TUGboat

Volume 28, Number 3 / 2007
TUG 2007 Conference Proceedings

TeX Users Group

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

Memberships and Subscriptions

2007 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, contact the TUG office or see our web site.

Addresses

General correspondence, payments, etc.
TeX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Delivery services, parcels, visitors
TeX Users Group
1466 NW Naito Parkway
Suite 3141
Portland, OR 97209-2820
U.S.A.

Telephone
+1 503 223-9994

Fax
+1 206 203-3960

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`

World Wide Web
`http://www.tug.org/`
`http://www.tug.org/TUGboat/`

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

[printing date: September 2007]

TUGboat

The Communications of the TEX Users Group

Volume 28, Number 3, 2007
TUG 2007 Conference Proceedings

TeX Users Group

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

Memberships and Subscriptions
2007 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, contact the TUG office or see our web site.

Addresses
General correspondence, payments, etc.
TeX Users Group
P. O. Box 2311
Portland, OR 97208-2311
U.S.A.

Delivery services, parcels, visitors
TeX Users Group
1466 NW Naito Parkway
Suite 3141
Portland, OR 97209-2820
U.S.A.

Telephone
+1 503 223-9994

Fax
+1 206 203-3960

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`

World Wide Web
`http://www.tug.org/`
`http://www.tug.org/TUGboat/`

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

[printing date: September 2007]

TUG 2007 Proceedings

San Diego State University
San Diego, California, USA
July 17–20, 2007

TUGboat

COMMUNICATIONS OF THE TEX USERS GROUP

TUGboat Editor Barbara Beeton
Proceedings Editor Karl Berry

Volume 28, Number 3 • 2007
Portland • Oregon • U.S.A.

TUG 2007 — Practicing TeX

July 17–July 20, 2007

San Diego State University ▪ San Diego, California, USA

Sponsors

TeX Users Group ▪ DANTE e.V.

Adobe Systems Inc. ▪ Addison-Wesley ▪ Carleton Production Centre ▪ Design Science Inc. ▪ Integre MacKichan Software ▪ NTG ▪ O'Reilly & Associates ▪ Springer ▪ von Hoerner & Sulger GmbH

Thanks to the sponsors, and to all the speakers, teachers, and participants, without whom there would be no conference. Special thanks to Sue DeMeritt for her help with local coordination, Wendy McKay for organizing the Mac OS X gatherings, and Duane Bibby for the (as always) excellent drawing.

Conference committee

Cheryl Ponchin ▪ Karl Berry ▪ Robin Laakso ▪ Sue DeMeritt

Participants

David Allen, University of Kentucky
Tim Arnold, SAS
Caleb Ashley, Gaithersburg, MD
Dave Bailey, MacKichan Software, Inc.
Kaveh Bazargan, Focal Image Ltd
Nelson Beebe, University of Utah
Barbara Beeton, American Mathematical Society
Karl Berry, TeX Users Group
Jon Breitenbucher, College of Wooster
Steve Brenton, Long Beach, CA
Philip Brown, Texas A & M Univ. at Galveston
Lance Carnes, Personal TeX Inc.
Robert Burgess, Cornell University
Dennis Claudio, Richmond, CA
Jennifer Claudio, St. Lawrence Academy
Don DeLand, Integre Technical Publishing Co.
Sue DeMeritt, Center for Communications Research, La Jolla, CA
Paulo Ney de Souza, UC Berkeley
Ron Fehd, Center for Disease Control and Prevention
Frances Felluca, INFORMS
Peter Flynn, Silmaril Consultants
August Gering, Duke University Press
Steve Grathwohl, Duke University Press
Eitan Gurari, Ohio State University
Hans Hagen, Pragma ADE
Michele Hake, American Physical Society
Idris Samawi Hamid, Colorado State University
William Hammond, SUNY Albany
Steven Harris, San Diego, CA
Jim Hefferon, St. Michael's College
Hartmut Henkel, von Hoerner & Sulger GmbH
Taco Hoekwater, Elvenkind BV
Klaus Höppner, DANTE e.V.
Morten Høgholm, LaTeX team and Technical University of Denmark
Roberto Ierusalimschy, Lua team, PUC-RIO
Calvin Jackson, Caltech
Mirko Janc, INFORMS
Shannon Jones, FRB Richmond
Jonathan Kew, SIL International
Richard Koch, University of Oregon
Martha Kummerer, University of Notre Dame
Robin Laakso, TeX Users Group
Richard Leigh, United Kingdom
Jenny Levine, Duke University Press
Barry MacKichan, MacKichan Software, Inc.
Barbara Mastrian, Rutgers
Wendy McKay, Caltech
Andrew Mertz, Eastern Illinois University
Jaime Moore, Decision & Sensor Analytics
Stephen Moye, American Mathematical Society
Brian Papa, American Meteorological Society
Oren Patashnik, BibTeX author
Cheryl Ponchin, Center for Communications Research, Princeton, NJ
Walter Reddall, Redondo Beach, CA
Joseph Riel, San Diego, CA
Leonard Rosenthol, Adobe Systems Inc.
Chris Rowley, Open University
Volker RW Schaa, DANTE e.V.
Martin Schröder, pdfTeX team
Herbert Schulz, Naperville, Illinois
Heidi Sestrich, Carnegie-Mellon University
William Slough, Eastern Illinois University
Lowell Smith, Salt Lake City, UT
Jon Stenerson, MacKichan Software, Inc.
Ari Stern, Caltech
Larry Thomas, Saint Peter's College
Paul Topping, Design Science, Inc.
Alan Wetmore, US Army
Peter Wilson, The Herries Press
Martin Woolstenhulme, San Diego, CA

TUG 2007 — program and information

Tuesday July 17	9 am–5 pm	track 1: *LaTeX workshop*, Sue DeMeritt & Cheryl Ponchin	
	9 am	track 2: *MetaPost workshop*, Hartmut Henkel	
	10:30–10:45 am	*break*	
	10:45 am	track 1 continues	
	10:45 am	track 2a: *Beamer & TikZ workshop*, William Slough & Andrew Mertz	
	10:45 am	track 2b: *ConTeXt workshop*, Hans Hagen	
	12:30–2 pm	*lunch*	
	2 pm	track 1 continues	
	2 pm	track 2a continues	
	2 pm	track 2b: *Lua and LuaTeX*, Roberto Ierusalimschy and Taco Hoekwater	
	3:30–3:45 pm	*break*	
	5–7 pm	*registration & reception,* at the Tula Community Center	
Wednesday July 18	8–9 am	*registration*	
	8:30 am	Karl Berry, TeX Users Group	*Welcome*
	8:35 am	Peter Wilson, Herries Press	Keynote: *Between then and now — A meandering memoir*
	9:30 am	Barbara Beeton, AMS & TUG	*STIX fonts and Unicode*
	9:55 am	Jonathan Kew, SIL	*SIL font projects*
	10:15 am	*break*	
	10:30 am	Dick Koch, Univ. of Oregon	*Multiple TeX distribution support in MacTeX*
	11:05 am	Jim Hefferon, St. Michael's College	*CTAN package sourcing*
	11:45 am	Jonathan Kew	*XeTeX Live*
	12:20 pm	Morten Høgholm, DTU	*LaTeX3 project update*
	12:45 pm	*lunch*	
	1:45 pm	Klaus Höppner, DANTE e.V. & TUG	*Typesetting tables with LaTeX*
	2:25 pm	David Allen, Univ. of Kentucky	*Three-dimensional graphics in LaTeX*
	3:05 pm	Morten Høgholm	*The breqn package: revised and revised*
	3:45 pm	*break*	
	4:00 pm	Ari Stern, Caltech	*Incorporating LaTeX text with LaTeXiT*
	4:20 pm	Leonard Rosenthol, Adobe Systems	*Everything you wanted to know about PDF but were afraid to ask*
	5 pm	q&a	
Thursday July 19	8:30 am	Robert Burgess, Cornell Univ.	*CrossTeX: A modern bibliography management tool*
	9:10 am	Andrew Mertz & William Slough, Eastern Illinois University	*Programming with PerlTeX*
	9:50 am	Chris Rowley, Open University	*Vistas for TeX*
	10:30 am	*break*	
	10:45 am	Paul Topping, Design Science	*MathType 6's TeX input for MS Word and Wikipedia*
	11:25 am	William Hammond, SUNY Albany	*Dual presentation with math from one source*
	12:05 am	Barry MacKichan, MacKichan Inc.	*Design decisions for a structured front end to LaTeX*
	12:45 pm	*lunch*	
	1:45 pm	Don DeLand, Integre	*From TeX to XML: The legacy of techexplorer and the future of math on the Web*
	2:25 pm	Eitan Gurari, Ohio State Univ.	*LaTeX conversion into normalized forms and speech*
	3:05 pm	Paulo Ney de Souza, UC Berkeley	*Long-time preservation strategies for TeX-sourced content*
	3:45 pm	*break*	
	4 pm	q&a, TUG meeting	
	7 pm	*banquet* (at the Aztec Center)	
Friday July 20	8:30 am	Roberto Ierusalimschy, PUC-RIO	*About Lua*
	9:30 am	Hans Hagen, Pragma ADE	*Introduction to the LuaTeX project*
	9:50 am	Taco Hoekwater, Elvenkind BV	*The Lua–TeX interface: Extra tables and callbacks*
	10:30 am	*break*	
	10:45 am	Hans Hagen	*LuaTeX attributes*
	11:25 am	Idris Hamid, Colorado State Univ.	*Arabic script typography*
	12:05 pm	Hans Hagen	*Zapfino as torture test*
	12:45 pm	*lunch*	
	1:45 pm	Nelson Beebe, Univ. of Utah	*Extending TeX and METAFONT with floating-point arithmetic*
	2:25 pm	Taco Hoekwater	*MPLib: Turning MetaPost into a reusable component*
	3:05 pm	Hans Hagen	*ConTeXt MkIV*
	3:45 pm	*break*	
	4 pm	Idris Hamid	*Critical editions*
	4:40 pm	*panel*	*Nelson Beebe, Taco Hoekwater, Jonathan Kew, Barry MacKichan, Oren Patashnik;* moderator: *Don DeLand*

TUG 2007: A few words

Tim Arnold
SAS

The 2007 TEX Users Group annual conference was a gathering of old friends and new faces, all concerned with and excited about the future of "TEX and friends".

The workshops preliminary to the conference were interesting — I only wish I could have attended them all. The chief architect of the Lua language (and its principal reference, the book *Programming in Lua*), Roberto Ierusalimschy, provided a great introduction to the language and its design and goals. Taco Hoekwater teamed up with him to give an overview and examples of how LuaTEX can work. For me this was a little like looking into the future with two knowledgeable guides pointing out the interesting features.

For me, there were several special highlights of the conference itself. In my daily work I am almost alone in my LATEX endeavors and often find myself in correspondence (asking for help!) with people around the globe. The contingent of participants from Europe was a delight as I got to meet people face-to-face who have been of great help to me. Their enthusiasm for LuaTEX was quite contagious and I am looking forward to experimenting with their results so far.

Another delight was Peter Wilson, the author of the memoir class. I have based all my publishing work on this class and it was quite a treat to meet the man behind the code. His keynote address covered the history of writing, from cuneiform to medieval script to documents of today. He brought several writing artifacts that he encouraged us to inspect while he talked. Afterwards I thanked him for his trust in us to respect the items from his collection. He responded confidently, "I know my audience."

Perhaps the best thing about these yearly conferences, aside from the knowledge shared from the podiums, is the indescribable feeling of community when listening to others describe their struggles and achievements, their concerns and excitement about the future. We often toil alone during the year, perhaps fending off arguments about why TEX is on the wane, that Word or some other format is going to make TEX obsolete, etc. Sharing time with others involved in the same work and challenges gives me a sense of "Well, I'm not crazy after all. This typesetting system really is amazing."

(A few photos from the conference follow, courtesy of Jennifer Claudio, Volker RW Schaa, and Hartmut Henkel. Many more are at `http://tug.org/tug2007/photos`. *Ed.*)

Roberto Ierusalimschy, Taco Hoekwater
(Mirko Janc in background)

Jonathan Kew, Jon Breitenbucher, Paulo Ney de Souza

Front: William Hammond, Leonard Rosenthol, Tim Arnold, Frances Felluca, Jenny Levine, Heidi Sestrich, Cheryl Ponchin, Larry Thomas.

Behind: Eitan Gurari, Paulo Ney de Souza, Mirko Janc, Paul Topping, Herb Schulz, Jon Breitenbucher, Steve Grathwohl, Richard Leigh, August Gering, Morten Høgholm, Kaveh Bazargan, Karl Berry, Ron Fehd, Barbara Beeton, Dave Bailey, Jim Hefferon, Steve Brenton.

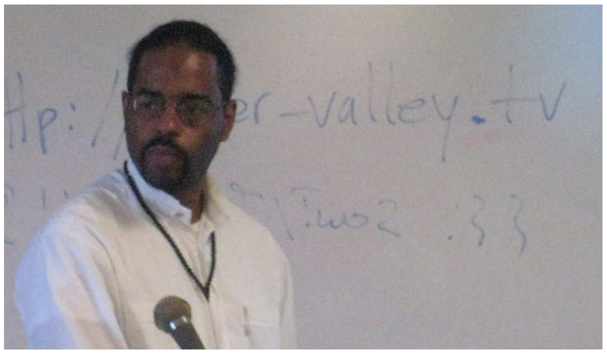
Dennis Claudio

Robin Laakso (Philip Brown in background)

Front: Martha Kummerer, Jennifer Claudio, Wendy McKay, Shannon Jones, Jonathan Kew, Hans Hagen, Roberto Ierusalimschy, Taco Hoekwater, Lowell Smith.

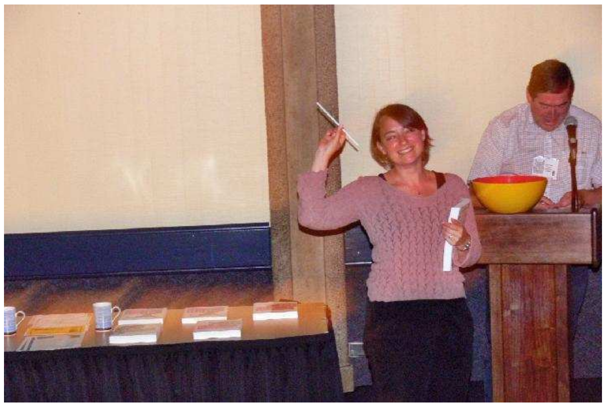
Behind: Robert Burgess, Don DeLand, Martin Schröder, Alan Wetmore, Walter Reddall, Nelson Beebe, Dick Koch, William Slough, Brian Papa, Andrew Mertz, Jon Stenerson, Stephen Moye, Peter Wilson, Peter Flynn, Chris Rowley, Barry MacKichan, Klaus Höppner, David Allen, Hartmut Henkel, Philip Brown.

Oren Patashnik, Nelson Beebe



Idris Hamid

Barbara Beeton


Jenny Levine, Peter Flynn

Volker RW Schaa, Hartmut Henkel, Martin Schröder, Klaus Höppner, Steve Grathwohl

Wendy McKay, Stephen Moye, Dick Koch, Jon Breitenbucher, Robin Laakso

Taco Hoekwater, Morten Høgholm

Ron Fehd, Caleb Ashley

The End
The Beatles
July/August 1969
[1969 = year of first edition of Donald Knuth's Seminumerical Algorithms (TAOCP volume 2)]
Nelson Beebe

Between then and now — A meandering memoir

Peter Wilson
Herries Press
18912 8th Ave. SW
Normandy Park, WA 98166
USA
herries dot press (at) earthlink dot net

Abstract

I was asked to talk about something interesting — perhaps how I came to develop the memoir class. Following this suggestion the first part is about how I became involved with LaTeX and friends and why the memoir class. To me all this is not particularly interesting as it falls into the personal 'been there, done that' category. What I find more interesting is how the written word has been presented. The second part briefly describes this, starting four millenia ago with Cuneiform and, with a few stops along the way, ending at recent times.

memoir, *n.* a fiction designed to flatter the subject and impress the reader.

With apologies to Ambrose Bierce

We are the inheritors of an ancient tradition, one that goes back for more than four thousand years. It has taken me a long time to start to appreciate it, and had it not been for LaTeX I never would have realised that it was there.

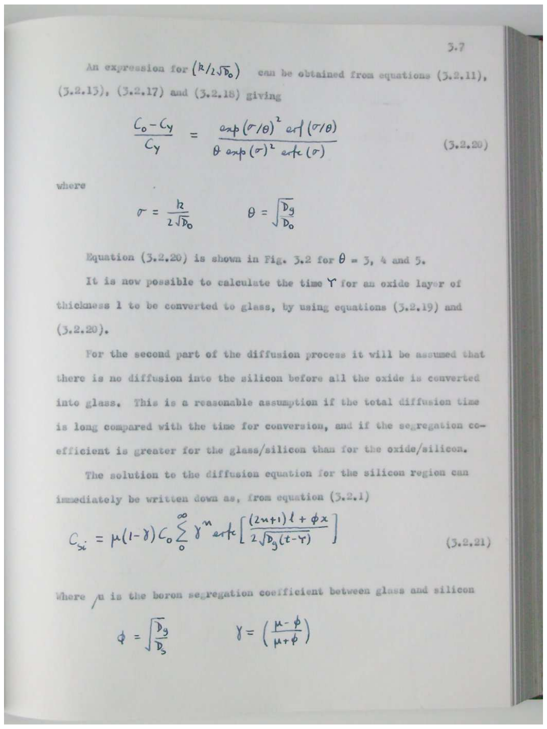
1 Neophyte

In 1973 I had to submit six bound copies of my thesis — one for my supervisor, another for the external examiner, the third for the University library, a fourth for myself, and two spare in case something untoward happened.[1] A very kind secretary typed it for me, one original and five carbon copies. I had to insert all the mathematics by hand (see Figure 1, original size 7½ by 10 inches), and in the last carbon copy that was about all that was legible.

Round about 1980 I came across a computer program called RUNOFF that would do a reasonable job of printing technical reports, provided you didn't mind adding in any mathematics by hand and you could overlook the fact that all we had was a dot matrix printer with too few dots.

Relief came in 1985 when I was introduced to LaTeX; no more hand insertions, justified text, different fonts, a professional look, and no looking back.

I used it for all my internal company reports and paper submission to journals — this was before we could ship documents around electronically so in



3.7

An expression for $(k/2\sqrt{D_o})$ can be obtained from equations (3.2.11), (3.2.13), (3.2.17) and (3.2.18) giving

$$\frac{C_o - C_y}{C_y} = \frac{\exp(\sigma/\theta)^2 \operatorname{erf}(\sigma/\theta)}{\theta \exp(\sigma)^2 \operatorname{erfc}(\sigma)} \qquad (3.2.20)$$

where

$$\sigma = \frac{k}{2\sqrt{D_o}} \qquad \theta = \sqrt{\frac{D_g}{D_o}}$$

Equation (3.2.20) is shown in Fig. 3.2 for θ = 3, 4 and 5.

It is now possible to calculate the time Υ for an oxide layer of thickness 1 to be converted to glass, by using equations (3.2.19) and (3.2.20).

For the second part of the diffusion process it will be assumed that there is no diffusion into the silicon before all the oxide is converted into glass. This is a reasonable assumption if the total diffusion time is long compared with the time for conversion, and if the segregation coefficient is greater for the glass/silicon than for the oxide/silicon.

The solution to the diffusion equation for the silicon region can immediately be written down as, from equation (3.2.1)

$$C_{si} = \mu(1-\gamma) C_o \sum_0^\infty \gamma^n \operatorname{erfc}\left[\frac{(2n+1)l + \phi x}{2\sqrt{D_g(t-\Upsilon)}}\right] \qquad (3.2.21)$$

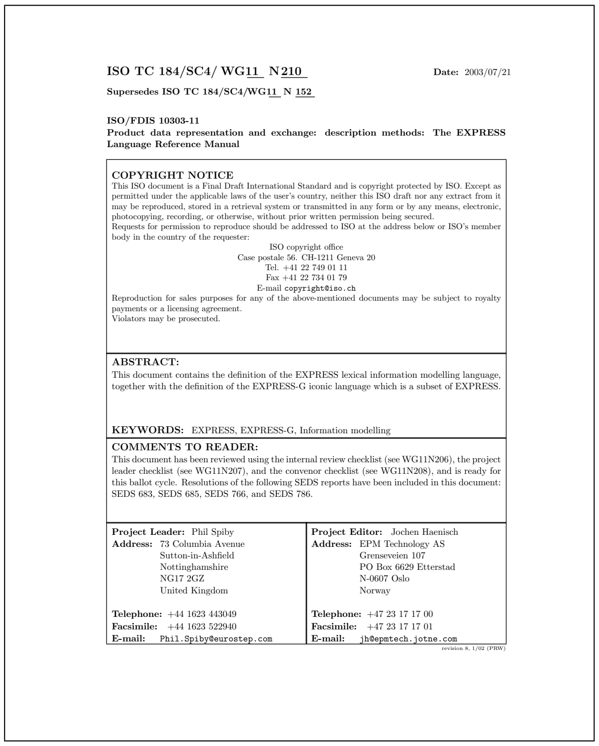
Where μ is the boron segregation coefficient between glass and silicon

$$\phi = \sqrt{\frac{D_g}{D_s}} \qquad \gamma = \left(\frac{\mu-\phi}{\mu+\phi}\right)$$

Herries Collection

Figure 1: Page from PhD thesis (1973)

[1] It did. The binder bound one copy with some pages upside down and others back to front!



ISO TC 184/SC4/ WG11 N210 **Date:** 2003/07/21

Supersedes ISO TC 184/SC4/WG11 N 152

ISO/FDIS 10303-11
Product data representation and exchange: description methods: The EXPRESS Language Reference Manual

COPYRIGHT NOTICE
This ISO document is a Final Draft International Standard and is copyright protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording, or otherwise, without prior written permission being secured.
Requests for permission to reproduce should be addressed to ISO at the address below or ISO's member body in the country of the requester:

ISO copyright office
Case postale 56. CH-1211 Geneva 20
Tel. +41 22 749 01 11
Fax +41 22 734 01 79
E-mail `copyright@iso.ch`

Reproduction for sales purposes for any of the above-mentioned documents may be subject to royalty payments or a licensing agreement.
Violators may be prosecuted.

ABSTRACT:
This document contains the definition of the EXPRESS lexical information modelling language, together with the definition of the EXPRESS-G iconic language which is a subset of EXPRESS.

