + ((3S + 9) \bmod 19)$. For instance, if $S = 19$, $1 + ((3 \times 19 + 9) \bmod 19) = 10$, so 19 is located on the 10th point. We can check that $\forall S < 19 : c_1[point_S(S)] = S$.

Column III: The last two digits of the year are also positioned in relation with the second column, for the same reason as for column I; we can notice that the years appear in the order (from top to bottom) 5, 16, 8, 19, 11, etc., exactly like the order of the values in column II; that is, $c_3[i] = (5 - 11i) \bmod 19$, where $c_3[i]$ is the smallest value in a circle in column III. For instance, on the first point, $c_3[1] = (5 - 11) \bmod 19 = 13$. On the second point $c_3[2] = 2$, etc. The point corresponding to the last two digits A of the year is $point_A(A) = 1 + ((15 + 12A) \bmod 19)$ and one can check that $\forall A < 19 : c_3[point_A(A)] = A$.

Linking columns I and III: If the centers of the circles in the first column are at coordinates $(0, i-1)$ where i is the point number (in the enumeration given above), and if the centers of the circles in the third column are at coordinates $(2, j)$ with $j = 0$ to 18 ($j = 0$ at the bottom), then the points in the second column are located at coordinates $(1, k/2)$ where $k = 0, 1, \ldots, 36$. This is the scheme shown in figure 2.

We can now check that linking the secular part and the last digits of the year indeed gives the Golden Number. Figure 3 shows columns I to III, and, for every value of S and A, we have added the value of $(100S)$ mod 19 (left of column I) and of A mod 19 (right of column III). We have also added the values of the Golden Number G in column II. We are in the conditions of figure 2, where $x_i = (4 + 8i) \bmod 19$, $y_i = (13 + 8i) \bmod 19$, and $z_i = (18 + 8i) \bmod 19$. The case $i = j = 0$ corresponds (for instance) to the year 1613, for which $G = (1613 \bmod 19) + 1 = 18$. Therefore, the triplet (x_0, y_0, z_0) is indeed a correct one. What we need to prove is that any three aligned points make a correct triplet. A correct triplet is of the form (x_i, y_j, z_{i+j}) (figure 2). This can be proved by induction. If we assume that the triplet (x_i, y_i, z_{i+j}) is a correct triplet, then, since $(x_{i+1} - x_i) \bmod 19 = (y_{i+1} - y_i) \bmod 19 =$

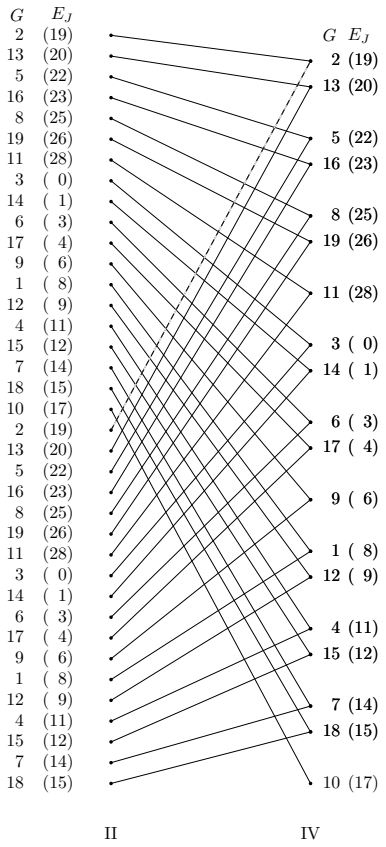


Figure 4: Rearranging the Golden Numbers (column II) according to the Julian epacts (column IV), but with gaps for the missing epacts.

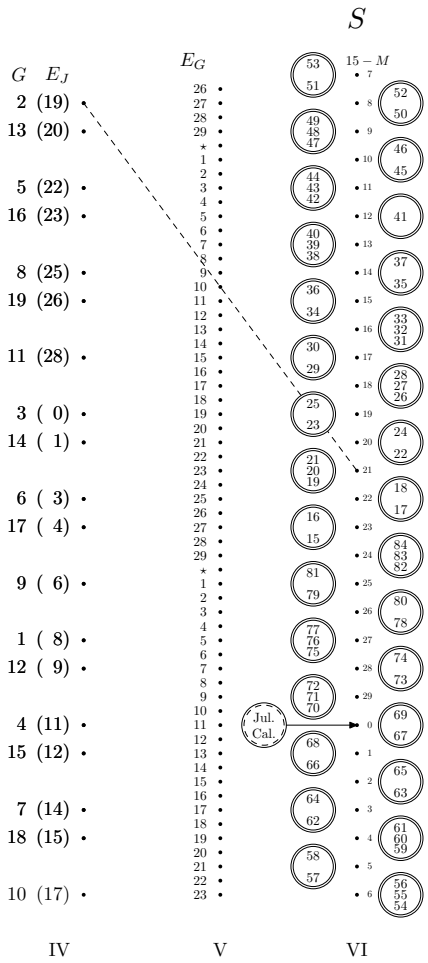


Figure 5: Finding the mean Gregorian epact (column V) using the Julian epact (column IV) and the value of M (column VI). If we add E_J to $15 - M$, we obtain the mean Gregorian epact $\overline{E_G}$. For $i = 0$, $E_J = 17$, $M = 9$, $15 - M = 6$ and $\overline{E_G} = E_J - (M - 15) = 23$. When M increases, $\overline{E_G}$ decreases. When E_J decreases, $\overline{E_G}$ decreases too. The left column shows $x_i = (17 - i) \bmod 30$, the right column shows $y_i = (6 - i) \bmod 30$ and the middle column shows $z_i = (23 - i) \bmod 30$.

$(z_{i+1} - z_i) \bmod 19 = 8$, it follows that $(x_{i+1}, y_i, z_{i+j+1})$ and $(x_i, y_{i+1}, z_{i+j+1})$ are also correct ones, because z_k increased by exactly as much as either the secular part or the last digits of the year contributed to the Golden Number, and therefore z_k must still be the Golden Number.

Garrigues could have designed the columns I–III more naturally, by putting the Golden Numbers in their natural order, but he chose to put them in the order of the Julian epacts without showing the values of those.

Columns IV–VI: (figure 5) The purpose of these three columns is to compute the mean Gregorian epact $\overline{E_G}$ from the Julian epact E_J.

Column V: This column is found halfway between columns IV and VI. Column IV represents the Julian epact E_J and column VI corresponds to the correction $15 - M$, with $M = (15 + S - \lfloor \frac{S}{4} \rfloor - \lfloor \frac{8S+13}{25} \rfloor) \bmod 30$, where $\lfloor ... \rfloor$ is the integer part. M is Gauss' correction (Gauss, 1816). The mean Gregorian epact $\overline{E_G}$ is given by $\overline{E_G} = (E_J - (S - \lfloor S/4 \rfloor - \lfloor (8S + 13)/25 \rfloor)) \bmod 30 = (E_J + (15 - M)) \bmod 30$. We call $\overline{E_G}$ the "mean Gregorian epact", because it is the epact considered without the corrections for epacts 24 and 25.

The real Gregorian epact E_G sometimes differs from the value of the mean Gregorian epact $\overline{E_G}$ which is obtained from the nomogram. $E_G = \overline{E_G} + 1$ if ($\overline{E_G} = 25$ and $G > 11$) or ($\overline{E_G} = 24$). This shifts the

paschal full moon one day earlier. We will come back to these exceptions when examining the Easter area between columns VII and XI.

Column V can therefore be seen as the sum of columns IV and VI, and will be the mean Gregorian epact $\overline{E_G}$. This column has 58 points, that is, as many as there are combinations between columns IV and VI. The values of the epact are shown in column VII. Epact 0 or 30 is usually written '⋆'.

Column IV: This column only serves for an addition with the value represented in column VI. There are 30 points in column IV if one includes the gaps. As mentioned above, the point numbers correspond to the Julian epacts on a 1–30 scale (hence the gaps). Each point corresponds to a value of the Julian epact, but the points are labeled with the Golden Number, since there is an exact correspondence between them. When the Golden Number is G, the Julian epact E_J is $(11G - 3) \bmod 30$ and E_J is on point $1 + ((17 + 29E_J) \bmod 30)$ hence on point $1 + ((20 + 19G) \bmod 30)$. $G = 1$ corresponds to point 10, $G = 2$ to point 29, $G = 3$ to point 18, $G = 4$ to point 7, etc. The first point on the top of column IV corresponds to $E_J = 19$ (for $G = 2$). The empty slot above it would correspond to Julian epact 18, but such a value does not exist in the Julian calendar. The second point from the top corresponds to $E_J = 20$ (for $G = 13$), the second empty slot to $E_J = 21$ (which does not exist), the next point is $E_J = 22$ (for $G = 5$), and so on, until Julian epact 17 (for $G = 10$) at the bottom.

So, column IV shows the Golden Number, but at positions corresponding to the Julian epacts.

Column VI: The secular parts S of the year, between 15 and 84, are positioned according to the values of $M = (15 + S - \lfloor S/4 \rfloor - \lfloor (8S + 13)/25 \rfloor) \bmod 30$; 30 points are in this column. The first point at the top corresponds to $M = 8$, the second point to $M = 7$, etc., until $M = 9$ at the bottom (for $S = 54$, 55, and 56). If S corresponds to a value M, it is put on the $(1 + (M + 21) \bmod 30)$-th point from the bottom. 30 different circles are put along that column, left and right of it, to save space. It was certainly this column which led Garrigues to stop the secular parts at 84, because $S = 85$ would have had to be added to the circle with $S = 15$ and $S = 16$, breaking the evenness of the distribution of S. Nevertheless, the nomogram could easily be extended if necessary.

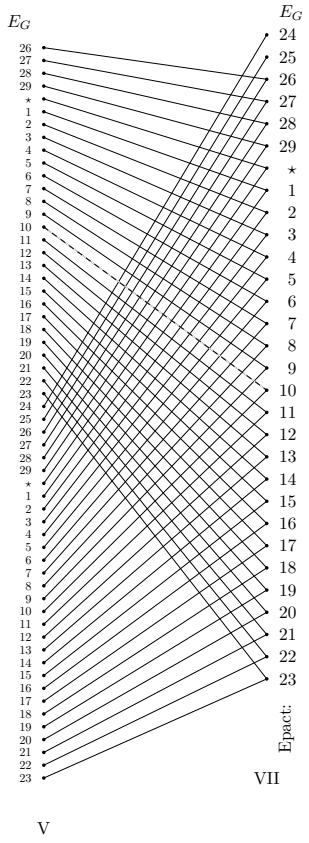


Figure 6: Rearranging the mean Gregorian epacts uniquely in column VII.

This column is used together with the Golden Number G (column IV) to obtain the epact (column V). The Julian calendar corresponds to $M = 15$ which goes with $S = 67$ and 69 in the Gregorian calendar.

Column VII: This column is like column V, but it will be used for the right-hand side of the drawing. The values do not start at the bottom, but this doesn't matter, as we have some freedom in the positioning of the points.

Columns VIII–X: (figure 7) The purpose of these three columns is to obtain the dominical letter of the year, using the secular part S and the last two digits A of the year.

Column IX: This column gives the series of dominical letters DL from bottom to top,

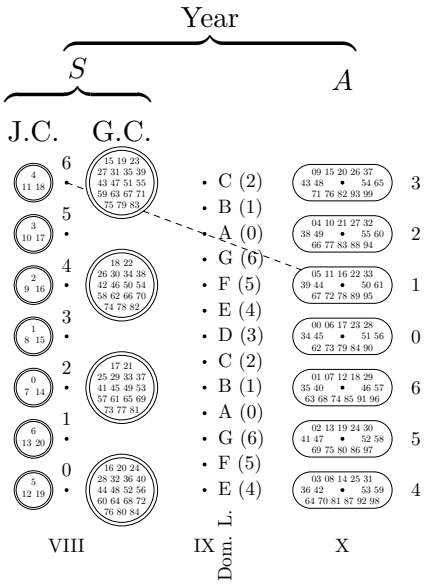


Figure 7: Finding the dominical letter (column IX) of a year, using its components S and A. The 1st of March 1900 was a Thursday (d.l. = G). Consequently, the 1st of March 2000 was a Wednesday (and d.l. = A). The 1st of March 2100 will be a Monday, and so on. If the vertical scale in column VIII is such that the days of the week go up from top to bottom, the centuries will be arranged as on the figure and the dominical letters must go in the opposite direction. In the Julian calendar, there is always a gap of 1 between two centuries. This figure shows the second dominical letters of the years $100S$ over the points of column VIII, with the convention $A \to 0$, ..., $G \to 6$. The first line corresponds to 2008 (for instance) and $z_0 = x_0 + y_0$.

starting with E. Each letter is associated with a number: $A \to 0$, ..., $G \to 6$.

Column VIII: For a year $Y = 100S + A$, column VIII corresponds to the day of the week for the 1st of March of year $100S$, and hence also to the second dominical letter of year $100S$ (the first dominical letter is for January and February). The case where March 1st is a Wednesday is put at the bottom and corresponds to the dominical letter A (because the letter associated with March 1st is D, and the previous Sunday is on the letter A). This is the case for 1600 and 2000, for instance. In the Julian calendar, since 100 years make up 36525 days ($\equiv 6 \bmod 7$), advancing 100 years means going one day backwards in the week and one letter forward in the dominical letters. This is shown in column VIII on the left when we go up when the century increases. In the Gregorian calendar, there are either 36525 days or 36524 days in a century. Hence, we go up by two days, except when going from $S = 19$ to 20, from $S = 23$ to 24, etc.

Column X: The values on the right of column X show the second dominical letters in the 21st century, with the same conventions as in column IX. 2000, for instance, had dominical letter A, and the last digits 0 of 2000 fall at position 0 (figure 7), 0 being associated with the dominical letter A.

Since the value associated at the bottom of column VIII is 0, and since the first value in column IX is 4, the years such as 2008 (with DL equal to E) are also put at the bottom of column X.

We can see how the other values are laid out: 365 days are a multiple of 7 plus 1. So every time we have a new year, we add one to the day of the week, except when we pass from a year like $4n - 1$ to $4n$ (for instance from 2007 to 2008). The year A goes on point $1 + (3 - A - \lfloor A/4 \rfloor) \bmod 7$ (counted from 1 at the bottom).

Linking columns VIII and X: Column VIII corresponds to the day of the week of the first March of the first year with secular number S, and column X corresponds to the shift introduced by the year within the 21st century. Adding the two gives the day of the week for the 1st of March of the year considered, because 2000 is at the bottom of column VIII, and hence gives the second dominical letter of the year. The values of column IX follow.

Column XI: This column reproduces the values of column IX, but avoiding the duplication.

Easter area: This is a table giving Easter using the mean Gregorian epact $\overline{E_G}$ and the dominical letter DL. Points B and C are used to draw lines towards the epact and dominical letter values, and the intersections fall in a slot. There are 35 possible days for Easter and therefore 35 slots in this area. Basically, the epact gives us the day of the pascal full moon, and the dominical letter gives us the day of the week of that full moon. The two together give the date of Easter. There are five Easter dates corresponding to each dominical letter.

This table must take the epact exceptions into account. As we have seen earlier, the value

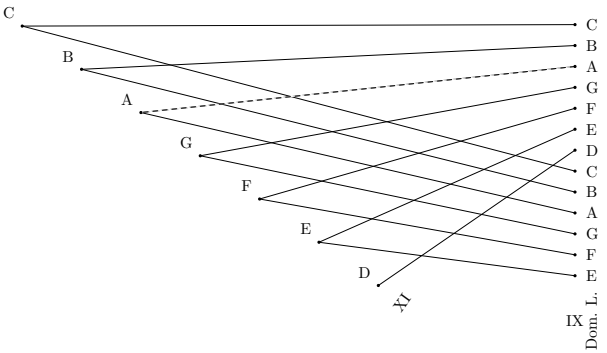


Figure 8: Rearranging the dominical letters for use in the Easter area (column XI).

of the mean Gregorian epact is subject to a correction in certain cases. If there are no corrections, a (mean) epact value of 24 corresponds to a paschal full moon on April 19, and a (mean) epact value of 25 corresponds to a paschal full moon on April 18. The corrections have as an effect to always shift the epact 24 full moon to April 18, and to move the epact 25 full moon to April 17 only when the Golden Number is greater than 11. This means that the cases $\overline{E_G} = 24$ and $(\overline{E_G} = 25) \wedge (G \leq 11)$ correspond to the same paschal full moon (April 18), and this is what is shown in figure 10. The only exception in the Easter area table is therefore the case $(\overline{E_G} = 25) \wedge (G > 11)$. In this case, the new paschal full moon is April 17, and this will only cause the date of Easter to move if April 18 happened to be a Sunday, hence if the dominical letter was C.

So, the table would give the date of Easter April 25 when the mean Gregorian epact is 25, the dominical letter is C, and the Golden Number is greater than 11. Between 1583 and 10000, this occurs only in 1954, 2049, 2106, 3165, 3260, 3317, 3852, 3909, 4004, 6399, 6551, 7086, 7143, 7238, 8202, 8297, 8354, 8449, and 9041. In these cases, the date of Easter should therefore be April 18 and not April 25 and this is what Garrigues indicated in a footnote.

4 Reproducing the nomogram with METAPOST

Reproducing Garrigues' nomogram in METAPOST is easy, once we have a good understanding of its structure. The complete reconstructed nomogram is shown in figure 9. We will in turn consider the positions of the points, the connections, the labels, and the Easter grid (between columns VII and XI).

4.1 METAPOST

METAPOST is the graphical programming language accompanying TEX. Graphics are expressed as programs where various points, lines and labels are defined. We will not describe the language here, and we refer the reader to the main references (Goossens, Mittelbach, Rahtz, Roegel, and Voss, 2008; Hobby, 2008). However, in the sequel, we will explain some of the interesting or particular constructions used in our code.

4.2 LATEX labels with the `latexmp` package

TEX labels are usually included in METAPOST using the `btex ... etex` construction, but this is a very inefficient solution, especially when labels are parameterized. A much better solution is to use the `latexmp` package which provides a macro `textext` taking a string representing some LATEX code. This is what we have been using throughout our code.

4.3 Auxiliary functions

The following macros are used in several places of the nomogram code and are described first.

The first macro `DL` (defined with `def` and taking i as a parameter) transforms an integer i from 1 to 7 into a character from A to G and is used to display the dominical letter:

```
def DL(expr i)=char(64+i) enddef;
```

The macro `gn_epact` returns a pair made of the Golden Number and the Julian epact associated to the i-th point in column II (see figure 4), 1 being at the bottom. For $i = 2$, for instance, this macro returns $(7, 14)$. The macro is defined with `vardef`, which is a variant of `def` making it possible for the `G` and `JE` variables to have only a local scope after their `save` declaration.

```
vardef gn_epact(expr i)=
  save G,JE;
  G=1+((9-11i) mod 19);
  JE=(11G-3) mod 30;
  (G,JE) % value returned
enddef;
```

The macro `gn_epactl` returns the value of the Golden Number and the point in column IV associated to the i-th point in column II, 1 being at the bottom (see figure 4). For $i = 2$, for instance, this macro returns $(7, 4)$, because it is the 4th point of column IV which is associated to $G = 7$ (the second point being empty).

```
vardef gn_epactl(expr i)=
  save G,JE,JEL;
  G=1+((9-11i) mod 19);
  JE=(11G-3) mod 30;
  JEL=30-((JE+12) mod 30);
```

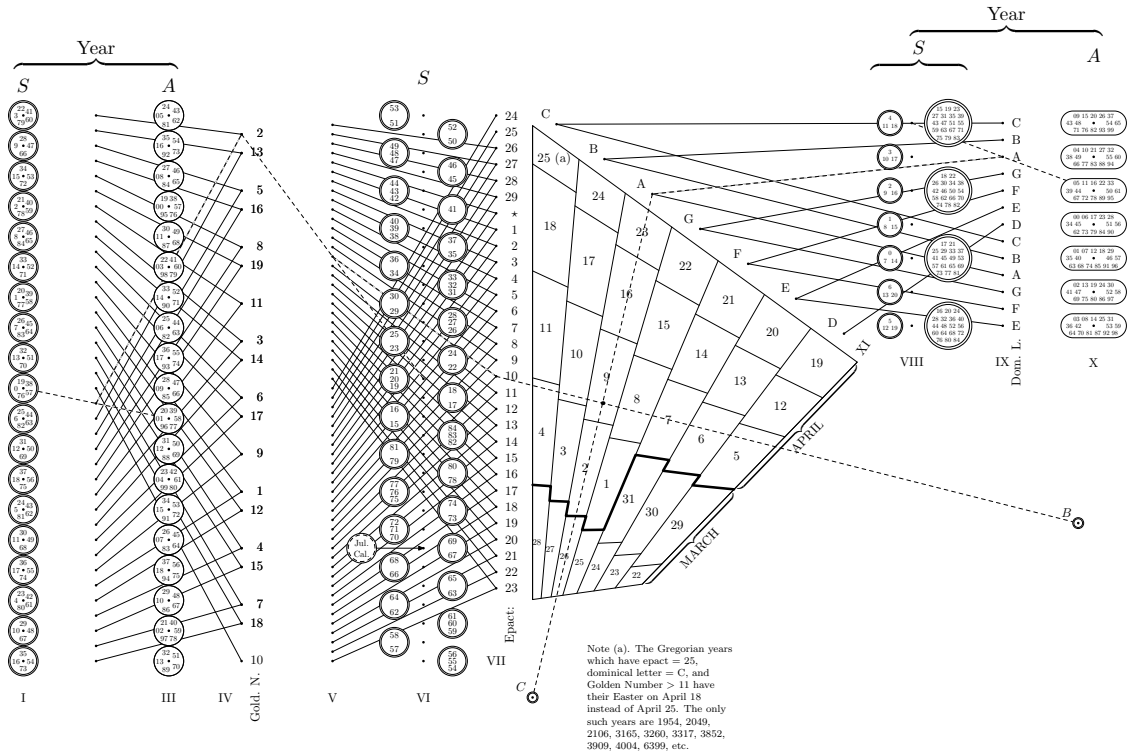


Figure 9: The complete METAPOST version of the nomogram.

```
  (G,JEL) % value returned
enddef;
```

`whatever` is a very useful METAPOST instruction which represents a yet undefined and *unnamed* scalar value. It is then possible to solve linear equations very easily, for instance finding point P such that $\overrightarrow{OP} = \overrightarrow{OA} + x\vec{u} = \overrightarrow{OB} + y\vec{v}$, x and y being the unknowns, with

```
OP=OA+whatever*u=OB+whatever*v;
```

`OP`, `OA`, `OB`, `u`, and `v` being points (or complex values).

Interestingly, the two values of `whatever` (corresponding to the unknowns x and y) are usually not equal, which is why `whatever` should not be viewed as the name of a variable. In the previous example, we were more interested in the position of P than in the values of x and y, and P is merely the intersection of two lines.

In our code, we use the macro `whateverpair` which is the equivalent of `whatever` for pairs. It defines a "fresh" pair of numerical values (which need not be equal, despite the way they are defined, for the reason given above).

```
def whateverpair=
  (whatever,whatever)
enddef;
```

The next three macros are defined for formatting purposes. The first macro `ep_st` formats the epact value so that it fits on two characters, and the value 0 is displayed as '$\star$'. `textext` is the main macro provided by the `latexmp` package.

```
def ep_st(expr i)=
  if i=0:
    textext("\phantom{0}$\star$")
  elseif i<10:
    textext("\phantom{0}"&decimal(i))
  else:
    textext(decimal(i))
  fi
enddef;
```

The second macro `gstring` is somewhat similar, but only formats a one or two-digit value with a two-digit width, forcing the value to have the same vertical size as an opening parenthesis, for alignment purposes.

```
def gstring(expr i)=
  if i<10:
    textext("\vphantom{(}\phantom{0}"
                &decimal(i))
  else:
    textext("\vphantom{(}"&decimal(i))
  fi
enddef;
```

The third macro `tddec` (two-digits decimal) formats a one or two-digit value as two digits, by possibly adding a 0 in front of it.

```
def tddec expr i=
  if i<10: "0" & decimal(i)
  else:
    decimal(i)
  fi
enddef;
```

4.4 Defining the points

In this section, we define the various points used in the construction of the nomogram.

4.4.1 Variables

For the points in the different columns, we mainly use two arrays of pairs:

```
pair col[][],col[]a[];
```

The points in column I will be stored in the variables `col[1][1]`, `col[1][2]`, `col[1][3]`, etc. The points in column II will be stored in `col[2][1]`, `col[2][2]`, `col[2][3]`, etc. In METAPOST, we can write `col1[1]` instead of `col[1][1]`, and this will simplify a little bit our code.

The second array (`col[]a[]`) is only used for the centers of the circles which are along column VI.

4.4.2 First points

The points in columns I to VII are set easily. The first and third columns have 19 points each, the second and fourth columns have 37 points each, column V has 58 points and columns VI and VII both have 30 points. All these points are linearly set and the points in columns II and V are obtained by bisecting segments linking points from adjacent columns.