KEYWORDS: EXPRESS, EXPRESS-G, Information modelling

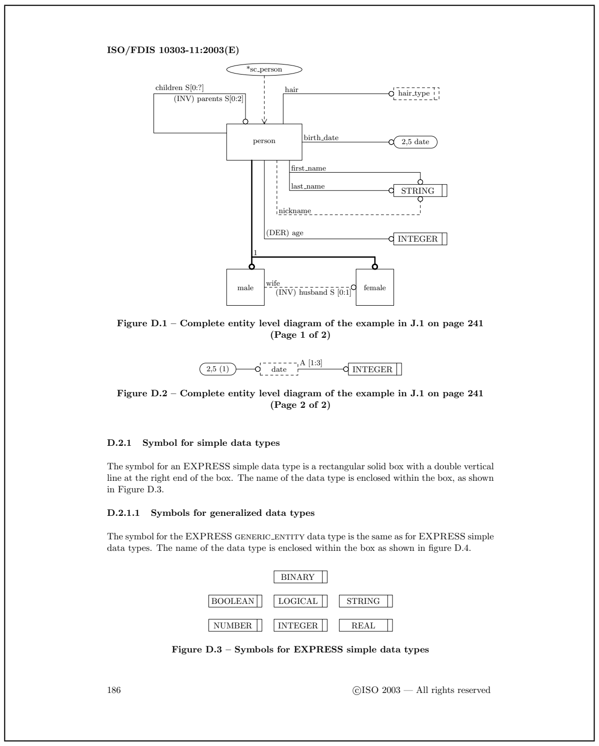
COMMENTS TO READER:
This document has been reviewed using the internal review checklist (see WG11N206), the project leader checklist (see WG11N207), and the convenor checklist (see WG11N208), and is ready for this ballot cycle. Resolutions of the following SEDS reports have been included in this document: SEDS 683, SEDS 685, SEDS 766, and SEDS 786.

Project Leader: Phil Spiby	**Project Editor:** Jochen Haenisch
Address: 73 Columbia Avenue Sutton-in-Ashfield Nottinghamshire NG17 2GZ United Kingdom	**Address:** EPM Technology AS Grenseveien 107 PO Box 6629 Etterstad N-0607 Oslo Norway
Telephone: +44 1623 443049	**Telephone:** +47 23 17 17 00
Facsimile: +44 1623 522940	**Facsimile:** +47 23 17 17 01
E-mail: `Phil.Spiby@eurostep.com`	**E-mail:** `jh@epmtech.jotne.com`

revision 8, 1/02 (PRW)

Herries Collection

Figure 2: Cover sheet for *ISO/FDIS 10303-11:2003*



ISO/FDIS 10303-11:2003(E)

*sc_person
children S[0:?]
(INV) parents S[0:2]
hair
hair_type
person
birth_date
2,5 date
first_name
last_name
STRING
nickname
(DER) age
INTEGER
1
male
wife
(INV) husband S [0:1]
female

Figure D.1 – Complete entity level diagram of the example in J.1 on page 241 (Page 1 of 2)

2,5 (1)
date
A [1:3]
INTEGER

Figure D.2 – Complete entity level diagram of the example in J.1 on page 241 (Page 2 of 2)

D.2.1 Symbol for simple data types

The symbol for an EXPRESS simple data type is a rectangular solid box with a double vertical line at the right end of the box. The name of the data type is enclosed within the box, as shown in Figure D.3.

D.2.1.1 Symbols for generalized data types

The symbol for the EXPRESS GENERIC_ENTITY data type is the same as for EXPRESS simple data types. The name of the data type is enclosed within the box as shown in figure D.4.

Figure D.3 – Symbols for EXPRESS simple data types

186 ©ISO 2003 — All rights reserved

Herries Collection

Figure 3: Page 186 from *ISO/FDIS 10303-11:2003*

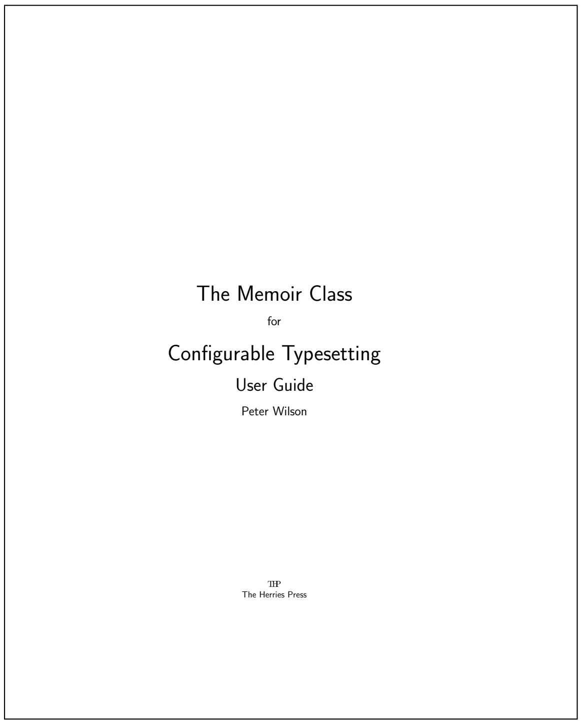
some sense it didn't matter what you used to create them as they would either be copied or retyped.

I became involved in the development of the International Standard 10303 *Industrial automation systems and integration — Product data representation and exchange*, commonly known as STEP, both as the editor and as a technical contributor. ISO had strict rules about the layout of the typewritten documents we would be submitting, which they would then retype for their publishing system, merrily adding typos as they went along. We managed to persuade them to take camera-ready copy so they could eliminate the typo introducing stage. We used LaTeX, of course, as it produced high quality output and, further, it was non-proprietary and we were working in a non-proprietary area.

The draft standard grew to about 2000 pages before we were allowed to split it up into parts to be published separately. Some part editors, for whatever reason, started to use wordprocessors instead of LaTeX. In the meantime I had developed a class for ISO standards in general (Wilson, 2002a), and ISO 10303 in particular (Wilson, 2002b).

Figure 2 shows the cover sheet for the part of the standard defining the EXPRESS and EXPRESS-G information modeling languages. The cover sheet was implemented using the `picture` environment and all that an author had to do was use a few macros for the text — rather like for the `\maketitle` command. Also as part of my work on STEP I developed the MetaPost `expressg` package (Wilson, 2004a) for drawing BLA (box, line, annotation) diagrams like the ones in Figure 3.

I eventually moved to the National Institute of Standards and Technology (NIST) in Maryland where the secretariat for STEP was based (Kemmerer, 1999). Someone up the management chain decided that the whole thing should be maintained as SGML documents (or portions thereof) in a database. As they were one of the major supporters of using wordprocessors I was surprised that they chose LaTeX as the publishing system and I spent a considerable time writing a LaTeX to SGML translator, and vice-versa. Unfortunately ISO kept changing their formatting requirements, LaTeX authors kept introducing their own macros, the SGML team kept changing their DTD, and the wordprocessor users were going to be involved at some indefinite date in the future. The experience made me really appreciative of Eitan Gurari's `TeX4ht` (Gurari, 2007). I left before any document made it through the system, which I think has died the death it deserved.



The Memoir Class

for

Configurable Typesetting

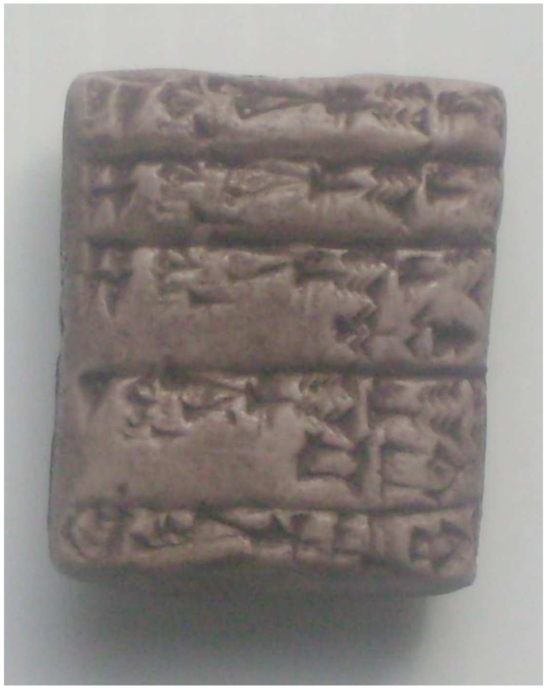
User Guide

Peter Wilson

TIP
The Herries Press

Herries Collection

Figure 4: Title page of the memoir class user manual



Herries Collection

Figure 5: Sumerian cuneiform tablet (circa 2112–2004 BC)

Some of the documents had got up to 1200 pages which caused enormous difficulties to the poor souls who had to use 'the' wordprocessor.

This led me on to the development of my LaTeX memoir class (Wilson, 2004b). I didn't want to be bitten by the ISO experience again, so I felt that a class that would let me change the document formatting easily without having to delve into its innards would be very useful. I had written a few packages that helped in formatting bits and pieces and decided to incorporate them into the class. Then there were other packages that I quite often used and integrating those, or their functionality, seemed reasonable, thus ensuring that they would all work well together. Then, like Topsy, it 'just growed'. Now it encompasses the functionality of more than 30 popular packages.

Putting everything together got me started on wondering how a document should be put together. This led to a long trail. One portion was trying to get a better idea about the typographer's craft. And as typographers deal with letter forms that led me to the history of the alphabet and the story of the letter forms that we use now.

2 Early writing

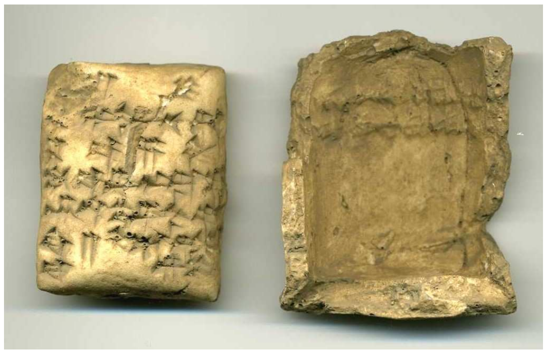
Writing was invented in ancient Mesopotamia, an area which roughly corresponds to modern day Iraq. The earliest recorded writings are by the Sumerians from around 3300 BC, who used pointed sticks or reeds to impress marks into wet clay tablets that were subsequently dried. The result is what we call Cuneiform.[2] We are still in the business of recording writing.

As the city states arose and society became more complex writing was necessary to help the bureaucrats and merchants keep track of things and so that tax collectors and others could go about their business in a fair manner.

Figure 5 shows a replica of a Sumerian cuneiform tablet dating back to between 2112 and 2004 BC, from the Third Dynasty of Ur, about the time of the Biblical Abraham. The original is 1¼ by 1¼ by ⅜ inches. The scribes would write on the front and the back of a tablet, and sometimes on the sides as well.

Cuneiform writing was adopted by the Babylonians even though their language was not like Sumerian, and Figure 6 shows a replica of a Babylonian

[2] From the Latin *cuneus* meaning wedge.



Herries Collection

Figure 6: Babylonian cuneiform tablet and envelope (circa 1790 BC)

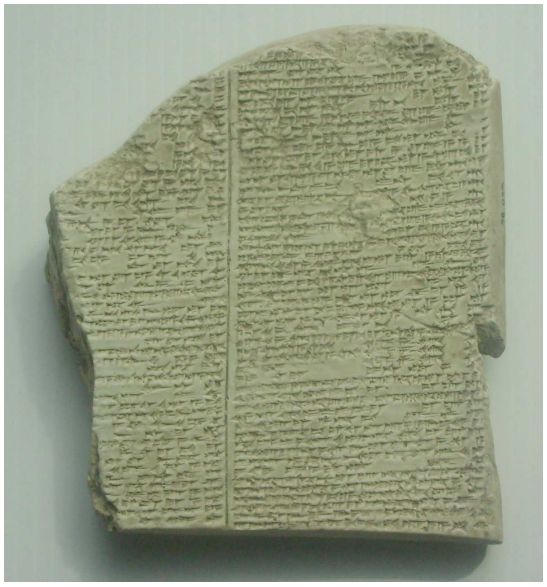
tablet and its clay envelope, from about 1790 BC. The tablet is 1 1/2 by 1 3/4 by 1/2 inches.

The package is a receipt for an amount of grain sufficient for one man for 6 months. The same text is on the outside of the clay envelope as on the tablet; if there was doubt about the external message then the envelope could be broken and the external and internal messages compared.

Writing evolved from that needed for simple record keeping to be able, for instance, to write people's names or to record the majestic deeds of the ruler. The earliest literary tablets containing parts of the *Epic of Gilgamesh*, which is by far the world's oldest epic, date back to about 2100 BC. The Gilgamesh story has been pieced together from thousands of pieces of broken cuneiform tablets (George, 2000). Figure 7 shows a replica of one of the many tablets found by Sir Austen Henry Layard in 1850–53 in the ruins of King Ashurbanipal's library at Ninevah which was destroyed in 612 BC. This particular one contains much of what is called 'Tablet 11' of the Epic which includes the best preserved story of a Deluge[3] or Flood, well pre-dating the Biblical version which was written around the 9th century BC.

The tablet, which is 5 3/4 by 5 3/4 by 1 1/4 inches, was first translated in 1872 by George Smith working at the British Museum. Wallis Budge (Budge, 1925) described the event like this:

> Smith took the tablet and began to read over the lines which Ready [the conservator who had cleaned the tablet] had brought to light; and when he saw that they contained the portion of the legend he had hoped to find there, he said, "I am the first man to read that after two thousand years of oblivion." Setting the tablet on the table, he jumped up and rushed about the room in great excitement, and, to the astonishment of those present, began to undress himself!



Herries Collection

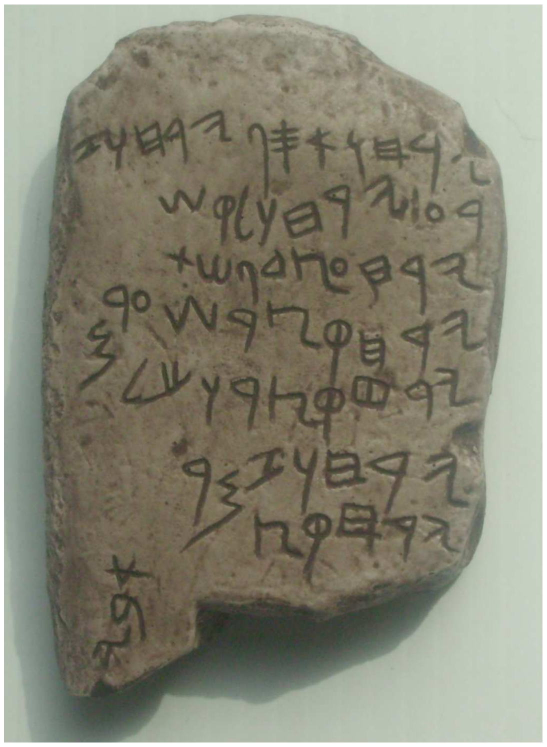
Figure 7: Epic of Gilgamesh, part of tablet 11 (circa 650 BC)

Figure 8 shows a replica of a soft piece of limestone rock from around 925 BC. This was found in 1908 by R.A.S. Macalister at Tell el-Jazari (the historic city of Gezer) about 20 miles NW of Jerusalem. The tablet is 3 by 4 1/2 by 5/8 inches. The text is written right to left in what some say is in a Proto-Hebrew script while others (Healey, 1990, p. 30) say it is in the Phoenician[4] script. It is a calendar of agricultural tasks and seasons. The tablet's inscription is:

𐤉𐤓𐤇𐤅𐤀𐤎𐤐𐤉𐤓𐤇𐤅𐤆
𐤓𐤏𐤉𐤓𐤇𐤅𐤋𐤒𐤔
𐤉𐤓𐤇𐤏𐤑𐤃𐤐𐤔𐤕
𐤉𐤓𐤇𐤒𐤑𐤓𐤔𐤏𐤓𐤌
𐤉𐤓𐤇𐤒𐤑𐤓𐤅𐤊𐤋
𐤉𐤓𐤇𐤅𐤆𐤌𐤓
𐤉𐤓𐤇𐤒𐤑
𐤀𐤁𐤉

[3] There is evidence that the catastrophe occurred around 7500 BC when the Black Sea's water level rose by 400 feet during the course of about a year (Ryan and Pitman, 2000).

[4] To me it looks remarkably like Phoenician.



Herries Collection

Figure 8: Gezer Calendar (circa 925 BC)

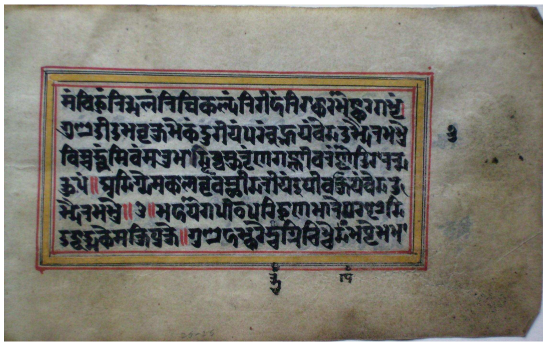
In the following transliteration I have added inter-word spaces that are not in the original. The first two lines on the tablet contain the first three lines of the calendrical information.

z wḥry ps' wḥry
šql wḥry 'r
tšp 'ṣ' ḥry
mr'š rṣq ḥry
lkw rṣq ḥry
rmz wḥry
ṣq ḥry
yb'

And a translation is:

Two months are [olive] harvest,
Two months are planting [grain],
Two months are late planting;
One month is hoeing up flax,
One month is harvest of barley,
One month is harvest and feasting;
Two months are vine tending,
One month is summer fruit.

It is signed in the bottom lefthand corner with the name 'Abijah'.



Herries Collection

Figure 9: Leaf from a copy of the *Bhagavad Gita*, Kashmir (circa 1800)

3 Manuscripts

Our modern alphabets date back to around 1600 BC, and in particular to the Phoenician script and alphabet. By various routes this spread out from the Middle East, changing as time went on to accommodate different languages (Wilson, 2005).

Throughout the ages scribes have always taken great care in the appearance of their work, especially with religious works.

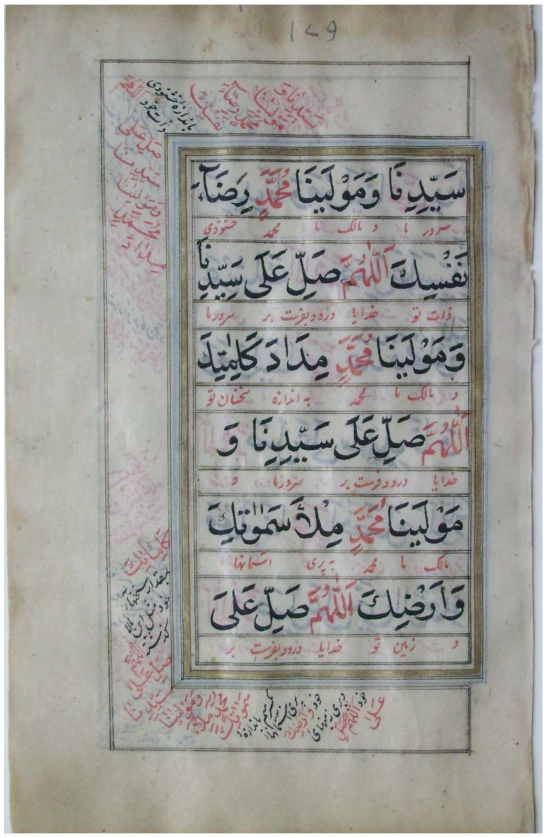
Figure 9 is a leaf from a Kashmiri copy of the *Bhagavad Gita*. The original is 5 1/2 by 3 1/4 inches overall in a black Devanagari script surrounded by a yellow, red and blue border, on burnished paper. It dates to the late 18th or early 19th century. The *Bhagavad Gita* (The Song of the Divine One) is a poem consisting of a dialogue between the warrior prince Arjuna and Lord Krishna (in the person of his charioteer), on the eve of the climactic battle at Kurukshetra. It forms part of the Hindu epic, the *Mahabharata* which dates back to the first millenium BC, while the *Gita* was written later, probably between the fifth and second centuries BC.

N. P. Davis (Davis, 1969) quotes J. Robert Oppenheimer after observing the first test of the atomic bomb on July 16, 1945, as saying:

> There floated through my mind a line from the *Bhagavad Gita* in which Krishna is trying to persuade the Prince that he should do his duty: 'I am become death: the shatterer of worlds'. I think we all had this feeling more or less.

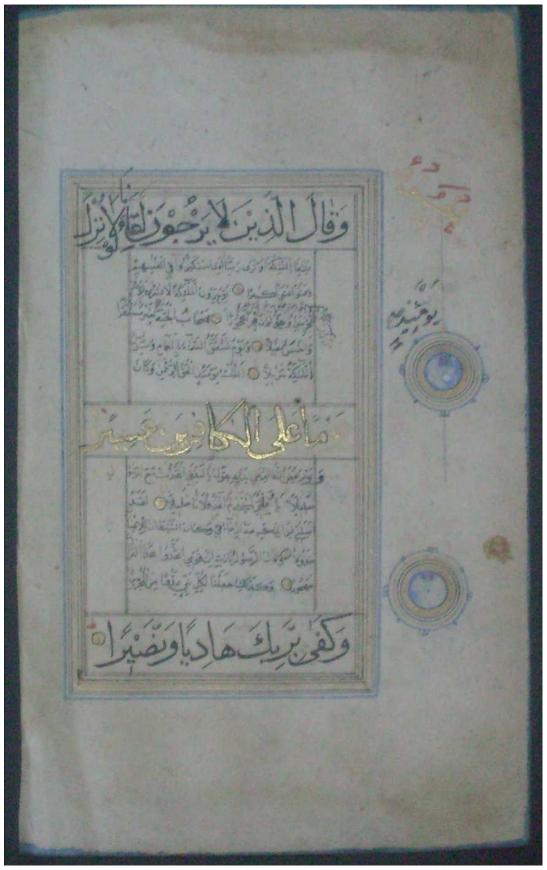
The preceding lines in the *Bhagavad Gita* are:

> If the radiance of a thousand suns
> Were to burst into the sky,
> that would be like
> the splendour of the Mighty One.