The first three columns are straightforward to set, using a `height` constant defined elsewhere (and not described in this article):

```
for i:=1 upto 19:
  col1[i]=(0,(i-1)*height/18);
endfor;
for i:=1 upto 19:
  col3[i]=(40u,(i-1)*height/18);
endfor;
for i:=1 upto 37:
  col2[i]=
    ((xpart(col1[1])+xpart(col3[1]))/2,
     .5*(i-1)*height/18);
endfor;
```

Column IV is a bit more tricky, and for each of the 37 points in column II, the macro `gn_epactl` returns a pair made of the Golden Number associated to this point, and of the corresponding point in column IV. The Golden Number is not used here. The value of `col4` is then set:

```
for i:=1 upto 37:
  E:=ypart(gn_epactl(i));
  col4[E]=(60u,(E-1)*height/29);
endfor;
```

Columns V to VII are also easily set. In the case of column VI, additional points `col6a` are defined for the positions of the circles offset in that column. The points in column VII are shifted upwards by a certain amount (here $20u$, u being here equal to 1 mm).

```
for i:=1 upto 30:
  col6[i]=(110u,(i-1)*height/29);
endfor;
for i:=1 upto 58:
  col5[i]=
     ((xpart(col4[1])+xpart(col6[1]))/2,
      .5*(i-1)*height/29);
endfor;
for i:=1 upto 30:
  if i mod 2=0:
    col6a[i]=col6[i]-(8u,0);
  else:
    col6a[i]=col6[i]+(8u,0);
  fi;
endfor;
for i:=1 upto 30:
  col7[i]=(130u,
           20u+(i-1)*(height-20u)/29);
endfor;
```

4.4.3 Easter table

The whole Easter table is obtained by setting points B and C, as well as four corners of the table. B and C can be positioned freely.

```
vardef define_easter_table=
  save corner,p;pair corner[];
  C=(xpart(col7[1])+10u,-10u);
  B=(xpart(C)+150u,ypart(col7[5]));
```

We define two additional points in column VII, one above the 30th (`col7[31]`), and one below the first (`col7[0]`):

```
  col7[1]-col7[0]=col7[31]-col7[30]
                 =col7[2]-col7[1];
```

The shape of area XI is defined by its four corners:

```
  corner1=whatever[col7[0],B]
         =C+whatever*up;
  corner3=.3[B,col7[31]];
  corner2=(C--corner3) intersectionpoint
          (B--corner1);
  corner4=whatever[B,col7[31]]
         =C+whatever*up;
```

Then, the whole area determined by the points `corner1`, `corner2`, `corner3`, and `corner4` is divided into eight slices, only seven of which will be drawn (figure 10). The first slice contains the Easter dates March 28, April 4, 11, 18 and 25. The second slice

contains the Easter dates March 27, April 3, 10, 17, and 24, and so on. The eighth slice is not drawn, but defined for practical reasons. These eight slices are limited by nine boundary lines. The first seven slices correspond to the dominical letters C, B, A, G, F, E, and D shown in column XI:

The slices are represented using two two-dimensional arrays, `s[]l[]` and `s[]r[]`. The Easter area is divided into slots, and each slot is a quadrilateral. Two of the vertices of each quadrilateral are located on one slice boundary, and the two others are located on another boundary. If we now consider a boundary between slices, which is a (more or less) vertical segment, this boundary contains points from which some segments go to the left, and other go to the right (more or less). In the former case, the points are given by the array `s[]l[]` ('l' for left), and in the latter case by the array `s[]r[]` ('r' for right). All the boundaries contain 10 points. The points of the second boundary, for instance, are `s2l0`, `s2l1`, `s2l2`, `s2l3`, `s2l4`, `s2l5`, `s2r0`, `s2r1`, `s2r2`, `s2r3`, `s2r4`, `s2r5`. `s2l0` is equal to `s2r0`, and `s2l5` to `s2r5`.

The first part of the code defines the beginnings and ends of each boundary line:

```
for i=1 upto 9:
  s[i]l0=s[i]r0
        =(((i-1)/8))[corner4,corner3];
  s[i]l5=s[i]r5
        =whatever[corner1,corner2]
        =whatever[s[i]l0,C];
endfor;
```

We then divide each of the eight boundaries four times. i is the boundary number and goes from left to right. Eight vertical lines enclose the 35 Easter slots. j varies over the horizontal inner divisions. A division is made so that the line going through B and the division falls exactly between two epact values in column VII:

```
for i=1 upto 8:
  for j=1 upto 4:
    p:=30-i-(j-1)*7;
    if i<8:
      % division leaving to the right
      %  of vertical line i
      s[i]r[j]=(s[i]l0--s[i]l5)
                intersectionpoint
            (B--.5[col7[p],col7[p-1]]);
    fi;
    if i>1:
      % division leaving to the left
      % of vertical line i
      s[i]l[j]=(s[i]l0--s[i]l5)
                intersectionpoint
            (B--.5[col7[p+1],col7[p]]);
    fi;
  endfor;
endfor;
enddef;
```

Finally, the points in column XI are obtained from the upper boundary of the Easter area. They are put on a line parallel to (`corner3`, `corner4`) and in the middle of the slices (as seen from C).

```
vardef define_dominical_letters=
  save shift;pair shift;
  shift=(3u,3u);
  for i=1 upto 8:
    col11[i]
      =whatever[C,.5[s[i]r0,s[i+1]l0]]
      =whatever[s1r0+shift,s8l0+shift];
  endfor;
enddef;
```

4.4.4 Last points

The points in columns VIII to X are determined as follows:

```
  for i=1 upto 7:
    col8[i]=s8l0
      +(15u,10u+(i-1)*ypart(s1l0-s8l0)/7);
    col10[i]=col8[i]+(50u,0);
  endfor;
  for i=1 upto 13:
    col9[i]=(xpart(col8[1]+col10[1])/2,
             ypart(col8[1])
                +(i-1)*(ypart(col8[7]
                         -col8[1]))/12);
  endfor;
```

4.5 Drawing the connections

Connections between columns II and IV are drawn by the following code:

```
  for i:=1 upto 37:
    draw col2[i]
          --col4[ypart(gn_epactl(i))];
  endfor;
```

Connections between columns V and VII are drawn by the following code:

```
  for i:=1 upto 58:
    draw col5[i]--col7[1+((i-1) mod 30)];
  endfor;
```

Connections between columns IX and XI are obtained by the following code:

```
  for i=1 upto 13:
    draw col9[i]--col11[1+(13-i) mod 7];
  endfor;
```

4.6 Drawing the circles

Double circles are drawn using a straightforward macro not described here. For column I, the circles are drawn with:

```
  for i:=1 upto 19:
```

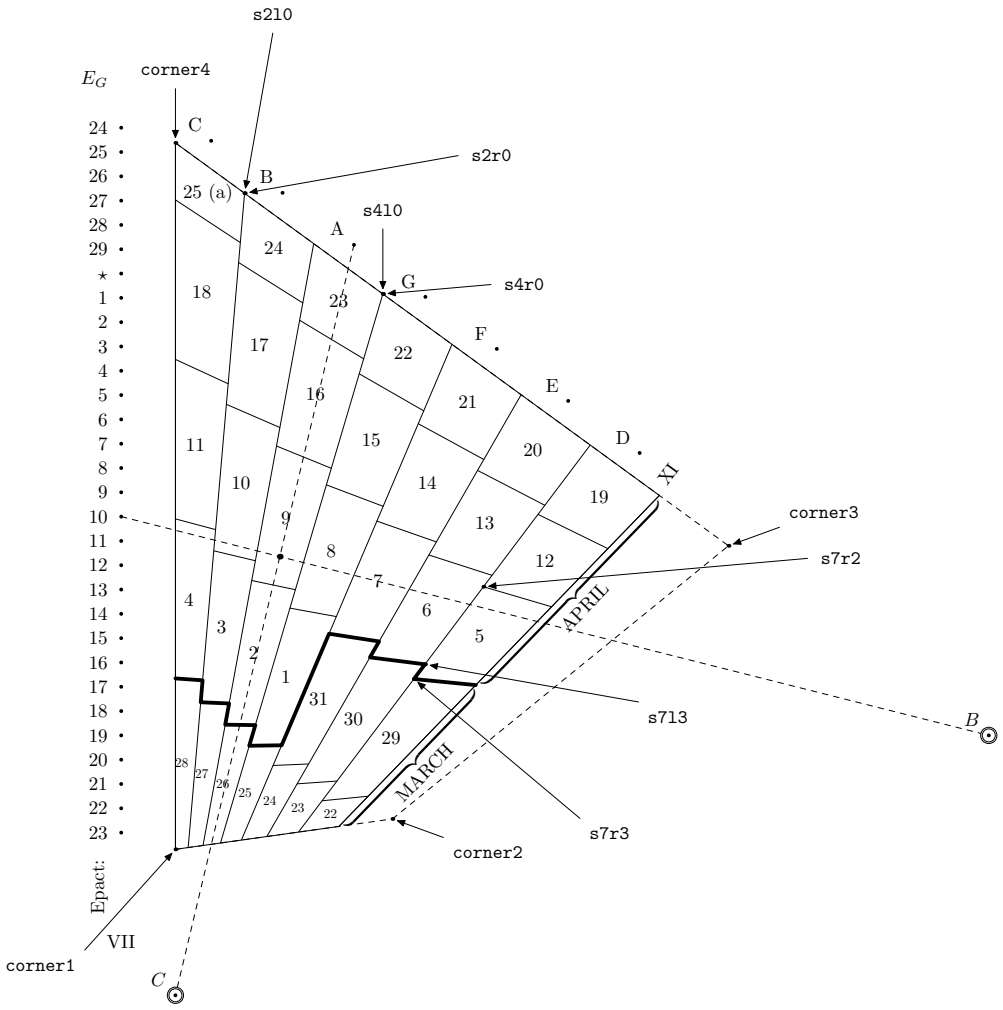


Figure 10: The Easter area, for the determination of Easter using the dominical letter and the mean Gregorian epact. This table shows one exception (note (a)), corresponding to the case $\overline{E_G} = 25$, $G > 11$ and $DL = C$. The cases $\overline{E_G} = 24$ and $(\overline{E_G} = 25) \wedge (G > 11)$ are gathered in the table. The figure shows that $DL = A$ and $\overline{E_G} = 10$ puts Easter on April 9.

```
    draw_dbl_circle(.9diam1,diam1,col1[i]);
  endfor;
```

The circles in the other columns are obtained similarly.

4.7 Labeling the points

For the labels of columns I, III and VI, we first build special strings which will be used later to typeset the labels. These strings are stored in the following variables:

```
string col[][]st;
```

4.7.1 Preparing the labels

Labels in the first column are defined as follows. We go through every secular part from 0 to 84 and find the position to which it belongs, using the formulæ found earlier. There are two cases, either the string was not yet defined (in which case `unknown col1[p]st` is true and we assign its first value, or it was already defined, and we append a new value with a comma in between. The comma will be useful later, when the string is analyzed.

```
vardef define_col_one_labels=
  save p;
  for i=0 upto 84:
    p:=1+(9+3i) mod 19;
    if unknown col1[p]st:
      col1[p]st=decimal(i);
    else:
      col1[p]st
        :=col1[p]st & "," & decimal(i);
```

```
    fi;
  endfor;
enddef;
```

Labels in the third and sixth columns are defined similarly:

```
vardef define_col_three_labels=
  save p;
  for i=0 upto 99:
    p:=1+(15+12i) mod 19;
    if unknown col3[p]st:
      col3[p]st=decimal(i);
    else:
      col3[p]st
        :=col3[p]st & "," & decimal(i);
    fi;
  endfor;
enddef;
```

In the sixth column, we use the value of the Gauss constant M and the computation is only done for values of the secular part between 15 and 84, since earlier centuries lead to a constant value of M.

```
vardef define_col_six_labels=
  save p,M;
  for i=15 upto 84:
    M:=(15+i-floor(i/4)
          -floor((8i+13)/25)) mod 30;
    p:=1+(M+21) mod 30;
    if unknown col6[p]st:
      col6[p]st=decimal(i);
    else:
      col6[p]st
        :=col6[p]st & "," & decimal(i);
    fi;
  endfor;
enddef;
```

4.7.2 Column I

Once the strings for the labels have been defined, these strings can be processed and the labels can be drawn. The macro processing the labels in columns I and III is `col_one_three_f`. This macro, as well as `col_six_f`, first counts the number of elements in the `list` parameter and stores it in `n`. It does so by analyzing the comma-separated string `list` with `scantokens`, which evaluates a string as if it were normal METAPOST code.

```
vardef col_one_three_f(expr list,l,c)=
  save n,i;n=0;
  for $=scantokens(list):
    n:=n+1;
  endfor;
  i=0;
  for $=scantokens(list):
    i:=i+1;
    label(textext(if c=3: (tddec $)
                  else: decimal $ fi)
              scaled .6,
            col[c][l]
              +((if c=3: 2.5u
                 else: 2u
                 fi,0)
                  rotated
                    (180-(i-1)*360/n)));
  endfor;
enddef;
```

Now, the labels are drawn with:

```
for i=1 upto 19:
  col_one_three_f(col1[i]st,i,1);
endfor;
label(textext("I"),col1[1]-(0,10u));
label(textext("$S$") scaled 1.5,
      col1[19]+(col1[19]-col1[18]));
```

4.7.3 Column III

The labels of column III are drawn using the same macro as for column I:

```
for i=1 upto 19:
  col_one_three_f(col3[i]st,i,3);
endfor;
label(textext("III"),col3[1]-(0,10u));
label(textext("$A$") scaled 1.5,
      col3[19]+(col3[19]-col3[18]));
```

4.7.4 Column IV

The labels in column IV are drawn using the macro `gn_epactl` seen above.

It should be noted that some of the values here are written twice, but this causes no harm.

```
pair GNE;
for i:=1 upto 37:
  GNE:=whateverpair;
  GNE=gn_epactl(i);
  label.rt(gstring(xpart(GNE)),
           col4[ypart(GNE)]);
endfor;
```

4.7.5 Column VI

The labels in the circles of column VI are drawn by processing the strings `col6[]st` which were prepared above. The postprocessing is done using the macro `col_six_f`:

```
vardef col_six_f(expr list,l)=
  save n,i;n=0;
  for $=scantokens(list):
    n:=n+1;
  endfor;
  i=0;
  for $=scantokens(list):
    i:=i+1;
    if n>1:
```

If there is more than one value, the extreme values are put at $2u$ below and above the center,

and the other values (if any) are spread evenly in-between:

```
      label(textext(decimal $) scaled .7,
            col6a[l]
             +(0,-2u+(i-1)*(4u/(n-1))));
    else:
```

If there is only one value, it is centered; there is only one such case:

```
      label(textext(decimal $) scaled .7,
            col6a[l]);
    fi;
  endfor;
enddef;
```

The labels are then drawn with:

```
for i=1 upto 30:
  col_six_f(col6[i]st,i);
endfor;
label(textext("VI"),col6[1]-(0,10u));
label(textext("$S$") scaled 1.5,
      col6[30]+2(col6[30]-col6[29]));
```

4.7.6 Column VII

In this column, we merely output the value of the epact.

```
for i:=1 upto 30:
  label.rt(ep_st((24-i) mod 30),col7[i]);
endfor;
label(textext("VII"),col7[1]-(0,20u));
label.rt(textext("Epact:") rotated 90,
         col7[1]-(0,10u));
```

4.7.7 Column VIII

In this column, there are labels for the Julian calendar on the left, and labels for the Gregorian calendar on the right. In the first case, there are always three values of S in each circle, and the labels can be produced by a simple loop.

```
for i=1 upto 7:
  for j=1 upto 3:
    v:=((4+i) mod 7)+(j-1)*7;
    label(textext(decimal(v)) scaled .5,
          col8[i]-(col_shift_eight_a,0)
           +((0,1.4u)
               rotated ((j-1)*120)));
  endfor;
endfor;
```

For the Gregorian calendar, there are some irregularities, and we have decided to explicit each line of the labels. The following lines could be parameterized, but it's not worth it.

```
secular_year(1,1)(16,20,24);
secular_year(1,2)(28,32,36,40);
secular_year(1,3)(44,48,52,56);
secular_year(1,4)(60,64,68,72);
secular_year(1,5)(76,80,84);
secular_year(2,1)(17,21);
secular_year(2,2)(25,29,33,37);
secular_year(2,3)(41,45,49,53);
secular_year(2,4)(57,61,65,69);
secular_year(2,5)(73,77,81);
...
```

The macro `secular_year` is defined as follows. It distributes all lines evenly, since there are always five of them:

```
vardef secular_year(expr i,j)(text sec)=
  save vd;
  % vertical shift of the first line
  vd=4u;
  label(textext(sval(sec)(decimal))
          scaled .5,
        col8[2i-1]+(10u,vd-(j-1)*.5vd));
enddef;
```

The `sval` macro builds a string with space-separated values:

```
vardef sval(text sec)(text f)=
  save s;string s;
  for $=sec:
    if unknown s:
      s=f $;
    else:
      s:=s & " " & f $;
    fi;
  endfor;
  s
enddef;
```

4.7.8 Column IX

In this column, we show all dominical letters resulting from the combination of the two parts of the year.

```
for i=1 upto 13:
  label.rt(textext(DL(1+(3+i) mod 7)),
           col9[i]);
endfor;
label(textext("IX"),col9[1]-(0,10u));
label.rt(textext("Dom. L.") rotated 90,
         col9[1]-(0,10u));
```

4.7.9 Column X

The labels in column X fit in rounded rectangles. In order to produce these "rectangles", we use the `rboxes` package and draw a rectangular box with rounded corners. There are seven boxes, `rb1` to `rb7`:

```
rbr=rbox_radius;
rbox_radius:=15pt;
for i=1 upto 7:
  rboxit.rb[i]("");
  rb[i].c=col10[i];
  rb[i].dx=9u;rb[i].dy=3.3u;
  unfill bpath(rb[i]);
  drawboxes(rb[i]);
```

```
  endfor;
  rbox_radius:=rbr;
```

Once the boxes are drawn, their contents can be added. Each box has three lines, the upper and lower ones extend on the whole width, and the middle one is split in two parts, one left of the center, and the other one right of the center.

The upper and lower lines are produced with the `yn` (year number) macro, whose first parameter is the point number, and whose second parameter is the line number within the label, the first line being here at the top. The middle line is produced with `yn_left` and `yn_right`.

```
  yn(1,1)(3,8,14,25,31);
  yn_left(1)(36,42);
  yn_right(1)(53,59);
  yn(1,3)(64,70,81,87,92,98);
  ...
```

Then, we have the three macros for setting a year.

`yn` puts the labels at symmetric positions in the two cases in which it is called.

```
vardef yn(expr i,j)(text y)=
  save vd;
  % vertical shift of the first line
  vd=2u;
  label(textext(sval(y)(tddec)) scaled .5,
        col10[i]+(0,vd-(j-1)*vd));
enddef;
```

`yn_left` and `yn_right` just put the label at symmetric positions on the left and right of the central point selected by i:

```
def yn_left(expr i)(text y)=
  label.rt(textext(sval(y)(tddec)) scaled .5,
           col10[i]+(-10u,0));
enddef;

def yn_right(expr i)(text y)=
  label.lft(
     textext(sval(y)(tddec)) scaled .5,
     col10[i]+(10u,0));
enddef;
```

4.7.10 Column XI

In this column, we output the seven dominical letters.

```
  for i=1 upto 7:
    label.ulft(textext(DL(1+(10-i) mod 7)),
               col11[i]);
  endfor;
  label(textext("XI")
          rotated (angle(s1r0-s8l0)-90),
        .6[col11[8],col11[7]]);
```

4.8 Drawing the Easter grid

Once the various slices of the Easter grid have been defined, the grid can be drawn easily. We first draw the slots, then the labels.

4.8.1 The slots

The Easter slots are drawn with the following macro:

```
vardef draw_easter_table_slices=
  save oldpen;
  oldpen=savepen;
  % divisions between slices:
  for i=1 upto 8:
    draw s[i]l0--s[i]l5;
  endfor;
  % external boundary:
  draw s8l5--s8l0--s1l0--s1l5--cycle;
  % internal divisions:
  for i=1 upto 7:
    for j=1 upto 4:
      draw s[i]r[j]--s[i+1]l[j];
    endfor;
  endfor;
  % March/April divisions:
  pickup pencircle scaled 2pt;
  draw s8l3--s7r3--s7l3--s6r3--s6l3--
       s5r3--s5l4--s4r4--s4l4--s3r4--
       s3l4--s2r4--s2l4--s1r4;
  pickup oldpen;
enddef;
```

4.8.2 Easter grid labels

For the labels inside the Easter grid, we first define an auxiliary macro. This macro takes a slice number x and a position y within the slice, and puts the label *lab* in the middle of the corresponding slot:

```
def label_easter_slot(expr x,y,lab)=
  label(lab,.5[s[x]r[y],s[x+1]l[y+1]]);
enddef;
```

Now, the main macro filling the Easter grid slots is the following. We first fill every slot with the appropriate number, and add a special case for April 25th ($\overline{E_G} = 25$, $G > 11$ and $DL = C$):

```
vardef draw_easter_table_labels=
  save laban,march,april,note,sl,j;
  string march,april,note;
  % 35 dates from March 22 till April 25
  for i=1 upto 35:
    sl:=1+(7-(i mod 7)) mod 7;
    j:=4-floor((i-1)/7);
    label_easter_slot(sl,j,
       textext(if i=35:"25 (a)"
               else:
                 decimal(if i>10:
                           i-10
                         else:
                           i+21
```

```
                        fi)
              fi)
      if i<8:
        scaled .7
      fi);
  endfor;
```

Then, we need to add two braces, as well as the footnote. This is done as follows:

```
  laban=angle(s8l0-s8l5);
  march="$\underbrace{\kern" &
     decimal(arclength(s8l3--s8l5)-5) &
        "bp}_{\hbox{MARCH}}$";
  april="$\underbrace{\kern" &
     decimal(arclength(s8l0--s8l3)-5) &
        "bp}_{\hbox{APRIL}}$";
  note="\footnotesize "&
         "\parbox{4cm}{\raggedright "&
         "Note (a)...}";
  label(textext(march) rotated laban,
        .5[s8l3,s8l5]
        +3u*unitvector((s8l0-s8l5)
                       rotated -90));
  label(textext(apri1) rotated laban,
        .5[s8l0,s8l3]
         +3u*unitvector((s8l0-s8l5)
                        rotated -90));
  label(textext(note),C+35u*right);
enddef;
```

5 Conclusion

We have eventually completed the analysis and reconstruction of Garrigues's nomogram. To some extent, the reconstruction was straightforward, and could have been achieved without a deep understanding of the nomogram, only by a mere observation. However, a good reconstruction almost always benefits from an initial analysis, and is useful if the structure has to be explained. Such conclusions had already been made in a previous work on a complex drawing in descriptive geometry (Roegel, 2007).

6 Acknowledgements

It is a pleasure to thank Damien Wyart who, many years ago, drew our attention to Garrigues' article and led us to redraw this nomogram.

References

Bien, Reinhold. "Gauß and Beyond: The Making of Easter Algorithms ". *Archive for history of exact sciences* **58**(5), 439–452, 2004.

Dershowitz, Nachum, and E. M. Reingold. *Calendrical calculations*. Cambridge: Cambridge University Press, 2008. 3rd edition.

d'Ocagne, Maurice. *Traité de nomographie : Étude générale de la représentation graphique cotée des équations à un nombre quelconque de variables, applications pratiques*. Paris: Gauthier-Villars, 1921. 2nd edition (first edition in 1899).

Garrigues, Damien. "Généralisation de la formule pascale de Gauss: Nomogramme du Calendrier perpétuel". *Annales françaises de chronométrie* pages 47–60, 1939.

Gauss, Karl Friedrich. "Berechnung des Osterfestes". *Monatliche Correspondenz zur Beförderung der Erd- und Himmels-Kunde, herausgegeben vom Freiherrn Franz Xavier G. von Zach* **2**, 121–130, 1800. In: (Gauss, 1973), volume 6, pages 73–79.

Gauss, Karl Friedrich. "Berichtigung zu dem Aufsatze: Berechnung des Osterfestes. Mon. Corr. 1800 Aug, S. 121". *Zeitschrift für Astronomie und verwandte Wissenschaften, herausgegeben von B. von Lindenau und J. G. F. Bohnenberger* **1**, 158, 1816. Correction of (Gauss, 1800). In: (Gauss, 1973), volume 11/1, page 201. Notes by Alfred Loewy.

Gauss, Karl Friedrich. *Werke*. New York: G. Olms, 1973. Reprint of the 1863–1933 edition.

Goossens, Michel, F. Mittelbach, S. Rahtz, D. Roegel, and H. Voss. *The LaTeX Graphics Companion, Second Edition*. Boston: Addison-Wesley, 2008.

Hobby, John. "METAPOST: A User's Manual". 2008. Updated version of the original manual; available at `http://tug.org/docs/metapost/mpman.pdf`.

Knuth, Donald Ervin. *Fundamental Algorithms, volume 1 of The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley, 1997.

Roegel, Denis. "The missing new moon of A.D. 16399 and other anomalies of the Gregorian calendar". 2004. Unpublished.

Roegel, Denis. "A complex drawing in descriptive geometry". *TUGboat* **28**(2), 218–228, 2007.

◇ Denis Roegel
LORIA — BP 239
54506 Vandœuvre-lès-Nancy cedex, France
`roegel (at) loria dot fr`
`http://www.loria.fr/~roegel`

LaTeX3 News

Issue 1, February 2009

Welcome to LaTeX3

Momentum is again starting to build behind the LaTeX3 project. For the last few releases of TeX Live, the experimental programming foundation for LaTeX3 has been available under the name expl3. Despite large warnings that the code would probably change in the future, we wanted to show that there was progress being made, no matter how slowly. Since then, some people have looked at the code, provided feedback, and — most importantly — actually tried using it. Although it is yet early days, we believe that the ideas behind the code are sound and there are only 'cosmetic improvements' that need to be made before expl3 is ready for the LaTeX package author masses.

What currently exists

The current LaTeX3 code consists of two main branches: the expl3 modules that define the underlying programming environment, and the 'xpackages', which are a suite of packages that are written with the expl3 programming interface and provide some higher-level functionality for what will one day become LaTeX3 proper. Both expl3 and parts of the xpackages are designed to be used *on top* of LaTeX 2$_\varepsilon$, so new packages can take advantage of the new features while still allowing to be used alongside many of the vast number of LaTeX 2$_\varepsilon$ packages on CTAN.

What's happening now

In preparation for a minor overhaul of the expl3 code, we are writing a comprehensive test suite for each module. These tests allow us to make implementation changes and then test if the code still works as before. They are also highlighting any minor shortcomings or omissions in the code. As the tests are being written, our assumptions about what should be called what and the underlying naming conventions for the functions and datatypes are being questioned, challenged, and noted for further rumination.

At the time of writing, we are approximately halfway through writing the test suite. Once this task is complete, which we plan for the first half of 2009, we will be ready to make changes without worrying about breaking anything.

What's happening soon

So what do we want to change? The current expl3 codebase has portions that date to the pre-LaTeX 2$_\varepsilon$ days, while other modules have been more recently conceived. It is quite apparent when reading through the sources that some unification and tidying up would improve the simplicity and consistency of the code. In many cases, such changes will mean nothing more than a tweak or a rename.

Beyond these minor changes, we are also re-thinking the exact notation behind the way functions are defined. There are currently a handful of different types of arguments that functions may be passed (from an untouched single token to a complete expansion of a token list) and we're not entirely happy with how the original choices have evolved now that the system has grown somewhat. We have received good feedback from several people on ways that we could improve the argument syntax, and as part of the upcoming changes to the expl3 packages we hope to address the problems that we currently perceive in the present syntax.

What's happening later

After the changes discussed above are finished, we will begin freezing the core interface of the expl3 modules, and we hope that more package authors will be interested in using the new ideas to write their own code. While the core functions will then remain unchanged, more features and new modules will be added as LaTeX3 starts to grow.

Some new and/or experimental packages will be changing to use the expl3 programming interface, including breqn, mathtools, empheq, fontspec, and unicode-math. (Which is one reason for the lack of progress in these latter two in recent times.) There will also be a version of the siunitx package written in expl3, in parallel to the current LaTeX 2$_\varepsilon$ version. These developments will provide improvements to everyday LaTeX users who haven't even heard of the LaTeX3 Project.

Looking towards the long term, LaTeX3 as a document preparation system needs to be written almost from scratch. A high-level user syntax needs to be designed and scores of packages will be used as inspiration for the 'out-of-the-box' default document templates. LaTeX 2$_\varepsilon$ has stood up to the test of time — some fifteen years and still going strong — and it is now time to write a successor that will survive another score.