Herries Collection

Figure 10: Leaf from a copy of *Delail al-Khayrat* Arabic/Persian (circa 1690)



Herries Collection

Figure 11: Leaf from a *Koran*, India (16th century)

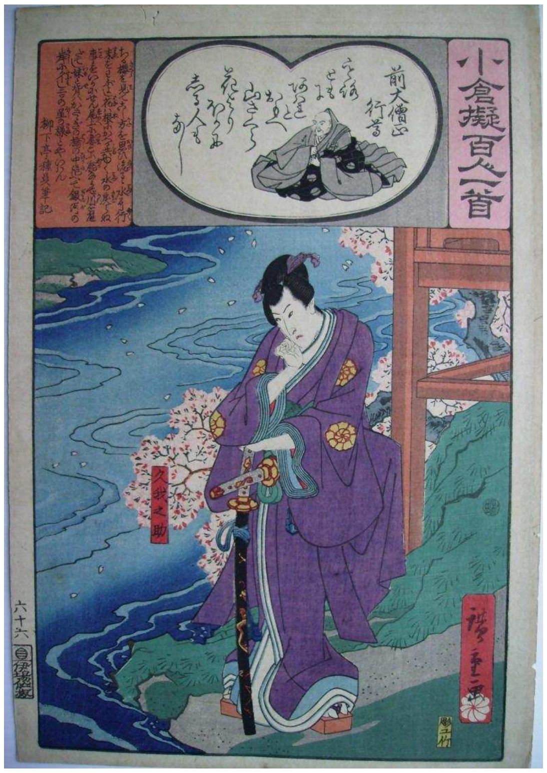
Figure 10 shows a leaf from a copy of *Delail al-Khayrat* — the book of *Blessings on the Prophet* composed by Muhammad ibn Sulayman al-Jazuli (d. 1465). The original of the leaf is $4^{11}/_{16}$ by $7^{5}/_{8}$ inches. It was written about 1690 by Mohammed Azeem for Nawab Sadullah Khan who was the Prime Minister of the Moghul emperor Shah Jehan — the builder of the Taj Mahal. The Arabic text is black with an interlinear Persian translation in red and a commentary in the margins around the main text. The border is in gold and a light blue.

Figure 11 is a leaf from a 16th century Indian copy of the *Koran*. The original is $3^{1}/_{2}$ by 6 inches overall. The Arabic script is in black ink, except for the central line which is in liquid gold, surrounded by a main border, $2^{1}/_{4}$ by $3^{3}/_{4}$ inches, in gold and blue. The marginal discs are also in gold and blue.

Arabic texts are famous for their calligraphy but there are other cultures as well where calligraphy is an esteemed art. Figure 12 is number 66 from the series of Japanese woodblock prints *Ogura Imitation of 100 Poets* illustrating a famous anthology of 100 poems by 100 poets that was assembled by the poet Fujiwara no Teiko in 1235. The woodblock print publisher Iba-ya Sensburō commissioned three artists — Kuniyoshi, Hiroshige and Kunisada — to produce the prints in the series which were published between 1845 and 1847. This one by Hiroshige illustrates a poem by Daisōjō Gyōson (1055–1136). The poem reads:

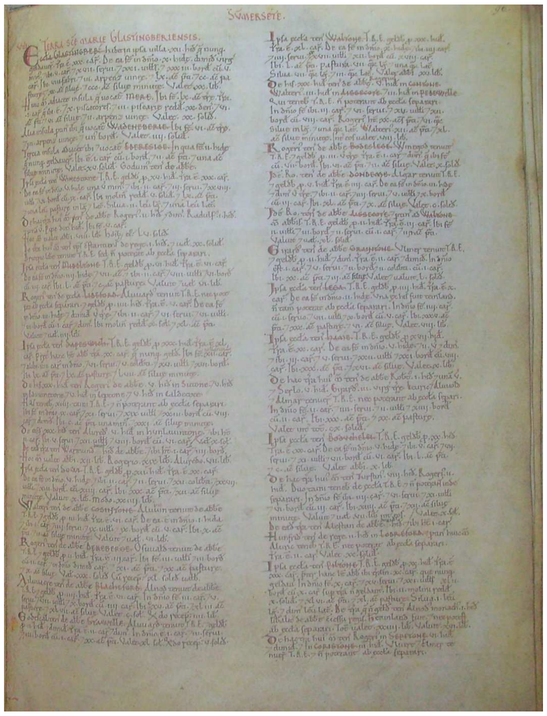
Morotomi ni	Let us, each for each
Aware to omoe	Pitying, hold tender thought,
Yamazakura	Mountain cherry flower!
Hana yori hoka ni	Other than thee, lonely flower,
Shiru hito mo nashi	There is none I know as friend.

The main illustration shows a contemplative Kuganosoke (the hero of the play *Imoseyama*) outside a pavilion on the bank of a river. The title of the series is at the top right in large kanji characters and at the top left is a description of the main illustration in smaller kanji. The lozenge contains a portrait of the poet and the poem itself in a highly calligraphic style. The original is in the standard



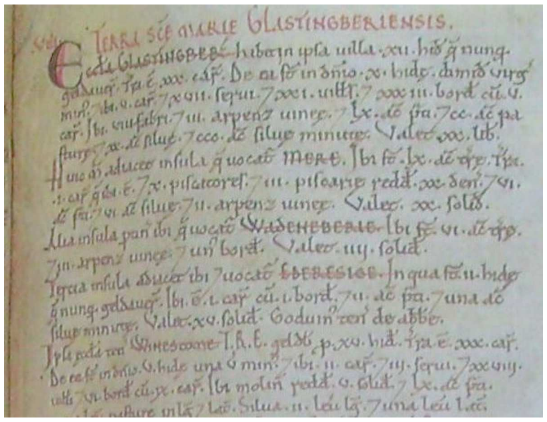
Herries Collection

Figure 12: *Ogura Imitation of 100 Poets* no. 66, by Hiroshige (circa 1846)



Herries Collection

Figure 13: Page from the Domesday Book, England (1086)



TERRA SCE MARIE GLASTINGBERIENSIS.

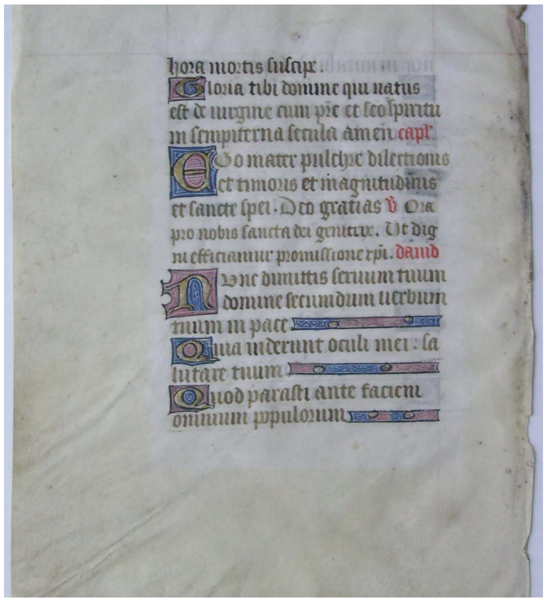
Eccla GLASTINGBERI habet in ipsa uilla .xii. hid' q̃ nunq̃
geldauer'. Tra ē .xxx. car'. De ea st in dnio .x. hidę. dimid uirg'
min'. Ibi .v. car'. 7 xvii. serui. 7 xxi. uitt'. 7 xxxiii. bord' cū .v.
car'. Ibi .viii. fabri. 7 iii. arpenz uineę. 7 lx. ac' pra. 7 cc. ac' pa
sturę. 7 xx. ac' siluę. 7 ccc. ac' siluę minutę. Valet .xxx. lib.
Huic m̃ adiacet insula q̃ uocat MERE. Ibi st .lx. ac' tre. Tra
.i. car' q̃ ibi ē. 7 x. piscatores. 7 iii. piscarię redd' .xx. den'. 7 vi.
ac' pa. 7 vi. ac' siluę. 7 ii. arpenz uineę. Valet .xx. solid.
Alia insula pat' ibi q̃ uocat WADENEBERIE. Ibi st .vi. ac' tre.
7 iii. arpenz uineę. 7 un' bord'. Valet .iiii. solid.
Tercia insula adiacet ibi 7 uocat ESTERESIGE. In qua st .ii. hidę
q̃ nunq̃ geldauer'. Ibi ē .i. car' cū .i. bord'. 7 ii. ac' pra. 7 una ac'
siluę minutę. Valet .xv. solid. Goduin' ten' de abbe.
Ipsa eccla ten' WINESCOME T.R.E. geldb p .xv. hid'. Tra ē .xxx. car'.
De ea st in dnio .v. hidę una v min'. 7 ibi .ii. car'. 7 iiii. serui. 7 xxviii.
uitt' 7 vi. bord' cū .x. car'. Ibi molin' redd' .v. solid. 7 lx. ac' pra.
[illegible] pasture in lg' 7 lat'. Silua .ii. leū lg'. 7 una leū l' [illegible]

Herries Collection

Figure 14: Domesday Book (enlarged), England (1086)

oban size of approximately 9½ by 14 inches. To print it there would have been one carved woodblock for each colour in the picture, with the picture being gradually built up one colour at a time. Registration between the individual blocks and with the paper is critical. Even seemingly simple pictures could require ten or more blocks.

Coming closer to home, European books were mainly written in Latin. Literacy was essentially confined to the Church, the Papal See and monasteries in particular, and to clerks in noble courts. Most works that have survived were religious in nature but rulers required administrative records of all kinds. One of the most famous is the *Domesday Book* that William the Conqueror (circa 1028–1087) ordered to be compiled in 1086. It is a survey of the newly conquered England, from Yorkshire to the South Coast, arranged by county, and listing all the landowners and the worth and taxes paid on their properties (Hinde, 1985). Figure 13 shows one page from the book that starts with information about Glastonbury in the County of Somerset. The text is in Latin, in two columns of 44 lines each, written in a Carolingian minuscule script. An enlarged view



Herries Collection

Figure 15: Book of Hours, France, (circa 1445)

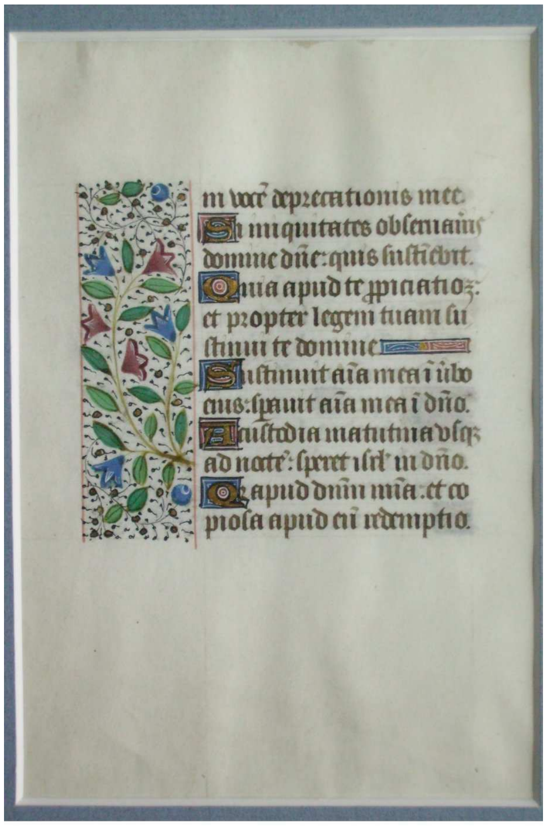
of the top of the left column is shown in Figure 14. Some headings are in red, but the text is not without errors.

Many beautiful manuscripts were written by scribes in monasteries, some for use by the Church and others for rich patrons. Many of the latter are elaborately decorated and illuminated.

Figure 15 is a leaf (verso) from a Benedictine Book of Hours produced in France around 1445. The original vellum leaf is 5 3/4 by 8 inches. The Latin text, 3 by 4 1/4 inches, is in the Gothic Textura Quadrata bookhand in a light brown ink. The versal initials are in liquid gold on grounds of red and blue with white tracery. The paragraph endings use the same style.

A more decorative example is shown in Figure 16 which is a leaf from a Book of Hours produced in France, perhaps at Rheims, around 1450 or maybe a little later. The original vellum leaf is 3 3/4 by 5 5/8 inches. The Latin text, 2 1/2 by 2 7/8 inches, is in the Gothic Textura Quadrata bookhand in a dark brown ink. The versals are in liquid gold with additional decoration in red and blue. The floriated decoration uses green as well as the other colours.

In a different vein, and a different script, Figure 17 is a page from Antonio Pigafetta's account of Magellan's circumnavigation (1519–1522), beautifully written in a humanist bookhand. There are four surviving manuscripts, one in the Venetian dialect of Italian, and three in French. Pigafetta probably completed his work in 1524 and it would then have been copied out by professional scribes. The manuscript now at the Bernicke Library at Yale University consists of 103 vellum leaves, measuring 7 1/2 by 11 1/4 inches, with 27 lines to a page (Pigafetta, 1969). The page in the illustration shows the end of chapter XVIII, a summary (in red) of the next chapter, and the title and first four lines of chapter XIX. The marginal notes, in red and blue, are a summary of the corresponding paragraphs in the main text.



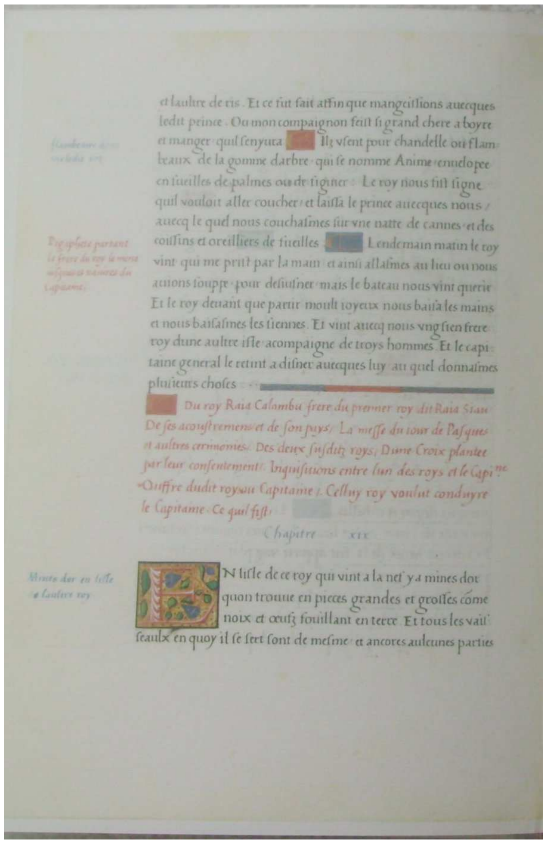
Herries Collection

Figure 16: Book of Hours, France, (circa 1450)

4 Printed books

In the West, printing using moveable type was invented by Johannes Gutenberg around 1440–1450, although the earliest printed book known is a 9th century Chinese woodblock printing of the *Diamond Sutra*. Gutenberg had to experiment to determine the formula for a suitable ink and also to discover a good metal alloy for the type itself. He came up with lead to which he added antimony for strength and hardness and tin for toughness.[5]

[5] This is still the basis for type today; Monotype casting machines use lead with 15–24% antimony and 6–12% tin.



[illegible] ledit roy

et laultre de ris. Et ce fut fait affin que mangeiſſions auecques
ledit prince. Ou mon compaignon feiſt ſi grand chere a boyre
et manger quil ſenyura. Ilz vſent pour chandelle ou flam
beaux de la gomme darbre qui ſe nomme Anime enuelopee
en fueilles de palmes ou de figuier. Le roy nous fiſt ſigne
quil vouloit aller coucher, et laiſſa le prince auecques nous,
auecq le quel nous couchaſmes ſur vne natte de cannes et des
couſſins et oreilliers de fueilles. Lendemain matin le roy
vint qui me priſt par la main et ainſi allaſmes au lieu ou nous
auions ſouppe pour deſiuſner mais le bateau nous vint querir
Et le roy deuant que partir moult ioyeux nous baiſa les mains
et nous baiſaſmes les ſiennes. Et vint auecq nous vng ſien frere
roy dune aultre iſle acompaigne de troys hommes. Et le capi
taine general le retint a diſner auecques luy au quel donnaſmes
pluſieurs choſes

Pigaphetta portant le frere du roy le mena [illegible] du Capitaine.

Du roy Raia Calambu frere du premier roy dit Raia Siau
De ſes acouſtremens et de ſon pays. La meſſe du iour de Paſques
et aultres cerimonies. Des deux ſuſditz roys. Dune Croix plantee
par leur conſentement. Inquiſitions entre lun des roys et le Capi^ne
Ouffre dudit roy au Capitaine. Celluy roy voulut conduyre
le Capitaine. Ce quil fiſt.

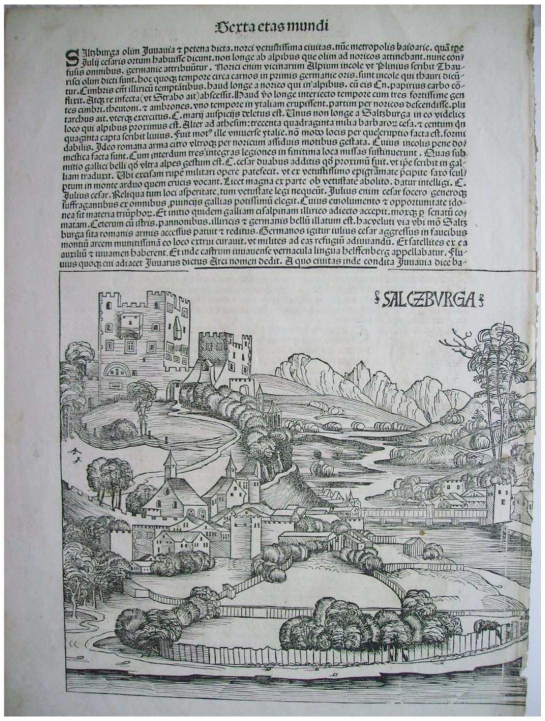
Chapitre xix

Mines dor en liſle de laultre roy

EN liſle de ce roy qui vint a la nef y a mines dor
quon trouue en pieces grandes et groſſes come
noix et oeufz fouillant en terre. Et tous les vaiſ
ſeaulx en quoy il ſe ſert ſont de meſme et ancores aulcunes parties

Herries Collection

Figure 17: Magellan's Voyage Around the World (1524)



Sexta etas mundi

Saltzburga olim Iuuauia ⁊ petena dicta. norici vetuſtiſſima ciuitas. nūc metropolis baioarie. quā tpe
Iulij ceſaris ortum habuiſſe dicunt. non longe ab alpibus que olim ad noricos attinebant. nunc con/
fuſis omnibus. germanie attribuūtur. Norici enim vicinarum Alpium incole vt Plinius ſcribit Thau-
riſci olim dicti ſunt. hoc quoqꝫ tempore circa carnos in primis germanie oris. ſunt incole qui thauri dicū-
tur. Cimbris em̄ illiricū temptātibus. haud longe a norico qui in alpibus. cū eis Cn. papirius carbo cō-
flixit. Atqꝫ re infecta (vt Strabo ait) abſceſſit. Haud vō longe interiecto tempore cum tres fortiſſime gen
tes cimbri. theutoni. ⁊ ambrones. vno tempore in ytaliam erupiſſent. partim per noricos deſcendiſſe. plu
tarchus ait. vterqꝫ exercitus. C. marij auſpicijs deletus eſt. Unus non longe a Saltzburga in eo videlicz
loco qui alpibus proximus eſt. Alter ad athesim: trecenta quadraginta milia barbaroꝝ ceſa. ⁊ centum qn
quaginta capta ſcribit liuius. Fuit mot⁹ ille vniuerſe ytalie. nō modo locis per que eruptio facta eſt. formi
dabilis. Ideo romana arma citro vltroqꝫ per noricum aſſiduis motibus geſtata. Cuius incolis pene do/
meſtica facta ſunt. Cum interdum tres integras legiones in finitima loca miſſas ſuſtinuerunt. Quas ſub
initio gallici belli qō vltra alpes geſtum eſt. C. ceſar duabus additis qō proximū fuit. vt ipē ſcribit in gal-
liam traduxit. Ubi exciſam rupē militari opere patefecit. vt ex vetuſtiſſimo epigrāmate pcipite ſaxo ſcul/
ptum in monte arduo quem crucis vocant. Licet magna ex parte ob vetuſtate abolito. datur intelligi. C.
Iulius ceſar. Reliqua tum loci aſperitate. tum vetuſtate legi nequeūt. Iulius enim ceſar ſocero generoqꝫ
ſuffragantibus ex omnibus puincijs gallias potiſſimū elegit. Cuius emolumento ⁊ opportunitate ido-
nea ſit materia triūphoꝝ. Et initio quidem galliam ciſalpinam illirico adiecto accepit. morqꝫ p ſenatū co/
matam. Ceterum cū iſtris. pannonibus. illiricis ⁊ germanis bellū illatum eſt. hac veluti via vbi mō Saltz
burga ſita romanis armis acceſſus patuit ⁊ reditus. Germanos igitur iulius ceſar aggreſſus in faucibus
montiū arcem munitiſſimā eo loco extrui curauit. vt milites ad eaꝫ refugiū adiuuandū. Et ſatellites ex ea
auxiliū ⁊ iuuamen haberent. Et inde caſtrum iuuauenſe vernacula lingua helffenberg appellabatur. Flu-
uius quoqꝫ cui adiacet Iuuarus dictus Arci nomen dedit. A quo ciuitas inde condita Iuuauia dice ba-

Herries Collection

Figure 18: *Nuremberg Chronicle*, Folio CLIIv (1493)

In order to be successful in the market, Gutenberg had to produce books that equaled those produced by the scribes, except that they did not necessarily have to be decorated so lavishly. The scribes, though, used many ligatures and other techniques to try and have non-ragged text blocks. To compete with them Gutenberg's font for his 42-line Bible, published around 1455, consisted of some 290 characters though all the text is in Latin which requires a basic character set of only forty letters — twenty lowercase letters and twenty caps — and some punctuation marks (Thorpe, 1999).

The 42-line Bible is set in two columns of 42 lines each. It is believed that about 135 copies were printed on paper and 40 on vellum. The page size was 12 by 16½ inches and it is estimated that more than five thousand calfskins were required for the vellum copies.

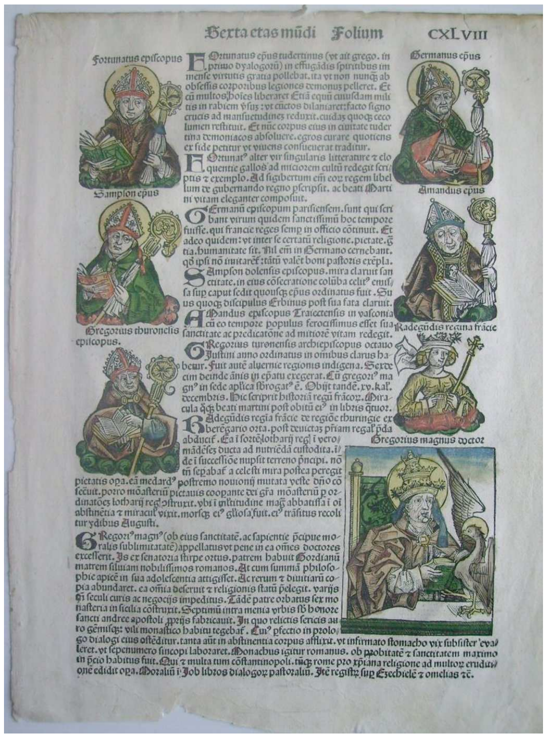
The *Nuremberg Chronicle* was published in 1493 in Nuremberg and was the first book to combine text with illustrations that illuminated the words (instead of using randomly selected woodblock engravings that happened to be at hand). As was usual then the book did not have a title page: Latin scholars call it the *Liber Chronicarum* and in German it is called *Die Schedelsche Weltchronik* after its author Hartmann Schedel. The book was printed and published by Anton Keberger with a print run of about 1500 Latin copies and 900 German ones. Around 400 Latin and 300 German copies have survived.