LaTeX3 News

Issue 2, June 2009

TeX Live and the expl3 code

TeX Live 2009 is almost upon us, and the LaTeX3 team have been readying a new release of the experimental LaTeX3 code for this. Very dramatic changes have occurred since the last public release of the code in TeX Live 2008; no backwards compatibility has been maintained (as warned in the beginning of the documentation) but we believe the changes made are all much for the better. Almost every single part of expl3 has been scrutinized, resulting in a far more coherent code base.

The expl3 code is now considered to be much more stable than it was before; a comprehensive test suite has been written that helps to ensure that we don't make any mistakes as we change things in the future. In the process of writing the test suite, many minor bugs were fixed; we recommend such test suites for all similar developmental projects! Some small underlying changes are still expected in the expl3 code, but major, disruptive, changes aren't planned.

Planned updates

Until now, the last update to CTAN of the expl3 bundle was for TeX Live 2008. Now that work on the code is happening on a semi-steady basis, we plan to keep updates rolling out to CTAN more frequently. This will allow anyone who wishes to experiment with the new code to use the TeX Live or MiKTeX updaters to install a recent version without having to 'check out' the SVN repository and install the packages manually.

New members

We didn't say anything about it in the last status update, but Joseph Wright and Will Robertson are now members of the LaTeX Team. They have been working fairly exclusively on the expl3 code.

It's worth repeating that LaTeX 2ε is essentially frozen in order to prevent any backwards compatibility problems. As desirable as it is to benefit from the new features offered by new engines XeTeX and LuaTeX, we cannot risk the stability of production servers running older versions of LaTeX 2ε which will inevitably end up processing documents written into the future.

LaTeX3 will not be inheriting the same restraints, so stay tuned.

Some specifics

Morten Høgholm will be presenting the recent changes in much more detail at TUG 2009. Here are some quick specifics for those interested. New code written and broad changes made to the expl3 modules:

More logical function names Many function names that were hold-outs from the TeX naming system have been changed to fit into the more logical scheme of expl3; e.g., `\def:Npn` and `\let:NN` are now `\cs_set:Npn` and `\cs_set_eq:NN`.

Defining functions and conditionals Much thought was put into new ways to define functions and conditionals with a minimum of code. See `\cs_set:Nn` and `\prg_set_conditional:Nnn`.

Smart comparisons Comparisons can be made much more easily now, with familiar notation such as `\intexpr_compare_p:n{ #1+3 != \l_tmpa_int }`.

Data from variables A new function argument specifier `V` has been added for extracting information from variables of different types, without needing to know the underlying variable structure. Some other tidy-ups on the argument specifiers offered, partially as a result of the addition of this new one.

l3msg New module to deal with communication between LaTeX3 code and the user (info messages, warnings, and errors), including message filtering partially inspired by the silence package.

The next six months

Having overhauled the expl3 code, we now plan to perform an analogous process with the foundations of the xpackages. These are the higher-level packages that will provide the basic needs such as control of the page layout and rich document-level interaction with the user. As the groundwork for this layer of the document processing matures, we will be able to start building more packages for a LaTeX3 kernel; these packages will also be usable on top of LaTeX 2ε and serve as broadly customisable templates for future document design.

As gaps in the functionality offered by expl3 are found (in some cases, we know that they exist already), the programming layer will be extended to support our needs. In other cases, wrappers around TeX functions that can be more usefully handled at a higher level will be written.

LaTeX3 programming: External perspectives

Joseph Wright

Abstract

The current experimental LaTeX3 packages provide a new, documented programming interface for TeX. The key ideas implemented in this new interface are highlighted in this article.

1 Introduction

Modifying the behaviour of LaTeX 2_ε often requires a combination of user macros, internal LaTeX macro and TeX primitives. This makes even trivial modifications of document layout potentially difficult, even for the experienced LaTeX user. The differing syntax used by TeX primitives and the LaTeX kernel only add to the confusion here.

The first step to develop a new LaTeX kernel is therefore to address how the underlying system is programmed. Rather than the current mix of LaTeX and TeX macros, the experimental LaTeX3 system provides its own consistent interface to all of the functions needed to control TeX. A key part of this work is to ensure that everything is documented, so that LaTeX users can work efficiently without needing to be familiar with the internal nature of the kernel or with plain TeX.

The current kernel also suffers from the mixing of design commands with structural code. Thus changing a layout element often requires modifying a kernel code block (or loading a package which provides an interface to achieve this). The second challenge for LaTeX3 is therefore separation of the basic tools of the kernel from the design of documents.

This short overview article highlights the key developments to date in LaTeX3. It is based on my own experience working with the new tools for writing packages, and a talk given recently to the UK TeX Users Group.

2 The components of LaTeX3

Currently, the experimental LaTeX3 packages are designed to be used "on top of" LaTeX 2_ε. This avoids needing to wait for the entire kernel to be finished before testing what is written.

The most developed part of the code is the expl3 ("experimental LaTeX3") bundle, the core of the new kernel providing the new programming interface. The new language is fully documented in the file `source3.pdf`, which contains some notes for the experienced (La)TeX programmer.

Built on top of expl3 is the xparse package. This is meant to be a "bridge" between the internal and user parts of the new kernel. The xparse package is used to create new user macros, in a much more controlled way than is possible using `\newcommand`.

More experimental than xparse are various other "xpackages". These are designed to explore new approaches to layout and document design for LaTeX3.

The most complete part of LaTeX3 is the expl3 bundle. The rest of this article is focussed mainly on the new internal syntax introduced in expl3.

3 A new internal syntax

LaTeX3 does not use `@` as a "letter" for defining internal macros. Instead, the symbols `_` and `:` are used in internal macro names to provide structure. In contrast to the plain TeX format and the LaTeX 2_ε kernel, these extra letters are used only between parts of a macro name (no strange vowel replacement).

LaTeX3 separates macros which do something (functions) from ones which only store data. The general form of an internal function in LaTeX3 is $\backslash\langle module\rangle\_\langle function\rangle:\langle arg\text{-}spec\rangle$.

- The $\langle module\rangle$ prefix is applied to almost all macros. For a package, it will typically be the package name; the kernel is split into a number of modules, each with its own name.
- The name of the $\langle function\rangle$ should give a good description of what it does: this may contain one or more `_` characters to divide the name into logical units.
- The concept of the $\langle arg\text{-}spec\rangle$ is potentially confusing to existing (La)TeX programmers. This *argument specifier* describes the arguments expected by the function. In most cases, each argument is represented by a single letter. The letter, including its case, conveys information about the type of argument required. The use of the $\langle arg\text{-}spec\rangle$ is illustrated later in this article.

3.1 Primitives renamed

All of the TeX primitives are given new names by expl3, although many are not intended to be used outside of the LaTeX3 kernel. Instead, a number of LaTeX wrappers for primitives are provided, so that the argument syntax is consistent.

At the most basic level, the `\fi` primitive becomes `\fi:`, indicating no arguments are required.

A more complex example is `\ifdefined` (an ε-TeX primitive), which becomes `\if_cs_exist:N`.

```
\if_cs_exist:N \Macro_One
  % Do Stuff
\fi:
```

Here, the $\langle arg\text{-}spec\rangle$ contains one letter, showing that only one argument is required. This argument is of

type N, meaning that it should be a single token *not* surrounded by braces.

3.2 Example kernel functions

Renaming primitives helps to keep the new syntax consistent, but does not show why the argument specifier is useful. This is perhaps best seen by looking at some of the functions provided by expl3.

By using the argument specifier, the new kernel provides families of related functions which avoid the need for complex \expandafter runs. For example, the TEX primitive \let can only be used with a macro name and a single token; no braces. In LATEX3, the family of \let-like macros contains:

```
\cs_set_eq:NN \Macro_One \Macro_Two
\cs_set_eq:Nc \Macro_One {Macro_Two}
\cs_set_eq:cN {Macro_One} \Macro_Two
\cs_set_eq:cc {Macro_One} {Macro_Two}
```

where an argument specified as c is to be given in braces and should expand to a csname. This is much clearer than the equivalent plain TEX constructions; taking \cs_set_eq:Nc as an example:

```
\expandafter\let\expandafter\Macro_One
  \csname Macro_Two\endcsname
```

The specifiers n (no expansion), o (expand once) and x (\edef-like full expansion) allow large families of related functions to be created easily, so that using the results is simplified. Thus we can create a macro \Macro_One:nn, then create \Macro_One:no, \Macro_One:xn and so on very rapidly. Later, we will see how the v and V argument specifiers add even more power to this concept.

The argument specifier concept also makes testing much easier. As an example, the new kernel provides three tests related to the \@ifundefined macro:

```
\cs_if_exist:cT  {csname} {true}
\cs_if_exist:cF  {csname} {false}
\cs_if_exist:cTF {csname} {true} {false}
```

In all three cases, the first argument will be converted to a csname (the c specifier). The first two functions then require one more argument, either T or F. As might be expected, these are executed if the test is true or false, respectively. The third function (ending :cTF) has both a true and false branch. By providing tests with the choice of T, F and TF arguments, empty groups in code can be avoided and meaning is much more obvious.

4 Data storage

In LATEX3, macros which carry out some process are called functions, and all contain an argument specifier. Macros used for storage are handled separately, to help to make code cleaner and easier to read. To further aid the programmer, expl3 defines several new data types:

- token lists (tl),
- comma lists (clist),
- property lists (prop),
- sequences (seq),

in addition to the existing types, which are renamed:

- boolean switches (bool),
- counters (int),
- skips (skip),

and so on.

The name "token list" may cause confusion, and so some background is useful. TEX works with tokens and lists of tokens, rather than characters. It provides two ways to store these token lists: within macros and as token registers (toks). LATEX3 retains the name "toks" for the later, and adopts the name "token lists" (tl) for macros used to store tokens. In most circumstances, the tl data type is more convenient for storing token lists.

The other new variable types are all essentially lists of items separated by a special token. The nature of the separator determines the type of variable and what functions apply. For example, a comma list is, as you might expect, a set of tokens separated by commas.

These are all created explicitly as either local or global, according to a prefix \l_ or \g_. For example, a local tl may be named:

```
\l_mymodule_myname_tl
```

while a global tl looks like this:

```
\g_mymodule_myname_tl
```

The other variable types follow the same pattern, with the appropriate type identified in the variable name.

As well as the new data types, expl3 provides a range of functions for manipulating data. Often, these had to be coded by hand when using LATEX 2_ε. For example, \tl_elt_count:N is available to count the number of elements (often characters) in a token list.

5 Expanding variables

When coding in (LA)TEX, the need to access data in variables is made more complicated by the different possibilities for recovering information later. For example, if three macros are defined as

```
\def\tempa{Some text}
\def\tempb{\tempa}
\def\tempc{\tempb}
```

then there are two likely scenarios for using the information in `\tempc`:

- Use of the value that `\tempc` contains (in this case `\tempb`);
- Exhaustive expansion of `\tempc` to use the unexpandable token list it represents (in this case "Some text").

The situation is further complicated as macros do not need an accessor function, whereas other TEX variables (toks, counts, skips) do. This leads to the need for carefully-constructed `\expandafter` runs in (LA)TEX, in order to get the content needed.

To avoid this, LATEX3 provides two argument specifiers which will always return the content of a variable. The `V` specifier requires the name of a variable, and returns the content. For example, if we define two variables, one of type `tl` and the other of type `toks`,

```
\toks_set:Nn \l_my_toks { Text \mymacro }
\tl_set:Nn  \l_my_tl  { Text \mymacro }
```

and pass them to some function `\foo_bar:V`,

```
\foo_bar:V \l_my_toks
\foo_bar:V \l_my_tl
```

both sets of input will result in "`Text \mymacro`" being passed as the argument to the "underlying" function (explained below) `\foo_bar:n`. The `V` specifier can be applied to any LATEX3 variable: this means that the programmer does not have to worry about how data is stored at a TEX level. A function using a `V` specifier will always receive the content of the variable passed.

The second "variable" specifier is `v`. This converts its argument to a csname, then recovers the content of the resulting variable and passes the content. Thus we might use a `\foo_bar:v` as:

```
\foo_bar:v { l_my_toks }
\foo_bar:v { l_my_tl }
```

with the same result as the previous example.

The two variable specifiers are very powerful. By using them, the programmer can almost entirely avoid the need to worry about the order of expansion when using stored information.

In LATEX3, functions which differ only in the argument specifier should carry out the same underlying operation: the only difference should be the processing of arguments *prior* to applying the function. Normally, the "underlying" function will act without argument expansion (taking `n` or `N` type arguments). Thus `\foo_bar:c` will normally be defined as expanding a csname and passing it to `\foo_bar:N`.

6 Other key features

The new kernel will require the ε-TEX extensions. Thus, those new primitives are always available when working with LATEX3. For example, `\unexpanded` is part of the expansion module, as `\exp_not:n`.

Boolean switches in TEX and LATEX 2ε use the `\iftrue` and `\iffalse` primitives. This can lead to problems nesting (`! Incomplete \if...`). To avoid this, LATEX3 does not create switches in the same way. This means that all of the switches use exclusively LATEX syntax, and require an "access" function.

```
\bool_if:NT  \l_example_bool { true code }
\bool_if:NF  \l_example_bool { false code }
\bool_if:NTF \l_example_bool { true code }
  { false code }
```

One of the most useful features of the new coding syntax is the treatment of white space. The literal space character ( ) is ignored inside code blocks, meaning that the text can be laid out to aid ease of reading. When a space is required in the output, a tilde (`~`) can be used. In this context, `~` is *not* a "hard" space, but a character with category code 10. The ability to finish lines without worrying about omitting or including `%` is highly welcome!

7 Conclusions

The current LATEX3 modules provide a new and powerful programming language for TEX. The full details of the language are collected in one place, and the language is much more logical than the current mix of TEX and LATEX 2ε. LATEX3 is therefore ready for serious use by (LA)TEX programmers.

At this stage, the document level of LATEX3 is much less defined. It seems likely that good separation of programming and document design will be made available. The new code syntax means that a number of ideas currently implemented as independent packages will need to be re-implemented either in the new kernel or as supported tools.

My own experience with LATEX3 convinces me that the kernel team need outsiders to use the code. The team has done a very good job so far, but everyone will bring new approaches to using the code. With the involvement of the wider TEX community, LATEX3 has the potential to be a major step forward for LATEX.

◇ Joseph Wright
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2 dot co dot uk

Macros

Implementing key–value input: An introduction

Joseph Wright and Christian Feuersänger

Abstract

The key–value system is justly popular as it greatly simplifies controlling packages for the user. Unfortunately, that ease of use is not transferred into setting up key–value systems for authors of pre-packaged TEX code. This article describes how to implement key–value controls for both TEX and LATEX authors, including a brief overview of how the underlying system works. As well as the original keyval package, the various extended keyval-based packages are covered, looking at the relative advantages of each system. Looking beyond keyval-based systems, an overview of the pgfkeys package is also given.

1 Introduction

The key–value method uses a comma-separated list of ⟨*key*⟩=⟨*value*⟩ to set one or more ⟨*key*⟩s. The code applied when a ⟨*key*⟩ is given can undertake a range of processing on the ⟨*value*⟩. Almost every (LA)TEX user will have come across the power of the this method for providing control values. The interface is increasingly widespread in controlling package and class behaviour. It offers a much cleaner method for managing large numbers of options or control values than defining multiple single-use macros and complex optional arguments.

The original keyval package (Carlisle, 1999) provides a core of functionality. This has been extended by xkeyval (Adriaens, 2008), kvoptions (Oberdiek, 2009a) and kvsetkeys (Oberdiek, 2009b), providing additional tools for the developer, and making key–value input available for LATEX package and class options.

Unfortunately, the ease of key–value input for the user has not translated into easy development of new uses of key–value syntax in package control. Many (even experienced) (LA)TEX code authors struggle to make a start with implementing key–value methods. This article aims to make key–value input more accessible. The major use of key–value syntax is controlling LATEX packages and classes, and this is reflected in the focus here. However, all of the key–value implementations are compatible to some extent with plain TEX. A short section on use with plain TEX is included here, and as far as possible all of the examples use only plain TEX macros. Throughout the article, "package" is used to refer to a LATEX package, LATEX class or other file using key–value input.

The pgfkeys system implements a key–value interface in a somewhat different manner from the various keyval-derived packages. As a result, it has unique strengths. Due to the differing approaches of the keyval-based systems and pgfkeys, the latter is covered in its own section. Many of the concepts from the keyval package and its derivatives apply to pgfkeys, and so the general introduction is useful even for users who have already decided on pgfkeys.

The various packages discussed have a range of features not covered in this article: in order to remain accessible, only the most widely-applicable concepts are discussed. Some simplifications have also been made where these will not impede the more advanced user. More detail can of course be found in the various package documentation. There is also a *TUGboat* article covering the design and some of the more advanced features of xkeyval (Adriaens and Kern, 2005).

2 How key–value works

There are two parts to using the key–value system: defining keys, and assigning values to keys. When using the keyval package itself, these tasks are handled by the macros `\define@key` and `\setkeys`, respectively.

The *key* in key–value input is the "name" of a data item. The model used by keyval divides keys into *families*: groups of keys that can be processed together. The `\define@key` macro is used to define keys. This requires three pieces of information: the key name, the family to which the key belongs, and a handler for the key. Consider a package fam defining a key `key`, which simply prints the value given:

```
\define@key{fam}{key}{#1}
```

As can be seen, `\define@key` takes three arguments, ⟨*family*⟩, ⟨*key*⟩ and ⟨*handler*⟩. The ⟨*handler*⟩ receives the value given for the key as macro argument `#1`, and can consist of any TEX code appropriate to process the *value* assigned to the key (the part after the equals sign).

How does `\define@key` work? A new macro \⟨*prefix*⟩`@`⟨*family*⟩`@`⟨*key*⟩ is defined, with expansion ⟨*handler*⟩. So in the example above, the following would achieve the same effect:

```
\def\KV@fam@key#1{#1}
```

Here, the *prefix* is a code added to the beginning of the key name, and acts as a family of families. The prefix is fixed with the value `KV`: only xkeyval allows this to be varied.

The \setkeys macro is then used to set key values, the second part of the key–value concept. The input to \setkeys is a comma-separated list: each comma-separated ⟨*key*⟩=⟨*value*⟩ pair is therefore processed in turn. Unlike the majority of TEX macros, this process ignores spaces between key–value pairs:

```
\setkeys{fam}{
  key one=value 1 ,
  key two=value2
}
```

consists of two key–value pairs "key␣one=value␣1" and "key␣two=value2". Notice that both the key name and the value can contain spaces. Braces must be used to protect literal "," and "=" characters inside \setkeys:

```
\setkeys{fam}{
  key three={value1,value2},
  key four={some=stuff}
}
```

For each pair found, \setkeys then attempts to separate the data into a key and a value, delimited by an equals sign. If there is no equals sign, an error will normally be raised. Assuming a value is found (even an empty one, if there is nothing after "="), \setkeys looks for a macro of the form \⟨*prefix*⟩@⟨*family*⟩@⟨*key*⟩ to handle the input. If such a macro exists, it is executed with the value as argument #1. If no macro is found, the key is regarded as undefined, and an error is raised. In the example earlier, the result of the \setkeys operation is to supply the key macro for key one with "value 1", and that for key two with "value2".

\setkeys passes the value to the processing macro as is. Thus macro names, *etc.*, can be used without worrying about expansion in the process.

3 Defining keys

As outlined in the previous section, a key is defined by creating a suitably-named macro. However, defining every key using \def or \newcommand would add considerably to the effort of using key–value input. All of the packages discussed here provide more convenient methods.

3.1 Using the keyval package

The \define@key macro for key definition is the only method that the original keyval package provides. However, this is the most powerful method for defining a key: the developer is completely free to code any handler required. One particularly common process is to store the value in a macro to be used later:

```
\define@key{fam}{key}{\def\fam@data{#1}}
```

This stores the value given for key in \fam@data. The definition of the storage macro does not occur until the key is used for the first time. Thus if the macro must be defined even if the key has not been used, an additional line is necessary:

```
\def\fam@data{initial}
\define@key{fam}{key}{%
  \def\fam@data{#1}%
}
```

Setting the key key will then redefine \fam@data to contain whatever value is passed to the key. Notice that here the key family has been used as the start of the storage macro name.

As was explained in Section 2, keys must have a value (even if this is empty). It is possible to specify a default value for a key, which is then used if the user does not supply one (this does *not* mean that the key is defined before it is first used!). A default value is supplied as an optional argument to the \define@key macro, which following the LATEX convention appears in square brackets:

```
\define@key{fam}{key}[default]{%
  \def\fam@data{#1}%
}
```

This means that

```
\setkeys{fam}{key}
```

is interpreted as though the user had written

```
\setkeys{fam}{key=default}
```

The handler macro receives the default value in exactly the same way as user-supplied data.

Using the "raw" \define@key macro rapidly becomes awkward when a large number of similar keys are required. Package authors can of course write short-cut macros to make the process easier. However, the other key–value packages seek to address this issue by making one or more common key definitions available directly.

3.2 Using kvsetkeys

Using kvsetkeys adds several "low-level" functions to keyval; those related to setting keys will be addressed later. kvsetkeys does not add any methods for processing *known* key names, and indeed relies on the explicit loading of keyval to define keys. It does, however, add a customised handler for key names which have not been defined.

When using the kvsetkeys package, a handler for unknown keys in a family is created using the macro \kv@set@family@handler. This allows data input for arbitrary key names, or perhaps simply a customised warning or error message. The name of the key used is available as #1. A simple warning could be given by:

```
\kv@set@family@handler{fam}{%
  \wlog{Warning: key '#1'
    unknown by package fam}
}
```

A more complex example might be to use the input to define a new macro. The value given for the key (if any) is available as `#2`. For example,

```
\kv@set@family@handler{fam}{%
  \expandafter\def\csname
    fam@user@#1\endcsname{#2}%
}
```

creates a new internal macro including the name of the unknown key to store the given value. Notice that the definition includes a marker that this is a user-provided key name (`\fam@user@`), as no check has been made for an existing definition.

3.3 Using kvoptions

As the package name indicates, kvoptions helps LaTeX developers use key–value input for package and class options. However, as we will see later, there is no fundamental difference between defining keys and defining key–value package options.

The kvoptions package makes life easier for the author by allowing the family value to be defined once, and then used in all subsequent key definitions. It also automatically generates various macros for the package author:

```
\SetupKeyvalOptions{
  family = fam,
  prefix = fam@
}
```

This defines the family as `fam`, and prefixes all new storage macros with `\fam@`. This does *not* affect the key prefix, used for the key macros themselves, which still start with `\KV@...`. Usually, the ⟨*prefix*⟩ given here will be simply ⟨*fam*⟩`@`, as this means all storage macros are defined as `\fam@...`. The rest of this section assumes this convention is used, and that the setup above applies. If no data has been supplied using `\SetupKeyvalOptions`, the family and macro prefix are taken from the name of the current package.

The kvoptions package provides macros for defining new keys (or options):

- `\DeclareBoolOption`;
- `\DeclareComplementaryOption`;
- `\DeclareStringOption`.

The names of the macros are a good guide to the general method key type they produce. kvoptions also provides methods applicable only to package options: these are discussed later.

`\DeclareBoolOption` creates a true/false key. Giving the key name alone is the same as giving it with the `true` value. A new switch is created which is named `\if`⟨*fam*⟩`@`⟨*key*⟩, which works in the same way as though created using `\newif`.

```
\DeclareBoolOption{active}
% Other code
\iffam@active
  % Do stuff
\else
  % Do nothing
\fi
```

`\DeclareComplementaryOption` creates a complementary key to an existing Boolean key. The most common example might be setting draft *versus* final:

```
\DeclareBoolOption{final}
\DeclareComplementaryOption
  {draft}{final}
% Other code
\iffam@final
  % Do final stuff
\else
  % Do draft stuff
\fi
```

In this way, the same switch may be set by keys with differing names.

`\DeclareStringOption` creates a new storage macro, to hold the data provided as the key value. This is similar to the `\define@key` method for saving to a macro given earlier.

```
\DeclareStringOption{key}
```

stores the value given in the macro `\fam@key`. An initial value can be provided for the option, so that `\fam@key` will be defined under all circumstances. This uses a LaTeX optional argument;

```
\DeclareStringOption[initial]{key}
```

has a similar result to

```
\def\fam@data{initial}
\define@key{fam}{key}{%
  \def\fam@key{#1}%
}
```

so that `\fam@key` will expand to "initial", until the key is set to an explicit value.

3.4 Extended keyval: xkeyval

The xkeyval package extends the key–value system further than any of the other packages. As a result, it has a much richer (and more complex) command syntax. The first point to note is that, unlike the other packages discussed, xkeyval allows the developer to alter the key prefix. This is achieved by adding an optional argument to `\define@key`:

```
\define@key{fam}{key}{#1}
\define@key[pre]{fam}{key}{#1}
```

The first command defines `\KV@fam@key` as the key-handling macro; the second defines `\pre@fam@key`.

If no explicit key prefix is given, the value `KV` is used. Of course, altering the key prefix means that `\setkeys` also needs to be modified to accommodate it. To set the two keys above, the appropriate `\setkeys` commands would be

```
\setkeys{fam}{key=input}
\setkeys[pre]{fam}{key=input}
```

Notice that, in contrast to kvoptions, there is no method to pre-set the family, *etc.* As a result, when defining a large number of keys it is often convenient to first create customised definition macros:

```
\def\fam@define@key{\define@key{fam}}
\def\fam@define@mykey
  {\define@key[pre]{fam}}
```

As is the case with kvoptions, xkeyval provides an extended set of key definition types:

- `\define@key`;
- `\define@boolkey`;
- `\define@boolkeys`;
- `\define@cmdkey`;
- `\define@cmdkeys`;
- `\define@choicekey`.

The extended version of `\define@key` has already been discussed. The concept of key prefix applies to all of the other key types, although the remaining examples all use the default `KV` prefix. If the prefix is given, it is always the first, optional, argument to the definition macro.

The `\define@boolkey` macro creates a single Boolean key. The key definition requires a function, even though this may be blank. To allow the key name alone to be used as equivalent to `key=true`, a default value is needed. This follows the LaTeX convention of appearing in square brackets, but is not the first argument given: instead, it follows the key name, for example,

```
\define@boolkey{opt}{key}[true]{}
```

creates a new switch `\ifKV@fam@key`, and a key-processing macro `\KV@fam@key` with no customised function attached: the `\if` is simply set appropriately. The name of the new switch can be altered using a second option argument to specify the macro prefix. This again appears in square brackets, between the family and key names:

```
\define@boolkey{opt}[fam@]
  {key}[true]{}
```

creates the switch `\iffam@key`, and is functionally equivalent to the `\DeclareBoolOption` macro from kvoptions.

Several Boolean keys can be created in one go using `\define@boolkeys`. Here, no custom function is needed (or indeed permitted). A default value is still needed to allow use of the key name alone:

```
\define@boolkeys{opt}[fam@]
  {key,key two,key three}[true]
```

Using `\define@cmdkey` creates a storage macro for the value given, along with a processing macro. This can become somewhat complicated, and so some examples are needed.

```
\define@cmdkey{fam}{key}{}
```

creates a new key macro `\KV@fam@key`, which will store the input in `\cmdKV@fam@key`. The name of the storage macro can be altered by adding a macro prefix argument, as with Boolean keys:

```
\define@cmdkey{fam}[fam@]{key}{}
```

The name of the *key* macro is unchanged, but the storage macro is now called `\fam@key`. Notice that both examples include a final processing argument: in these examples this is blank as storage of the input alone is required. A default can be given for a command key, as an optional argument after the key name:

```
\define@cmdkey{fam}[fam@]{key}
  [default]{}
```

The `\define@cmdkeys` macro allows the creation of several keys at one go, using a comma-separated list. Only one default is available for all of the commands, and a custom function cannot be given. In many cases, this will not be an issue as the stored value is the aim of the key. For example, to create three command keys `key`, `key two` and `key three`:

```
\define@cmdkeys{fam}[fam@]
  {key,key two,key three}
```

For large numbers of storage keys, this method is preferable to multiple calls to `\define@cmdkey`.

Finally, `\define@choicekey` allows creation of a key with a limited number of valid input values from an arbitrary list. This key type has several optional arguments which make it somewhat difficult to set up without experimentation. At the most basic, the value is checked by xkeyval and is then passed to key handler function:

```
\define@choicekey{fam}{key}
  {val1,val2,val3}
  {You chose: #1}
```

Here, the `key` key can take only the values `val1`, `val2` and `val3`. The `*` modifier makes the comparison by `\define@choicekey` case-insensitive.

```
\define@choicekey*{fam}{key}
  {val1,Val2,VAL3}
  {You chose: #1}
```

will match `key=val1`, `key=Val1`, *etc.* In these examples, the processing macro simply displays the

user's choice. Further processing of keywords is possible in this argument, for example to set several switches based on a keyword. Adding the + modifier to \define@choicekey makes a second handler available for items not on the list:

```
\define@choicekey+*{fam}{key}
  {val1,val2,val3}
  {You chose: #1}
  {\wlog{Invalid choice '#1': you
    must put 'key=val1', 'key=val2'
    or 'key=val3'}%
  }
```

Here, valid choices act as in the previous example. Any other value will use the second handler, which in this case simply writes a warning to the log.

The macros outlined above all have more extended syntax, with additional optional arguments. This more complex area has been covered by the authors of xkeyval (Adriaens and Kern, 2005).

4 Setting keys: user interface

As described in Section 2, the keyval package sets key values using the \setkeys macro. The same is true for kvoptions and xkeyval (the latter overloads its own modified version of the macro). In contrast, kvsetkeys uses the \kvsetkeys macro; this is designed to be more robust than \setkeys as defined by keyval, and to cope better with altered catcodes for "," and "=".

The \kvsetkeys macro can also set keys from the other packages, provided they use the key prefix KV. Thus the only keys that cannot be set by \kvsetkeys are those produced using xkeyval with a non-standard key prefix. In the following discussion, \setkeys could therefore be replaced by \kvsetkeys.

The \setkeys macro needs to know the family (and potentially prefix) to which keys belong. Often, and especially when developing a package, a user macro which already contains this information is desirable. The usual method is to define a custom setup macro:

```
\def\famsetup#1{\setkeys{fam}{#1}}
```

An optional key–value argument to user macros is often defined, so that settings apply only to that instance of the macro. Provided the processing of the macro occurs inside a group, this is easy to achieve (using LaTeX for convenience):

```
\newcommand*{\mycmd}[2][]{%
  % #1 is the optional keyval argument
  % #2 is a mandatory argument
  \begingroup
    \setkeys{fam}{#1}%
    % Do stuff with #2
  \endgroup}
```

4.1 \kvsetkeys *versus* \setkeys

Using \kvsetkeys adds three major refinements to the keyval \setkeys macro. Firstly, \kvsetkeys reliably sets keys when the catcodes for "," and "=" are non-standard. This is important when using packages that make the equals sign active, for example the turkish option of babel. The xkeyval version of \setkeys also handles these cases correctly.

Secondly, both \kvsetkeys and \setkeys remove some braces from value input. \kvsetkeys aims to be more predictable. It removes only one set of curly braces, whereas \setkeys may remove one or two sets of braces, depending on circumstances.

Finally, \kvsetkeys supports the unknown key handler. This will be many authors' motivation to use kvsetkeys: handling unknown keys otherwise requires adding custom low-level code.

5 LaTeX package and class options

The preceding sections apply to using key–value methods in a wide variety of situations. One of the most common aims of authors considering key–value input is to use it for processing LaTeX package or class options. This has particular points to consider, and therefore specialised macros have been made available for this area.

Any key defined when processing occurs is available as an option. This means that options can be created using \define@key or any of the higher-level macros listed here. It also means that any key–value option is also a valid key. This may not always be desirable, and is considered further in Section 6.1.

Before using key–value options, the careful developer should know the limitations of the system. Before package options are passed to the key–value system, they are processed by LaTeX. The kernel removes all unprotected spaces from the input, which means that key names' spaces will be rendered useless. Secondly, unlike direct use of \setkeys, the kernel will expand the input. This means that some keys should *not* be given as options to a package.

Although patches exist to deal with these problems, these are not generally useful: the patches must be loaded before input of the package or class requiring them! This leaves the package author with two options. The first approach is to abandon key–value load-time options, with a setup macro used only after loading the package. More commonly, the options can be designed to minimise the impact of the problem. Design steps to achieve this include:

- Avoiding any key names containing spaces;
- For keys which will receive values containing spaces, initially defining the key to gobble the

value with a warning, then redefining it after processing options to the real meaning (see Section 6.1);
- For keys that will require a single macro, requiring the csname rather than the macro itself, then using `\csname` ... `\endcsname` in the implementation.

To allow key–value syntax to be used in package options, the standard LaTeX method for handling option input has to be modified. This can be done directly, but copy–pasting code is not normally considered good programming. xkeyval and kvoptions both provide suitable macro definitions.

5.1 Using kvoptions

When using kvoptions, option processing takes place using the `\ProcessKeyvalOptions` macro. This has to be supplied with the family of keys to be processed:

```
\ProcessKeyvalOptions{fam}
```

To make handling certain styles of option easier, kvoptions provides two key-defining macros which are very focussed on package options. Options acting in the normal LaTeX manner are created by the `\DeclareVoidOption` macro. The key is to be used alone, but if a value is given it is ignored with a warning. As this is essentially a standard LaTeX option, the normal need to provide an action exists:

```
\DeclareVoidOption{old}{%
  \PackageInfo{fam}{You gave the 'old' option}%
}
```

`\DeclareDefaultOption` is used to process unknown options, in the manner of the LaTeX kernel `\DeclareOption*` macro. The result is that `\CurrentOptionKey` stores the current key name, with `\CurrentOptionValue` holding any value which was given, or `\relax` if there is no value.

```
\DeclareDefaultOption{%
  \PackageInfo{fam}{%
    You gave the '\CurrentOptionKey' option,
    with value '\CurrentOptionValue'
  }%
}
```

5.2 Using xkeyval

The `\ProcessOptionsX` macro is used to process xkeyval options. As might be expected, this takes an optional prefix and mandatory family argument. The family has to be given in angle brackets, for example

```
\ProcessOptionsX<fam>
```

Loading xkeyval provides `\DeclareOptionX` for handling package options which may have no value. Values *can* be accepted, and are available as `#1`. This macro does not require a key family, although one can be given as an optional argument, again in angle brackets.

```
\DeclareOptionX<fam>{letter}{%
  \PassOptionsToPackage{geometry}
    {letter}%
}
\DeclareOptionX<fam>{date}
  {\renewcommand*{\date}{#1}}
```

The `\DeclareOptionX*` macro works like the kernel's `\DeclareOption*` macro, but no error is raised if the option is in ⟨*key*⟩=⟨*value*⟩ format. In contrast to kvoptions, the entire unknown input (key, plus potentially an equals sign and a value) is stored as `\CurrentOption`.

```
\DeclareOptionX*{%
  \PackageWarning{fam}
    {'\CurrentOption' invalid}}
```

6 Additional considerations

6.1 Redefining and disabling keys

Keys can be (re)defined at any point using any of the key-defining macros discussed here. Thus keys can be defined to only give a warning, then redefined later to carry out a function. This is particularly useful for LaTeX package options, where the key may not be appropriate at load time but may be later.

Conversely, some keys are appropriate only before some action (such as loading a file) takes place. Disabling a key simply requires that the key is defined to do nothing:

```
\define@key{fam}{key}{\wlog{Key 'key' ignored}}
```

If a key (re)definition occurs inside a group (such as `\begingroup` ... `\endgroup` or `{`...`}`), the definition applies only inside that group. There is no `\global` prefix to `\define@key`, and so to ensure that a key is globally disabled, the low-level TeX `\gdef` must be used:

```
\gdef\KV@fam@key#1{\wlog{Key 'key' ignored}}
```

Both kvoptions and xkeyval provide high level methods for disabling keys. kvoptions defines the `\DisableKeyvalOption` macro, which requires only the family and key name:

```
\DisableKeyvalOption{fam}{key}
```

This macro takes an optional argument which can be used to control the result of attempting to use a disabled key (warning, error, ignore, *etc.*). The use of the optional argument is illustrated in Section 7. xkeyval provides the similar `\disable@keys`:

```
\disable@keys{fam}{key}
```

In this case, the macro can accept the usual xkeyval optional argument for the key prefix.

6.2 Setting one key from another

There are occasions when the setting of one key affects another. Usually, this can be accommodated using `\setkeys` within `\define@key` (or a derivative, if using xkeyval):

```
\define@key{fam}{key}{#1}
\define@key{fam}{key two}{%
  You said: \setkeys{fam}{key=#1}%
}
```

If two keys should function in an identical manner, it is sometimes easier to `\let` one to the definition of the other. Be careful about default values: only the key defined using `\define@key` will have one using this method! This issue can be avoided by first declaring the keys as normal, then carrying out the `\let`.

```
\define@key{fam}{key}[default]{#1}
\define@key{fam}{key two}[default]{}
\expandafter\let\csname
  KV@fam@key two\endcsname\KV@fam@key
```

gives two identical keys, `key` and `key two`, with the same default.

The use of these methods to allow alternative spellings for setting a key, to set a storage macro and a TEX `\if...`, are illustrated in Section 7.

6.3 Interaction between the different key–value packages

The xkeyval, kvoptions and kvsetkeys packages all use unique macro names (both user and internal). All three can therefore be loaded without issue. Provided the standard key prefix `KV` is used, the keys generated are also cross-compatible.

Neither kvoptions nor kvsetkeys define any of the macros from the keyval package itself. This means that they require keyval, and that they do not affect its functions. xkeyval works differently, using its own definition of the core keyval macros, and under LATEX prevents subsequent loading of the keyval package. xkeyval aims to make these changes backward-compatible; however, under certain circumstances some macros may behave differently. The latest version of xkeyval fixes a number of differences in behaviour between keyval and xkeyval.

The following short LATEX document can be used as a test to show the differences in behaviour between older versions of xkeyval and the keyval package. With keyval or the latest version of xkeyval this document compiles correctly. However, older versions of xkeyval give errors.

```
\documentclass{article}
\usepackage{keyval}
%\usepackage{xkeyval}
\makeatletter
\define@key{w}{cmd}
  {\def\test##1{#1}}
\makeatother
\setkeys{w}{cmd={--#1--}}
\begin{document}
[\test{ee}]
\end{document}
```

It is therefore strongly recommended that any package using key–value should be tested with xkeyval loaded, even if it is not being used. In this way, if other packages load xkeyval problems should be avoided.

6.4 Using key–value with plain TEX

All of the key–value packages are compatible to some extent with plain TEX. Both kvoptions and kvsetkeys are designed to auto-detect whether TEX or LATEX is in use. A minimal set of LATEX macros are defined only if they are not otherwise available. Thus both can be used directly in plain TEX.

```
\input kvoptions.sty
\input kvsetkeys.sty
```

The xkeyval bundle is designed in a modular fashion. The file `xkeyval.sty` contains the LATEX code (including processing code for package options), whereas the code for defining and setting keys is contained in `xkeyval.tex`. As plain TEX users need only the latter, using xkeyval is simply:

```
\input xkeyval
```

The keyval package itself is not designed for use with plain TEX. It therefore requires a small but non-zero number of LATEX macros. These are conveniently provided by miniltx.

```
\input miniltx
\input keyval.sty
```

The file `keyval.sty` is also loaded by kvoptions, which ensures that the necessary macros are defined.

7 Putting it all together: a short example

The various methods outlined above will be sufficient for many people implementing a key–value interface. However, putting everything together can still be challenging. A short, and not entirely trivial, example will illustrate the steps needed.

Consider the following situation. You have been asked by an inexperienced LATEX user to produce a small package providing one user macro, `\xmph`, which will act as an enhanced version of `\emph`. As well as italic, it should be able to make its argument bold, coloured or a combination of all of these. This should be controllable on loading the package, or during the document. Finally, a de-activation setting is requested, so that `\xmph` acts exactly like

\emph. This latter setting should be available only in the preamble, so that it will apply to the entire document body.

Looking at the problem, you first decide to call the package xmph, and to use the xmph@ prefix for internal macros. The settings requested all look relatively easy to handle using the kvoptions package, so you choose that for key–value support. You decide on the following options/settings:

- inactive, a key with no value, which can be given only in the preamble;
- useitalic, a Boolean option for making the text italic;
- usebold and usecolour, two more Boolean options with obvious meanings
- colour, a string option to set the colour to use when the usecolour option is true.

You also anticipate that US users would prefer the option names usecolor and color, and so you decide to implement them as well.

As well as the \xmph macro, you decide to create a document body setup macro \xmphsetup. Both \xmph and \xmphsetup will take a single, mandatory argument. This keeps everything easy to explain, and means there is not too much work to do with arguments and so on.

With the design decisions made, you can write the package. The options and so on come first. Most of the keys are defined using high-level kvoptions macros, although two low-level methods are used. Initial settings for the package are set up by a \setkeys instruction *before* processing any package options.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{xmph}
  [2008/03/17 v1.0 Extended emph]
\RequirePackage{color,kvoptions}
\SetupKeyvalOptions{
  family=xmph,
  prefix=xmph@}
\DeclareBoolOption{useitalic}
\DeclareBoolOption{usebold}
\DeclareBoolOption{usecolour}
\DeclareBoolOption{usecolor}
\let\KV@xmph@usecolor
  \KV@xmph@usecolour
\DeclareStringOption{colour}
\define@key{xmph}{color}
  {\setkeys{xmph}{colour=#1}}
\DeclareVoidOption{inactive}{%
  \PackageInfo{xmph}
    {Package inactive}%
  \AtEndOfPackage{\let\xmph\emph}%
}
\setkeys{xmph}{useitalic,colour=red}
\ProcessKeyvalOptions{xmph}
\define@key{xmph}{inactive}
  {\PackageInfo{xmph}
     {Package inactive}
  \let\xmph\emph
}
\AtBeginDocument{
  \DisableKeyvalOption[
    action=warning,
    package=xmph]
    {xmph}{inactive}
}
\newcommand*{\xmphsetup}
  {\setkeys{xmph}%
}
```

The user macros are then defined; by keeping the two parts separate, it will be easier to alter the method for managing the keys, if needed. Later, we will see how this enables switching from keyval-based keys to pgfkeys without altering the core of the package at all.

```
\newcommand*{\xmph}[1]{%
  \xmph@emph{\xmph@bold{%
    {\xmph@colourtext{#1}}}}%
}
\newcommand*{\xmph@emph}{%
  \ifxmph@useitalic \expandafter\emph
  \else \expandafter\@firstofone
  \fi}
\newcommand*{\xmph@bold}{%
  \ifxmph@usebold \expandafter\textbf
  \else \expandafter\@firstofone
  \fi}
\newcommand*{\xmph@colourtext}{%
  \ifxmph@usecolour \expandafter\textcolor
  \else \expandafter\@secondoftwo
  \fi
  {\xmph@colour}}
```

The actions of the new package are shown by the following short example LaTeX file. The use of the disabled key inactive will result in a warning entry in the log.

```
\documentclass{article}
\usepackage[
  usecolour,
  usebold]{xmph}
\begin{document}
  Some text \xmph{text}
  \xmphsetup{
    usecolor=false,
    usebold=false,
```

```
    useitalic=false}%
  \xmph{more text}
  \xmphsetup{inactive}
\end{document}
```

8 A different approach: pgfkeys

All of the packages discussed so far are built on the keyval approach. Keys are part of families, and further subdivision (at least beyond altering the key prefix) is not readily achieved. An alternative approach is taken by the pgfkeys package (Tantau, 2008). This package uses the ⟨*key*⟩=⟨*value*⟩ input format, but the underlying implementation is not derived from keyval; the pgfkeys package therefore uses a unique key management model. Thus, while for the user pgfkeys and keyval are very similar, for the developer they require different approaches. However, many of the ideas of keys with differing behaviours carry through from the earlier discussion.

8.1 How key–value works with pgfkeys

In principle, pgfkeys works in the same ways as described in Section 2: there are two parts of the key–value system, defining keys and assigning values to keys. However, pgfkeys requires just one command for both parts: the `\pgfkeys` macro.

The definition requires the use of special suffixes, the so-called key handlers. Here, the term *handler* is used slightly differently than in the other packages. For example, the statement

```
\pgfkeys{/path/key/.code={#1}}
```

defines a key named `/path/key`. The `.code` statement defines a macro which expands to the TEX code in the arguments (in our case, the TEX code is simply the argument itself, "`#1`"). Hence, using the key will just print its value:

```
\pgfkeys{/path/key=value}
```

yields "`value`". The `/path` plays a similar role to ⟨*prefix*⟩ and ⟨*family*⟩ for keyval and friends: it associates `key` with a sub-tree.

As with the key–value syntax in Section 2, spaces in key and path names are allowed, and spaces between keys and their values and different keys are ignored. Also, literal "`,`" and "`=`" characters need to be protected by braces:

```
\pgfkeys{
  /path/key three={value1,value2},
  /path/keyfour={some=stuff}
}
```

In contrast to keyval and friends, pgfkeys uses a different concept to manage key prefixes and key suffixes: the key *tree*.

8.2 The key tree

In the pgfkeys model, keys are organised hierarchically, similar to the Unix file system; subdivisions are generated using slashes. For example, `/path/sub/key` is a key named `key`, which belongs to the subtree `/path/sub` which is in turn located inside `/path`. The slash "`/`" defines the tree's root. A statement like

```
\pgfkeys{
  /path/sub/key = value,
  /path/key two = value2
}
```

sets both of these keys, showing that keys belonging to different subtrees can be set in one statement.

It is not necessary to fully qualify keys: a default path is considered for every key without a full path. For example,

```
\pgfkeys{
  key = value of key,
  key two = value of key two,
  sub/key three = value3
}
```

will search for `key`, `key two` and `sub/key three` in the current default path. Default paths can be set using a *change directory* command, using the `.cd` handler which will be discussed below. The initial setting is "`/`", which means any unqualified key name like `key` will be changed to `/key` implicitly.

8.3 Using pgfkeys

In contrast to the keyval approach, pgfkeys uses a single macro to define and set keys, namely `\pgfkeys`. At its heart, pgfkeys works with three different types of keys: keys which store their values directly, command keys and keys which are handled. Key definitions, assignments and other key types are composed of these three building blocks.

Key type 1: direct keys

Direct keys simply store their values as character sequences. A pgfkeys direct key is thus similar to a xkeyval command key (one defined using `\define@cmdkey`). For example,

```
\pgfkeys{/path/key/.initial = value}
```

defines the key `/path/key` and assigns `value`. After this, the value can be changed with assignments:

```
\pgfkeys{/path/key = new value}
```

Direct keys are stored in a way which is not directly accessible to end users. Instead, the command `\pgfkeysgetvalue` is used to get a direct key's current value into a (temporary) macro. For example, the statement

```
\pgfkeysgetvalue{/path/key}{\macro}
```

will get the current value of /path/key and copy it into \macro. The macro will be (re-)defined if necessary without affecting the stored key's value.

Putting these things together, direct keys can be used as in the following example. The code

```
\pgfkeys{/path/key/.initial = value}
\pgfkeysgetvalue{/path/key}{\macro}
After definition: ``\macro''.

\pgfkeys{/path/key = new value}
\pgfkeysgetvalue{/path/key}{\macro}
After setting: ``\macro''
```

will define /path/key with an initial value, copy the value to \macro and typeset the result. Afterwards, it changes the current value, copies the new value to \macro and typesets it again. Here's the output:

```
After definition:  ``value''.
After setting:  ``new value''.
```

Key type 2: command keys

The second type of pgfkeys-keys are command keys. Here, pgfkeys uses a slightly different terminology than keyval. Command keys with pgfkeys are very similar to the keys defined by \define@key: they are TEX commands with (usually) one argument replacing "#1" with the assigned value. So, what pgfkeys calls a "command key" is a "key handler" in the terminology of keyval and friends.

The usual way to define command keys is to append /.code={⟨*TEX code*⟩} to the key's name. Thus,

```
\pgfkeys{/path/cmd key/.code = {(value=#1)}}
```

defines a command key /path/cmd key which typesets "(value={⟨*its value*⟩})" whenever it is assigned. For example, the listing

```
\pgfkeys{/path/cmd key/.code = {(value=#1)}}
\pgfkeys{/path/cmd key=cmd value}
```

yields "(value=cmd value)".

As with direct keys, command keys are stored in a manner which is not directly accessible by end users. In fact, pgfkeys creates a temporary macro with \def and stores this macro into a direct key /path/cmd key/.@cmd whenever it creates a new command key.

So, command keys are TEX macros which operate on some input argument (the value) using "#1". Useful examples of command keys are

```
\pgfkeys{/path/store key/.code =
    {\def\myPkgOption{#1}
}
```

to store the input into a macro \myPkgOption or

```
\pgfkeys{/path/call key/.code = {\call{#1}}}
```

to invoke another macro \call{#1} with the value. These keys can be used with

```
\pgfkeys{
  /path/store key = value,
  /path/call key = value2
}
```

Since some processing methods are generally useful, pgfkeys provides easier ways to assign them. For example, our example of a command key which simply stores its value into a macro can equivalently be defined using

```
\pgfkeys{
  /path/store key/.store in=
    \myPkgOption
}
```

The suffix .store in, and also the suffix .code, are *key handlers*, the third type of pgfkeys options.

Key type 3: handled keys

The third type of pgfkeys-keys are handled keys.[1] If \pgfkeys encounters a key which is neither a direct option nor a command key, it splits the key into key path (everything up to the last "/") and key name (everything after the last "/"). Then, pgfkeys looks in the special /handlers/ subtree for a key called key name. This is then passed both the current path and the value given. For example,

```
\pgfkeys{/path/cmd key/.code = {(value=#1)}}
```

is a handled key with key name .code and key path /path/cmd key because

1. there is no direct key /path/cmd key/.code;
2. there is no command option by this name;
3. there *is* a command key /handlers/.code.

The predefined handler .code creates a new command key named according to the current key's path (in our case, /path/cmd key).

So, key handlers take a key path and a value as input and perform some kind of action with it. They can define new key types (for example storage keys, Boolean keys or choice keys as we will see in the next section), they can check whether a key is defined, they can change default paths and more. Much of the strength of the pgfkeys package comes from its key handlers.

8.4 Predefined key handlers

pgfkeys provides many predefined key handlers, most of which are used to define more or less special command keys. Here are some common key handlers:

[1] Again, pgfkeys uses a slightly different terminology. Its handled keys are not to be mistaken with the "handlers" defined by \define@key; those are called "command keys" in pgfkeys.

`.cd` A "change directory" command:

```
\pgfkeys{/path/.cd,A=a,B=b}
```

sets the default path to `/path` and will thus set `/path/A=a` and `/path/B=b`. We will later see that the command `\pgfqkeys` also changes the default path, thus

```
\pgfqkeys{/path}{A=a,B=b}
```

will also set `/path/A=a` and `/path/B=b`.

`.default={`⟨*value*⟩`}` Determines a value to be used if no "=" sign is given:

```
\pgfkeys{/path/A/.default=true}
\pgfkeys{/path/A}
```

is the same as if we had written

```
\pgfkeys{/path/A=true}
```

`.code={`⟨*code*⟩`}` Defines a new command key which expands to the value of `.code`. The resulting command key takes one argument.

`.is if={`⟨*TEX-Boolean*⟩`}` Creates a new Boolean key which sets a TEX Boolean to either true or false:

```
\newif\ifcoloured
\pgfkeys{
  /path/coloured/.is if = coloured
}
% set \colouredtrue:
\pgfkeys{/path/coloured=true}
% set \colouredfalse:
\pgfkeys{/path/coloured=false}
```

An error message is raised if the supplied value is neither `true` nor `false`. pgfkeys does not call `\newif` automatically, and the leading "`if`" must not be included in the argument of `.is if`, *i.e.* `coloured/.is if=ifcoloured` would be wrong.

`.is choice` Creates a new choice key, with the available choices given as subkeys of the current one:

```
\pgfkeys{
  /path/op/.is choice,
  /path/op/plus/.code={+},
  /path/op/minus/.code={-},
  /path/op/nop/.code={nothing}
}
% invokes /path/op/plus
\pgfkeys{/path/op=plus}
```

An error results if the user gives an unknown choice.

`.store in={`⟨*\macro*⟩`}` Defines a command key that simply stores its value into a macro:

```
\pgfkeys{/path/key/.store in=
  \keyvalue}
\pgfkeys{/path/key=my value}
Result is `\keyvalue'
```

Expands to "`Result is 'my value'`". Such a key is very similar to a *direct key*, see above.

`.style` Creates a new *style* key, which contains a list of other options. Whenever a style key is set, it sets all of its options:

```
\pgfkeys{
  /text/readable/.style=
    {font=large,color=pink},
  /text/unreadable/.style=
    {font=small,color=black}
}
\pgfkeys{/text/readable}
```

will set the additional options `/text/font=large` and `/text/color=pink` (using the default path since they have no full path).

`.append style` Appends more options to an already existing style key. Given the example above,

```
\pgfkeys{
  /text/readable/.append style=
    {underlined=true}}
```

has the same effect as writing

```
\pgfkeys{/text/readable/.style=
  {font=large,color=pink,
    underlined=true}}
```

Since style keys can be defined and changed easily, they provide much flexibility for package users.

8.5 pgfkeys in action — an example

We will now realise our example LATEX package of Section 7 with pgfkeys. We use the same option names and the same user interface, with one exception: pgfkeys does not support LATEX package options (although see Section 8.6). Any configuration has to be done with `\xmphsetup`.

We do not need to change our implementation for `\xmph` and we can keep its helper macros `\xmph@bold`, `\xmph@emph` and `\xmph@colourtext` as well. We only need to change the option declaration, which is shown in the following listing.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{xmph}
  [2009/03/17 v1.0 Extended emph]
\RequirePackage{color,pgfkeys}
\newif\ifxmph@useitalic
\newif\ifxmph@usebold
\newif\ifxmph@usecolour
\pgfkeys{
  /xmph/.cd,
  useitalic/.is if = xmph@useitalic,
  usebold/.is if = xmph@usebold,
  usecolour/.is if = xmph@usecolour,
  usecolor/.is if = xmph@usecolour,
  useitalic/.default = true,
```

```
  usebold/.default = true,
  usecolour/.default = true,
  usecolor/.style = {usecolour=#1},
  colour/.store in = \xmph@colour,
  color/.style = {colour=#1},
  inactive/.code = {%
    \let\xmph\emph
    \PackageInfo{xmph}
      {Package inactive}%
    }
}
\pgfkeys{
  /xmph/.cd,
  useitalic,
  colour = red
}
\newcommand*{\xmphsetup}{%
  \pgfqkeys{/xmph}%
}
\AtBeginDocument{
  \pgfkeys{
    /xmph/inactive/.code = {%
      \PackageInfo{xmph}{%
        Option 'inactive' only
        available in preamble
      }%
    }
  }
}
```

The command `\pgfqkeys` occurring in the last listing is a variant of `\pgfkeys` which sets the default path directly, without a `.cd` statement. The command

```
\pgfqkeys{/xmph}{
  colored = false,
  bold    = true
}
```

thus uses `/xmph` as its default path.