There are 1809 woodcut illustrations printed from 645 originals, so many were used multiple times, usually portraits. For example a single woodcut was used to represent Alcuin, Cato, Dante, Paris and Plutarch on different pages. The woodcuts were created by Michael Wolgemut and Hans Pleydenwurff, with perhaps one or two by Albrecht Dürer who was apprenticed to Wolgemut at the time.

The pages are large, 12 by 17½ inches. Views of cities were printed as a double spread. Spaces were left in the text for the woodcuts; in the more luxurious volumes the woodcuts were hand coloured.

The Chronicle divides the history of the world into seven ages:

1. Creation to the Deluge
2. ends with the birth of Abraham
3. ends with the reign of King David
4. ends with the Babylonian captivity



Herries Collection

Figure 19: *Nuremberg Chronicle*, Folio CXLVIIIr (1493)

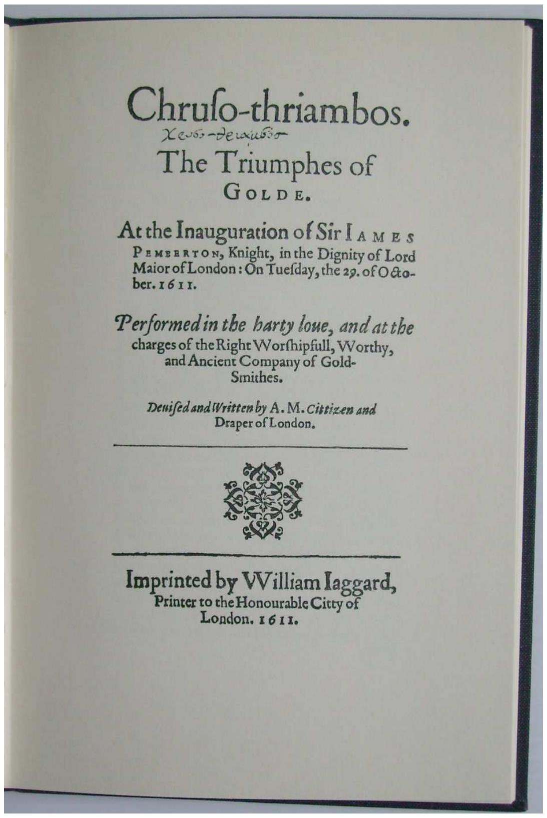
5. ends with the Incarnations of Jesus
6. from the birth of Christ to the end of the world
7. the age of the Anti-Christ
8. the Last Judgement

Beloit College has an extensive web site (`http://www.beloit.edu/~nurember`) devoted to their copy of the *Nuremberg Chronicle* which has coloured illustrations.

Figure 18 is Folio CLII (verso) from the *Nuremberg Chronicle*. At the bottom is half of a double spread picture of Salzburg (the other half is on the recto of Folio CLIII).

Figure 19 shows Folio CXLVIII (recto) from the *Nuremberg Chronicle*. The hand coloured pictures are of various ecclesiastical personages and at the lower right a queen (Radegudis regina fracie) and a doctor (Gregorius magnus doctor). The original for this picture is 12 by 15½ inches (over the years 2 inches have disappeared from the lower margin).

Books had, of course, been made and sold long before Gutenberg. In London, for example, the publishing trade was regulated by the Guild of Stationers which was incorporated in 1403. At that time stationers were either booksellers who sold manuscripts that they had copied; or illuminators who illustrated and decorated manuscripts; or bookbinders who bound manuscripts. Stationers would also sell the materials that they used. Unless you were a member of the Guild you could do none of these things.


Chruſo-thriambos.

Χρυσο-θριαμβος

The Triumphes of GOLDE.

At the Inauguration of Sir IAMES PEMBERTON, Knight, in the Dignity of Lord Maior of London: On Tueſday, the 29. of October. 1611.

Performed in the harty loue, and at the charges of the Right Worſhipfull, Worthy, and Ancient Company of Gold-Smithes.

Deuiſed and Written by A. M. *Cittizen and* Draper of London.

Imprinted by William Iaggard, Printer to the Honourable Citty of London. 1611.

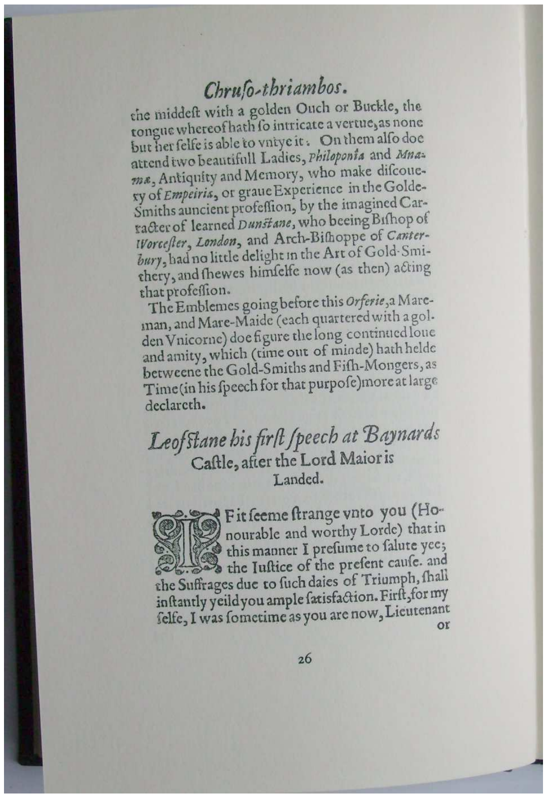
Herries Collection

Figure 20: *Chruso-thriambos:* Title page (1611)

Following Gutenberg, printing rapidly spread out over much of Europe. In England, for example, Caxton set up his shop in 1476, Theoderic Rood was printing in Oxford between 1478 and 1485, and John Sieberch in Cambridge in 1520. The Stationers Guild received a royal charter in 1557 and was responsible for regulating the printing industry over all the country, which meant that they had a monopoly on book production — once a member asserted ownership of a text (or ‘copy’) no other member could publish it. This is the origin of the term ‘copyright’.

In Germany books were usually printed in a gothic type but the rest of Europe moved to types based on the humanist tradition that had been maintained in Italy.

Figure 20 is the title page of a reprint of *Chruso-thriambos* or *The Triumphs of Golde* by Anthony Mundy, published in 1611. The original is 6 by 9



Chruſo-thriambos.

the middeſt with a golden Ouch or Buckle, the tongue whereof hath ſo intricate a vertue, as none but her ſelfe is able to vntye it. On them alſo doe attend two beautifull Ladies, *Philoponia* and *Mnamæ*, Antiquity and Memory, who make diſcouery of *Empeiria*, or graue Experience in the Golde-Smiths auncient profeſſion, by the imagined Caracter of learned *Dunstane*, who beeing Biſhop of *Worceſter*, *London*, and Arch-Biſhoppe of *Canterbury*, had no little delight in the Art of Gold-Smithery, and ſhewes himſelfe now (as then) acting that profeſſion.

The Emblemes going before this *Orferie*, a Mareman, and Mare-Maide (each quartered with a golden Vnicorne) doe figure the long continued loue and amity, which (time out of minde) hath helde betweene the Gold-Smiths and Fiſh-Mongers, as Time (in his ſpeech for that purpoſe) more at large declareth.

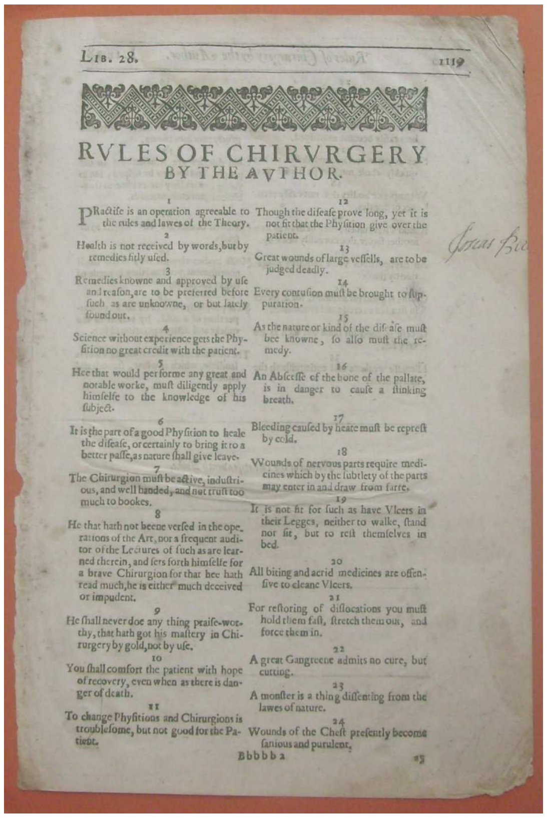
Leofstane his firſt ſpeech at Baynards Caſtle, after the Lord Maior is Landed.

IF it ſeeme ſtrange vnto you (Honourable and worthy Lorde) that in this manner I preſume to ſalute yee; the Iuſtice of the preſent cauſe. and the Suffrages due to ſuch daies of Triumph, ſhall inſtantly yeild you ample ſatisfaction. Firſt, for my ſelfe, I was ſometime as you are now, Lieutenant or

26

Herries Collection

Figure 21: *Chruso-thriambos:* page 8 (1611)



LIB. 28. 1119

RVLES OF CHIRVRGERY
BY THE AVTHOR.

1
PRactiſe is an operation agreeable to the rules and lawes of the Theory.

2
Health is not received by words, but by remedies fitly uſed.

3
Remedies knowne and approved by uſe and reaſon, are to be preferred before ſuch as are unknowne, or but lately found out.

4
Science without experience gets the Phyſition no great credit with the patient.

5
Hee that would performe any great and notable worke, muſt diligently apply himſelfe to the knowledge of his ſubject.

6
It is the part of a good Phyſition to heale the diſeaſe, or certainly to bring it to a better paſſe, as nature ſhall give leave.

7
The Chirurgion muſt be active, induſtrious, and well handed, and not truſt too much to bookes.

8
He that hath not beene verſed in the operations of the Art, nor a frequent auditor of the Lectures of ſuch as are learned therein, and ſets forth himſelfe for a brave Chirurgion for that hee hath read much, he is either much deceived or impudent.

9
He ſhall never doe any thing praiſe-worthy, that hath got his maſtery in Chirurgery by gold, not by uſe.

10
You ſhall comfort the patient with hope of recovery, even when as there is danger of death.

11
To change Phyſitions and Chirurgions is troubleſome, but not good for the Patient.

12
Though the diſeaſe prove long, yet it is not fit that the Phyſition give over the patient.

13
Great wounds of large veſſells, are to be judged deadly.

14
Every contuſion muſt be brought to ſuppuration.

15
As the nature or kind of the diſeaſe muſt bee knowne, ſo alſo muſt the remedy.

16
An Abſceſſe of the bone of the pallate, is in danger to cauſe a ſtinking breath.

17
Bleeding cauſed by heate muſt be repreſt by cold.

18
Wounds of nervous parts require medicines which by the ſubtlety of the parts may enter in and draw from farre.

19
It is not fit for ſuch as have Vlcers in their Legges, neither to walke, ſtand nor ſit, but to reſt themſelves in bed.

20
All biting and acrid medicines are offenſive to cleane Vlcers.

21
For reſtoring of diſlocations you muſt hold them faſt, ſtretch them out, and force them in.

22
A great Gangreene admits no cure, but cutting.

23
A monſter is a thing diſſenting from the lawes of nature.

24
Wounds of the Cheſt preſently become ſanious and purulent,

Bbbbb 2 25

Herries Collection

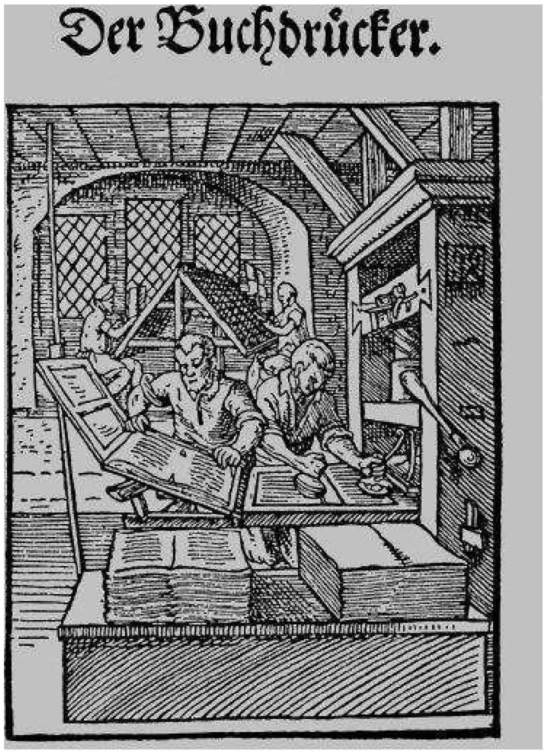
Figure 22: Page from the first English translation of Ambroise Paré's works (1634)

inches. The pageant *Chruso-thriambos* was written and produced at the request and charge of the Worshipful Company of Goldsmiths in honour of Sir James Pemberton, a goldsmith, the newly elected Lord Mayor of London. Page 8 (numbered 26 in the book containing the reprint) from the body is shown in Figure 21.

Ambroise Paré (1510–1590) served as the official royal surgeon for kings Henry II, Francis II, Charles IX and Henry III of France, and did much to advance medical procedures, particularly surgery. A page from the first English translation of his major work, by Thomas Johnson and printed in 1634 by Th. Cotes and R. Young, is shown in Figure 22. The original is 8 by 12½ inches and is set using an Oldstyle type, possibly Garamond. Paré's major contributions included the abandonment of boiling oil for the treatment of gunshot wounds in favour of egg yolk, oil of roses and turpentine which worked far better. He also introduced the use of ligatures instead of cauterisation during amputations, and was especially adept at devising ingenious and efficient artificial limbs and new surgical instruments. All in all he seems to have been afflicted with a great deal of common sense.

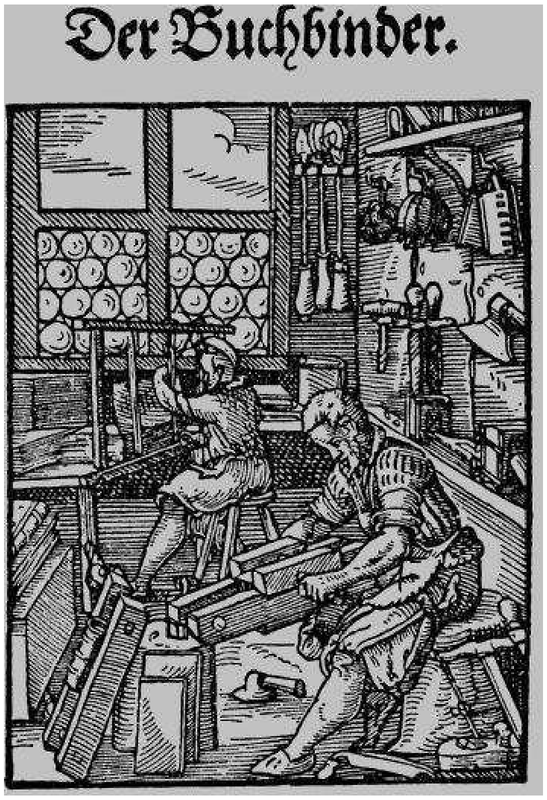
A book by Hans Sachs, *Eygentliche Beschreibung Aller Stände auff Erden* about 16th century trades, was published in Frankfurt in 1568 which included several woodcuts from drawings by Jost Amman. Figure 23 is one of these showing a printing shop. The two men in the background are setting type, taking the characters from the type cases in front of them. The men in the foreground are operating the printing press. The one on the left is removing a sheet of paper that has just been printed and the one on the right is using two circular pads to ink the type for the next sheet. A fresh sheet of paper will replace the one being removed. The flap at the left, with the cutouts, will be folded down to hold the paper in place, then the assembly folded over to lie on top of the type. The final assembly is slid into the press, the lever pulled to press the paper onto the type, the assembly slid out from the press and the printed page removed.

Figure 24 is another of the woodcuts, this time showing a book bindery. In the background there is



Herries Collection

Figure 23: 16th century printing shop



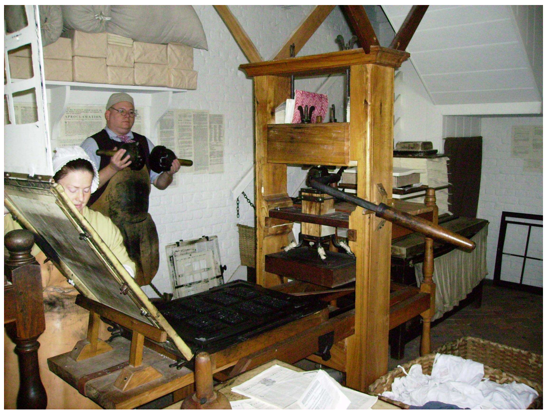
Herries Collection

Figure 24: 16th century book bindery

a sewing frame with a book and the man is sewing the sheets together. In the foreground there is a book in a lying press at the left and at the right the man is trimming the edges of the pages in a sewn book, which is in another lying press, before the covers will be put on. In those days books were often sold without covers so that clients could select the kind they wanted.

Little changed in the manufacture of books until the middle of the 19th century when some of the processes began to be mechanized (Chappell and Bringhurst, 1999). Figure 25 is a reconstructed 18th century print shop in Williamsburg, Virginia, 2007. James Mosley, who for 42 years was the Librarian at the St Bride Printing Library in London, said that it was ‘the most perfect and accurate working reconstruction of an 18th-century office’ that he had ever seen (Mosley, 2003). The paper holder is at the left, the type in the center and the press itself, a so-called *English Common Press*, at the right. The man is preparing to ink the type.

Also at Williamsburg is a reconstructed 18th century bindery, shown in Figure 26. Two sewing frames are in the foreground and a large standing press is in the semi-background.



Herries Collection

Figure 25: Reconstructed 18th century print shop (Williamsburg 2007)

Herries Collection

Figure 26: Reconstructed 18th century bindery (Williamsburg 2007)

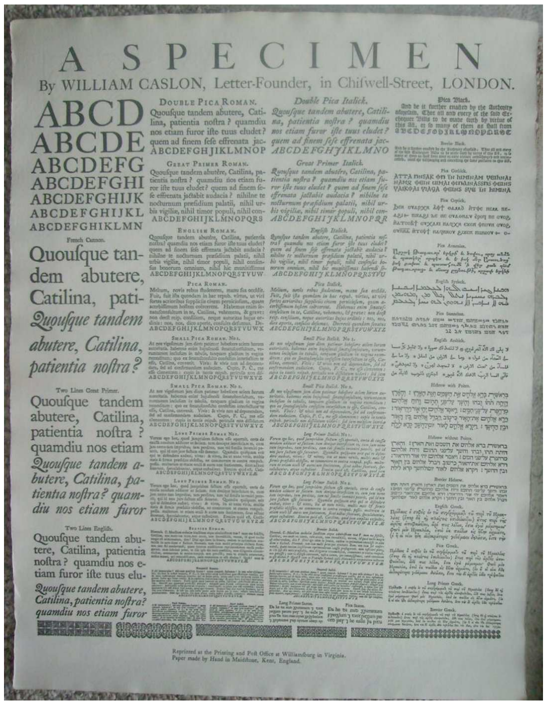
In the days of the American Colonies, printing was not encouraged. Sir William Berkeley, who was the governer of Virginia for 1642 to 1652 and again from 1660 to 1677, spoke for many officials when he said,

> But, I thank God, there are no free schools nor printing, and I hope we shall not have these for hundreds of years; for learning has brought disobedience, and heresy, and sects into the world, and printing has divulged them, and libels against the best government. God keep us from both.

However, using type purchased from England, such as those of William Caslon (1692–1766), printing became a thriving business. Figure 27 is Caslon's first specimen sheet, originally printed in 1734. The original is 15½ by 20½ inches. As well as the expected roman, italic, and blackletter, the specimens include fonts for the Saxon, Gothic, Coptic, Armenian, Syriac, Samaritan, Arabic, Hebrew (both with and without points), and Greek alphabets. The roman ranges in size from Canon to Pearl although examples of 6- and 8-line Pica are also shown; the exotics mostly come in a single size although there are three sizes of Greek. There are also several typographic ornaments.

Nowadays, the size of a font is expressed in points but originally names were used. The more common sizes are given in Table 1.

Caslon's type was used in Philadelphia by John Dunlap for the first printing of *The Declaration of Inpependence* in 1776. A more prosaic example of the kind of work done by Colonial printers is Figure 28 showing the title page of *Every Man his own Doctor: or, The Poor Planter's Physician* as printed



A SPECIMEN

By WILLIAM CASLON, Letter-Founder, in Chiſwell-Street, LONDON.

ABCD
ABCDE
ABCDEFG
ABCDEFGHI
ABCDEFGHIJK
ABCDEFGHIJKL
ABCDEFGHIKLMN

French Cannon.

Quouſque tandem abutere, Catilina, pati-
Quouſque tandem abutere, Catilina, patientia noſtra?

Two Lines Great Primer.

Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam
Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam furor

Two Lines Engliſh.

Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus elu-
Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam furor

Double Pica Roman.

Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus eludet? quem ad finem ſeſe effrenata jac-
ABCDEFGHJIKLMNOP

Great Primer Roman.

Quouſque tandem abutère, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus eludet? quem ad finem ſeſe effrenata jactabit audacia? nihilne te nocturnum præſidium palatii, nihil urbis vigiliæ, nihil timor populi, nihil con-
ABCDEFGHIJKLMNOPQRS

Engliſh Roman.

Quouſque tandem abutère, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus eludet? quem ad finem ſeſe effrenata jactabit audacia? nihilne te nocturnum præſidium palatii, nihil urbis vigiliæ, nihil timor populi, nihil conſenſus bonorum omnium, nihil hic munitiſſimus
ABCDEFGHIJKLMNOPQRSTVUW

Pica Roman.

Melium, novis rebus ſtudentem, manu ſua occidit. Fuit, fuit iſta quondam in hac repub. virtus, ut viri fortes acrioribus ſuppliciis civem perniciofum, quam acerbiſſimum hoſtem coërcerent. Habemus enim ſenatuſconſultum in te, Catilina, vehemens, & grave: non deeſt reip. conſilium, neque autoritas hujus ordinis: nos, nos, dico aperte, conſules deſumus. De-
ABCDEFGHIJKLMNOPQRSTVUWX

Small Pica Roman. No 1.

[illegible]
ABCDEFGHIJKLMNOPQRSTVUWXYZ

Small Pica Roman. No 2.

[illegible]
ABCDEFGHIJKLMNOPQRSTVUWXYZ

Long Primer Roman. No 1.

[illegible]
ABCDEFGHIJKLMNOPQRSTVUWXYZÆ

Long Primer Roman. No 2.

[illegible]
ABCDEFGHIJKLMNOPQRSTVUWXYZÆ

Bourgeois Roman.

[illegible]

Double Pica Italick.

Quouſque tandem abutere, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus eludet? quem ad finem ſeſe effrenata jac-
ABCDEFGHJIKLMNO

Great Primer Italick.

Quouſque tandem abutère, Catilina, patientia noſtra? quamdiu nos etiam furor iſte tuus eludet? quem ad finem ſeſe effrenata jactabit audacia? nihilne te nocturnum præſidium palatii, nihil urbis vigiliæ, nihil timor populi, nihil con-
ABCDEFGHIJKLMNOPQR

Engliſh Italick.

[illegible]
ABCDEFGHIJKLMNOPQRSTVU

Pica Italick.

Melium, novis rebus ſtudentem, manu ſua occidit. Fuit, fuit iſta quondam in hac repub. virtus, ut viri fortes acrioribus ſuppliciis civem perniciofum, quam acerbiſſimum hoſtem coërcerent. Habemus enim ſenatuſconſultum in te, Catilina, vehemens, & grave: non deeſt reip. conſilium, neque autoritas hujus ordinis: nos, nos, dico aperte, conſules deſumus. Decrevit quondam ſenatus
ABCDEFGHIJKLMNOPQRSTVUWXYZ

Small Pica Italick. No 1.

[illegible]

Small Pica Italick. No 2.

[illegible]

Long Primer Italick. No 1.

[illegible]

Long Primer Italick. No 2.

[illegible]

Bourgeois Italick.

[illegible]

Pica Black.

And be it further enacted by the Authority aforeſaid, That all and every of the ſaid Exchequer Bills to be made forth by virtue of this Act, or ſo many of them as ſhall from
ABCDEFGHIJKLMNOPQRST

Brevier Black.

[illegible]

Pica Gothick.

[illegible]

Pica Coptick.

[illegible]

Pica Armenian.

[illegible]

Engliſh Syriack.

[illegible]

Pica Samaritan.

[illegible]

Engliſh Arabick.

[illegible]

Hebrew with Points.

[illegible]

Hebrew without Points.

[illegible]

Brevier Hebrew.

[illegible]

Engliſh Greek.

[illegible]

Pica Greek.

[illegible]

Long Primer Greek.

[illegible]

Brevier Greek.

[illegible]

Long Primer Saxon.

[illegible]

Pica Saxon.

[illegible]

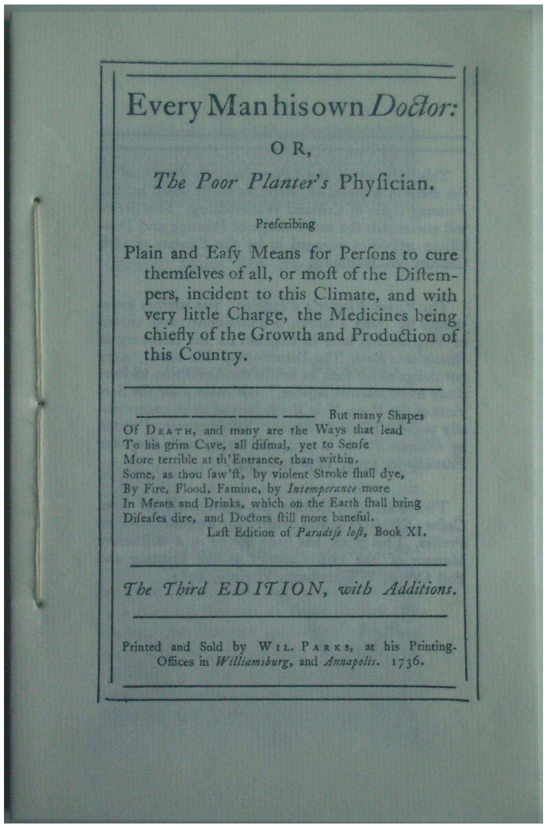
Reprinted at the Printing and Poſt Office at Williamſburg in Virginia.
Paper made by Hand in Maidſtone, Kent, England.

Herries Collection

Figure 27: Specimen sheet of Caslon types (1734)

Table 1: Traditional font size designations

Points	Name
3	Excelsior
3½	Brilliant
4	Diamond
5	Pearl
5½	Agate
6	Nonpareil
6½	Mignonette
7	Minion
8	Brevier
9	Bourgeois
10	Long Primer
11	Small Pica
12	Pica
14	English
18	Great Primer
24	Double (or Two Line) Pica
28	Double (or Two Line) English
36	Double (or Two Line) Great Primer
48	French Canon (or Four Line Pica)
60	Five Line Pica
72	Six line Pica
96	Eight Line Pica



Every Man his own *Doctor:*

O R,

The Poor Planter's Phyſician.

Preſcribing

Plain and Eaſy Means for Perſons to cure themſelves of all, or moſt of the Diſtempers, incident to this Climate, and with very little Charge, the Medicines being chiefly of the Growth and Production of this Country.

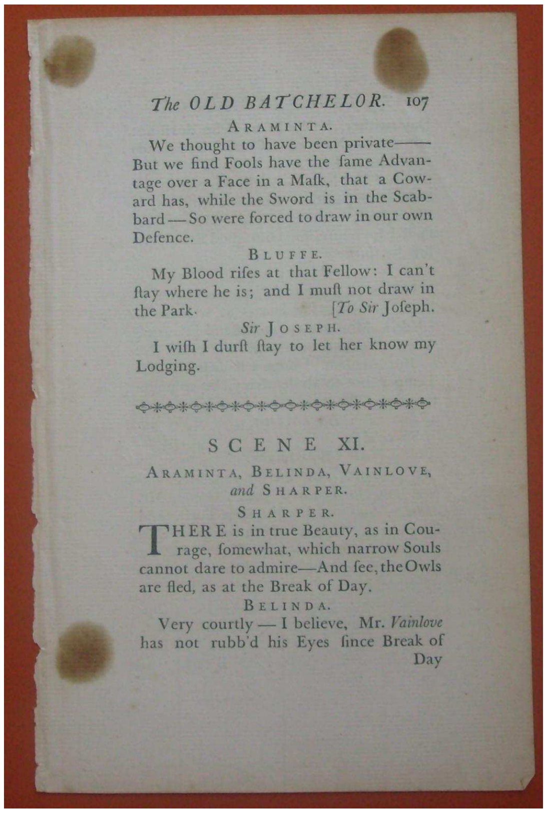
———— ———— ———— —— But many Shapes
Of Death, and many are the Ways that lead
To his grim Cave, all diſmal, yet to Senſe
More terrible at th'Entrance, than within.
Some, as thou ſaw'ſt, by violent Stroke ſhall dye,
By Fire, Flood, Famine, by *Intemperance* more
In Meats and Drinks, which on the Earth ſhall bring
Diſeaſes dire, and Doctors ſtill more baneful.
Laſt Edition of *Paradiſe loſt*, Book XI.

The Third EDITION, with Additions.

Printed and Sold by Wil. Parks, at his Printing-Offices in *Williamsburg*, and *Annapolis*. 1736.

Herries Collection

Figure 28: Title page of *Every Man his own Doctor*, Williamsburg, VA (1736)



The OLD BATCHELOR. 107

Araminta.

We thought to have been private——
But we find Fools have the ſame Advantage over a Face in a Maſk, that a Coward has, while the Sword is in the Scabbard — So were forced to draw in our own Defence.

Bluffe.

My Blood riſes at that Fellow: I can't ſtay where he is; and I muſt not draw in the Park. [*To Sir* Joſeph.

Sir Joseph.

I wiſh I durſt ſtay to let her know my Lodging.

SCENE XI.

Araminta, Belinda, Vainlove, *and* Sharper.

Sharper.

THERE is in true Beauty, as in Courage, ſomewhat, which narrow Souls cannot dare to admire—And ſee, the Owls are fled, as at the Break of Day.

Belinda.

Very courtly — I believe, Mr. *Vainlove* has not rubb'd his Eyes ſince Break of Day

Herries Collection

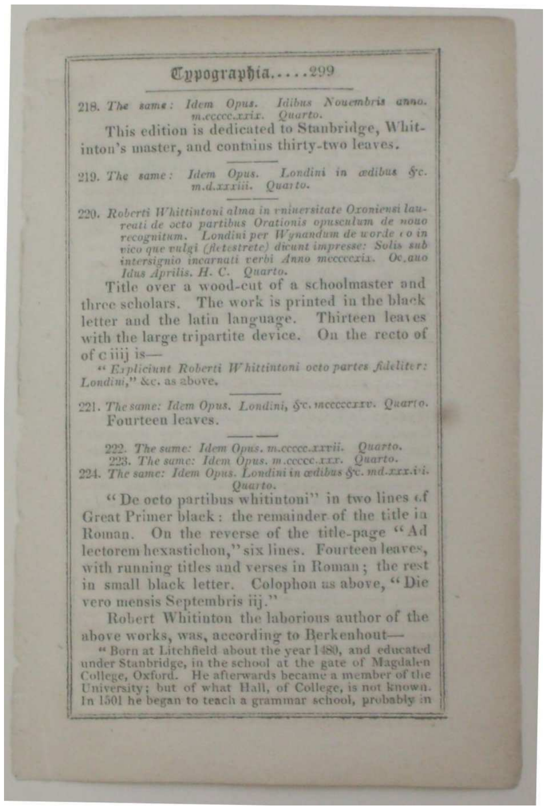
Figure 29: Page from Baskerville's edition of *The Plays and Poems of William Congreve* (1761)

in Williamsburg by William Parks in 1736. This edition is hand set with Caslon Oldstyle Type. The original is 5 by 7½ inches. The binding of such publications was very easy as the sheets were simply sewn together along the lines of Japanese stab bindings, but not so attractively.

The book was very popular; two editions were printed by William Parks, and Benjamin Franklin printed three editions between 1734 and 1737. The reprinted version notes that 'The Directions in the Book "were not designed for such as are in the Condition to Purchase more learned Advice" but mainly for the Services of the Poor'. The Directions mainly seemed aimed at making the patient so uncomfortable that it was better to be well than ill. The recommended treatments for almost everything except physical injuries seemed to involve the letting of copious amounts of blood accompanied by potions aimed at purging anything the patient may have eaten or drunk over the previous couple of days.

Like Caslon, John Baskerville (1706–1775) came from the Birmingham area in England. He printed his first book, Virgil's *Georgics*, in 1757. Not only did he design his type but he also improved on the printing press of the day and experimented with the formula for ink to produce one that was blacker and more uniform, and also dried quicker which improved the overall efficiency of the printing process. A page from his 1761 edition of *The Plays and Poems of William Congreve* is shown in Figure 29; the original is 5¾ by 8⅞ inches. He invented, and used, a new kind of paper called *wove* rather than the normal *laid* paper. His type had greater contrast between the thick and thin strokes than Caslon's and was more open. His work was not much appreciated in his native England as it was felt to be too brilliant, or bright, thus hurting the eyes. However he had a major influence on continental type designers such as Fournier, Didot and Bodoni.

John Johnson (1777–1848) produced an exhaustive survey of typography and printing in his two volume, 1300 page *Typographia, or the Printers' Instructor* published in 1824. The work was produced in four sizes, the largest being royal octavo (6⅛ by 9⅞ inches) and the smallest, as shown in Figure 29,



Typographia.....299

218. *The same: Idem Opus. Idibus Nouembris anno. m.ccccc.xxix. Quarto.*

This edition is dedicated to Stanbridge, Whitinton's master, and contains thirty-two leaves.

219. *The same: Idem Opus. Londini in ædibus &c. m.d.xxxiii. Quarto.*

220. *Roberti Whittintoni alma in vniuersitate Oxoniensi laureati de octo partibus Orationis opusculum de nouo recognitum. Londini per Wynandum de worde co in vico que vulgi (fletestrete) dicunt impresse: Solis sub intersignio incarnati verbi Anno mccccxix. Oc,auo Idus Aprilis. H. C. Quarto.*

Title over a wood-cut of a schoolmaster and three scholars. The work is printed in the black letter and the latin language. Thirteen leaves with the large tripartite device. On the recto of of c iiij is—

"*Expliciunt Roberti Whittintoni octo partes fideliter: Londini,*" &c. as above.

221. *The same: Idem Opus. Londini, &c. mcccccxxv. Quarto.* Fourteen leaves.

222. *The same: Idem Opus. m.ccccc.xxvii. Quarto.*
223. *The same: Idem Opus. m.ccccc.xxx. Quarto.*
224. *The same: Idem Opus. Londini in ædibus &c. md.xxx.vi. Quarto.*

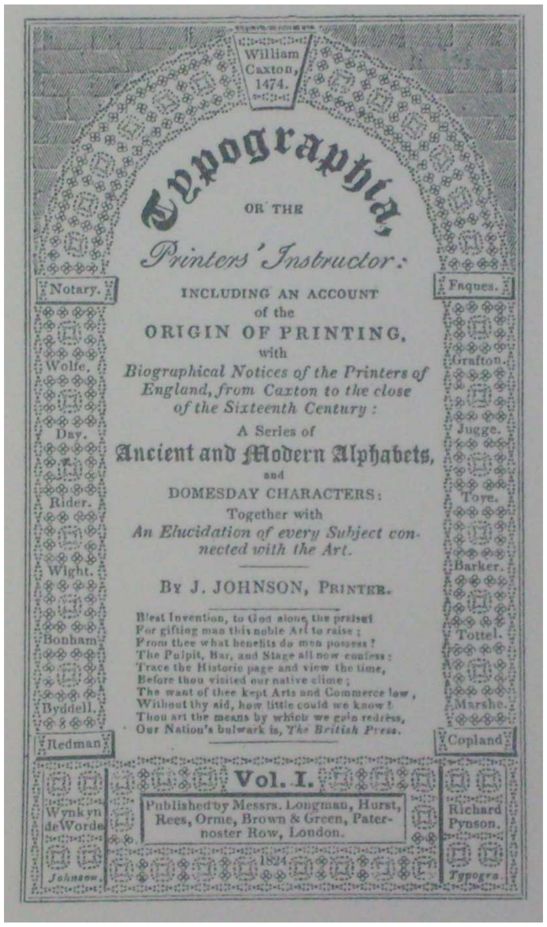
"De octo partibus whitintoni" in two lines of Great Primer black: the remainder of the title in Roman. On the reverse of the title-page "Ad lectorem hexastichon," six lines. Fourteen leaves, with running titles and verses in Roman; the rest in small black letter. Colophon as above, "Die vero mensis Septembris iij."

Robert Whitinton the laborious author of the above works, was, according to Berkenhout—

"Born at Litchfield about the year 1480, and educated under Stanbridge, in the school at the gate of Magdalen College, Oxford. He afterwards became a member of the University; but of what Hall, of College, is not known. In 1501 he began to teach a grammar school, probably in

Herries Collection

Figure 30: Page from John Johnson's *Typographia* (1824)



William Caxton, 1474.

Typographia, or the *Printers' Instructor:*

INCLUDING AN ACCOUNT of the ORIGIN OF PRINTING, with *Biographical Notices of the Printers of England, from Caxton to the close of the Sixteenth Century:*

A Series of Ancient and Modern Alphabets, and DOMESDAY CHARACTERS: Together with *An Elucidation of every Subject connected with the Art.*

BY J. JOHNSON, PRINTER.

Blest Invention, to God alone the praise!
For gifting man this noble Art to raise;
From thee what benefits do men possess?
The Pulpit, Bar, and Stage all now confess:
Trace the Historic page and view the time,
Before thou visited our native clime;
The want of thee kept Arts and Commerce low,
Without thy aid, how little could we know!
Thou art the means by which we gain redress,
Our Nation's bulwark is, *The British Press.*

Notary. Wolfe. Day. Rider. Wight. Bonham Byddell. Redman Wynkyn deWorde Johnson.

Faques. Grafton. Jugge. Toye. Barker. Tottel. Marshe. Copland Richard Pynson. Typogra.

Vol. I.

Published by Messrs. Longman, Hurst, Rees, Orme, Brown & Green, Paternoster Row, London.

1824

Herries Collection

Figure 31: Title page of John Johnson's *Typographia* (1824)

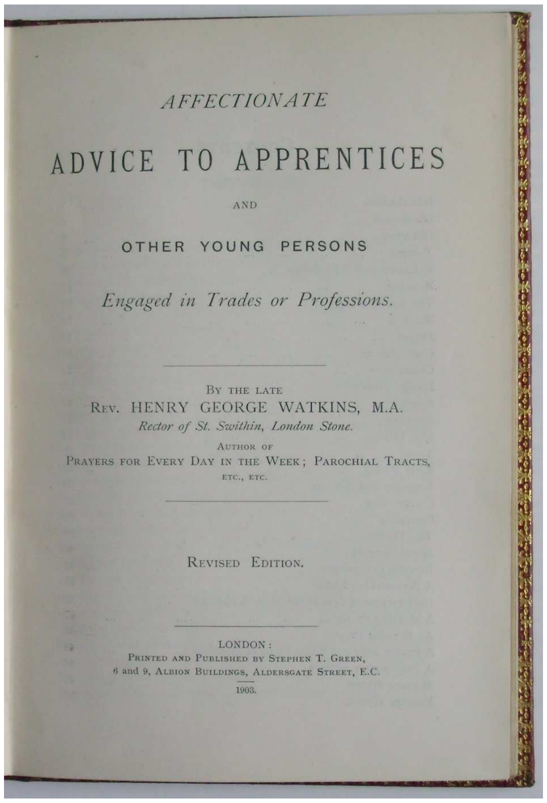
being thirty-twomo (3¼ by 4⅞ inches) (Wulling, 1967). The latter is not easy to read because of the small size of the print, from 8pt down to 4pt, but it must have been infinitely more difficult to typeset and proofread the half a million words in the two volumes. The title pages alone, one of which is shown in Figure 31 enlarged slightly, were built up using over a thousand flowers and rules. Included in the two volumes are sixty exotic alphabets assembled from the learned and commercial presses in England.

Many nineteenth century printers seem to have felt the need to show off their collection of fonts, often choosing a book's title page as the ideal place for this. Johnson's title page is an amazing piece of printing, but most certainly is not at all representative of the general style. Figure 32 is the title page from *Affectionate Advice to Apprentices*, written in 1827 by the Rector of St. Swithin's at London Stone, for the then Lord Mayor of London. It was distributed widely to many of the young people learning their crafts within the City.[6] This copy was reprinted in 1903. The original size is 4¾ by 7 inches. The Victorian lifestyle comes through very clearly: work, obey, learn, and pray. There is no mention of having fun but plenty of advice about avoiding sinful pleasures like going to the theatre to see a play. There is one telling remark, though.

> Our Creator, in great mercy to working people, has commanded every seventh day to be kept to the end of the world as a day of holy rest. If God had not appointed this rest, masters would never in the first instance have thought of giving it to their workpeople.

When clearing out my late father-in-law's Lincolnshire farmhouse I came across an old recipe book tucked away at the back of a cupboard. It covered a

[6] The Worshipful Company of Goldsmiths, chartered in 1327, still presents it to those seeking to become Freemen of the Company; other Livery Companies may do so as well.



AFFECTIONATE

ADVICE TO APPRENTICES

AND

OTHER YOUNG PERSONS

Engaged in Trades or Professions.

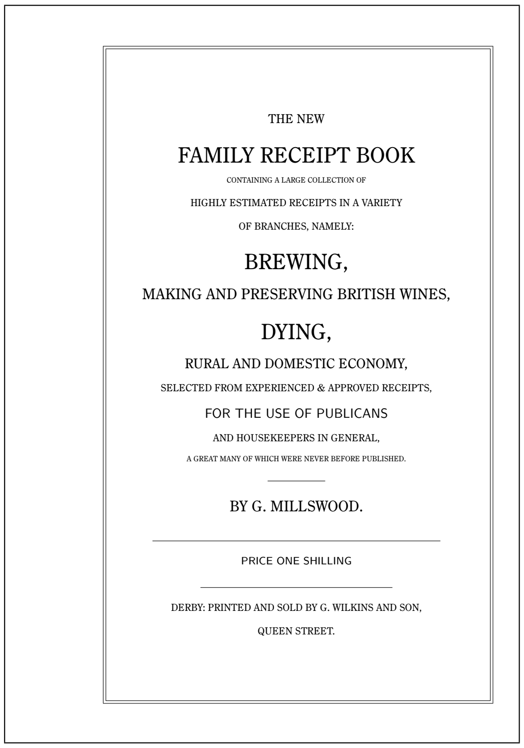
By the late
Rev. HENRY GEORGE WATKINS, M.A.
Rector of St. Swithin, London Stone.
Author of
Prayers for Every Day in the Week; Parochial Tracts,
etc., etc.

Revised Edition.

LONDON:
Printed and Published by Stephen T. Green,
6 and 9, Albion Buildings, Aldersgate Street, E.C.
1903.

Herries Collection

Figure 32: *Affectionate Advice to Apprentices* (1827)



THE NEW

FAMILY RECEIPT BOOK

CONTAINING A LARGE COLLECTION OF
HIGHLY ESTIMATED RECEIPTS IN A VARIETY
OF BRANCHES, NAMELY:

BREWING,

MAKING AND PRESERVING BRITISH WINES,

DYING,

RURAL AND DOMESTIC ECONOMY,
SELECTED FROM EXPERIENCED & APPROVED RECEIPTS,
FOR THE USE OF PUBLICANS
AND HOUSEKEEPERS IN GENERAL,
A GREAT MANY OF WHICH WERE NEVER BEFORE PUBLISHED.

BY G. MILLSWOOD.

PRICE ONE SHILLING

DERBY: PRINTED AND SOLD BY G. WILKINS AND SON,
QUEEN STREET.