8.6 pgfkeys for LaTeX package options

The pgfkeys package does not include any native functionality for processing LaTeX package and class options. However, the pgfopts package (Wright, 2008) adds this ability, using a modified copy of the functionality in kvoptions.

The pgfopts package provides only a single user macro, `\ProcessPgfOptions`. Keys are created using the pgfkeys interface discussed above, and can then be used as package (or class) options using the `\ProcessPgfOptions` macro. The requirement to have *no* spaces in the key names for this to work remains exactly the same as for xkeyval or kvoptions processing of options.

9 Conclusions

There are a number of methods for the author wanting to make a start using key–value input. The pgfkeys package has much to recommend it. The interface has been well designed, and it is very strong in handling a wide range of situations (well illustrated in the user documentation). For large-scale projects in particular, the tree concept makes option management much easier. By loading pgfopts, LaTeX option processing is also possible with pgfkeys.

For users who wish to handle LaTeX package options using key–value input, most authors will want to load either kvoptions or xkeyval, rather than coding the option handler directly. Both handle the core issue of providing key–value package options well. Each packages has some advantages, depending on the job at hand.

xkeyval provides a rich set of macros for defining almost every possible type of key. The additional graduation of keys made available by the variable prefix is welcome. The package has a very large number of features which have not been discussed here. However, the package has been criticised for modifying keyval internals. More importantly for many, it suffers from the very problem of complex optional arguments that the key–value method is supposed to avoid.

On the other hand, kvoptions provides a smaller, but more focussed, set of additional key types. The input syntax is much less complex than that of xkeyval, and the provision of `\SetupKeyvalOptions` is particularly welcome. Using the kvoptions method does make it more likely that ambitious package authors will have to become familiar creating customised functions with `\define@key`. However, the clearer syntax make kvoptions a better choice for rapidly making progress with using key–value input.

10 Acknowledgments

Thanks to Didier Verna and Morten Høgholm for helpful suggestions when drafting this manuscript, and Will Robertson for the example of the keyval *versus* xkeyval problem.

References

Adriaens, Hendri. "The xkeyval package". Available from CTAN, `macros/latex/contrib/xkeyval`, 2008.

Adriaens, Hendri, and U. Kern. "xkeyval — new developments and mechanisms in key processing". *TUGboat* **25**(2), 194–199, 2005.

Carlisle, David. "The keyval package". Part of the graphics bundle, available from CTAN, macros/latex/required/tools, 1999.

Oberdiek, Heiko. "The kvoptions package". Part of the oberdiek bundle, available from CTAN, macros/latex/contrib/oberdiek, 2009a.

Oberdiek, Heiko. "The kvsetkeys package". Part of the oberdiek bundle, available from CTAN, macros/latex/contrib/oberdiek, 2009b.

Tantau, Till. "pgfkeys". Part of the Ti*k*Z and PGF bundle, available from CTAN, graphics/pgf, 2008.

Wright, Joseph. "pgfopts — LaTeX package options with pgfkeys". Available from CTAN, macros/latex/contrib/pgfopts, 2008.

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2 dot co dot uk

◇ Christian Feuersänger
Institute for Numerical Simulation
Wegelerstraße 6
53115 Bonn
Germany
ludewich (at) users dot
sourceforge dot net

Current typesetting position in pdfTEX

Vít Zýka

Abstract

Among the relatively little-known pdfTEX extensions is a possibility of obtaining the current typesetting position. It can be used later for placing objects. This article gives a description of related primitives and shows an example of usage.

1 Motivation

TEX proceeds sequentially when building a page. It places an object (box, char, rule, space) next to another one and it shifts the current typesetting point. If we need to stack up several objects (e.g. color background, stamps) we have to proceed in four steps: 1. remember a current position, 2. move to the required position, 3. place the object here, and 4. return back to the starting position. The current typesetting point must not be influenced when processing these four steps. This does not limit us in most cases but it is a bit unwieldy.

We face a worse case when placing/scaling of the object depends on two or even more points. Imagine a color background or a frame surrounding several paragraphs with nonzero vertical stretchability between, or drawing a diagram with an arrow between cells, or connecting two words inside a paragraph by line. For all these applications we need to remember the typesetting points for later drawing of the dependent object. Classical TEX has no instrument for these tasks. The only way used to be a cooperation with a specific driver (e.g. via PostScript operators), which makes the document driver-dependent. PdfTEX has a more straightforward and portable solution — it provides new primitives for obtaining a current typesetting position.

2 New primitives

There are three new primitives for working with the current position. The first is `\pdfsavepos`, which saves a mark in the main vertical list. After the page formatting, during `\shipout` operation, the mark is processed and the absolute typesetting position is saved in relation to the left bottom page corner. Afterwards, this (x, y) position can be read by primitives `\pdflastxpos` and `\pdflastypos`. Each of them returns an integer value representing the distance in scaled points (`sp`).

Since `\pdfsavepos` is processed at `\shipout` time, when typesetting is done, we need to write the position values to a file, then read and use them in the next TEX run.

This usage is not simple and therefore we will show it in an example. Our goal will be to connect two words inside a paragraph by a line.

3 Example: Drawing a line inside a paragraph

Let us solve this task: *draw a line from place A to place B*. The places are to be marked by writing `\posMark{place_label}` anywhere on a single page, even inside a paragraph. The two marks could be used to draw a line between them. The example illustrating the idea is shown inside this paragraph. It was typeset by the following code:

```
Let us solve this task:
{\it draw a line\posMark{A} ...
... to draw a line\posMark{B} between ...
```

How does it work? Let us start our description with helping macros.[1] First we need to draw a line. Let's make that line be gray and 2 bp wide. Low level PDF or PostScript operators solve this simple drawing, so we can avoid loading large vector drawing packages such as Ti*k*Z or PSTricks. The main output drivers pdfTEX and Dvips are distinguished by the `\ifpdf` macro:

```
\ifpdf
  % 1,2=start x y; 3,4=stop x y <bp unit>
  \def\Line#1,#2--#3,#4{%
    \pdfliteral page
      {0.7 G 2 w #1 #2 m #3 #4 l S }}
\else
  \def\Line#1,#2--#3,#4{%
    \special{" 0.7 setgray 2 setlinewidth
      #1 #2 moveto #3 #4 lineto stroke }}
\fi
```

The next macro cuts off the unit *pt* from a dimension:

```
{\catcode`\p=12 \catcode`\t=12
 \gdef\removePT#1pt{#1}}
```

Our last helping action is conversion from *sp* units to *bp*, which is a base unit in PDF or PostScript:

```
% 1=identificator 2=number <sp>
\def\defBPfromSP#1#2{%
  \bgroup
  \dimen0=#2sp
  \dimen0=.013837\dimen0
  \dimen0=72\dimen0
  \expandafter\xdef\csname#1\endcsname{%
```

First published in *Zpravodaj* 17:2 (2007), pp. 67–72, as "Používáme pdfTEX V: aktuální pozice sazby"; translation by the author. Reprinted with permission.

[1] We use LATEX packages or macros for commands not closely related to our topic: `eso-pic`, `ifpdf`, `afterpage` and `\InputIfFileExists`.

```
    \expandafter\removePT\the\dimen0 }%
  \egroup}
```

Now we are prepared to proceed to our topic — the current typesetting position. As we mentioned before, the position is not known until `\shipout` and thus it has to be saved to an auxiliary file. We name this file with the main file name and a `.pos` extension. The following macros open this file for writing at the document beginning and close it at the document end:

```
\newwrite\posHandle
\def\posFile{\jobname.pos }

\def\posOpen{\openout\posHandle=\posFile}
\def\posClose{\closeout\posHandle}

\AtBeginDocument{\posLoad\posOpen}
\AtEndDocument{\posClose}
```

The user macro `\posMark` writes the position to the file. It uses all three new pdfTEX primitives:

```
\def\posMark#1{% 1=place_label
  \pdfsavepos
  \write\posHandle{%
    \string\posDef\string{#1\string}%
    \string{\the\pdflastxpos\string}%
    \string{\the\pdflastypos\string}}}
```

After the first LATEX run the following file is created:

```
\posDef{A}{10597449}{27447688}
\posDef{B}{24506216}{25133596}
```

This file is loaded by the `\posLoad` call at line 31:

```
\def\posLoad{\InputIfFileExists{\posFile}{}{}}
```

Loading the `.pos` file only if it exists avoids an error in the first LATEX pass.

The task of the internal `\posDef` macro, which is passed the label name and the (x, y) position, is to create two macros `\pos-x-sp-place_label` and `\pos-y-sp-place_label` with the values in *sp* and corresponding macros `\pos-x-bp-place_label` and `\pos-y-bp-place_label` in *bp*:

```
% 1=place_label 2=x-pos 3=y-pos
\def\posDef#1#2#3{%
  \expandafter
    \def\csname pos-x-sp-#1\endcsname{#2}%
  \posDefXbp{#1}%
  \expandafter
    \def\csname pos-y-sp-#1\endcsname{#3}%
  \posDefYbp{#1}}
```

Unit conversion is done by:

```
\def\posDefXbp#1% 1=place_label
  {\defBPfromSP{pos-x-bp-#1}{\posGetX{#1}}}
\def\posDefYbp#1% 1=place_label
  {\defBPfromSP{pos-y-bp-#1}{\posGetY{#1}}}
```

Macro calling is simplified by these definitions:

```
\def\posGetXY#1{\expandafter% 1=full_label
  \ifx\csname #1\endcsname\relax0
  \else\csname #1\endcsname\fi}
\def\posGetX#1{\posGetXY{pos-x-sp-#1}}
\def\posGetY#1{\posGetXY{pos-y-sp-#1}}
\def\posGetXbp#1{\posGetXY{pos-x-bp-#1}}
\def\posGetYbp#1{\posGetXY{pos-y-bp-#1}}
```

These previously defined absolute coordinates enter the macro for the line drawing. Its suitable placement is inside `\shipout`, when the base coordinate system is on. This is simplified by the package `eso-pic`:

```
% 1,2=start x y; 3,4=stop x y <bp unit>
\def\AbsLine#1,#2--#3,#4{%
  \AddToShipoutPicture{%
    \AtPageLowerLeft{\Line#1,#2--#3,#4}}}
```

And finally, we have the top-level drawing macro `\AbsLineFromTwoMarks` with the place labels as arguments:

```
% 1=place_label_A 2=place_label_B
\def\AbsLineFromTwoMarks#1#2{%
  \AbsLine(\posGetXbp{#1},\posGetYbp{#1}--%
           \posGetXbp{#2},\posGetYbp{#2})}
```

Now we can see that adding the next two lines after our illustrative paragraph will draw the line:

```
\AbsLineFromTwoMarks{A}{B}
\afterpage{\ClearShipoutPicture}
```

The last line avoids repeating the drawing on every subsequent page.

4 Conclusion

The current typesetting position is a useful pdfTEX extension. It works in both PDF output and DVI output from pdfTEX. Many graphical tricks such as framing word(s)/sentence(s)/paragraph(s) or surrounding them by backgrounds, emphasizing page parts by a vertical line in the margin, visualization of page elements relationship, and tabular cell placement can benefit from it. Here is a list of some LATEX packages that utilize this feature: `changebar`, `marginnote`, `t-angles`, `pdfsync`, `tabularht`. ConTEXt employs it throughout.

[Editor's note: This is one of a series of articles by Dr. Zyka on pdfTEX primitives. We hope to reprint other installments in future issues.]

◇ Vít Zýka
TYPOkvítek
Prague, Czech Republic
vit dot zyka (at) seznam dot cz

Letters

In response to "mathematical formulæ"

Kaihsu Tai

I welcome Massimo Guiggiani and Lapo Mori's helpful style guide "Suggestions on how *not* to mishandle mathematical formulæ" [1]. However, there are a few points which the authors might have got wrong.

At §5.2, the authors said "walk *at most* 2 *km north*" is the "correct form". But in fact the correct form, as specified by §6.1.1 of the excellent NIST advice [4], is that "Unit symbols are printed in roman (upright) type regardless of the type used in the surrounding text", giving "walk *at most* 2 km *north*". This can perhaps be achieved by

```
walk \emph{at most $2~\mathrm{km}$ north}.
```

The NIST guide also advised (§10.5.3) "digits should be separated into groups of three, counting from the decimal marker towards the left and right, by the use of a thin, fixed space." Example: "43 279.168 29". This should apply even in the English language.

At §5.4, the authors said that "round brackets can be used in tables and graphs when units appear next to a symbol of the corresponding physical quantity instead of the numeric value to which they refer". However, this is inferior to the NIST suggestion (§7.1): "to eliminate the possibility of misunderstanding, an axis of a graph or the heading of a column of a table can be labeled 't/°C' instead of 't (°C)' or 'Temperature (°C).' Similarly, an axis or column heading can be labeled 'E/(V/m)' instead of 'E (V/m)' or 'Electric field strength (V/m).'" There is a mnemonic rationale to this: Let's say we see a number "36.8" under the heading "τ/s". This stands for the (incorrect) formula "$\tau/\mathrm{s} = 36.8$", which can be converted into the correct expression $\tau = 36.8$ s.

While I still have the gentle readers' attention, may I mention a few more items. First, the international standard IEC 80000-13 [2] introduces binary prefixes. So now we should speak of "two mebioctets" (2 Mio) rather than "two megabytes". ("Mebi-" is exactly 2^{20}, not 10^6 "mega-"; the byte has not always been defined as 8 bits.)

Second, we should use the correct SI unit "gigagram" (1 Gg) rather than the "megaton" ("1 Mt") when measuring things like greenhouse gas emission (the horror of "1 $\mathrm{MtCO_2}$"!) or explosive energy in TNT equivalents.

Third, I would like to start a trend of using ISO 4217 [3] currency codes with SI prefixes; for example, "38 kEUR". This is convenient and avoids creating a new currency symbol (and a new typographical problem) whenever a new currency is introduced (a recent example being the euro).

References

[1] Massimo Guiggiani and Lapo Mori. Suggestions on how *not* to mishandle mathematical formulæ. *TUGboat*, 29(2):255–263, 2008.

[2] International Electrotechnical Commission. *IEC 80000-13:2008 Quantities and units — Part 13: Information science and technology*. Genève, Switzerland, 2008.

[3] International Organization for Standardization. *ISO 4217:2008 Codes for the representation of currencies and funds*. Genève, Switzerland, 2008.

[4] Ambler Thompson and Barry N. Taylor. *NIST Special Publication 811: Guide for the Use of the International System of Units (SI)*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 2008.

◇ Kaihsu Tai
Department of Biochemistry
University of Oxford
South Parks Road
Oxford OX1 3QU
Great Britain
k (at) kauha dot eu
http://kauha.eu/

In response to Kaihsu Tai

Massimo Guiggiani and Lapo F. Mori

The authors would like to thank Kaihsu Tai for his comments (Tai, 2009) on our paper about mathematical formulæ (Guiggiani and Mori, 2008). We also would like to thank Claudio Beccari for his comments and suggestions (Beccari, 2009).

In particular we agree that the correct form for writing unit symbols is always in upright roman. The example given in §5.2 should however be achieved with

```
walk \emph{at most $2~\mathrm{km}$} north
```

since the word "north" was, and should remain, outside the emphasis.

We also agree that in the English language digits should be separated into groups of three by the use of a thin fixed space. This is clearly required by ISO 31-0 (1992) and NIST Special Publication 811 (2008). In our article we were, however, noticing that the babel package separates groups of three digits by a space or a comma according to the current language. In particular when the English language is selected, the variable `\thousandsep` is defined as a comma. This probably comes from the widespread usage in the English language of a comma. This behavior, which should be avoided, is sometimes even required by manuals of style (American Psychological Association, 2001).

We do not agree with the suggestion of dividing physical quantities by their units, although this follows NIST Special Publication 811 (2008). This form is not widely used and is not required or even suggested by the ISO standards. We believe that a reader would find it far easier to understand that v (m/s) indicates a physical quantity named v expressed in meters per second, rather than $v/(\mathrm{m/s})$. In the second case the reader could be confused by the notation and interpret it as a dimensionless physical quantity vs/m.

As noted by Tai, ISO 4217 (2008) requires the use of the SI prefixes with the international three-letter currency codes. We believe that this should be strictly followed by documents focused on currencies but not necessarily by less specialized documents. New currencies are not created that often and, when this happens, their symbol becomes quickly available in most word processing applications. For instance we believe that in non-financial documents it is better to use the Euro sign € defined by the European Commission (1997), instead of the three-letter currency code EUR. The use of SI prefixes with monetary units is still very uncommon.

In the end, we would like to remind that the focus of our paper was on gross mistakes, too often found in scientific writings, which may severely impair readability.

Bibliography

American Psychological Association. *Publication Manual of the American Psychological Association*. 5th edition, 2001.

Beccari, C. "Private communication". 2009.

European Commission. *Communication from the Commission — The use of the Euro symbol*. 1997.

ISO 31-0. *Quantities and units — Part 0: General principles*. International Organization for Standardization, Geneva, 3rd edition, 1992.

ISO 4217. *Codes for the representation of currencies and funds*. International Organization for Standardization, Geneva, 7th edition, 2008.

NIST Special Publication 811. *Guide for the Use of the International System of Units (SI)*. National Institute of Standards and Technology, Gaithersburg, MD, USA, 2008.

Guiggiani, M., and L. F. Mori. "Suggestions on how *not* to mishandle mathematical formulæ". *TUGboat* **29**(2), 255–263, 2008.

Tai, K. "In response to 'mathematical formulæ'". *TUGboat* **30**(1), 2009.

◇ Massimo Guiggiani
Dipartimento di
Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
Pisa, Italy
guiggiani (at) ing dot unipi dot it

◇ Lapo F. Mori
Dipartimento di
Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
Pisa, Italy
lapo dot mori (at) ing dot unipi dot it

The Treasure Chest

This is a list of selected new packages posted to CTAN (`http://www.ctan.org`) from June 2008–June 2009, with descriptions based on the announcements and edited for brevity.

Entries are listed alphabetically within CTAN directories. A few entries which the editors subjectively believed to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

We hope this column and its companions will help to make CTAN a more accessible resource to the TEX community. Comments are welcome, as always.

◇ Karl Berry
`http://tug.org/ctan.html`

biblio

`chembst` in `biblio/bibtex/contrib`
: BibTEX style files for chemistry journals.

dviware

`dviasm` in `dviware`
: Script for disassembling and reassembling DVI files, including adding preprint numbers and watermarks.

fonts

`boisik` in `fonts`
: A Metafont font inspired by Baskerville.

`dozenal` in `fonts`
: Fonts and macros for typesetting documents in base 12 (Dozenal).

* `gentium` in `fonts`
: TrueType fonts from SIL, pdfTEX files for many encodings (`agr`, `t2a`, `ec/T1`, `texnansi`, `l7x`, `qx`, `t5`), and support files for ConTEXt.

`inconsolata` in `fonts`
: Original monospace font with LATEX support for several encodings.

`junicode` in `fonts`
: TrueType font for medievalists with many OpenType features.

`libris` in `fonts`
: Sans-serif family similar to the well-known Lydian.

`phaistos` in `fonts/archaic`
: All symbols from the Disc of Phaistos (produced via punches in clay, probably around 1700 BCE).

`pigpen` in `fonts`
: Font and macros for Pigpen (masonic) ciphers.

`shuffle` in `fonts`
: A symbol for the shuffle product.

`tolkienfonts` in `fonts`
: Virtual fonts for writing English, Quenya, and Sindarin with various free Tolkien-world fonts.

graphics

* `asymptote` in `graphics`
: Vector graphics language for technical drawing, inspired by MetaPost but with a C++-like syntax.

`autoarea` in `graphics/pictex/addons`
: Have PiCTEX recognize lines and arcs in determining bounding boxes.

`bclogo` in `graphics/pstricks/contrib`
: PSTricks macros for colorful boxes with a title and logo.

`circuitikz` in `graphics/pgf/contrib`
: Drawing electrical circuits with PGF/Ti*k*Z.

`jpgfdraw` in `graphics`
: Graphics application written in Java providing integration with a variety of TEX packages as well as basic drawing capabilities.

`metago` in `graphics/metapost/contrib/macros`
: MetaPost package for Go game positions.

`psbao` in `graphics/pstricks/contrib`
: PSTricks macros for drawing Bao (game) diagrams.

`pst-bezier` in `graphics/pstricks/contrib`
: PSTricks macros for drawing a Bezier curve with full control.

`pst-bspline` in `graphics/pstricks/contrib`
: Draws uniform Bspline curves and interpolations.

`pst-gantt` in `graphics/pstricks/contrib`
: PSTricks macros for drawing Gantt charts.

`pst-sigsys` in `graphics/pstricks/contrib`
: PSTricks macros for disciplines related to signal processing.

`pst-support` in `graphics/pstricks/contrib`
: Support for PSTricks in Distiller and TEXnicCenter.

`schemabloc` in `graphics/pgf/contrib`
: PGF/Ti*k*Z macros for block diagrams.

`tikz-inet` in `graphics/pgf/contrib`
: Ti*k*Z macros and shapes for interaction nets.

`tikz-timing` in `graphics/pgf/contrib`
: Generating timing diagrams with Ti*k*Z.

`vaucanson-g` in `graphics/pstricks/contrib`
: PSTricks macros for automata.

* `xetex-pstricks` in `graphics`
: Configuration files to use PSTricks with XƎTEX.

info

`amslatex/primer` in `info`
: Getting up and running with AMSLATEX.

`Aro-Bend` in `info/challenges`
Collection of the TEX macro programming challenges "Around the Bend", by Michael Downes.

`Doc-PiCTeX.txt` in `info/pictex`
Concise reference information on PiCTEX.

* `first-latex-doc` in `info`
Guide to trying out LATEX for the first time, using mathematical documents.

`intro-scientific` in `info`
Introduction to typesetting scientific or mathematical documents using LATEX.

`latex-course` in `info`
A LATEX course as a Beamer slide presentation, based on the German `tex-kurs`.

* `latex-doc-ptr` in `info`
Pointers to major LATEX guides in most areas.

* `latexcheat` in `macros/latex/contrib`
Single-sheet LATEX reference.

`lnotes` in `info`
Introduction to (LA)TEX in Chinese.

`lshort-chinese` in `info/lshort`
Chinese translation of the `lshort` document.

`lshort-mongol` in `info/lshort/mongolian`
Mongolian translation of the `lshort` document.

* `Math_into_LaTeX-4` in `info`
George Grätzer's series of video presentations introducing LATEX.

`mpman-ru` in `info/metapost/doc/russian`
Russian translation of the MetaPost user manual.

`pstdoc` in `info`
PSTricks help system using Python.

language

`casyl` in `language`
Cree/Inuktitut in Canadian Aboriginal Syllabics.

`dehyph-exptl` in `language/hyphenation`
Experimental hyphenation patterns for German, covering traditional and reformed orthography.

`hyph-utf8` in `language`
TEX hyphenation patterns converted to UTF-8 while retaining compatibility with previous patterns.

`georgian` in `language`
Georgian language support for (LA)TEX.

`lithuanian` in `language`
Lithuanian language support: hyphenation, `babel`, fonts, and more.

macros/generic

`encxvlna` in `macros/generic`
Insert nonbreakable spaces required by Czech and Slovak typographical rules, based on encTEX.

`tex-ewd` in `macros/generic`
Typeset calculational proofs and programs in Dijkstra's "guarded command language".

macros/latex/contrib

`alterqcm` in `macros/latex/contrib`
Creating multiple choice questionnaires.

`analogclock` in `macros/latex/contrib`
A ticking analog clock for pdfLATEX documents and presentations.

`arsclassica` in `macros/latex/contrib`
Reproduces the look of the Italian guide 'The art of writing with LATEX'.

`asyfig` in `macros/latex/contrib`
Support for standalone `.asy` figure files.

`beamerposter` in `macros/latex/contrib`
Create custom-sized LATEX posters, e.g., double DIN-A0 size in landscape or portrait orientation. Poster fonts can be scaled.

`bezos` in `macros/latex/contrib`
New `subdocs` style sharing `.aux` files between parts.

`blowup` in `macros/latex/contrib`
Scale all pages of a document up or down, similar to plain TEX's `\magnification`.

`calctable` in `macros/latex/contrib`
Typeset accounting and other tables with automatic sum and percentage computations.

`chletter` in `macros/latex/contrib`
Typesetting letters and small documents according to Swiss rules.

`codedoc` in `macros/latex/contrib`
Produce LATEX code and documentation in a single file with ordinary LATEX syntax.

`collref` in `macros/latex/contrib`
Merge blocks of references into one `\bibitem`.

`cpssp` in `macros/latex/contrib`
Draw protein secondary structures.

`csbulletin` in `macros/latex/contrib`
Zpravodaj (CSTUG bulletin) article style.

`diagmac2` in `macros/latex/contrib`
Reynolds diagram macros supporting any line slope.

`drac` in `macros/latex/contrib`
Declare "robust active characters", an active character which can be used in moving arguments.

`ean13isbn` in `macros/latex/contrib`
Generate ISBN barcodes in EAN13 format, as used after January 1, 2007.

`easylist` in `macros/latex/contrib`
Creating customizable lists of numbered items with a single active character.

`elsarticle` in `macros/latex/contrib`
Document class for articles submitted to Elsevier.