Herries Collection

Figure 33: Title page of a Recipe book (1830)

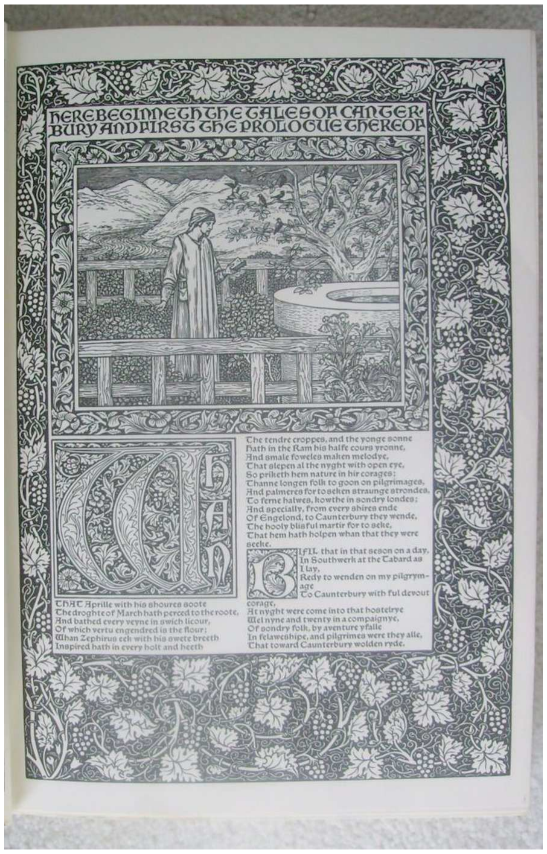
fascinating collection of topics ranging from brewing beer and adulterating rum, through dyeing cloth, to destroying vermin. As the pages were very fragile I reprinted it using the memoir class and the Century Old Style fonts from Christopher League's fontsite package (League, 2003). The title page is shown in Figure 33 and the LaTeX code, among many other examples for title pages, is given in Wilson (2007). The original had been printed by George Wilkins and Son, Derby, in 1830.

William Morris, one of the founders of the Arts and Crafts movement, disliked the erosion of craftsmanship by machines, and in 1891 he established the Kelmscott Press to produce hand made books of the highest quality.

Among others, he produced what is known as the Kelmscott Chaucer, his best known book, consisting of Chaucer's *Canterbury Tales* and all his other works — a total of 31 altogether — which include *The Romaunt of the Rose*, *Troilus and Cressida* and *A Treatise on the Astrolabe*. Although Morris designed the type (Chaucer) and the borders and the decorative initials, 87 woodcuts by Edward Burne-Jones were used as well. The book was published as a limited edition in 1896. There were 425 copies on paper, forty-eight of which were bound in pigskin by Thomas Cobden-Sanderson of the Doves Bindery (later the Doves Press). There were also thirteen copies on vellum. As the pages are 11 3/8 by 16 5/8 inches it is not a book for light reading.

Figure 34 is the opening page of the Prologue to Chaucer's *Canterbury Tales* from a facsimile of the Kelmscott Chaucer. The facsimile is 'slightly reduced in size' where the pages are only 8 5/8 by 12 7/8 inches and weighs 6 1/2 lbs (3 kg).

Morris believed that the factors in bookmaking were all interdependent, that is, the type, paper, ink, imposition and impression all had to be considered together. He also declared that a double spread must always be considered as a whole unit, as demonstrated in Figure 35. Although it has been said (Chappell and Bringhurst, 1999, p. 226) that his style has 'an abundance of thickets and undergrowth', he started people considering a book as a work of art, not as simply words on pages, and was instrumental in initiating the move away from the excesses of the Victorian printers.

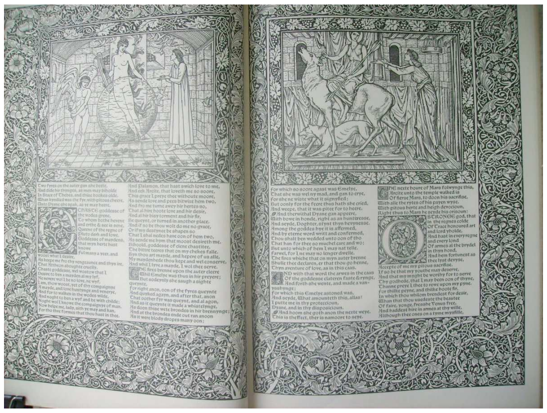


HEREBEGINNETH THE TALES OF CANTERBURY AND FIRST THE PROLOGUE THEREOF

WHAN THAT Aprille with his shoures soote
The droghte of March hath perced to the roote,
And bathed every veyne in swich licour,
Of which vertu engendred is the flour;
Whan Zephirus eek with his swete breeth
Inspired hath in every holt and heeth
The tendre croppes, and the yonge sonne
Hath in the Ram his halfe cours yronne,
And smale foweles maken melodye,
That slepen al the nyght with open eye,
So priketh hem nature in hir corages;
Thanne longen folk to goon on pilgrimages,
And palmeres for to seken straunge strondes,
To ferne halwes, kowthe in sondry londes;
And specially, from every shires ende
Of Engelond, to Caunterbury they wende,
The hooly blisful martir for to seke,
That hem hath holpen whan that they were seeke.
BIFIL that in that seson on a day,
In Southwerk at the Tabard as I lay,
Redy to wenden on my pilgrymage
To Caunterbury with ful devout corage,
At nyght were come into that hostelrye
Wel nyne and twenty in a compaignye,
Of sondry folk, by aventure yfalle
In felaweshipe, and pilgrimes were they alle,
That toward Caunterbury wolden ryde.

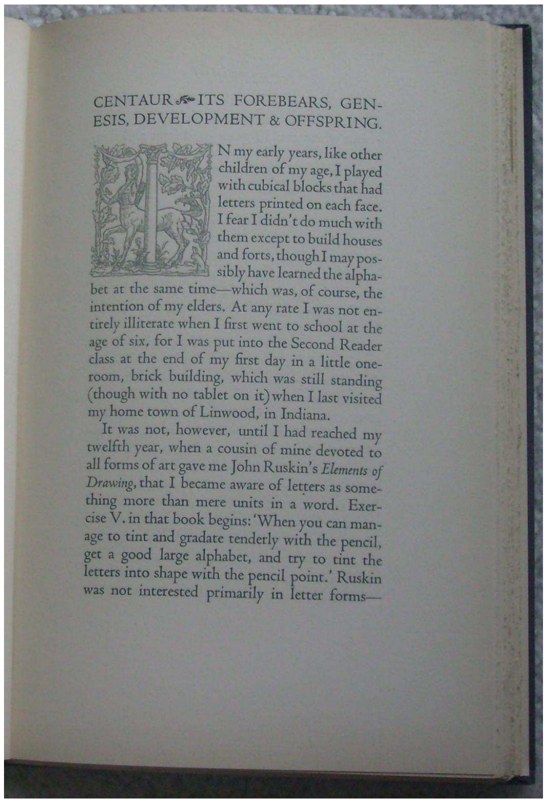
Herries Collection

Figure 34: Opening page of the Kelmscott Chaucer's *Prologue* (1896)



Herries Collection

Figure 35: Double spread from the Kelmscott Chaucer (1896)



CENTAUR ❧ ITS FOREBEARS, GENESIS, DEVELOPMENT & OFFSPRING.

IN my early years, like other children of my age, I played with cubical blocks that had letters printed on each face. I fear I didn't do much with them except to build houses and forts, though I may possibly have learned the alphabet at the same time—which was, of course, the intention of my elders. At any rate I was not entirely illiterate when I first went to school at the age of six, for I was put into the Second Reader class at the end of my first day in a little one-room, brick building, which was still standing (though with no tablet on it) when I last visited my home town of Linwood, in Indiana.

It was not, however, until I had reached my twelfth year, when a cousin of mine devoted to all forms of art gave me John Ruskin's *Elements of Drawing*, that I became aware of letters as something more than mere units in a word. Exercise V. in that book begins: 'When you can manage to tint and gradate tenderly with the pencil, get a good large alphabet, and try to tint the letters into shape with the pencil point.' Ruskin was not interested primarily in letter forms—

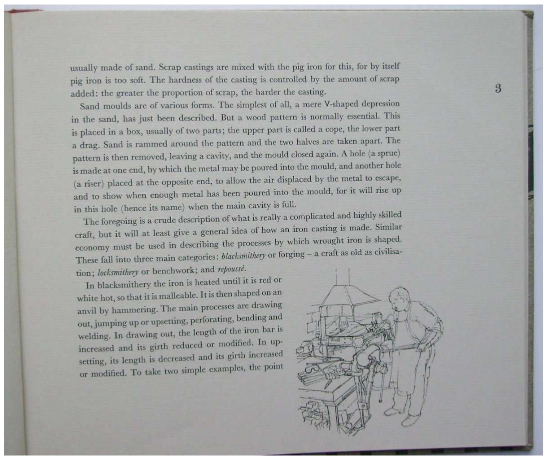
Herries Collection

Figure 36: *The Centaur Types* (1949)

5 Almost today

The traditions that started to be established in the 16th century are still seen today. Although books are not so lavishly decorated as some from the early days of printing, in general they have calmed down from the freneticism that occurred during the 19th century.

Manuscripts tended to emphasise the capital letter at the start of a paragraph (see Figures 15 and 16), and especially at the start of a major piece of the text as in Figure 17. Versals are still used, as shown in Figure 36 which is the opening page of *The Centaur Types* (Rogers, 1949), but much more rarely than in medieval times. Bruce Rogers (1870–1957) is said to be the 'most accomplished book designer that America has yet produced' (Lawson, 1990, p. 62). He was also the designer of the Centaur type which 'has been one of the widely praised roman types of our time' (ibid, p. 72). Rogers described how he came to design Centaur in his book *The Centaur Types*, which, of course, is set in Centaur and also includes exact size reproductions of the engraver's patterns. The original size is 6 1/4 by 9 1/2 inches.



usually made of sand. Scrap castings are mixed with the pig iron for this, for by itself pig iron is too soft. The hardness of the casting is controlled by the amount of scrap added: the greater the proportion of scrap, the harder the casting.

Sand moulds are of various forms. The simplest of all, a mere V-shaped depression in the sand, has just been described. But a wood pattern is normally essential. This is placed in a box, usually of two parts; the upper part is called a cope, the lower part a drag. Sand is rammed around the pattern and the two halves are taken apart. The pattern is then removed, leaving a cavity, and the mould closed again. A hole (a sprue) is made at one end, by which the metal may be poured into the mould, and another hole (a riser) placed at the opposite end, to allow the air displaced by the metal to escape, and to show when enough metal has been poured into the mould, for it will rise up in this hole (hence its name) when the main cavity is full.

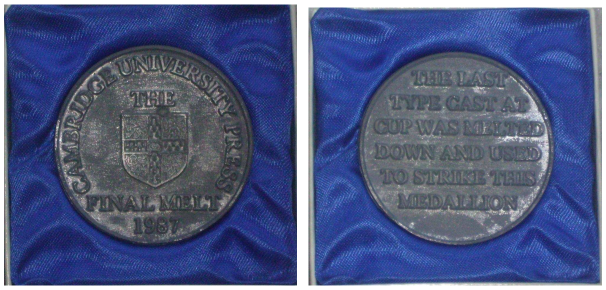
The foregoing is a crude description of what is really a complicated and highly skilled craft, but it will at least give a general idea of how an iron casting is made. Similar economy must be used in describing the processes by which wrought iron is shaped. These fall into three main categories: *blacksmithery* or forging – a craft as old as civilisation; *locksmithery* or benchwork; and *repoussé*.

In blacksmithery the iron is heated until it is red or white hot, so that it is malleable. It is then shaped on an anvil by hammering. The main processes are drawing out, jumping up or upsetting, perforating, bending and welding. In drawing out, the length of the iron bar is increased and its girth reduced or modified. In upsetting, its length is decreased and its girth increased or modified. To take two simple examples, the point

3

Herries Collection

Figure 37: *Hammer and Hand* (1969)


CAMBRIDGE UNIVERSITY PRESS
THE
FINAL MELT
1987
THE LAST TYPE CAST AT CUP WAS MELTED DOWN AND USED TO STRIKE THIS MEDALLION

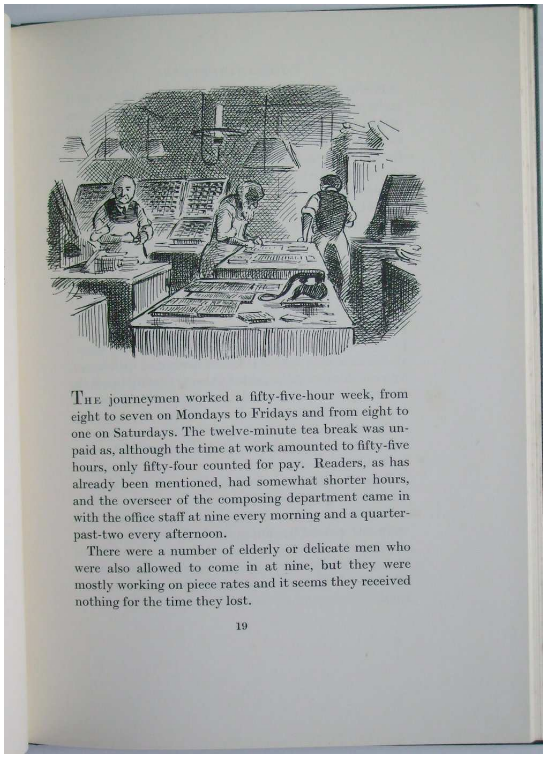
Herries Collection

Figure 39: Type metal medallion (1987)

The *Nuremberg Chronicle*, as in Figure 19, put woodcuts into cutouts in the text. The same idea can be seen in Figure 37 which shows page 3 from *Hammer and Hand* by Raymond Lister with drawings by Richard Bawden (Lister, 1969). The book is a long essay on the ironwork of Cambridge, principally the colleges' wrought iron gates. It was the Cambridge University Printer's Christmas book for 1969. The original page size is 9 3/4 by 8 3/8, and unusually it is printed on beige paper.

Another element in the design of the *Nuremberg Chronicle* is putting full width illustrations at the top of a page or, as in Figure 18, at the bottom. Figure 38 shows page 19 from *A Stickful of Nonpareil* by George Scurfield and illustrated by Edward Ardizzone (Scurfield, 1956). It was the Cambridge University Printer's Christmas book for 1956. The original is 6 1/2 by 9 inches. 'Nonpareil' is an old printers name for a particular size (6pt) of type, and the book consists of recollections of working at the Cambridge University Press around the end of the nineteenth century. The illustration shows a part of the composing room which is not all that different from the composing area in Jost Amman's 16th century view (Figure 23).

There are, of course, the inevitable changes, both in fashion and, more significantly, in technology. For example, the Cambridge University Press used metal types when it was founded in 1584 and since then all was set by hand until a Monotype composing machine was introduced in 1913 (Black, 1988). Computer-aided phototypesetting and lithographic printing were introduced in the early 1970s. Finally, after four centuries, the last vestiges of the traditional techniques vanished in 1987 when the types that remained in use were finally melted down and cast into commemorative medallions, shown in Figure 39.



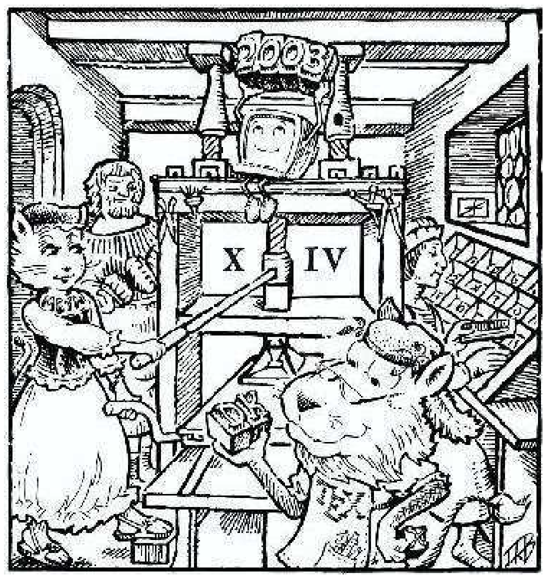
THE journeymen worked a fifty-five-hour week, from eight to seven on Mondays to Fridays and from eight to one on Saturdays. The twelve-minute tea break was unpaid as, although the time at work amounted to fifty-five hours, only fifty-four counted for pay. Readers, as has already been mentioned, had somewhat shorter hours, and the overseer of the composing department came in with the office staff at nine every morning and a quarter-past-two every afternoon.

There were a number of elderly or delicate men who were also allowed to come in at nine, but they were mostly working on piece rates and it seems they received nothing for the time they lost.

19

Herries Collection

Figure 38: *A Stickful of Nonpareil:* page 19 (1956)



Duane Bibby (EuroTEX 2003)

Figure 40: The TEX print shop, 2003

On the other hand, Duane Bibby's drawing for the EuroTEX 2003 conference (Figure 40) shows that the spirit of the tradition lives on.

References

Black, M. H. *Cambridge University Press 1584–1984*. Cambridge University Press, 1988.

Budge, E. A. Wallis. *The Rise and Progress of Assyriology*. London, 1925.

Chappell, Warren, and R. Bringhurst. *A Short History of the Printed Word*. Hartley & Marks, 1999.

Davis, N. P. *Lawrence and Oppenheimer*. Cape, 1969.

George, Andrew. *The Epic of Gilgamesh*. Penguin Classics, 2000.

Gurari, Eitan. "TeX4ht: LaTeX and TeX for Hypertext". 2007. Available from `http://www.cse.ohio-state.edu/~gurari/TeX4ht`.

Healey, John F. *The Early Alphabet*. Reading the Past. University of California Press, 1990.

Hinde, Thomas, editor. *The Domesday Book: England's Heritage, Then and Now*. Guild Publishing London, 1985.

Kemmerer, Sharon J., editor. *STEP: The Grand Experience*. Number 939 in Special Publications. National Institute of Standards and Technology, 1999.

Lawson, Alexander. *Anatomy of a Typeface*. David R. Godine, 1990.

League, Christopher. "TEX support for the FontSite 500 CD". 2003. Available from `http://contrapunctus.net/fs500tex`.

Lister, Raymond. *Hammer and Hand: An essay on the Ironwork of Cambridge*. Cambridge University Printer, 1969. Drawings by Richard Bawden.

Mosley, James. "The American Printing History Association 2003 Individual Award: Acceptance Remarks". 2003. Available from `http://www.printinghistory.org/htm/misc/awards/2003-james-mosley.htm`.

Pigafetta, Antonio. *Magellan's Voyage: A Narrative Account of the First Circumnavigation*. Yale University Press, 1969. In two volumes. Translated and edited by R. A. Skelton from the manuscript in the Beinecke Rare Book and Manuscript Library of Yale University.

Rogers, Bruce. *The Centaur Types*. October House, 1949.

Ryan, William, and W. Pitman. *Noah's Flood: The New Scientific Discoveries About the Event that Changed History*. Touchstone, 2000.

Scurfield, George. *A Stickful of Nonpareil*. Cambridge University Printer, 1956. Illustrated by Edward Ardizzone.

Thorpe, James. *The Gutenberg Bible: Landmark in Learning*. Huntington Library, 1999.

Wilson, Peter. "LATEX for ISO Standards". 2002a. Available from `ctan/macros/latex/contrib/isostds/iso`.

Wilson, Peter. "LATEX Package Files for ISO 10303". 2002b. Available from `ctan/macros/latex/contrib/isostds/iso10303`.

Wilson, Peter. "The Expressg MetaPost package for drawing box-line-annotation diagrams". 2004a. Available from `ctan/graphics/metapost/contrib/macros/expressg`.

Wilson, Peter. "The Memoir Class for Configurable Typesetting". 2004b. Available from `ctan/macros/latex/contrib/memoir`.

Wilson, Peter. "The Alphabet Tree". *TUGboat* **26**(3), 199–214, 2005.

Wilson, Peter. *Some Examples of Title Pages*. Herries Press, 2007. Available from `ctan/info/latex-samples/titlepages.pdf`.

Wulling, Emerson G. *J. Johnson, Typ.* Sumac Press, La Crosse, Wisconsin, 1967.

The STIX Project — From Unicode to fonts

Barbara Beeton
American Mathematical Society
201 Charles Street
Providence, RI 02904-2294
USA
bnb at ams dot org

Abstract

The goal of the STIX project is to provide fonts usable with other existing tools to make it possible to communicate mathematics and similar technical material in a natural way on the World Wide Web. This has involved two major efforts: enlarging Unicode to recognize the symbols of the mathematical language, and creating the fonts necessary to convert encoded texts into readable images.

This ten-year effort is finally resulting in fonts that can actually be used for the intended purpose.

1 Introduction: What is Unicode?

According to the Unicode manual, the original goal of the effort was "to unify the many hundreds of conflicting ways to encode characters, replacing them with a single, universal standard."

Unicode is thus an encoding system capable of representing all the world's languages in a way that will enable any person to interact with a computer in his own language. Nearly all modern computer operating systems are based on Unicode.

The three principal components of Unicode are the character, the block and the plane. A character is the smallest unit, carrying a semantic value. A character may represent a letter, a digit, or some other symbol or function.

A block consists of 256 characters — the number of characters that can be addressed by eight binary digits, addressed as 00–FF. A plane is composed of 256 blocks, for a total of 65,536 characters; there are 17 planes for a capacity of 1,114,112 in all [4, p. 2].

The first plane, Plane 0, is referred to as the Basic Multilingual Plane (BMP); if a piece of software claims to support Unicode, it should be able to access every character in the BMP.

Characters are assigned to blocks with the most heavily used given the lowest addresses; assignments are made in half-block (128-byte) chunks.

The first half of block 00 is the basic character set known as ASCII (the formal name is "C0 Controls and Basic Latin"). This contains the upper- and lowercase Latin alphabet, ten digits, various punctuation marks, and a number of control functions; it is the set of characters found on most computer keyboards. The second half of block 00 is known as "Latin 1", and includes many accented letters found in western European languages, as well as additional punctuation marks and control characters.

The next few blocks contain:

- the Greek and Cyrillic alphabets;
- a collection of diacritics to be used to compose accented letters not accommodated by Latin 1;
- Hebrew;
- Arabic;
- the scripts for many of the languages of India and southeast Asia.

Each script is allotted a half or full block as needed.

Blocks from 10 to 1F accommodate more language scripts, including extensions for Latin and Greek. Except for very basic symbols such as plus (+) or asterisk (*), non-language characters aren't included until blocks beginning at 20.

2 Who is responsible for Unicode, and how do things get added?

Unicode was developed and is maintained by the Unicode Technical Committee (UTC) an arm of the Unicode Consortium. Members of the consortium include most computer hardware manufacturers and software vendors. To align Unicode with ISO 10646, the standard on which hardware and software are actually based, the UTC works closely with the standardization subcommittee for coded character sets of the International Organization for Standardization.