`eltex` in `macros/latex/contrib`
Draw electric circuit diagrams according to IEC 617.

`emptypage` in `macros/latex/contrib`
Suppress page numbers and headings on empty pages.

`eukdate` in `macros/latex/contrib`
Change `\today` to the UK format, including weekday, as in 'Friday, 25 June 2008'.

exp-testopt in macros/latex/contrib
Expandable \@testopt.

figbas in macros/latex/contrib
Mini-fonts for figured-bass music notation.

forarray in macros/latex/contrib
Process lists and arrays in LaTeX, including nesting.

gmdoc-enhance in macros/latex/contrib
Enhancements for gmdoc.

gmverse in macros/latex/contrib
Typesetting (short) poems.

getfiledate in macros/latex/contrib
Fetch and format the last modification time of a local file.

greekdates in macros/latex/contrib
Support ancient Greek names of days, months, etc.

*grid in macros/latex/contrib
Grid-based typesetting in double-column documents.

hypdvips in macros/latex/contrib
Improve hyperref support with the Dvips driver.

javadoc in macros/latex/contrib
Document source code.

inlinedef in macros/latex/contrib
Selective expansion within a definition.

ionumbers in macros/latex/contrib
Restyle numbers in math mode.

isomath in macros/latex/contrib
Typeset math according to ISO 31.

liturg in macros/latex/contrib
Typeset Catholic liturgical texts, particularly Missal and Breviary texts.

logical-markup-utils in macros/latex/contrib
Language-dependent inline quotes and dashes.

macqassign in macros/latex/contrib
Typeset Macquarie University assignments.

makebarcode in macros/latex/contrib
Produce various 2/5 and Code 39 barcodes, using only \vrule.

metalogo in macros/latex/contrib
Expose spacing parameters for TeX logos, so they can be optimized for different fonts.

minibox in macros/latex/contrib
Boxes allowing manual line breaks and shrinking to the maximum natural line width.

modref in macros/latex/contrib
Customize cross-references in LaTeX.

multiobjective in macros/latex/contrib
Provide operators used in fields related to multiobjective optimization.

nicetext in macros/latex/contrib
Minimal markup syntax for simple wiki-style text.

pagecont in macros/latex/contrib
Page numbering continuation between documents.

pdfcomment in macros/latex/contrib
Friendly interface to PDF annotations.

pdfmarginpar in macros/latex/contrib
Improved \marginpar with read-only PDF annotations.

*pdfx in macros/latex/contrib
PDF/X-1a and PDF/A-1b support for pdfTeX.

pgfopts in macros/latex/contrib
Extends pgfkeys to handle LaTeX class and package options, much as kvoptions extends keyval.

*pstool in macros/latex/contrib
Optimized conversion of PSTricks figures to PDF, with psfrag support.

psu-thesis in macros/latex/contrib
Thesis package for Penn State University.

rcs-multi in macros/latex/contrib
Typesetting RCS or CVS keywords, with support for multi-file documents.

sageep in macros/latex/contrib
LaTeX style for papers at the Environmental and Engineering Geophysical Society's Annual Meeting.

*silence in macros/latex/contrib
Selective filtering of error messages and warnings.

siunitx in macros/latex/contrib
A comprehensive (SI) units package.

steinmetz in macros/latex/contrib
Produce the electrotechnics Steinmetz notation.

svn-prov in macros/latex/contrib
Variants of \ProvidesPackage, etc., with information automatically determined from Subversion.

syllogism in macros/latex/contrib
Typeset syllogisms and syllogistic-style arguments.

tabularcalc in macros/latex/contrib
Automatic calculation of values in a numeric table.

tabularew in macros/latex/contrib
Handle centering of multicolumn headings.

tdclock in macros/latex/contrib
A ticking digital clock for pdfLaTeX documents and presentations.

tdsfrmath in macros/latex/contrib
Facilitate use of LaTeX for French math teachers.

termlist in macros/latex/contrib
Label any kind of term with an increasing sequence of numbers, as with equation numbers.

theoremref in macros/latex/contrib
Automatically typeset theorem names in references.

tkz-doc in macros/latex/contrib
Documentation macros for tkz-* packages.

tkz-linknodes in macros/latex/contrib
Based on PGF/TikZ, provides for linking elements of amsmath environments such as \align.

todonotes in macros/latex/contrib
Let authors mark things to do in a LaTeX document.

totcount in macros/latex/contrib
Compute and display the last value of counters (sections, pages, etc.).

`tufte-latex` in `macros/latex/contrib`
Document classes inspired by the books and work of Edward Tufte.

`ucdavisthesis` in `macros/latex/contrib`
Thesis/dissertation class for UC Davis.

`ulqda` in `macros/latex/contrib`
Support for the field of qualitative data analysis.

`verbatimbox` in `macros/latex/contrib`
Store verbatim text in a LaTeX box, for use in places where the `verbatim` environment is not allowed.

`vwcol` in `macros/latex/contrib`
Typesetting multicolumn paragraph text with different column widths on a single page.

`xstring` in `macros/latex/contrib`
String manipulation: tests, substrings, substitutions, length, and more.

`yagusylo` in `macros/latex/contrib`
An extended `pifont`, with macros for obtaining one glyph, drawing lines and filling, list environments.

`zwgetfdate` in `macros/latex/contrib`
Fetch dates of used packages and files for macros.

`zwpagelayout` in `macros/latex/contrib`
Page layout, cropmarks, and reflected pages.

macros/latex/exptl

`biblatex-apa` in `macros/latex/exptl/biblatex-contrib`
APA citation and reference style for `biblatex`.

`biblatex-chem` in `macros/latex/exptl/biblatex-contrib`
Experimental chemistry styles for `biblatex`.

`biblatex-chicago-notes-df` in `macros/latex/exptl/biblatex-contrib`
Chicago "notes & bibliography" style files.

`biblatex-historian` in `macros/latex/exptl/biblatex-contrib`
A `biblatex` style based on the Turabian Manual.

`biblatex-jura` in `macros/latex/exptl/biblatex-contrib`
A `biblatex` style for German legal literature.

`biblatex-nature` in `macros/latex/exptl/biblatex-contrib`
A `biblatex` style for *Nature*.

`biblatex-zeitschrift` in `macros/latex/exptl/biblatex-contrib`
A `biblatex` style for *Historische Zeitschrift*.

`cfr-lm` in `macros/latex/exptl`
Enhanced support for GUST's Latin Modern fonts.

`keys3` in `macros/latex/exptl`
Key management for LaTeX3.

macros/luatex

`luainputenc` in `macros/luatex/latex`
Standard `inputenc` package adapted for LuaTeX.

`luamcallbacks` in `macros/luatex/generic`
Register multiple functions in LuaTeX callback.

*`luamplib` in `macros/luatex/generic`
Use MetaPost natively from LuaTeX.

*`luaotfload` in `macros/luatex/generic`
OpenType font loading for (LA)TeX, with syntax similar to XeTeX.

`luatextra` in `macros/luatex/generic`
Core additional functionality for LuaTeX.

macros/xetex

`harvardkyoto` in `macros/xetex/generic`
Harvard/Kyoto input mapping for XeTeX Unicode Devanagari (0900–097F).

`fontwrap` in `macros/xetex/latex`
Bind fonts to Unicode blocks, for automatic font tagging of multilingual text.

`mathspec` in `macros/xetex/latex`
Typeset math in XeLaTeX using any text font.

*`polyglossia` in `macros/latex/contrib`
Multilingual XeLaTeX, with over 50 languages.

`xecjk` in `macros/xetex/latex`
Typeset Chinese/Japanese/Korean with XeLaTeX.

`xecolour` in `macros/xetex/latex`
Defines many colors for use in XeTeX, including in bidirectional text.

`xelibertine` in `macros/xetex/latex`
Support the OpenType font Libertine.

`xepersian` in `macros/xetex/latex`
Typeset Persian and Arabic with XeLaTeX.

`xetexfontinfo` in `macros/xetex/plain`
Query fonts for their supported features.

support

`acroreloadpdf` in `support`
JavaScript to add reload support to Adobe Reader, under Unix-ish systems.

`ctanify` in `support`
Prepare a LaTeX package for upload to CTAN.

`ctantools` in `support`
Search LaTeX packages on CTAN from the command line.

`firefox_ctan_plugins` in `support`
CTAN search plugins for Firefox.

`fragmaster` in `support`
Produce PDF from EPS with `psfrag` substitutions applied.

`meper` in `support`
Java program for editing and previewing MetaPost.

`texdirflatten` in `support`
Perl script that recursively follows a LaTeX document, outputting all graphics and other files into a single directory.

`texloganalyser` in `support`
Perl script to display selected parts of a LaTeX log.

ArsTEXnica #5–7 (2008–09)

Editor's note: *ArsTEXnica* is the journal of GUIT, the Italian TEX user group (`http://www.guit.sssup.it/`).

ArsTEXnica #5, April 2008

Gianluca Pignalberi, Editoriale [From the editor]; p. 3

A short overview of the present issue.

Massimo Guiggiani and Lapo F. Mori, Consigli su come non maltrattare le formule matematiche [How to avoid abusing mathematical formulae]; pp. 5–14

[Published in *TUGboat* 29:2.]

Massimiliano Dominici, Introduzione a XETEX [Introduction to XETEX]; pp. 15–26

Unicode and smart font technologies are the current de facto standard in digital typography. This article should explain how they can be incorporated, with XETEX, in a TEX-based typesetting system.

Claudio Beccari, Macroistruzioni con argomenti delimitati [Macros with delimited arguments]; pp. 27–34

The macro package commonly known as LATEX does not describe any means for defining macros with delimited arguments; furthermore it offers a very small number of them to the user. By means of the primitive commands of the TEX interpreter it is easy to define macros with delimited arguments that may be used also while using the other LATEX macros. This kind of macros may be very useful in some instances, particularly when writing class or package files, where they make it easy to identify the function that any argument plays in the macro expansion. The subject is described with the aid of a practical problem: the Lecture Log.

Enrico Gregorio, HyPlain, più lingue insieme anche in Plain [HyPlain, several languages together under Plain]; pp. 35–42

We describe a system for enabling hyphenation in several languages under Plain TEX, along with an interface to define the used languages and their conventions.

ArsTEXnica #6, October 2008

Gianluca Pignalberi and Massimiliano Dominici, Editoriale [From the editor]; pp. 3–4

A short note about the fifth meeting of the Italian TEX user group (GUIT).

Klaus Höppner, A short introduction to MetaPost; pp. 5–9

MetaPost is strongly related to Knuth's original Metafont. It uses nearly the same graphics language and syntax, but instead of bitmap fonts it produces PostScript output. So it can be used to create high quality graphics. In MetaPost, points and paths may be described by a set of linear equations that are solved by the program. Thus, MetaPost becomes unique among other tools like PSTricks or commercial applications (e.g., CorelDraw).

Additionally, the PostScript subset created by MetaPost can be interpreted by pdfTEX. So MetaPost figures can be directly included with e.g., the standard LATEX graphics package, while normal EPS images have to be converted first to be usable with pdfLATEX.

Agostino De Marco, Gestione avanzata delle figure in LATEX: l'annotazione di illustrazioni e grafici con psfrag/PSTricks e PGF/Ti*k*Z [Advanced graphics handling under LATEX: annotation of figures and graphs with psfrag/PSTricks and PGF/Ti*k*Z]; pp. 10–27

This article shows how the combination of LATEX with the package PSTricks or with PGF/Ti*k*Z can be used to produce advanced, nice-looking illustrations and plots. This subject is dealt with at a technical level biased towards intermediate/advanced users. The aim of the work is presenting a number of examples of how a figure, that is a bitmapped or vector image, might be annotated according to the typographic style of the main LATEX document and of the displayed math.

Roberto Giacomelli, Una tabella che fa calcoli [A computing table]; pp. 28–36

One of main advantages of LATEX is an easy markup syntax of the text. The user is able to build up the informative content of the document without worrying about how it will appear on the page.

This paper aims at testing this remarkable characteristic of LATEX, both for the author's productivity and the quality of his work, building, with particular attention to the syntax project, a new environment of type table for invoices, expense notes and liquidation.

The linguistic aspects will be first discussed, taking into account a testing carried out on a group of users working in the business sector. Then the complete code of the new environment called calctab, will be proposed, starting with the list of the powerful LATEX packages it is based upon, and the numeric TEX capability.

In the end some hints about methods will be

given in order to spread LaTeX more widely in business and professional offices.

Lapo F. Mori, Gestire la bibliografia con LaTeX [Managing bibliographies with LaTeX]; pp. 37–51
[Published in this issue of *TUGboat*.]

Lorenzo Pantieri, Introduzione allo stile ClassicThesis [Introduction to the ClassicThesis style]; pp. 52–66

The purpose of this work is to provide the Italian LaTeX users some tools to write documents using the ClassicThesis style, by André Miede, inspired by Robert Bringhurst's masterpiece *The Elements of Typographic Style* (1992).

This aim is pursued by introducing my personal reworking of the style documentation (Miede, 2007) and analyzing the typical problems faced during the writing of an academic or professional publication, especially in the Italian language, while showing the solutions I think better.

Norbert Preining, TeX Live 2008 and the TeX Live Manager; pp. 67–75
[See abstract in *Die TeXnische Komödie* 2009/2.]

Gianluca Pignalberi and Enrico Bini, LaTeX e grammatiche (la faccia triste dell'informatica) [LaTeX and grammars]; pp. 76–85

Grammars, syntax diagrams and automata are computer science's basic topics; thanks to them our computers can do what they do. LaTeX has to be able to typeset them. In this paper we'll give an overview of which tools we can use to add those elements to our documents.

Agostino De Marco and Massimiliano Dominici, longmedal: un pacchetto per medaglioni divisi su più pagine [longmedal: a package for floating framed boxes spanning several pages]; pp. 86–92

Some textbooks organize different kinds of advanced or secondary material in framed boxes, possibly spanning several pages. The longmedal package aims at providing an easy interface to reproduce such objects in a LaTeX document.

Jean-Michel Hufflen, Specifying translated works in bibliographies; pp. 93–97

First we recall the layout recommended within a bibliography for a translation of a document. Then we explain why entries for translated works cannot be specified nicely if we use BibTeX. A solution is proposed for future implementation in MlBibTeX.

ArsTeXnica #7, April 2009

Gianluca Pignalberi, Editoriale [From the editor]; p. 3
Overview of this issue.

Gianluca Pignalberi and Massimiliano Dominici, Intervista eSamizdat Simone Guagnelli [Interview with eSamizdat founder Simone Guagnelli]; pp. 4–7

Usually TeX is linked to the typesetting of scientific texts, where a considerable use of formulae, and mathematical notation in general, is required. Yet it may prove a useful tool also in different kinds of publications, as shown in this interview with Simone Guagnelli, editor and founder, together with Alessandro Catalano, of eSamizdat.

Giovanni Maschio, TeX per i ciechi e per gli ipovedenti [TeX for blind and vision-impaired people]; pp. 8–12

We present here a number of reasons to choose an existing way to code mathematics, suitable for blind people, avoiding creating new methods.

Claudio Beccari, Il formato archiviabile dei file PDF [The archive format for PDF files]; pp. 13–24

Archiving electronic documents requires a special format called PDF/A-1 by the ISO regulation 19005-1. This paper shows how to obtain this result with the main and subsidiary programs of the TeX system. Some difficulties that are encountered in this process will be also highlighted; some solutions to the above problems will be suggested.

Claudio Beccari, George Kamel, Typesetting Coptic liturgy in Bohairic; pp. 25–31

This paper describes what the authors have done in order to typeset some Coptic texts with LaTeX, mainly in the Bohairic variant used in liturgy. This implied the creation of suitable fonts, the macros for typesetting special liturgical symbols, the hyphenation patterns necessary to typeset with the Coptic alphabet and the rules used by the Bohairic variant.

Riccardo Nisi, Il PostScript in LaTeX [PostScript in LaTeX]; p. 32

PSTricks, along with its links to PostScript, gave me the chance to take an interest in this language, searching for further occasions to integrate it with LaTeX, and enrich its application scope.

[Received from Massimiliano Dominici and Gianluca Pignalberi.]

***Baskerville* 10.1, May 2009**

Editor's note: *Baskerville* is the journal of the UK-TEX Users' Group (`http://uk.tug.org`).

Jonathan Webley, Editorial and survey; p. 2
Welcome to the revived *Baskerville*, and results of the survey regarding publication formats.

Jonathan Webley, Events; pp. 3–4
Announcement of various events in 2009: BachoTEX, Mathematics and Fiction (Knuth being one of the contributors, speaking on *Surreal Numbers*), EuroTEX and TUG meetings, and the UKUUG (UK Unix and Open Systems User Group) summer conference.

Jonathan Webley, The Hound; p. 4
A "somewhat easy", cryptic crossword; solution on p.7.

Jonathan Webley, Currency symbols in LaTeX; pp. 5–6
Commands for generating a wide variety of currency symbols.

Jonathan Webley, Wikibooks; pp. 6–8
Background and TEX support in wikis and wikibooks, focusing on the LaTeX wikibook at `http://en.wikibooks.org/wiki/LaTeX`.

***Eutypon* 21, October 2008**

Editor's note: *Eutypon* is the journal of the Greek TEX Friends (`http://www.eutypon.gr`).

Claudio Beccari, The CB Greek fonts; pp. 1–13
This paper takes its origin from the documentation accompanying a revision of the CB Greek fonts completed on 1st January 2008, but it goes into deeper detail with comments on many font features that are commonly overlooked. It tells the story of the CB Greek fonts and describes the new features associated with this new distribution. (*Article in English.*)

Dimitrios Filippou, A discussion with Claudio Beccari, TEXie and book lover; pp. 15–26
Claudio Beccari reveals how he got involved with TEX, how he started creating Greek fonts with METAFONT, and his ultimate love for books. (*Article in English without abstract.*)

Artemios G. Voyiatzis, Typesetting a diploma thesis with XeLaTeX; pp. 27–34
Writing a diploma thesis, at undergraduate or graduate level, is a painful exercise. Particularly in the field of theoretical and applied sciences, the choice of a suitable tool for the presentation of the thesis is an essential — and often irreversible — point in the process of thesis writing. This article presents the author's experiences in writing theses with LaTeX, LaTeX 2ε and more recently with XeLaTeX. The author of this article hopes his experiences to be of some help for other theses writers in the Greek academic world. (*Article in Greek with English abstract.*)

Elia Koumi, Stories in print at the museum of the newspaper *Chaniotika Nea*; pp. 35–39
The Museum of Typography of the newspaper *Chaniotika Nea* has been in operation in Chania (Crete, Greece) since 2005. Founder and soul of the museum is the publisher and editor of the newspaper, Giannis Garedakis, who during his 40-year career had the opportunity to experience the evolution of 20th century typography. The museum, unique in Greece, is attracting many visitors, and aspires to expand with new exhibits. (*Article in Greek with English abstract.*)

Dimitrios Filippou, Is TEX dying?; pp. 41–49
Some statistics show that the interest for TEX and other similar systems of electronic typesetting is on the decline. The postings on the newsgroup `comp.text.tex` showed a peak in 2002 and are decreasing ever since. The publication of new books on TEX and similar systems has almost ceased. A simple statistical analysis shows that within few years TEX will be dead. Only a radical change — a new successor — will renew the interest for electronic typesetting based on TEX. (*Article in Greek with English abstract.*)

Apostolos Syropoulos, OpenType fonts: a short presentation; pp. 51–56
OpenType fonts are not a recent technological development, yet it is not well known what they really are, what their relationship is to other font formats, or what their advantages are in comparison to other font formats lile TrueType and Type1. It would not be an exaggeration to say that the OpenType format is just a superset of the Type1 and TrueType formats. (*Article in Greek with English abstract.*)

[Received from Apostolos Syropoulos]

Die TEXnische Komödie 2008/2–2009/2

Editor's note: *Die TEXnische Komödie* is the journal of DANTE e.V., the German-language TEX user group (`http://www.dante.de`).

DTK 2008/2

Christine Römer, PSTricks for linguistic texts

PSTricks offers to all areas of linguistics the option to illustrate the relevant phenomena according to the usual "factual" practice. Many ways to new concepts of rich visualization are opened. The use of macros in the PSTricks family is advantageous compared to that of single linguistic packages. When a package, like `pst-jtree`, is particularly designed for linguistics the act of writing is clearly minimized because of specific shortcuts. Hopefully this article with its examples of use can motivate bringing into the PSTricks family further linguistic packages.

Dominik Wassenhoven, Managing your bibliography with BibLATEX (part one)

This article gives an overview of the LATEX package BibLATEX in two parts. Part one focuses on how to use BibLATEX, with an emphasis on the differences and the advantages as compared with standard BibTEX. Part two, which will be published in the next edition of DTK, shows how to create your own styles for both citations, and bibliographies. As BibLATEX provides a wealth of opportunities, no attempt is made to give an all-comprehensive introduction. Please refer to the package documentation.

Michael Stötzel, A web based data base for central template management with LATEX connection

Writing letters in perfect form with LATEX is no mystery. But what to do when one wants centrally managed templates which are filled-in with distinct data from a data base? A central web server with PHP and MySQL can help out with this cumbersome task.

DTK 2008/3

Ulrike Fischer, First steps with XeLATEX

The following article gives a brief (at least it was brief at the beginning) introduction to using XeTEX with the LATEX format ("XeLATEX").

Uwe Ziegenhagen, Document management with LATEX and Subversion

Version control systems provide quite a number of advantages to programmers and authors in their daily work. Collaboration in a team is simplified drastically, as the laborious and error-prone exchange of files via FTP or e-mail is dropped. Older versions of a file can be restored without problems, and joining different versions is simplified. Another advantage that should not be underestimated is the possibility to create backups "in passing". Subversion is a modern version control system running on all common platforms and requiring not much time to become acquainted with configuring and using it. This article describes the usage of Subversion with LATEX. It will be explained how to install and configure it on Windows and Linux systems. Finally, some packages will be discussed which facilitate a convenient integration of information provided by subversion into LATEX.

DTK 2008/4

Dominik Wassenhoven, Managing your bibliography with BibLATEX (part two)

In part two of this introduction to the BibLATEX package, an example from the humanities is chosen to illustrate how to create your own citation and bibliography styles. As BibLATEX provides a wealth of opportunities, no attempt is made to give an all-comprehensive introduction. Please refer to the package documentation. The article is based on BibLATEX 0.7, while version 0.8 has been released in the meantime.

Uwe Siart, An introduction to BibTEX for managing your bibliographies

Although BibTEX has been available for managing your bibliography for many years, using it still causes trouble for most LATEX users. I constantly receive BibTEX databases containing severe syntax errors. Drawing from the kind of errors I have come across, users seem not to have attained even a basic understanding of how BibTEX works. This article provides a brief practical introduction to the BibTEX system, urging new users to drop a document-related approach of managing your bibliography in favour of a more generally-minded one, while advanced BibTEX users can revise their general command of the system.

DTK 2009/1

Stanislav Jan Šarman, DEK shorthand script with METAFONT and LATEX

This article presents Text2DEK, a METAFONT and LATEX-based web application which reproduces German text as "Verkehrsschrift" shorthand notes. By using the example of "DEK" stenography is outlined and afterwards it is described how to model in METAFONT shorthand characters that interconnect, giving shorthand glyphs for words (stenemes?),

which is described in a meta language that implements them in METAFONT as characters. A delineation of the system architecture and an abstract of shorthand history and systems completes the article.

[See also the author's article in *TUGboat* 29:3.]

Hàn Thế Thành, TrueType fonts for pdfTEX

[Published in this issue of *TUGboat*.]

Philipp H. Poll and Michael Niedermair, The "Linux Libertine" font and XeTEX

The article shows in a historical digest how the font "Linux Libertine" evolved, which thoughts, ideas, etc., were integrated, and its potential when used in XeTEX.

Uwe Ziegenhagen, Conference management with LaTeX

To organise events like conferences, congresses, or workshops requires a whole set of bills, lists of participants, name badges, and other documents. LaTeX provides for all these sorts of document types suitable class files, and via the package `datatool`, by Nicola Talbot, access to CSV (comma separated values) files is made possible. This article uses a fictitious example to describe the various packages and their interrelation.

Dominik Wagenführ, Compilation of periodicals with LaTeX

LaTeX can set many documents. In particular, of course, scientific papers, but letters or presentations are also no problem. This article will show that it even can compile a PDF magazine and, above all, how recurring problems can be solved.

Adelheid Gross, The package todonotes

Some time ago the idea came up of presenting a continuing series of smaller (La)TeX packages which either could be helpful or just for fun. This article will deliver an insight into the scope of application the use of LaTeX brings.

Gerd Neugebauer, Fooling with TeX logos in HTML

Although TeX and friends can produce almost perfect results in the print area they could not make their breakthrough on the Web. So still many sites exist created with HTML. Writing about TeX, automatically the question arises how to set the logo best. This article will give some answers to this.

DTK 2009/2

Norbert Preining, TeX Live 2008 and TeX Live Manager

TeX Live 2008 was the first release of TeX Live that came with a new program called TeX Live Manager, or `tlmgr` for short. TeX Live Manager takes care of some tasks hitherto covered by `texconfig`, which itself has never been available for Windows. It also brings a number of new features to TeX Live, including a long-standing demand for continuous online updates of the entire TeX distribution. This article presents the new TeX Live installer called TeX Live Manager, and describes some more news in TeX Live 2008.

Rolf Niepraschk, Installing TeX Live on Linux

In additon to Norbert Preining's article on TeX Live in this issue of DTK, this article describes how to install TeX Live on Linux. It also gives some advice on how to use this TeX distribution. The author deals with openSUSE in particular, but as Linux/Unix platforms are very much the same, his presentation can be drawn upon by users of other platforms as well.

Stefan Kottwitz, TeX Live on netbooks under Ubuntu Linux

Netbooks, or mini notebooks, equipped with up-to-date hardware are becoming ever more popular, as they are quite portable, yet performing sufficiently for working with (La)TeX efficiently. Most models are shipped with the now obsolete operating system Windows XP. So, this article deals with Ubuntu Linux and TeX Live 2008 as a dual boot system, offering a Free alternative.

Uwe Ziegenhagen, Counting words in LaTeX documents

In most word processors, it is quite easy to have the number of words and paragraphs counted. Some LaTeX editors such as Kile also offer this feature on mouse-click. However, apart from Kile and its brethren we rely on external tools for this, some of which will be presented in this article.

Herbert Voss, Converting colour graphics to grayscale

In most cases, a document containing pictures in colour will be printed in black and white, only. Depending on the printing process you might like to convert colour pictures to grayscale in advance as this task is better not left to the printer (and his software).

Rolf Niepraschk, Additional cutting edge

If a document is printed on professional printing machines, it can be a problem when the page has a color area which runs to (bleeds off) the border of the document. In such a case the color border should be a little bit greater to be sure that when the pages are trimmed, there are no white lines.