The UTC members are individuals with various areas of expertise. Most have a strong background in computer software. Many are skilled as well in languages and linguistic-related areas. However, there are very few practicing physical scientists.

If something isn't in Unicode, there is a standard proposal form. This asks for a number of items:

- the repertoire of characters being requested, including character names;
- the context in which the proposed characters are used;
- references to authoritative published sources where the characters have been used;
- relationships the proposed characters bear to characters already encoded;
- contact information for the supplier of a computerized font to be used in printing the standard;
- names and addresses of contacts within national or user organizations.

The on-line description of the proposal review process warns that

- international standardization requires a significant effort on the part of the submitter;
- it frequently takes years to move from an initial proposal to final standardization;
- submitters should be prepared to become involved in the process.

In the case of the STIX proposal, all these warnings were true, in spades.

3 Initial conditions

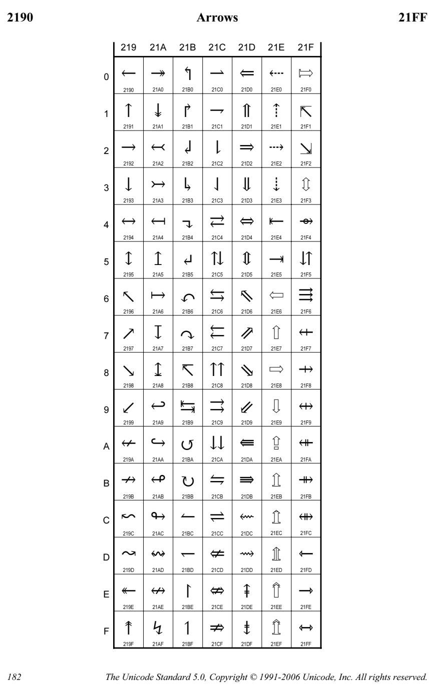
In 1997, when the STIX project began, Unicode was at version 2.0. It contained several blocks of interest for mathematics:

- combining diacritics (first half of block 03 for text; the last three 16-cell columns of block 20 for diacritics used with symbols)
- Greek (last half of block 03)
- arrows (last half of block 21) (Figure 1)
- mathematical operators (block 22) (Figure 2)
- miscellaneous technical (first half of block 23)
- geometric shapes (last half of block 25)

None of these blocks was entirely full at that time.

4 Character ≠ glyph

Unicode encodes *characters*. Each character has a designated, well-defined meaning. It appears in the Unicode charts as a representative glyph, or image. However, since the purpose of Unicode is to convey meaning, the shape of the glyph may vary. To take a trivial example, in text, an "A" has the same code whether it is upright Roman, italic (A), bold ($\mathbf{A}$),

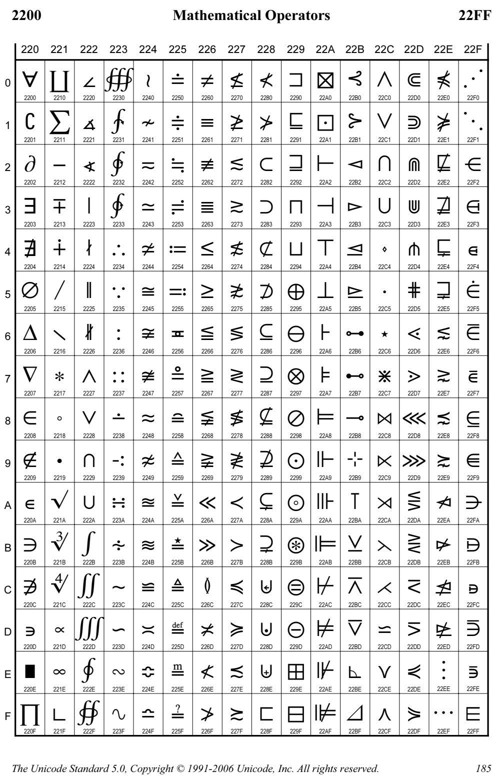


2190 Arrows 21FF

	219	21A	21B	21C	21D	21E	21F
0	← 2190	↠ 21A0	↰ 21B0	⇀ 21C0	⇐ 21D0	⇠ 21E0	⇰ 21F0
1	↑ 2191	↡ 21A1	↱ 21B1	⇁ 21C1	⇑ 21D1	⇡ 21E1	⇱ 21F1
2	→ 2192	↢ 21A2	↲ 21B2	⇂ 21C2	⇒ 21D2	⇢ 21E2	⇲ 21F2
3	↓ 2193	↣ 21A3	↳ 21B3	⇃ 21C3	⇓ 21D3	⇣ 21E3	⇳ 21F3
4	↔ 2194	↤ 21A4	↴ 21B4	⇄ 21C4	⇔ 21D4	⇤ 21E4	⇴ 21F4
5	↕ 2195	↥ 21A5	↵ 21B5	⇅ 21C5	⇕ 21D5	⇥ 21E5	⇵ 21F5
6	↖ 2196	↦ 21A6	↶ 21B6	⇆ 21C6	⇖ 21D6	⇦ 21E6	⇶ 21F6
7	↗ 2197	↧ 21A7	↷ 21B7	⇇ 21C7	⇗ 21D7	⇧ 21E7	⇷ 21F7
8	↘ 2198	↨ 21A8	↸ 21B8	⇈ 21C8	⇘ 21D8	⇨ 21E8	⇸ 21F8
9	↙ 2199	↩ 21A9	↹ 21B9	⇉ 21C9	⇙ 21D9	⇩ 21E9	⇹ 21F9
A	↚ 219A	↪ 21AA	↺ 21BA	⇊ 21CA	⇚ 21DA	⇪ 21EA	⇺ 21FA
B	↛ 219B	↫ 21AB	↻ 21BB	⇋ 21CB	⇛ 21DB	⇫ 21EB	⇻ 21FB
C	↜ 219C	↬ 21AC	↼ 21BC	⇌ 21CC	⇜ 21DC	⇬ 21EC	⇼ 21FC
D	↝ 219D	↭ 21AD	↽ 21BD	⇍ 21CD	⇝ 21DD	⇭ 21ED	⇽ 21FD
E	↞ 219E	↮ 21AE	↾ 21BE	⇎ 21CE	⇞ 21DE	⇮ 21EE	⇾ 21FE
F	↟ 219F	↯ 21AF	↿ 21BF	⇏ 21CF	⇟ 21DF	⇯ 21EF	⇿ 21FF

182 *The Unicode Standard 5.0, Copyright © 1991-2006 Unicode, Inc. All rights reserved.*

Figure 1: When the STIX project began, positions 21EB–21FF were empty. Copyright Unicode, used by permission.



2200 Mathematical Operators 22FF

	220	221	222	223	224	225	226	227	228	229	22A	22B	22C	22D	22E	22F
0	∀ 2200	∐ 2210	∠ 2220	∰ 2230	≀ 2240	≐ 2250	≠ 2260	≰ 2270	⊀ 2280	⊐ 2290	⊠ 22A0	⊰ 22B0	⋀ 22C0	⋐ 22D0	⋠ 22E0	⋰ 22F0
1	∁ 2201	∑ 2211	∡ 2221	∱ 2231	≁ 2241	≑ 2251	≡ 2261	≱ 2271	⊁ 2281	⊑ 2291	⊡ 22A1	⊱ 22B1	⋁ 22C1	⋑ 22D1	⋡ 22E1	⋱ 22F1
2	∂ 2202	− 2212	∢ 2222	∲ 2232	≂ 2242	≒ 2252	≢ 2262	≲ 2272	⊂ 2282	⊒ 2292	⊢ 22A2	⊲ 22B2	⋂ 22C2	⋒ 22D2	⋢ 22E2	⋲ 22F2
3	∃ 2203	∓ 2213	∣ 2223	∳ 2233	≃ 2243	≓ 2253	≣ 2263	≳ 2273	⊃ 2283	⊓ 2293	⊣ 22A3	⊳ 22B3	⋃ 22C3	⋓ 22D3	⋣ 22E3	⋳ 22F3
4	∄ 2204	∔ 2214	∤ 2224	∴ 2234	≄ 2244	≔ 2254	≤ 2264	≴ 2274	⊄ 2284	⊔ 2294	⊤ 22A4	⊴ 22B4	⋄ 22C4	⋔ 22D4	⋤ 22E4	⋴ 22F4
5	∅ 2205	∕ 2215	∥ 2225	∵ 2235	≅ 2245	≕ 2255	≥ 2265	≵ 2275	⊅ 2285	⊕ 2295	⊥ 22A5	⊵ 22B5	⋅ 22C5	⋕ 22D5	⋥ 22E5	⋵ 22F5
6	∆ 2206	∖ 2216	∦ 2226	∶ 2236	≆ 2246	≖ 2256	≦ 2266	≶ 2276	⊆ 2286	⊖ 2296	⊦ 22A6	⊶ 22B6	⋆ 22C6	⋖ 22D6	⋦ 22E6	⋶ 22F6
7	∇ 2207	∗ 2217	∧ 2227	∷ 2237	≇ 2247	≗ 2257	≧ 2267	≷ 2277	⊇ 2287	⊗ 2297	⊧ 22A7	⊷ 22B7	⋇ 22C7	⋗ 22D7	⋧ 22E7	⋷ 22F7
8	∈ 2208	∘ 2218	∨ 2228	∸ 2238	≈ 2248	≘ 2258	≨ 2268	≸ 2278	⊈ 2288	⊘ 2298	⊨ 22A8	⊸ 22B8	⋈ 22C8	⋘ 22D8	⋨ 22E8	⋸ 22F8
9	∉ 2209	∙ 2219	∩ 2229	∹ 2239	≉ 2249	≙ 2259	≩ 2269	≹ 2279	⊉ 2289	⊙ 2299	⊩ 22A9	⊹ 22B9	⋉ 22C9	⋙ 22D9	⋩ 22E9	⋹ 22F9
A	∊ 220A	√ 221A	∪ 222A	∺ 223A	≊ 224A	≚ 225A	≪ 226A	≺ 227A	⊊ 228A	⊚ 229A	⊪ 22AA	⊺ 22BA	⋊ 22CA	⋚ 22DA	⋪ 22EA	⋺ 22FA
B	∋ 220B	∛ 221B	∫ 222B	∻ 223B	≋ 224B	≛ 225B	≫ 226B	≻ 227B	⊋ 228B	⊛ 229B	⊫ 22AB	⊻ 22BB	⋋ 22CB	⋛ 22DB	⋫ 22EB	⋻ 22FB
C	∌ 220C	∜ 221C	∬ 222C	∼ 223C	≌ 224C	≜ 225C	≬ 226C	≼ 227C	⊌ 228C	⊜ 229C	⊬ 22AC	⊼ 22BC	⋌ 22CC	⋜ 22DC	⋬ 22EC	⋼ 22FC
D	∍ 220D	∝ 221D	∭ 222D	∽ 223D	≍ 224D	≝ 225D	≭ 226D	≽ 227D	⊍ 228D	⊝ 229D	⊭ 22AD	⊽ 22BD	⋍ 22CD	⋝ 22DD	⋭ 22ED	⋽ 22FD
E	∎ 220E	∞ 221E	∮ 222E	∾ 223E	≎ 224E	≞ 225E	≮ 226E	≾ 227E	⊎ 228E	⊞ 229E	⊮ 22AE	⊾ 22BE	⋎ 22CE	⋞ 22DE	⋮ 22EE	⋾ 22FE
F	∏ 220F	∟ 221F	∯ 222F	∿ 223F	≏ 224F	≟ 225F	≯ 226F	≿ 227F	⊏ 228F	⊟ 229F	⊯ 22AF	⊿ 22BF	⋏ 22CF	⋟ 22DF	⋯ 22EF	⋿ 22FF

The Unicode Standard 5.0, Copyright © 1991-2006 Unicode, Inc. All rights reserved. 185

Figure 2: Math operators in Unicode; at the start of the STIX project, the last code assigned was 22F1. Copyright Unicode, used by permission.

or sans serif (A). Similarly, an accented "é" can be represented either by one code or by a combination of the letter "e" and the combining diacritic "´".

This is not true for math notation, however. The same letter in different styles (italic, script, Fraktur, bold, ...) means different things. This is illustrated by the Hamiltonian equation from physics:

$$\mathcal{H} = \int d\tau(\varepsilon E^2 + \mu H^2)$$

In 1997, at the beginning of the STIX project, there was no way to unambiguously identify the script $\mathcal{H}$. Based only on the encoding, it was indistinguishable from the H on the right side of the equation:

$$H = \int d\tau(\varepsilon E^2 + \mu H^2)$$

Something more was needed; early proposals by UTC members recommended *markup* (e.g., font changes), such as provided by XML or MathML. However, it was realized that a physicist might wish to search for this entity in a corpus or database, and searching would be much more reliable if it could be done using an unambiguous code.

The UTC solution was to incorporate a substantial set of *mathematical alphanumerics*, about 1,000 characters. These variations on the Latin and Greek alphabets fill four complete blocks (U+1D40–U+1D7F) in Plane 1. Placement outside the BMP was meant to discourage casual users from using these special alphabets for things such as wedding invitations, where stylistic markup is more appropriate.

Another facet of the character/glyph dichotomy is the use in math notation of different-sized operators in text vs. display environments — the size used in text is generally smaller; compare $\sum_{i=0}^{\infty} x_i$ and

$$\sum_{i=0}^{\infty} x_i\,.$$

The sum symbol is just a single character in Unicode. Delimiters (parentheses, brackets, etc.) are also considered to be single characters, but they must be provided in many sizes, including segments suitable for piecing together to span multiple lines.

Unicode takes the position that such substitutions are the responsibility of the application.

5 Requesting additions to Unicode

In addition to the approximately 1,000 mathematical alphanumerics already mentioned, the STIX collection identified roughly 1,000 non-alphanumeric symbols that couldn't be found in Unicode version 2. These were assigned provisional identifiers in the Unicode Private Use Area (PUA) in order to keep track of them. IDs were assigned in order of accession, rather than by shape, usage, or other rational system.

Because of the large number of characters being requested, the UTC invited a representative of STIX to present the proposal in person at a regular UTC meeting, to answer questions directly, rather than carrying on an extensive paper and e-mail interchange. The fact that the proposal was backed by five professional societies and a technical publisher, based on actual experience in their publications, probably lessened the usual requirement for extensive examples. This did not mean that there was no requirement to justify every symbol; it did, however, allow symbols to be considered in groups rather than individually — if one member of a coherent symbol group (e.g., arrows with a triple stem pointing in several directions) was accepted, the rest of the group was accepted as well.

As noted earlier, Unicode assigns characters in blocks, preferably of groups with some inherent relationships. The UTC experts, acting on usage information provided with the proposed characters, classified them into groups that corresponded to the existing symbol blocks: operators, arrows, geometrics, and so forth. Then began the process of shoehorning them into the code space. First, the gaps in existing blocks were filled with appropriate items. Next, the number of characters in each category was tallied, and new blocks of appropriate sizes assigned. The bulk of the math additions first appeared (on line) in Unicode version 3.2, with the first paper publication in version 4.0.

As of Unicode version 5.0, these new blocks have been added:

- miscellaneous mathematical symbols A (U+2700–U+27EF)
- supplemental arrows A (U+27F0–U+27FF)
- supplemental arrows B (U+2900–U+297F)
- miscellaneous mathematical symbols B (U+2980–U+29FF)
- supplemental mathematical operators (U+2A00–U+2AFF)
- miscellaneous symbols and arrows (U+2B00–U+2BFF)

Not all of these blocks are filled yet, but space has been left where experience has shown growth is likely to occur.

One other key feature was adopted: a *variation selector* — a one-character code (U+FE00 for math symbols) identifying the preceding character as having the same meaning, but an alternate shape which cannot be composed from a base character plus a combining diacritic. An example is the relation $\gneqq$ (U+2269) vs. $\gvertneqq$ (U+2269,U+FE00). The use of the variation selector is very tightly controlled; all characters using it must be accepted explicitly by the UTC. Other shape variations must be indicated by markup and recognized by the application software.

Some important decisions were made during the course of this exercise that should make future submissions progress more smoothly.

First and foremost, it was accepted that mathematics is a language, and that symbols used in this context are as essential as the letter "e" is to English. Another "given" is that math notation is open-ended — mathematicians and other scientists will continue to invent and adopt new symbols, so the job isn't done, and may never be.

Just within the past few months, a mathematician from Morocco has submitted documentation of mathematical notation in Arabic — it is a mirror image of what we see in European language contexts. This generated a flurry of activity in the UTC to adopt a rational collection of right-to-left symbols to complement the basically left-to-right symbols already present. The new material will appear in Unicode version 5.1.

6 What wasn't accepted, and why not?

In spite of the generally high level of acceptance of characters proposed by STIX, the UTC rejected some symbols. The reason for most rejections was that they weren't "math". Symbols used by other disciplines (astronomy, meteorology) were not considered to be relevant to the STIX request; it was suggested that an organization involved in those disciplines should make a separate submission, at which time it would be considered on its own merits.

Some symbols were rejected because they were easily constructed as compounds of existing characters and combining diacritics; this includes any negated relations that hadn't already been encoded.

For some symbols, in particular ones that were identified after the initial proposal, the available documentation was deemed insufficient for acceptance. However, when a suitable in-context published example is found, acceptance of these stragglers is very likely.

Finally, some items in the STIX collection aren't considered independent symbols; they are partial glyphs used for constructing larger symbols such as multi-line parentheses or braces, or extenders for arrows. These weren't even submitted to the UTC since they fall into the area that is the responsibility of application software.

7 Okay, Unicodes have been assigned; how can we print them?

Assignment of Unicodes, while necessary, is not sufficient for use of these symbols in electronic or paper communication. It is also necessary to be able to generate images that can be understood by someone trying to read them. Here is where fonts come in.

A popular font for typesetting of math is Times Roman or one of its variants. This font, originally designed for newspaper use, is compact (a lot of material can be squeezed onto a page), and is legible at small sizes. Its adoption for technical material means that a large number of symbols have been designed to be compatible. Times Roman was the overwhelming choice of the STIX organizations as the base font around which the new STIX fonts would be created.

There are some very specific design criteria for a font intended for math:

- Each letter must be unambiguously recognizable in isolation; for Times, this means that a substitute must be provided for the italic v, since the usual Times shape is too easily confused with the Greek letter nu ν.
- Hairlines must be thick enough to keep shapes from breaking up in sub- and superscripts, and to withstand multiple photocopy runs.
- Normal weight must be readily distinguishable from bold.
- An alphabet intended for use as symbols need not be usable for continuous text; in fact, it is often desirable for a math alphabet to look a bit peculiar if used for text.

Implementation of the STIX glyphs was contracted out. The working list was a database in order of provisional ID; assignment of new Unicodes was still in the future. Glyphs were implemented in blocks, which were returned to the STIX Technical Review Committee for comments; any problem glyphs were returned to the contractor for repair.

The random ordering of the glyphs in the working list meant that glyphs intended to be used together, or supposed to be the same shape or weight, often weren't designed in the same batches, and weren't available for review at the same time. This meant that a final design review would be essential.

The random ordering also meant that the fonts couldn't yet be used for anything practical. Among other things, it was necessary to have a well-defined naming scheme. Because the fonts were delivered in Adobe Type 1 form, it was decided to assign glyph names according to the Adobe guidelines.

Except for a relatively small core of glyphs — essentially those representing the ASCII and Latin 1 blocks and some additional punctuation — the recommended form of a name was based on the Unicode, with extensions to indicate compounding or size and shape variations. This name begins with either "`uni`" or "`U`" for glyphs corresponding to characters in Unicode Plane 0 or Plane 1 respectively, or with "`stix`" for (the fewer than 256) glyphs with no corresponding Unicode.

8 Bookkeeping, bookkeeping

In order to keep track of what was happening, master tables or databases were maintained in several places. Tim Ingoldsby (of AIP, the overall project manager) started with the same database as used by the font contractor. To this he added, as phases were delivered, information about what glyphs were delivered in which phase, and the font and position in the font where each was located.

I maintained a list based on the original collection information, sorted by Unicode or provisional ID. This initially included sources, the names by which the sources refer to each glyph, the number of instances required (for weight, posture, size, etc.), and a glyph description. As new information became available, or was defined, it was added to the table:

- newly assigned Unicodes, with cross-references to and from the provisional ID;
- Type 1 glyph names;
- "TeX names", since several of the STIX organizations use that typesetting system;
- MathML entity names.

When delivery of the glyphs was nearing completion, Tim reprocessed my list, merged the differences into his database, and produced a file for checking. In this process we identified items that had been overlooked, and made a final list for completion of the deliveries.

That left only a few tasks:

- design review;
- shape and content corrections;
- packaging and user documentation;
- beta testing;
- (LaTeX)TeX support;
- final coordination of MathML entity names.

9 The design review

One more rearrangement was necessary — organizing the glyphs into groups that reflected shape categories, irrespective of identifier value. Since alphabets are ordered logically within Unicode, they had already been reviewed and corrected, and it was not necessary to look at them again. The other categories included

- diacritics;
- punctuation;
- geometric shapes (circles, squares, diamonds and lozenges, triangles, other polygons);
- arrows;
- relations (equals, greater/less, sub/supersets, others);
- binary operators (cups/caps, and/or, plus/times, other);
- large operators (integrals, other);
- delimiters and fences;
- other shapes.

Within each category, glyphs were arranged by similarity of shape and size (Figure 3). Making sure that everything was accounted for involved one more sweep through the entire STIX master table. This turned up some residual errors, which were corrected so that the permanent documentation would be accurate.