[Received from Herbert Voß.]

MAPS 36–37 (2008)

MAPS is the publication of NTG, the Dutch language TEX user group (`http://www.ntg.nl`).

MAPS 36 (Spring 2008)

Taco Hoekwater, ConTEXt conference 2008; p. 65

Announcement of the ConTEXt annual meeting for 2008 in Bohinj, Slovenia.

Taco Hoekwater and Hans Hagen, MetaPost library project; pp. 66–68

This paper documents the target and implementation milestones of the MetaPost library project (MPlib). [Related material was published in *TUGboat* 29:3.]

Taco Hoekwater and Hans Hagen, The MetaPost library; pp. 69–81

[Published in *TUGboat* 29:3.]

Hans Hagen and Taco Hoekwater and Volker Schaa, Reshaping Euler; pp. 82–84

[Published in *TUGboat* 29:2.]

Hans van der Meer, Blocks and arrows with MetaPost; pp. 85–89

Typesetting of blocks and arrows in ConTEXt with MetaPost.

***MAPS* 37 (Fall 2008)**

Taco Hoekwater, Redactioneel [From the editor]; p. 1

Overview.

Taco Hoekwater, TUG conference 2009; p. 2

Announcement of the TUG annual meeting for 2009 at the University of Notre Dame, Indiana, USA.

Hans Hagen, The TEX–Lua mix; pp. 3–11

[Published in *TUGboat* 29:3.]

Mojca Miklavec and Arthur Reutenauer, Putting the Cork back in the bottle; pp. 12–16

[Published in *TUGboat* 29:3.]

Taco Hoekwater, PDF genereren voor e-readers [PDF generation for e-readers]; pp. 17–24

Notudoc is a commercial Internet application that ConTEXt uses for the on-the-fly generation of PDF documents, for the e-readers from IREX Technologies, and more. This article gives a look behind the scenes.

[Translation of Dutch abstract.]

Hans Hagen, Dealing with XML in ConTEXt MkIV; pp. 25–39

The tree-based method of handling XML in MkIV.

Willi Egger, Printing labels with ConTEXt; pp. 45–47

Sometimes one needs to print a single label which will be glued onto a package, a large envelope or for the identification of a box. In certain situations one wants to produce a series of identical labels or one needs to typeset whole databases of addresses. ConTEXt offers the possibility of using an XY-arranging procedure to print on each of the labels being present on a sheet. Here a possible approach is presented for labels of the size 105 x 42.3mm, i.e. 14 labels on an A4 sheet. It is shown how to print a single label but also how to get multiple copies of the same content and how to prepare sheets of labels containing the addresses of a database.

Hans van der Meer, CD and DVD covers in ConTEXt; pp. 48–54

Production of CD and DVD covers in several variations using ConTEXt.

Taco Hoekwater and Hans Hagen, Punk from Metafont to MetaPost; pp. 55–58

To make Knuth's punk font usable with ConTEXt MkIV, it had to be converted from Metafont to MetaPost input. This article highlights the most important changes that had to be made in the conversion process.

Hans Hagen and Taco Hoekwater, How to convince Don and Hermann to use LuaTEX; pp. 59–66

Using the newly randomized punk font.

Hans Hagen, The Punk module; pp. 67–69

The Punk module in ConTEXt.

Jonathan Kew, TEXworks: Lowering the barrier to entry; pp. 70–72

[Published in *TUGboat* 29:3.]

Norbert Preining, TEX Live 2008 and the TEX Live Manager; pp. 73–89

TEX Live 2008 has been released recently, and the DVDs are ready to go gold. This is the first release of TEX Live shipping the TEX Live Manager, tlmgr for short. Besides taking over some of the tasks from texconfig (which has never been available for Windows) it finally brings many new features to the TEX Live world, most importantly the option for dynamic updates. This article will present the new TEX Live Installer, the TEX Live Manager, and at the end lists other changes in TEX Live 2008.

Taco Hoekwater, EuroTEX conference 2008; p. 90

Announcement of the EuroTEX (and ConTEXt) 2009 meeting in The Hague, The Netherlands.

[Received from Wybo Dekker]

The PracTEX Journal 2008-2–2008-3

The PracTEX Journal is an online publication of the TEX Users Group. Its web site is `http://tug.org/pracjourn`. All articles are available there.

The PracTEX Journal 2008-2, August 2007

Issue theme: Class and style packages.

Lance Carnes, From the Editor

From the Readers, Feedback

The Editors, News from Around: TUG 2008; User group news; Math font videos

The Editors, Class & Style — An introduction

Wentao Zheng, Go game positions with MetaPost

This article introduces a method of drawing Go game positions with MetaPost. It begins with how the Go game is modeled in the MetaPost language, then explains the detailed implementation, and ends with some examples of Go game positions.

Lars Madsen, Page styles on steroids (or, memoir makes page styling easy)

Designing a page style has long been a pain for novice users. Some parts are easy, others need good LATEX knowledge. In this article we will present the Memoir way of dealing with page styles including new code added to the recent version of Memoir, that will reduce the pain to a mild annoyance. We will end the article with a series of common scenarios and how to solve these.

Lance Carnes, Opinion: Enduring LATEX documents

The title of this opinion piece may seem a little strange. After all, if I keep my document source files safely stored away, and have a LATEX system to format them, they should always work, right? Well, sometimes. More often than not, though, a set of LATEX files more than a few years old will probably not format the same today as they did in the original edition.

David Walden, Travels in TEX Land: A bigger experiment with ConTEXt

In this column in each issue I muse on my wanderings around the TEX world. In my column in the 2007-2 issue (`http://tug.org/pracjourn/2007-2/walden/`) I tried a small experiment with using ConTEXt. In this issue I describe an additional, quite extensive, effort to use ConTEXt — to create a picture book for a "slide show" I was involved in creating a number of years ago.

The Editors, Ask Nelly: How do I center only the last line of a paragraph? How do I get the first and last entry of each index page in its header?

The Editors, Distractions: Fun packages — sudoku solvers

The PracTEX Journal 2008-3, December 2008

Issue theme: LATEX and TEX on the Web.

Lance Carnes and Paul Blaga, From the Editor

From the Readers, Feedback

The Editors, News from Around: What is new in LATEX; User group news — three print journal releases

George Grätzer, A gentle learning curve for LATEX

Is there an easy way to get started in LATEX? I suggest that there is.

Tomas Morales de Luna, Writing posters in LATEX

LATEX is an excellent editor for the creation of poster presentations. When writing a poster with LATEX, several options are available. Here we would like to present some of these options and in particular the a0poster class and Brian Amberg's poster template. We shall introduce the basics as well as some useful packages and techniques to make your poster look nice. You can even choose to write your poster sequentially or up from different text blocks positioned absolutely or relatively within the page.

Paul A. Thompson, Clinical trials management on the Internet — I. Using LATEX and SAS to produce customized forms

In clinical trials, forms are used to gather data which is then entered into a database. Paper-based forms are still the standard for data collection, due to portability, stability, and storage considerations. In producing forms, SI (a SAS product which works with the Internet) is used to facilitate the entry of information about participants in a clinical trial over the Internet. Using LATEX, the forms are then processed to produce a `.pdf` file. The `.pdf` is returned to the requesting party using a return page on the web browser. The entire process takes about 20 seconds. The system allows highly customized forms to be produced, in which values are inserted into appropriate locations on the forms. LATEX is important due to its superior scripting capabilities, while SAS provides a very flexible database from which to pull information to be inserted into the forms, as well

as providing a method for scripting up the entire transaction. The code required for the process and general approach is outlined.

Paul A. Thompson, Clinical trials management on the Internet — II. Using LaTeX, PostScript, and SAS to produce barcode label sheets

In clinical trials, it is often necessary to print labels with barcodes to identify samples. The availability of open-source tools for barcode management is still somewhat limited. Until recently, no LaTeX tools existed for the manipulation and encoding of barcodes. Using direct PostScript, barcodes can be defined for strings to be printed on labels. Using LaTeX, the labels can be queued up into appropriate sizes for specific label sheets, and tnen converted into .pdf files. Using SAS, the label sheets can be ordered in a web environment, queued up into appropriate files, and returned to users in a printable file.

Tim Arnold, plasTeX: Converting LaTeX documents to other markup languages

This article introduces plasTeX, a software package for converting LaTeX documents to other markup languages. It begins with usage details including examples of how to create HTML and DocBook XML from LaTeX sources. Then, it describes development details: how plasTeX works and how developers can use it to create or extend a publishing workflow in a production setting. Finally, it ends with some examples of customizing the parser and renderer as well as suggestions of how others can contribute to this open source project.

David Walden, Travels in TeX Land: A sidebar for a book

In this column in each issue I muse on my wanderings around the TeX world. In this issue I describe a small effort to typeset a sidebar for a book project.

The Editors, Book reviews: *More Math into LaTeX*, *Tout ce que vous avez toujours voulu savoir sur LaTeX sans jamais oser le demander*

English translation of the second book's title: *Everything you always wanted to know about LaTeX but were afraid to ask.*

The Editors, LaTeX & TeX web sites

The Editors, Ask Nelly: How do I replace one overlay with another on a Beamer slide? How do I typeset ancient Greek quotations

The Editors, Distractions: Writing recipes with LaTeX

TEXemplares 8 (2006)

Editor's note: *TEXemplares* is the publication of CervanTeX, the Spanish TeX user group (`http://www.cervantex.es`).

atopos, What and why LaTeX; pp. 4–9

[Translation of the introduction: "In the guise of an appeal".]

Anyone who comes here with no knowledge of the matter I'll address, and who, however, is confusedly attracted by the implications of the title of this paper, might feel let down by the discovery that my contribution has nothing to do with the promotion of some magic product with erogenous powers hitherto unknown about a certain part of our fleshy bodily constitution. Or perhaps, on the contrary, will feel, against any predictions, gladly surprised at the corroboration that the computer program I will present — yes, that's what's at stake — is capable of eliciting an intense and lasting intellectual pleasure, that has little to envy from those other pleasures that he might have been thinking of when entering the room.

What is, then, this "LaTeX" that makes it to your booklets? How does this puzzling entity relate to those other entities, no less puzzling, known us "the humanities"?

[The article includes a section 2: LaTeX and its family: of writers and printers; and a section 3: The linguistic model and the visual metaphor. A more historical and conceptual than technical discussion of LaTeX, citing sources from Turing to Wittgenstein.]

Javier Bezos, Word hyphenation; pp. 10–19

Word hyphenation is an orthographic problem that still raises problems, both theoretical and practical. This article analyzes possible criteria for hyphenation in Spanish, with a set of rules, and afterwards studies the way in which such rules can be implemented in TeX. Among other things, considered are the 1999 and, especially, the 2005 rules of the *Real Academica Española*, with comments about their requirements.

The article is divided in two parts. The first is devoted to the analysis of the rules and their relationship to the first two criteria. The second part presents the rest of the criteria, commentary on some sources, a brief history of the patterns, and a discussion of their current implementation.

[Compiled by Federico Garcia]

***Zpravodaj* 16(2–4)–18(4), 2006–2008**

Editor's note: *Zpravodaj* is the bulletin of ČSTUG, the TEX user group for the Czech and Slovak languages (`http://www.cstug.cz`).

***Zpravodaj* 16(2–4), 2006**

Ton Otten and Hans Hagen, PRAGMA ADE, Exkurze do ConTEXtu, česká verze [ConTEXt, an excursion, Czech version; translated from English by Vít Zýka, Ján Buša, Jiří Hrbek, Martina Plachá and Petr Tesařík]; pp. 57–224

This issue presents a Czech translation of the manual available at `http://www.pragma-ade.com/general/manuals/mp-cb-en.pdf`.

***Zpravodaj* 17(1), 2007**

Jaromír Kuben, Dopis předsedy ČSTUGu [Opening letter from the ČSTUG President]; pp. 1–2

Petr Tesařík, S češtinou a slovenštinou do Babylónu [Czech and Slovak languages into the Babel package]; pp. 2–11

This article is one of the ČSTUG grant results which presents a new Czech and Slovak implementation for the Babel package according to the ČSLATEX requirements.

Zdeněk Wagner, Babylón mluví hindsky [Babel speaks Hindi]; pp. 12–20

Babel provides a unified interface for creation of multilingual documents. Unfortunately none of the Indic languages is currently supported. Typesetting in Indic languages is based on specialised packages. The most advanced of them is Velthuis Devanāgarī for TEX because it already provides Hindi captions as well as a macro for a European style date. A language definition file for plugging Hindi into Babel has therefore been recently developed.

The second part of the paper explains differences between Unicode and Velthuis transliteration. This is important for understanding the tool that can convert Hindi and Sanskrit documents from Microsoft Word and OpenOffice.org into TEX via an XSLT 2.0 processor and a Perl script as well as a method of making the PDF files searchable.

Finally the paper discusses some possibilities of further development, mainly the advantages offered by XETEX and by forthcoming integration of Lua into pdfTEX.

Source code and notes are available on the author's web page, `http://icebearsoft.euweb.cz/tex/`.

Zdeněk Wagner, Babylón v TEX Live 2007 [Babel in TEX Live 2007]; pp. 21–23

With the inclusion of XETEX into TEX Live the structure of the `language.dat` file has been changed slightly. Due to this fact the new Czech and Slovak module, which is not yet distributed with official Babel, cannot be installed smoothly. The article introduces an installation package of the new module not only for TEX Live but also for other well known TEX distributions. Functionality of XELATEX is also preserved.

Petr Březina, Abecední řazení a sestavování rejstříků výhradně pomocí makrojazyka TEXu [Alphabetical sorting and creation of indexes exclusively by means of the TEX language]; pp. 23–30

The article presents the macro package index for creation of indexes. An important part of this package is the TEX macro "sort" for alphabetical sorting. It uses a four-pass sorting algorithm which can be accommodated to different languages via a sorting table. Its memory requirements are independent of the length of the list of entries to be sorted. The treatment of page numbers in index entries is sophisticated. The package index is available, including French documentation, on the author's home page, `http://www.volny.cz/petr-brezina/`.

Pavel Stříž, Proměnné záhlaví a zápatí [Variable headings and footings]; pp. 31–59

This article deals with the typesetting of headings and footings. It describes basic opportunities and ways to typeset them. It uses the standard package fancyhdr in nearly all examples. It shows the ways how to prepare variable objects which are usually page dependent. In case variable objects have additional dependencies themselves, the article introduces a method which generates a part or whole of a TEX document using PHP and MySQL tools.

Zdeněk Wagner, Marrákéš 2006, krátká reportáž [Brief Report on the 27th TUG annual meeting]; pp. 60–64

***Zpravodaj* 17(2), 2007**

Jaromír Kuben, Úvodníček předsedy [Welcome to the issue by the ČSTUG President]; pp. 65–66

Vít Zýka, Používáme pdfTEX V: aktuální pozice sazby [Using pdfTEX V: Current typesetting position]; pp. 67–72

[Published in this issue of *TUGboat*.]

Robert Mařík, Vkládání JavaScriptů pdfLATEXem prakticky [Inserting JavaScripts with pdfLATEX in practice]; pp. 72–83

This article describes a few possibilities of using the JavaScript language available in the Adobe Reader browser to enhance possibilities and effects in the PDF files created by pdfLaTeX. Among other things, we briefly describe the technical background of some related LaTeX packages available on CTAN.

An accompanying file is available at `http://user.mendelu.cz/marik/latex/testjs.zip`.

Jozef Říha and Pavel Stříž, Prezentačný software pre LaTeX [Presentation software for LaTeX]; pp. 84–95

The article is an introduction to the preparation of presentations. In the first part, it gives information about the general problems of preparing presentations. In the second part it points out the TeX classes Slides and Prosper, plus the FoilTeX package. In the next part it briefly mentions the existence of the packages UwmSlide and TeXPower. The last-discussed package is Beamer. The Beamer package is a fully featured tool for creating presentations in TeX and this will be discussed in another issue in more detail. In the last part, the authors mention a few tips and hints for better presentations and recommend Internet sources for given topics.

Jozef Říha and Pavel Stříž, Príprava posteru [Scientific poster preparation]; pp. 95–103

The article is an introduction to the preparation of scientific posters. In the first part, it deals with the definition, specification and sizes of posters. It points to proper TeX programmes and packages, dealing mainly with A0poster, Sciposter, Poster and Epssplit. It also briefly mentions non-TeX tools, such as OpenOffice.org Impress, Microsoft PowerPoint and its templates plus suitable software products. In the second part, it discusses printing and the price of posters. The last part gives Internet links to some real-world galleries of posters and some other recommended sources.

Josef Tkadlec, Opakování operací a relací při zlomu řádku [Repeating operations and relations at line breaks]; pp. 103–105

Two solutions for repeating of operations and relations in line breaks are presented, depending on whether the relation or operation is given by a command or by a character.

Zápis z Valné hromady CSTUGu ze dne 17. 11. 2007, Brno [Report from the CSTUG general assembly of 17 November 2007]; pp. 106–107

Zpráva o činnosti CSTUGu [Report on CSTUG Activities]; pp. 107–109

Podporované projekty [Supported projects by CSTUG]; pp. 109–112

***Zpravodaj* 18(1–2), 2008**

Jaromír Kuben, Zdeněk Wagner Úvodník a Opravenka [Introduction and errata]; pp. 1–2

Karel Píška, Testování LM-fontů s ohledem na českou a slovenskou sazbu [Latin Modern fonts testing with regard to Czechoslovak typesetting requirements]; pp. 3–43

This extended article presents grant results with the author's major findings and results. It makes recommendations for changes to the LM-font creators after performing comparisons of fonts such as CM, LM, CS and EC in the Type 1 format. The article also makes comparisons based on metric and graphical data. The tested parameters were the widths of the letters, kernings, differences in LM, CS and CM fonts, and finally the technical quality of the glyphs.

The tools used during the testing were FontForge and also MetaType1. The testing scripts were done in .bat and AWK and are published on the author's websites. The author proposes some changes to improve the actual state of the fonts, commenting on this in depth, including illustrations and tables.

They also presented their thoughts on the creation of a new OpenType font and rewriting testing scripts in LuaTeX. Some findings were presented at BachoTeX 2006, EuroTeX 2006 and the EuroBachoTeX 2007 conferences and their proceedings have been published. The author's notes can be found on `http://www-hep2.fzu.cz/~piska/`.

Luboš Prchal and Pavel Schlesinger, Poster v TeXu [Posters in TeX]; pp. 44–55

The creation of a poster in this article is done using the A0poster class. It includes examples of packages such as multicol, color, fancybox, graphics, epsf, picinpar and psfrag. In the next section of the article the authors discuss the settings for the layout of the poster. In the last section the authors present a template for a poster to be created with A0poster. Two real-world posters are inserted in the conclusion. A style-sheet for a poster can be downloaded from `http://www.karlin.mff.cuni.cz/~antoch/`.

Luboš Prchal and Pavel Schlesinger, Prezentace v TeXu [Presentations in TeX]; pp. 56–63

In this article, the authors share their knowledge, notes and experience with the Beamer presentation class. The article includes installation notes and the first steps in Beamer, the `pause` command, `<+->`, `+` and `-` options, generating a title page and a table of contents. It also explains how to change the design of a presentation by setting `\use*theme{value}` and `\setbeamer*{element}{value}`. In the conclusion

of the article, the authors recommend the Beamer user's guide and a few Internet resources for further reading. The Beamer template of the authors is published independently on their website. A stylesheet for a Beamer presentation can be downloaded from http://www.karlin.mff.cuni.cz/~antoch/.

Pavel Stříž and Michal Polášek, Ukázky prezentací [Examples of presentations]; pp. 63–75

In the first part, the article presents a simple presentation created both with the PDFSlide and PDFScreen packages and with the Beamer class. In the second part, the article discusses the Beamer class in a real-world presentation in more detail. It starts with the creation of METAPOST graphics. After that it comments on some settings of the Beamer deisgn, generating a title page and a section table of contents. Next follows an example of \alert and \convertMPtoPDF commands. The generated output is a PDF file. This file is converted for printing purposes using the pdfpages package. At the conclusion of the article a selection of individually named title pages is prepared with printed materials which are to be given to the members of the committee, e.g., before a thesis defense.

Roman Plch and Petra Šarmanová, Interaktivní 3D grafika v HTML a PDF dokumentech [Interactive 3D Graphics in HTML and PDF Documents]; pp. 76–92

The paper presents the authors' experience with including interactive 3D objects into HTML and PDF documents, starting with modifying 3D graphics in Maple by means of the library JavaViewLib, followed by its export into the MPL or JVX format and finishing with web integration. In the second part, the authors describe exporting Maple 3D graphics into the VRML format, then its conversion to U3D with the use of Deep Exploration, and finish with its embedding into a PDF document by means of pdfTEX and the movie15 package. This procedure preserves the possibility of the user's interaction with 3D objects even in the final PDF document without the necessity of the local installation of Maple or other graphical programs.

Zdeněk Hlávka, Velkovýroba tabulek pomocí AWK [Large-scale production of tables in AWK]; pp. 93–95

This short article demonstrates the capabilities of AWK when producing LaTeX tables and formatting them in large quantities.

První oznámení a informace k TEXperience 2008 [Invitation to the TEXperience 2008 conference]; p. 96

***Zpravodaj* 18(3), 2008**

Sborník z TEXperience 2008 [From the TEXperience 2008 Conference Committee and About the Venue of the Conference]; pp. 97–101

Program TEXperience 2008 [The Scientific and Social Programmes of the TEXperience 2008 Conference]; pp. 102–103

Jiří Rybička, Typografie a TEX [TEX and typography]; pp. 104–109

Computer typesetting is a very widespread application commonly used with personal computers. It is necessary to handle appropriate programs but it is also very important to apply typographic rules. This paper deals with the question of how to solve this problem in TEX and its formats? Quo vadis typography in TEX?

Jan Přichystal, Inovace a rozšíření systému TEXonWeb [Innovation and enhancement of the system TEXonWeb]; pp. 110–115

[Published in this issue of *TUGboat*.]

Petr Sojka and Michal Růžička, Publikování z jednoho zdroje v odlišných formátech pro různá výstupní zařízení [Parallel electronic publications]; pp. 116–129

TEX is traditionally used as an authoring tool for the paper publishing of scientific texts and textbooks. Parallel electronic publications that are meant for on-screen viewing and web delivery are also demanded by readers for many reasons today. This paper discusses the ways to single-source author publishing from a LaTeX source file, and it shows examples of several textbooks published by this approach. Special attention is given to the web document generation either to HTML or XHTML markup with a notation translated to MathML. Also discussed is a personalised automated document generation for a digital library project DML-CZ, http://dml.cz/.

Petr Olšák, DocBy.TEX – dokumentování zdrojových textů TEXem [DocBy.TEX — Documenting source code with TEX]; pp. 130–141

DocBy.TEX (web site http://www.olsak.net/docbytex.html) is a TEX macro software product which gives the possibility of documenting source code written in various programming languages, for example written in C. You can include parts of your source code into your documentation. All occurrences of documented words in your included source code are automatically made active links if encTEX and pdfTEX are active. To make PDF ourput, you need no more than pdfTEX with encTEX. The table

of contents and the index are also created automatically. The sorting of the words in the index is implemented at the TEX macro level.

Tomáš Hála, Značkovací styl pro rychlou sazbu bibliografických citací [Markup style for fast typesetting of bibliographic references]; pp. 142–150

The paper deals with the typesetting of bibliographic references. The introduction covers some important methods of styles and the systems for processing and typesetting bibliographic references. Basic problems of the proceedings in typesetting are dealt with. No one method alone is suitable for the typesetting of proceedings. A basic style and some extensions focus on designing cross references. Sophisticated database methods use up a lot of time while the database is being prepared and it can only be used once. In conclusion a new style for faster markup and typesetting is created.

The input conditions are: (a) no database usage, (b) a simple interface for authors and/or typesetters, (c) complete markup in LATEX macros, (d) extendable and modifiable when necessary, (e) the result does not need detailed proof.

The style `bib.sty`, `http://konvoj.cz/styly/biblio/2.30/bib.sty`, contains macros for the most frequently used types of bibliographic references and for elements of references. Some additional macros are described and electronic documents are also included.

Petra Talandová, Možnosti tabulkové sazby [Typesetting possibilities for tables]; pp. 151–160

This paper deals with the typesetting of tables. It briefly describes packages prepared for tables and for the modification of individual characteristics of tables. Selected packages that can contribute most to the typesetting are described, and an analysis of their compatibility is done. Almost all chosen packages work together and extend the possibilities of typesetting. Examples of typesetting with and without these packages show the potential of table typesetting.

Zdeněk Wagner, LATEX v sazečské praxi [LATEX in the typographer's profession]; pp. 161–174

TEX is known mainly in the academic world and is used for writing technical publications. Many people are aware of the possibility of creating high-quality typesetting with TEX. However, these days when programs with graphical user interfaces hiding important information prevail, it is difficult to find instructions on how to prepare with TEX a file for a phototypesetter or a digital printer. The article demonstrates the methods of using LATEX in practice. A few macro packages that prepare leaflets and invitation cards are discussed. Also the typesetting of books including their covers.

Author's notes on his LATEX packages are available on `http://icebearsoft.euweb.cz/tex/`.

Sbohem TEXperience 2008! Buď vítána TEXperience 2009! [Good-bye TEXperience 2008 and welcome to TEXperience 2009!]; p. 175

***Zpravodaj* 18(4), 2008**

Zdeněk Wagner, Úvodník [From the editor]; p. 177

Vít Zýka, Příprava dokumentů pro formátování [Document preparation for typesetting]; pp. 178–199

In this article we express the general principles of a good document and we pose the requirements for their editing, processing and visualisation. Based on these requirements we show that an appropriate format is a structurally-marked document. We explain what structure marking is and describe its features. Finally we mention the tools for manipulating structure-marked documents and we sketch the ways they are formatted by TEX.

Vít Zýka, Článek a logo ConTEXtem: tutoriály [Article and logo by ConTEXt: Tutorials]; pp. 200–211

In this tutorial we show how to create a technical article using ConTEXt. The resulting text will be a shortened version of the real article, and so it will contain most of the elements of this kind of document.

In the second tutorial we show how to create a PDF vector figure by ConTEXt. The figure is rather primitive but illustrative. Although the drawing uses METAPOST, its language description is not our goal. We are focusing on a step-by-step demonstration of ConTEXt infrastructure for this kind of work.

Tutorials are available on author's website `http://www.zyka.net/?id=typography&lang=en`.

Petr Březina, Zrcadlová sazba [Parallel typesetting]; pp. 212–226

The article presents an efficient solution to the problem of typesetting two texts in parallel on facing pages in bilingual editions. The solution assumes that the two texts are saved separately in two files and that they are divided into small sections, as the Bible is divided into verses. This division makes it possible to synchronize the texts automatically. Each of the two texts can have its own footnotes, illustrations and other insertions as if it were an ordinary document, but the texts are broken into

individual pages simultaneously in such a way that each odd page contains the same sections as the corresponding even page. The presented macros are available on the author's web site, `http://www.volny.cz/petr-brezina/`.

Petr Březina, Sazba trojjazyčné knihy [Typesetting of a trilingual book]; pp. 227–236

TEX has an insertion mechanism that makes it possible to handle several texts simultaneously. It can be used in preparation of multilingual books. In this article, the author describes how he has typeset a Latin-Greek-Czech edition of *The Dream of Scipio* where each double page contains the text simultaneously in the three languages. The described macros are available on the author's home page, `http://www.volny.cz/petr-brezina/`.

Vít Zýka, Postřehy ze setkání TEXperience 2008 a ConTEXt 2008 [Impressions from the TEXperience 2008 conference and ConTEXt meeting 2008]; pp. 237–242

Pozvánka na TEXperience 2009, EuroTEX 2009 a třetí setkání uživatelů ConTEXt, TUG 2009 [Invitations to TEXperience 2009, EuroTEX 2009 and TUG 2009]; pp. 243–247

[Received from Pavel Stříž.]

TUG Institutional Members

American Mathematical Society, *Providence*, *Rhode Island*

Aware Software, Inc., *Midland Park*, *New Jersey*

Banca d'Italia, *Roma*, *Italy*

Center for Computing Sciences, *Bowie*, *Maryland*

Certicom Corp., *Mississauga*, *Ontario*, *Canada*

CSTUG, *Praha*, *Czech Republic*

Florida State University, School of Computational Science and Information Technology, *Tallahassee*, *Florida*

IBM Corporation, T J Watson Research Center, *Yorktown*, *New York*

Institute for Defense Analyses, Center for Communications Research, *Princeton*, *New Jersey*

MacKichan Software, Inc., *Washington/New Mexico*, *USA*

Marquette University, Department of Mathematics, Statistics and Computer Science, *Milwaukee*, *Wisconsin*

Masaryk University, Faculty of Informatics, *Brno*, *Czech Republic*

Moravian College, Department of Mathematics and Computer Science, *Bethlehem*, *Pennsylvania*

MOSEK ApS, *Copenhagen*, *Denmark*

New York University, Academic Computing Facility, *New York*, *New York*

Princeton University, Department of Mathematics, *Princeton*, *New Jersey*

Springer-Verlag Heidelberg, *Heidelberg*, *Germany*

Stanford University, Computer Science Department, *Stanford*, *California*

Stockholm University, Department of Mathematics, *Stockholm*, *Sweden*

University College, Cork, Computer Centre, *Cork*, *Ireland*

University of Delaware, Computing and Network Services, *Newark*, *Delaware*

Université Laval, *Ste-Foy*, *Québec*, *Canada*

University of Oslo, Institute of Informatics, *Blindern*, *Oslo*, *Norway*

TUG Business

TUG 2009 election report

TUG 2009 election report

Nominations for TUG President and the Board of Directors in 2009 have been received and validated. Because there is a single nomination for the office of President, and because there are fewer nominations for Board of Directors than there are open seats, there will be no requirement for a ballot this election.

For President, Karl Berry was nominated. As there were no other nominees, he is duly elected and will serve for another two years.

For the Board of Directors, the following individuals were nominated: Jonathan Fine, Steve Grathwohl, Jim Hefferon, Klaus Höppner, Steve Peter, David Walden. As there were fewer nominations than open positions, all the nominees are duly elected.

Terms for both President and members of the Board of Directors will begin with the Annual Meeting at the University of Notre Dame. Congratulations to all.

Board members Dick Koch, Martha Kummerer and Arthur Ogawa have decided to step down at the end of their terms. On behalf of the Board, I wish to thank them for their service, and for their continued participation through July.

Statements for all the candidates, both for President and for the Board, are appended (in alphabetical order). They are also available online at `http://www.tug.org/election`, along with announcements and results of previous elections.

◇ Barbara Beeton
for the Elections Committee

Karl Berry

Biography:

I have served as TUG president since 2003 and was a board member for two terms prior to that. During my term as president, we've enacted new initiatives, expanding the scope of the special member and institutional memberships. We've also partnered with Addison-Wesley for online book sales, with Bigelow&Holmes for making the Lucida fonts available through TUG and with Adobe making the Utopia typeface family freely available, among others.

As president, I coordinate the formal and informal meetings of the Board, provide direction and oversight to the Executive Director, and monitor TUG's financial transactions. I also serve on the conference committee, and thus have been one of the principal organizers for all TUG-sponsored conferences since 2004, both the annual meetings and the Practical TEX conferences, including web site and program creation, coordination of publicity, and so forth.

I have been on the TUG technical council for many years. I co-sponsored the creation of the TEX Development Fund in 2002, and am one of the primary system administrators and webmasters for the TUG servers. I'm also one of the production staff for the *TUGboat* journal.

On the TEX development side, I'm currently editor of TEX Live, the largest free software TEX distribution, and thus coordinate with other TEX projects around the world, such as CTAN, LATEX, and pdfTEX. I developed and still maintain Web2c (Unix TEX) and its basic library Kpathsea, a freely redistributable library for path searching, Eplain (a macro package extending plain TEX), GNU Texinfo, and many other projects. I am also a co-author of *TEX for the Impatient*, an early comprehensive book on TEX, now freely available. I first encountered and installed TEX in 1982, as a college undergraduate.

Personal statement:

I believe TUG can best serve its members and the general TEX community by working in partnership with the other TEX user groups worldwide, and sponsoring projects and conferences that will increase interest in and use of TEX. I've been fortunate enough to be able to work essentially full time, pro bono, on TUG and TEX activities the past several years, and plan to continue doing so if re-elected.

Jonathan Fine

I work for the Open University (the UK's leading provider of distance education) as a TEX expert for print media. I'm also halfway through a two-year project on putting mathematics on web pages.

In 2006–7 I set up MathTran, which now provides typesetting of TEX-notation formulas to images as a public web service, serving about a million images a month.

MathTran shows the value of TEX as a web service, which I'd like to extend to whole documents. Installing and configuring TEX can be slow and difficult. Using TEX through a web browser will help beginners.

Part of my math-on-web project is a page where students can interactively create a TEX-notation formula, say for putting on a web page or in a word-processor document.

I have a doctorate in Mathematics and although not my career I still have research interests. I have been using TEX for over 20 years, and joined TUG in 1989. For the past two years I've been Chair of the UK TEX Users Group, and have recently been re-elected for another two years.

The past three years have seen UK TUG come out of a long period of inactivity and decline. The credit for this of course belongs to the Committee and the members, and not simply myself. We've organised three successful meetings, adopted a new constitution, and set up a website with links to UK TEX resources.

As a board member I would bring to TUG a focus on a key core community, namely those who write material with lots of mathematics. I have a particular interest in providing help and support, particularly through web pages.

TUG, by virtue of TEX being a typesetting program, rightly has a focus on print media. But to flourish we must also use new media effectively. The Open University faces the same challenge, and my experience there will help TUG.

You can comment on this statement, and read the comments of others, at `http://jonathanfine.wordpress.com/2009/01/31/tug-board-election/`.

TUG has a special responsibility, to publicise TEX and related fonts, programs, documentation and other resources.

I'd like TUG to offer more to institutional members. In particular, we should help them share user support experience and resources. Supporting TEX can be daunting without outside help.

When I joined TUG there were over 150 institutional members. There are now just 27. The loss I feel the most is the Library of Congress.

Steve Grathwohl

Biography:

I have used TEX since 1986, first as a hobby, and then "professionally" after I joined Duke University Press in 1983 on the staff of the *Duke Mathematical Journal*. Eventually I supervised the production of the journal (for both print and online incarnations), and I wrote and maintained the class files for typesetting. Since 2005 I have been responsible for loading content for our 35 journals onto multiple platforms as well as being TEXnical liaison for Duke to Project Euclid, a hosting service for over 50 independent mathematics journals. My current work involves a significant amount of work with XML content and metadata schemas as well as being the in-house TEX specialist.

Personal statement:

TEX has proved to be an astoundingly robust piece of software, and the continuing development of projects like LATEX3, LuaTEX and XETEXhelps insure TeX's vitality into the future. I would like to see the TUG board continue to support these and others (like TEX Gyre and TEXworks) that contribute to a 21st-century TEX.

Jim Hefferon

I have enjoyed working on the Board, trying to promote the interests of TEX and friends. In the future I would like to continue to do so, trying to balance fiscal prudence with taking the opportunities that arise.

Klaus Höppner

Biography:

I got a PhD in Physics in 1997. After some post-doctoral fellowships I have been working working in the

Control Systems group of an accelerator center in Darmstadt, Germany, since 2002. My first contact to LaTeX was in 1991, using it frequently since then.

I was preparing the CTAN snapshot on CD, distributed to the members of many user groups, from 1999 until 2002. I was heavily involved in the organization of several DANTE conferences and EuroTeX 2005. Since 2000, I am a member of the DANTE board, acting as president since 2006.

Personal statement:

In the years since Karl Berry's presidency the cooperation of TUG and European user groups improved a lot. My candidacy is in the hopes of helping to continue this trend. Projects like TeX Live and CTAN owe their success to the work of active volunteers, but also to the support and cooperation of the user groups.

Steve Peter

Biography:

I am a linguist and publisher originally from Illinois, but now living in New Jersey. I first encountered TeX as a technical writer documenting Mathematica. Now I use TeX and friends (these days, lots of ConTeXt) for a majority of my publishing work, and occasionally consult on it. I am especially interested in multilingual typography and finding a sane way to typeset all of those crazy symbolisms linguists create. As if that weren't bad enough, I've recently begun studying typeface design.

I got involved in TUG via translations for *TUGboat*, where I also work on the production team. This past year, I was on the organizing committee for PracTeX San Francisco, co-edited the TUG 2004 conference pre-proceedings, and was appointed to the TUG Board (thanks, Karl!). Working with and for the community has been so rewarding that I've decided to run for a regular term on the board.

Personal statement:

The future of TeX and TUG lies in communication and working together to promote and sustain the amazing typographic quality associated with TeX and friends. I am especially interested in having TUG support various projects (technical and artistic) that will serve to bolster TeX and TUG's visibility in the world at large.

David Walden

Biography:

I was supposed to be studying math as an undergraduate at San Francisco State College; but, from my junior year I was hacking on the school's IBM 1620 computer. While working as a computer programmer at MIT's Lincoln Laboratory, I did the course work for a master's degree in computer science at MIT. Most of my career was at Bolt Beranek and Newman Inc. (BBN) in Cambridge, Massachusetts, where I was, in turn, a computer programmer, technical manager, and general manager. At BBN, I had the good fortune to be part of BBN's small ARPANET development team. Later I was involved in a variety of high tech professional services and product businesses, working in a variety of roles (technical, operations, business, and customer oriented).

Throughout my business career and now during my so-called retirement years, I have always done considerable writing and editing. This led to my involvement since the late 1990s with TeX, becoming a member of TUG and now as a TUG volunteer. I have served as a member of the TUG Board for the last three years and also served in the role of Treasurer (I know bookkeeping from my business career). I have used LaTeX to write three published books and numerous articles. I have contributed to *The PracTeX Journal* since its inception, I founded TUG's Interview Corner, and I have helped behind the scenes with the *TUGboat* web site.

You can learn more about me at: `http://www.walden-family.com` and `http://tug.org/interviews/interview-files/dave-walden.html`.

Personal statement:

I am interested in continuing to serve on the TUG Board for three reasons:

1. To more explicitly serve the community that has so generously served me via `comp.text.tex`, CTAN, *TUGboat*, etc.
2. As a way of helping maintain the viability for years to come of TeX and the TeX world, entities I would call "national treasures" except for their world wide nature.
3. Because rubbing shoulders more closely with various TUG members will help me learn more about TeX faster.

As a TUG Board member, my frame of mind has been to get things done quickly and pragmatically with enough generality so evolution is possible.

TEX Development Fund 2009 report

TEX Development Fund committee

In *TUGboat* 28:3 (September 2007), we presented a roadmap for future TEX development, focusing on three major projects: the LuaTEX extension, the TEX Gyre fonts, and the TEXworks front end. Since then, considerable progress has been made on all three. We have also supported additional projects.

Major projects

LuaTEX (`http://luatex.org`): MetaPost has been rewritten as a library as of MetaPost 1.100, thus greatly facilitating graphics support in LuaTEX. The next major task is enhanced math support; the proposal is available online at `http://tug.org/tc/devfund/luamath08.pdf`. Arabic support in the collateral Oriental TEX project sponsored by Dr. Idris Hamid at Colorado State University is also ongoing.

TEX Gyre (`http://gust.org.pl/projects/e-foundry/tex-gyre`): new releases with additional glyphs and other features continue apace. As noted elsewhere in this issue, the Gyre project is now on firm legal footing: URW++ has made the base 35 PostScript fonts available under the LPPL.

TEXworks (`http://tug.org/texworks`): TEX Live 2009 is expected to contain a TEXworks binary for Windows, with binaries for other systems available from the TEXworks web site. Development and documentation support continue, with other contributors joining Jonathan Kew, the principal author.

The following smaller projects, some completed and some ongoing, have also been supported recently.

SVG output in MetaPost

Applicant: Taco Hoekwater, The Netherlands, `http://tug.org/metapost`.
Amount: US$1500; acceptance date: 28 Nov 2008.

Implement SVG output as a backend in MetaPost version 1.200. This project was co-sponsored by a generous contribution to TUG made by David Crossland.

Bulaq Press Arabic font development

Applicant: Khaled Hosny, Egypt.
Amount: US$1000; acceptance date: 5 Nov 2008.

Bulaq Press, established in Cairo in 1820, has developed one of the most widely used Arabic typefaces that has been a standard in Arabic printing for more than 150 years. However, no fully conformant digitized version of that typeface is available; only a few proprietary fonts come close.

This project aims to digitize the Bulaq (Amiriya) Press typeface in the form of an OpenType font that implements all contextual features of the original typeface as well. Also, the project will work on extending it to cover other languages using the Arabic script. Finally, the project will consider writing any macro packages or support files needed to use the font in Arabic-capable TEX engines such as XETEX and LuaTEX.

Free (libre) font initiative

Applicant: David Crossland, Great Britain, `http://tug.org/fonts/librefontfund.html`.
Amount: US$500; date: 3 Sep 2008.

A project to collect and create free (libre) fonts and tools, and make them widely available for general use. TUG is also contributing administrative support to this project. A report from January 2009 is available at `http://tug.org/tc/devfund/fontfund09.pdf`.

Obyknovennaya Novaya font development

Applicant: Basil Solomykov, Russia.
Amount: US$1000; date: 7 Aug 2008.

The Obyknovennaya Novaya ("Ordinary New Face") typeface was widely used in the former USSR for scientific and technical publications, as well as for textbooks. The current implementation is in Metafont (`http://litwr.boom.ru/obnov.html`); the author aims to provide an outline version as well.

Inconsolata

Applicant: Raph Levien, USA, `http://levien.com/type/myfonts/inconsolata.html`.
Amount: US$1000; acceptance date: 30 Nov 2005.

The Inconsolata design is nearly final at this writing, and has been available from the web page above for some time. (LA)TEX support has also been written, and is available from `http://mirror.ctan.org/fonts/inconsolata`.

– – * – –

The TEX Development Fund was created by the TEX Users Group in 2003, under the aegis of the TUG Technical Council, to foster growth of TEX-related technical projects.

As always, we remain most appreciative of the ongoing support from individuals and institutions, which have made the recent grants possible. Contributions are always welcome!

For application information, the complete list of projects, and more, please see the web site.

◇ TEX Development Fund committee
`http://tug.org/tc/devfund`

TUG Business

TUG financial statements for 2008

David Walden, TUG treasurer

The financial statements for 2008 have been reviewed by the TUG board but have not been audited. They may change slightly when the final 2008 tax return is filed. As a US tax-exempt organization, TUG's annual information returns are publicly available on our web site: `http://www.tug.org/tax-exempt`.

Revenue (income) highlights

Membership dues revenue was slightly up from 2007 to 2008 (at the end of December 2008 we had 1,549 paid members); conference income was substantially down; and interest income was down somewhat. Product sales income was down; contributions income was up about $1,500. Altogether, revenue decreased 9 percent from 2007 to 2008.

Cost of Goods Sold and Expenses highlights

Payroll, office expenses, and *TUGboat* production and mailing continue to be the major expense items in 2008.

The production and shipping expense of the 2008 TEX Collection software includes $1,400 for 1600 DVD's, $350 for 1500 mailers, and $2,161 for postage and labor. The 'Postage/Delivery — Members' item is the mailing cost for individual *TUGboat* issues sent from the office, instead of the bulk mailing house; this expense was down in 2008 from 2007 on account of the timing and contents of the first *TUGboat* issue in the respective years. Direct *TUGboat* expenses were up in 2008 from 2007 because two issues were larger than usual (the first issue was a EuroTEX proceedings) and included color.

Overall, expenses are up about $8K in 2008 because of a modest cost-of-living increase in payroll, overhead (credit card and bank charges), and a significant increase in contributions made by TUG.

The bottom line

Subtracting 'Cost of Goods Sold' from 'Income', gross profit is down from 2007 to 2008. As expenses are up about $8K, the net income for 2008 is a loss of about $8K, compared to a profit of about $15K in 2007. This is pretty much as budgeted. The year 2007 was the first year of an increase in fees, and the annual conference in San Diego made an unusually large profit. Thus, for 2008 (and 2009) we budgeted no increase in fees, essentially spreading the 2007 surplus over 2008 (and 2009).

Often we have a prior year adjustment that takes place early in the year to compensate for something that had to be estimated at the time the books were closed at year end; however, at this time there are no known prior year adjustments for 2008.

Balance sheet highlights

TUG's end-of-year asset level is essentially the same from 2007 to 2008.

The 'Committed Funds' come to TUG specifically for designated projects: the LATEX project, the TEX Development fund, and so forth. They have been allocated accordingly and are disbursed as the projects progress. TUG charges no overhead for administering these funds.

'Prepaid Member Income' is member dues that were paid in 2008 for 2009 and beyond. Most of this liability (the 2009 portion) was converted to 'Membership Dues' for 2009 on January 2009. The payroll liabilities are for 2008 state and federal taxes due January 15, 2009.

Because of the large decrease in year-to-year profit, the Total Equity is also down significantly.

Summary

TUG remained financially solid as we entered 2009, such that we again budgeted no fee increase for 2009, continuing to use the carry over surplus from 2007. This cannot go on indefinitely.

TUG continues to work closely with the other TEX user groups and ad hoc committees on many activities to benefit the TEX community.

TUG 12/31/2008 (versus 2007) Balance Sheet

	Dec 31, 08	Dec 31, 07
ASSETS		
Current Assets		
Total Checking/Savings	162,709	160,490
Accounts Receivable	95	254
Other Current Assets	1,315	1,327
Total Current Assets	164,119	162,071
Fixed Assets	2,396	3,726
TOTAL ASSETS	**166,515**	**165,797**
LIABILITIES & EQUITY		
Liabilities		
Committed Funds	33,569	24,413
Prepaid member income	2,905	4,075
Payroll Liabilities	1,096	1,080
Total Current Liabilities	37,570	29,568
TOTAL LIABILITIES	**37,570**	**29,568**
Equity		
Unrestricted	136,230	120,820
Net Income	-7,285	15,409
Total Equity	128,945	136,229
TOTAL LIABILITIES & EQUITY	**166,515**	**165,797**

TUG 2008 (versus 2007) Revenue and Expenses

	Jan - Dec 08	Jan - Dec 07
Ordinary Income/Expense		
Income		
Membership Dues	103,171	101,956
Product Sales	5,809	7,667
Contributions Income	6,987	5,423
Annual Conference	-1,339	6,827
Interest Income	5,341	5,901
Advertising Income	405	230
Total Income	120,374	128,004
Cost of Goods Sold		
TUGboat Prod/Mailing	31,401	25,130
Software Production/Mailing	3,911	1,111
Postage/Delivery - Members	3,164	6,296
Conf Expense, office + overhead	1,036	1,164
JMM supplies/shipping	829	
Member Renewal	408	335
Copy/Printing for members	30	55
Total COGS	40,779	34,091
Gross Profit	79,595	93,913
Expense		
Contributions made by TUG	10,525	5,750
Office Overhead	12,595	11,653
Payroll Exp	62,200	59,863
Professional Fees	230	200
Depreciation Expense	1,330	1,498
Total Expense	86,880	78,964
Net Ordinary Income	-7,285	14,949
Other Income/Expense		
Other Income		
Prior year adjust	0	459
Total Other Income	0	459
Net Other Income	0	459
Net Income	-7,285	15,408

Calendar

2009

May 15 – Aug 15 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. Minnesota Center for Book Arts, Minneapolis. Sites and dates are listed at palimpsest.stanford.edu/byorg/gbw

Jun 24 – 27 DANTE: Exhibitor at LinuxTag, Berlin, Germany. www.dante.de

Jul 5 – 25 Wells College Book Arts Center, Summer Institute, Aurora, New York. www.wells.edu/bkarts/summer2009.htm

Jul 8 – 9 Workshop: Towards a Digital Mathematics Library (DML 2009), Grand Bend, Ontario, Canada. www.fi.muni.cz/~sojka/dml-2009.html

Jul 14 – 19 TypeCon 2009: "Rhythm", Atlanta, Georgia. www.typecon.com

---

TUG 2009
University of Notre Dame, Notre Dame, Indiana

Jul 28 – 31 The 30th annual meeting of the TEX Users Group. tug.org/tug2009

Aug 3 – 7 SIGGRAPH 2009, "Network Your Senses", New Orleans, Louisiana. www.siggraph.org/s2009

Aug 7 – 9 UKUUG Summer Conference 2009, UKs Unix & Open Systems User Group, Birmingham, UK. ukuug.org/events/summer2009

Aug 11 – 14 Balisage: The Markup Conference, Montréal, Québec. www.balisage.net

Aug 31 – Sep 4 EuroTEX 2009 and 3rd ConTEXt meeting, "Educational Uses of TEX", Den Haag, The Netherlands. www.ntg.nl/EuroTeX2009

Sep 1 – 4 Book history workshop, École de l'institut d'histoire du livre, Lyon, France. ihl.enssib.fr

Sep 6 – Nov 23 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. San Francisco Public Library, San Francisco, California. Sites and dates are listed at palimpsest.stanford.edu/byorg/gbw

Sep 9 Workshop on mathematical content in electronic media, Open University, Milton Keynes, UK. groups.google.com/group/uk-math-content-2009/web/home

Sep 15 – 18 ACM Symposium on Document Engineering, Munich, Germany. www.documentengineering.org

Sep 25 "A Short History of Printing": lecture by Frank Romano, Museum of Printing, North Andover, Massachusetts. www.museumofprinting.org/txp/Events

Sep 30 – Oct 1 XML-in-Practice 2009, Arlington, Virginia. www.idealliance.org/conferences_and_events

Oct 16 – 18 American Printing History Association's 34th annual conference, "The Book Beautiful", Newport, Rhode Island. www.printinghistory.org/htm/conference/2009/CFP-2009.htm

Oct 16 – 18 The Seventh International Conference on the Book, University of Edinburgh, Scotland. booksandpublishing.com/conference-2009

Oct 17 GuIT meeting 2009 (Gruppo utilizzatori Italiani di TEX), Pisa, Italy. www.guit.sssup.it/guitmeeting/2009

Oct 26 – 30 Association Typographique Internationale (ATypI) annual conference, "The Heart of the Letter", Mexico City. www.atypi.org

Oct 26 – Nov 13 "Late letterpress: The work of Desmond Jeffrey", exhibition, with a talk on Oct 27, St Bride Library, London, England. stbride.org/events

Oct 29 – 31 Guild of Book Workers, Standards of Excellence Annual Seminar, San Francisco, California. palimpsest.stanford.edu/byorg/gbw

Nov 21 Journée GUTenberg & Assemblée générale, Centre FIAP, Paris, France. www.gutenberg.eu.org/manifestations

Dec 7 – Feb 19 "Marking Time": A traveling juried exhibition of books by members of the Guild of Book Workers. Allen Library, University of Washington, Seattle. Sites and dates are listed at palimpsest.stanford.edu/byorg/gbw

Status as of 15 June 2009

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 206 203-3960, e-mail: office@tug.org). For events sponsored by other organizations, please use the contact address provided.

An updated version of this calendar is online at www.tug.org/calendar.

TEX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at `http://tug.org/consultants.html`. If you'd like to be listed, please see that web page.
To place a larger ad in *TUGboat*, please see `http://tug.org/TUGboat/advertising.html`.

Dangerous Curve
PO Box 532281
Los Angeles, CA 90053
+1 213-617-8483
Email: `typesetting (at) dangerouscurve.org`
Web: `http://dangerouscurve.org/tex.html`
We are your macro specialists for TEX or LATEX fine typography specs beyond those of the average LATEX macro package. If you use XETEX, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical TEX and LATEX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a TEX book.

Martinez, Mercè Aicart
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) ono.com`
Web: `http://www.edilatex.com`
We provide, at reasonable low cost, TEX and LATEX typesetting services to authors or publishers worldwide. We have been in business since the beginning of 1990. For more information visit our web site.

Peter, Steve
New Jersey, USA
+1 732 287-5392
Email: `speter (at) mac.com`
Specializing in foreign language, linguistic, and technical typesetting using TEX, LATEX, and ConTEXt, I have typeset books for Oxford University Press, Routledge, and Kluwer, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. I have extensive experience in editing, proofreading, and writing documentation. I also tweak and design fonts. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Shanmugam, R.
No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.
Choolaimedu, Chennai-600094, Tamilnadu, India
+91 9841061058
Email: `rshanmugam (at) yahoo.com`
As a Consultant I provide consultation, technical training, and full service support to the individuals, authors, corporates, typesetters, publishers, organizations, institutions, etc. and I also support to leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level of automation in the field of Typesetting/Prepress/Composition, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc. I have sound knowledge in creating Macros/Styles/Templates/Scripts and Conversions with automation using latest softwares in industry.

Sievers, Martin
Im Treff 8, 54296 Trier, Germany
+49 651 81009-780
Email: `info (at) schoenerpublizieren.com`
Web: `http://www.schoenerpublizieren.com`
As a mathematician I offer TEX and LATEX services and consulting for the whole academic sector and everybody looking for a high-quality output. From setting up entire book projects to last-minute help, from creating citation styles to typesetting your math, tables or graphics — just contact me with information on your project.

Veytsman, Boris
46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: `borisv (at) lk.net`
Web: `http://www.borisv.lk.net`
TEX and LATEX consulting, training and seminars. Integration with databases, automated document preparation, custom LATEX packages, conversions and much more. I have about fourteen years of experience in TEX and twenty-seven years of experience in teaching & training. I have authored several packages on CTAN, published papers in TEX related journals, and conducted several workshops on TEX and related subjects.

TUGboat

Volume 30, Number 1 / 2009