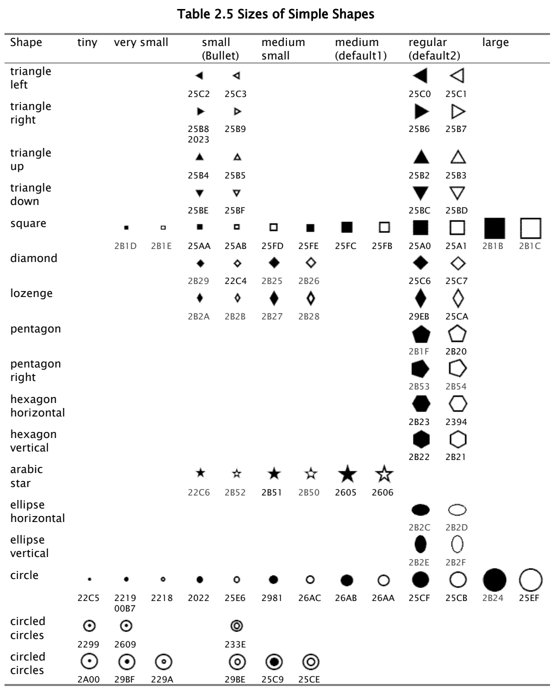
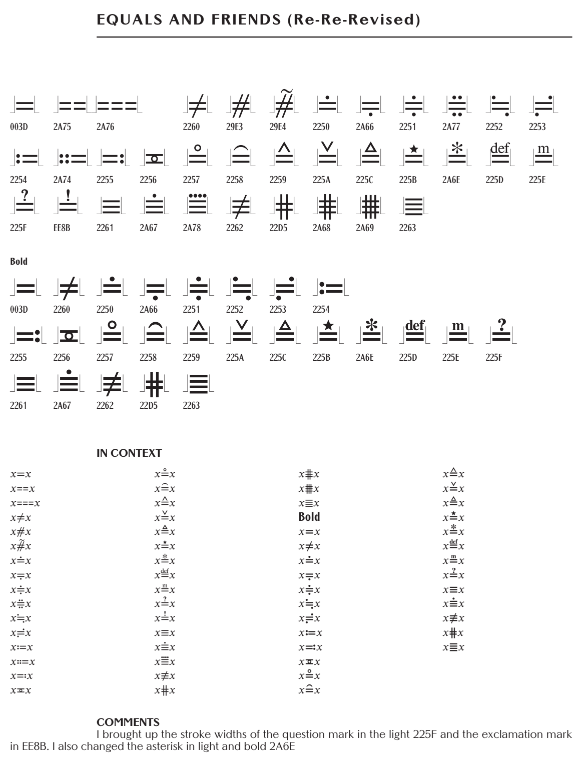


Table 2.5 Sizes of Simple Shapes

Shape	tiny	very small		small (Bullet)		medium small		medium (default1)		regular (default2)		large	
triangle left				◂ 25C2	◃ 25C3					◀ 25C0	◁ 25C1		
triangle right				▸ 25B8 2023	▹ 25B9					▶ 25B6	▷ 25B7		
triangle up				▴ 25B4	▵ 25B5					▲ 25B2	△ 25B3		
triangle down				▾ 25BE	▿ 25BF					▼ 25BC	▽ 25BD		
square		⬝ 2B1D	⬞ 2B1E	▪ 25AA	▫ 25AB	◽ 25FD	◾ 25FE	◼ 25FC	◻ 25FB	■ 25A0	□ 25A1	⬛ 2B1B	⬜ 2B1C
diamond				⬩ 2B29	⋄ 22C4	⬥ 2B25	⬦ 2B26			◆ 25C6	◇ 25C7		
lozenge				⬪ 2B2A	⬫ 2B2B	⬧ 2B27	⬨ 2B28			⧫ 29EB	◊ 25CA		
pentagon										⬟ 2B1F	⬠ 2B20		
pentagon right										⭓ 2B53	⭔ 2B54		
hexagon horizontal										⬣ 2B23	⎔ 2394		
hexagon vertical										⬢ 2B22	⬡ 2B21		
arabic star				⋆ 22C6	⭒ 2B52	⭑ 2B51	⭐ 2B50	★ 2605	☆ 2606				
ellipse horizontal										⬬ 2B2C	⬭ 2B2D		
ellipse vertical										⬮ 2B2E	⬯ 2B2F		
circle	⋅ 22C5	∙ 2219 00B7	∘ 2218	• 2022	◦ 25E6	⦁ 2981	⚬ 26AC	⚫ 26AB	⚪ 26AA	● 25CF	○ 25CB	⬤ 2B24	◯ 25EF
circled circles	⊙ 2299	☉ 2609			⌾ 233E								
circled circles	⨀ 2A00	⦿ 29BF	⊚ 229A		⦾ 29BE	◉ 25C9	◎ 25CE						

Figure 3: For geometric shapes, Unicode does make a distinction by size. From Unicode Technical Report #25 [1]; copyright Unicode, used by permission.


EQUALS AND FRIENDS (Re-Re-Revised)
Bold
IN CONTEXT
COMMENTS

I brought up the stroke widths of the question mark in the light 225F and the exclamation mark in EE8B. I also changed the asterisk in light and bold 2A6E

Figure 4: A proof sheet for the glyphs based on or related to equal signs.

For each group of symbols, proof sheets were generated (Figure 4), reviewed, and comments forwarded to the font specialist making the corrections. No category was accepted the first time around, but most required no more than two cycles for approval.

10 LaTeX support

For non-TeX use, the fonts will be delivered as OpenType. However, some of the STIX organizations are still using TeX implementations that don't even support the use of virtual fonts. For this reason, a set of Type 1 fonts will be provided as well, re-encoded to access the glyphs in 256-glyph chunks.

Most "TeX names" will not change for glyphs that were already available for TeX; Scott Pakin's comprehensive symbols list [2] has been an invaluable resource in the naming effort. For symbols that were not already generally available, new names have been assigned according to established naming principles, being careful to avoid conflicts with existing names. Actual coding of the LaTeX support package will be done by an experienced LaTeX programmer.

LaTeX support will not be included for the initial beta release, which is expected early in the Fall, but it should be available soon afterwards, and we anticipate that it will be ready to go by the time the fonts are posted for general release.

11 The future

We expect that a few more problems will be identified during beta testing, but in general, we believe that our efforts have resulted in a collection of fonts that will make it possible to represent nearly all mathematical expression both on paper and on computer screens. How this is actually done does depend on application developers, but since support of Unicode beyond just Plane 0 is beginning to be viewed as necessary by browser distributors, we are optimistic.

As we mentioned earlier, mathematical notation is open-ended. The mechanism for adding this notation to Unicode is now in place. The only question open is, how will new glyphs become part of these fonts. Presumably the STIX organizations will address that question after they've all had a well deserved rest.

References

[1] Barbara Beeton, Asmus Freytag and Murray Sargent III, Unicode Technical Report #25, Unicode Support for Mathematics, 2007. `http://www.unicode.org/reports/tr25`

[2] Scott Pakin, *The Comprehensive LaTeX Symbol List*, 2005. CTAN: `info/symbols/comprehensive`

[3] The Unicode Consortium, *The Unicode Standard, Version 2.0*, Addison-Wesley Developers Press, Reading, MA, 1996.

[4] The Unicode Consortium, *The Unicode Standard, Version 5.0*, edited by Julie D. Allen et al., Addison-Wesley, Upper Saddle River, NJ / Boston, 2006.

Fonts for every language: SIL's font projects and the Open Font License

Jonathan Kew
(based on slides by Victor Gaultney)
`jonathan_kew (at) sil (dot) org`

People around the world speak over 6000 languages; to communicate in writing, these language communities use hundreds of variations on dozens of different scripts. Written communication is fundamental to today's information-based world, and an ever-increasing proportion of the world's writing and communication is done through computer systems.

While the writing systems of major national languages are generally well supported, millions of people still have no way to type their language on a computer, because there are no adequate fonts for their writing system. Minority language communities may use a well-known script such as Latin, Cyrillic, Arabic, or Devanagari, but with particular extensions or variations that were not known to mainstream developers, and thus are not supported in standard fonts or software. In other cases, entire scripts used by minority groups are as yet not fully documented or encoded in Unicode.

Without adequate support for their language in computer systems, people are excluded from the "information society", or have access to it only through a second or third language, seriously hampering their ability to participate in the modern world and to access the resources they need to develop their own communities, as well as to benefit from current technology to enhance their own literature and cultural heritage.

Several factors may play a part in cutting language communities off from adequate computing solutions:

Availability Is there even a font for their script or alphabet?

Completeness Does that font support that particular language, with its unique variations on the script?

Complexity Does the technology support the rules of the writing system?

Quality Is the result both correct and attractive, in the eyes of the user community?

Accessibility Does everyone have access to that solution?

For many of the world's minorities, at least one (often several) of these factors severely limits their access to digital information and modern communication systems.

The situation is changing, albeit gradually. A growing number of individuals and groups are coming together to provide solutions, and SIL International is privileged to play a part. By linking experienced type designers and technicians with minority communities that have particular needs, we have been able to provide high-quality fonts that meet the requirements of a number of minority language groups. Graphite, an open-source font layout engine developed by SIL, makes it possible to program complex script behavior into fonts without waiting for the (sometimes lengthy) process of standardization and then implementation in engines like Uniscribe, Pango, or ICU. And XƎTEX provides a document formatting system able to work with complex scripts, whether implemented via OpenType layout or Graphite.

For solutions to be useful, they must be available. And to be maximally useful, they must also be adaptable, as no single developer can hope to meet the needs of all potential users. We therefore encourage font developers to consider releasing their work under the SIL Open Font License. This license is specifically designed for fonts and related software. To quote from the web site (`http://scripts.sil.org/OFL`), "The OFL is designed to be in tune with the FLOSS (Free/Libre and Open Source Software) culture. It builds upon good ideas already in existence in some free and open projects but by bringing our extensive font design experience and linguistic software engineering know-how into the mix, we believe we are able to make a font-specific license better, simpler, more human-readable, neutral and reusable."

It is our view that every community should have at least one high-quality font, working with standard software, suitable for their language. We invite the partnership of all who are interested in making this a reality.

Dual presentation with math from one source using GELLMU

William F. Hammond
Department of Mathematics & Statistics
University at Albany
Albany, New York 12222 USA
`http://www.albany.edu/~hammond/`

Abstract

Two traditional approaches for achieving simultaneous print and HTML output from a single marked-up source are relevant to the TEX community.

1. Write a LATEX article, and use a program that translates to HTML.
2. Write an article in an author-level XML document type, and use standard XML translation methods to generate both LATEX and HTML.

This article addresses a hybrid approach: the use of "generalized LATEX", as implemented in the GELLMU Project, to produce dual presentation from a single LATEX-like source. The method combines the reliability of XML document transformation with many of the conveniences traditionally available to LATEX authors.

1 Introduction

A contemporary author writing an article for "dual presentation" has in mind both the classical printed presentation of an article and the online form of an article formatted in HTML. There are particular challenges when mathematics is involved because it is moderately difficult to produce correct mathematical markup for modern HTML documents.

While mathematics is a principal concern in this article, most of what is said here is relevant, though less critically so, for documents that do not involve mathematics.

There are two main approaches for achieving dual presentation that are relevant to the TEX community. (Texinfo, the language of the GNU documentation system, also provides a route for dual presentation of articles without mathematical markup.)

1. Write a LATEX article, and use a program that translates to HTML.

2. Write an article in a suitable XML document type, such as DocBook, and use standard software for generating LATEX and HTML.

Both methods present challenges to authors who have been accustomed to using LATEX. In particular, if mathematics is involved, there are no widely deployed XML document types that support author-level mathematical markup. Since mid-2002 mathematical content in the second-generation form of HTML has been supported by the two most widely deployed web browsers, but not many articles seem to have appeared on the web in this form so far. The most likely reason is difficulty of creation.

This article addresses the use of "generalized LATEX", as implemented in the GELLMU Project, to produce dual content from a single LATEX-like source. (The overall system design in GELLMU is one for multiple outputs although at this time the standard implementation provides (i) printed output via standard LATEX and (ii) HTML.)

A generalized LATEX article under the GELLMU Project is essentially equivalent to an XML document under an author-level document type that may be called "GELLMU article". Preparing documents this way combines the reliability of XML document transformation with most of the conveniences, such as *newcommand* macro substitutions, the use of blank lines for paragraph boundaries, and cross-referencing, that are available when writing LATEX markup.

It should be emphasized that the GELLMU Project does not provide translation of classical LATEX to HTML or to any XML document type. Of course, one may open a classical LATEX document in an editor and invest time and energy to "port" it to GELLMU source, but there is no automation for this.

The task of translating legacy documents to HTML and to XML document types is difficult. In 2007 we are witness to more than 10 years of effort in this direction, and we still have no easy path, particularly when mathematics is involved. Nonetheless, translation, to the extent that it is reasonably possible, is very important because we have 30 years of legacy documents. (Most of the past translation efforts have been aimed at standard structured formats like HTML and the *DocBook* XML document type. It would be interesting to see if the translation task toward such standard formats can be improved by first translating toward an author-level document type, such as GELLMU *article*, that models structured LaTeX rather closely and then translating from there toward the original target.)

When a contemporary author has dual presentation in mind for new documents, writing classical LaTeX is no longer the best way to proceed because of the difficulty of translation from LaTeX to other formats. Writing for a suitable author-level XML document type is a much better way to have seamless dual presentation, and using the LaTeX-like front end offered by GELLMU makes it seem to the author very much like writing LaTeX.

In his talk preceding mine at TUG 2007 Chris Rowley ventured the idea that "peak TeX", like "peak oil", lies in the near future. In saying that I hope he is speaking of a peak in relative use by authors as markup source of the print-focused typesetting language that we have known. The other side of this is that *if*, as a community, we come to appreciate the usefulness of LaTeX-like markup, as opposed to classical LaTeX, as a general author front end for writing structured documents and then come to understand that practice as part of the TeX world and hone our techniques of formatting these structured documents for classical TeX-based typesetting, the future I imagine is one of more, not less, TeX.

2 Writing source markup

This is GELLMU source for a short paragraph with a relatively simple mathematical display.

```
The following identity may be
regarded as a formulation of the
Weierstrass product for the Gamma
function.
\[ \int_{0}^{\infty} t^x
   e^{-t} \frac{dt}{t} \int:
   = \frac{1}{x}
   \prod_{k=1}^{\infty}
   \frac{\bal{1 + \frac{1}{k}}^x
    }{\bal{1 + \frac{x}{k}}}
   \prod: \]
Understanding the derivation of
this identity is reasonable for a
bright student of first year
undergraduate calculus in the
United States.
```

This source compiles to:

> The following identity may be regarded as a formulation of the Weierstrass product for the Gamma function.
>
> $$\int_0^\infty t^x e^{-t}\frac{dt}{t} \;=\; \frac{1}{x}\prod_{k=1}^{\infty}\frac{\left(1+\frac{1}{k}\right)^x}{\left(1+\frac{x}{k}\right)}$$
>
> Understanding the derivation of this identity is reasonable for a bright student of first year undergraduate calculus in the United States.

The markup looks like classical LaTeX. In fact, except for the use of the zone closers `\int:` and `\prod:`, it would be classical LaTeX. It is *generalized LaTeX*.

The mandatory use of zone closers arises from the fact that the GELLMU system is not monolithic. Rather than being a single program, it is a suite of cross-platform component programs, each with a well-defined task, managed with a driver script. The first stage of processing operates at the level of syntax with almost no knowledge of markup vocabulary.[1] Because of this, GELLMU source, like Texinfo source, has stricter syntactic requirements than plain TeX and classical LaTeX.

Other ways in which GELLMU source differs from classical LaTeX source include:

1. Command arguments must be explicitly braced.
2. There may be no white space between a command name and the delimiter (a brace or bracket) for its first argument or option.
3. There may be no white space separating the delimiters of the successive arguments and options of a command.
4. Braces for the argument of a superscript or subscript may be omitted only if the argument is a single character.
5. The semi-colon at the end of a command name (such as `\latex;` above) indicates that the command does not introduce content. Often this type of semi-colon may be omitted, and, beyond that for most purposes `\foo;` may be regarded as shorthand for `\foo{}`.
6. The command vocabulary is somewhat different.

3 Another example

Figure 1 provides a GELLMU rendition of an example posted to the UseNet newsgroup `sci.math.research` on 29 October 2002[2] by David Madore of ENS, comparing TeX markup to MathML markup in order to illustrate the undisputed point that no

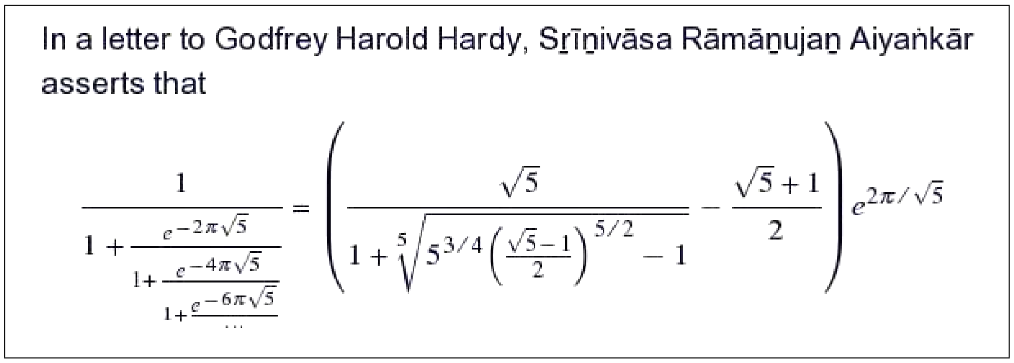
[1] It can be given lists of names of commands with shared syntactic properties.

[2] Message id: `apmpvn$bpb$1.repost@nef.ens.fr`

In a letter to Godfrey Harold Hardy, Sr̲īn̲ivāsa Rāmān̲ujan̲ Aiyaṅkār asserts that

$$\frac{1}{1+\frac{e^{-2\pi\sqrt{5}}}{1+\frac{e^{-4\pi\sqrt{5}}}{1+\frac{e^{-6\pi\sqrt{5}}}{\cdots}}}} = \left(\frac{\sqrt{5}}{1+\sqrt[5]{5^{3/4}\left(\frac{\sqrt{5}-1}{2}\right)^{5/2}-1}} - \frac{\sqrt{5}+1}{2}\right)e^{2\pi/\sqrt{5}}$$

Figure 1



In a letter to Godfrey Harold Hardy, Sr̲īn̲ivāsa Rāmān̲ujan̲ Aiyaṅkār asserts that

$$\frac{1}{1+\frac{e^{-2\pi\sqrt{5}}}{1+\frac{e^{-4\pi\sqrt{5}}}{1+\frac{e^{-6\pi\sqrt{5}}}{\cdots}}}} = \left(\frac{\sqrt{5}}{1+\sqrt[5]{5^{3/4}\left(\frac{\sqrt{5}-1}{2}\right)^{5/2}-1}} - \frac{\sqrt{5}+1}{2}\right)e^{2\pi/\sqrt{5}}$$

Figure 2

author would ever regularly want to write MathML directly.[3]

This is the GELLMU source underlying figure 1:

```
\macro{\=}{\ovbar}
\macro{\.}{\ovdot}
\newcommand{\b}[1]{\unbar{#1}}
In a letter to Godfrey Harold
Hardy, S\b{r}\={\i}\b{n}iv\={a}sa
R\={a}m\={a}\b{n}uja\b{n}
Aiya\.{n}k\={a}r asserts that
\[ \frac{1
 }{1+\frac{e^{-2\pi\sqrt{5}}
 }{1+\frac{e^{-4\pi\sqrt{5}}
 }{1+\frac{e^{-6\pi\sqrt{5}}
 }{\ldots}}}}
=
\bal{\frac{\sqrt{5}
 }{
 1+\sqrt[5]{5^{3/4}
 \bal{
  \frac{\sqrt{5}-1}{2}
 }^{5/2}-1}}
-\frac{\sqrt{5}+1}{2}}
e^{2\pi/\sqrt{5}} \]
```

In this markup, note first the use of GELLMU's `\macro` facility to provide emulation of classical LaTeX algorithmic accents with names that are not formed with letters. Further note the use of `\bal{ ... }` in place of the LaTeX usage `\left( ... \right)`. GELLMU has various balancers of this type and will eventually have more. This is related to the fact that the markup is simply a "front" for an SGML document type where a name is needed. The processor for XHTML + MathML output will not tolerate unbalanced balancing characters in a math zone except as provided through these balancers and also through the list generator

`\vect[...]{...}{...} ... {...}` .

The kinds of weak enforcement of mathematical semantics represented by such balancing provisions and by the requirement for explicit ending of *sum*, *int*, and *prod* containers is a prelude to future optional incorporation of stronger mathematical semantics in the markup.

Madore is correct in suggesting that one doesn't want to look at the MathML markup for this, but the rendering by *Firefox*, somewhat enlarged, is captured in the screenshot that is figure 2.

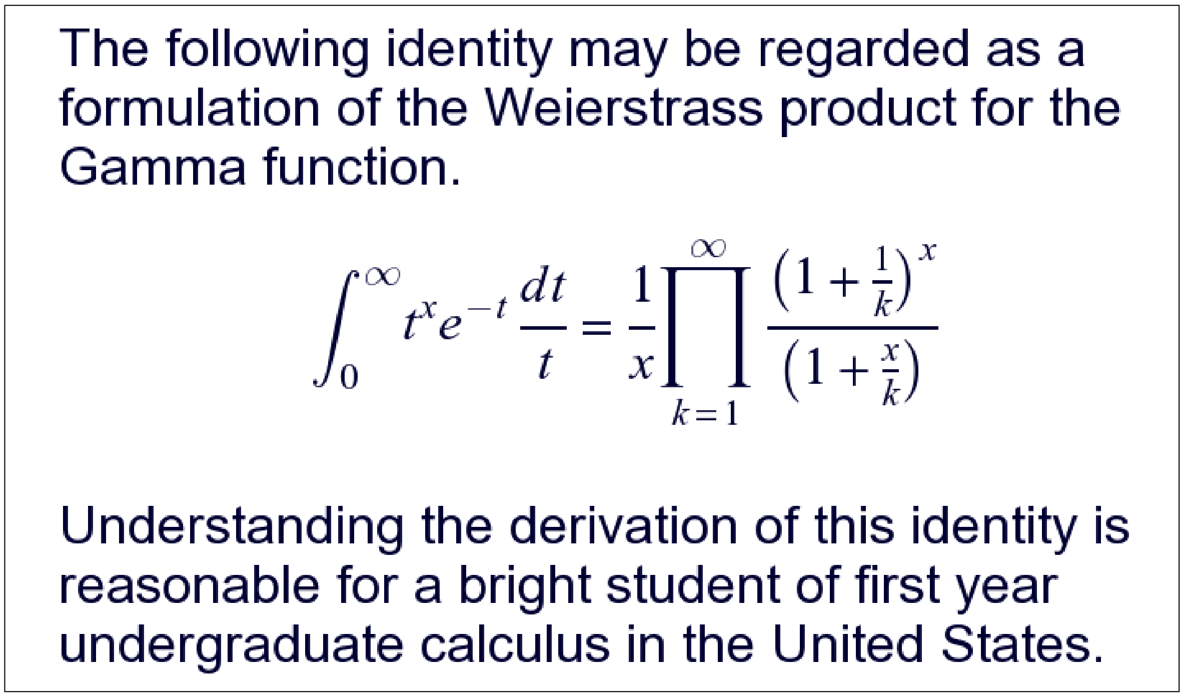
4 The importance of XHTML + MathML

There are several reasons why it is important to have articles and course materials with mathematical content online in modern HTML, i.e., XHTML + MathML.

4.1 Public relations for mathematics

- To a young person XHTML + MathML represents "math on the web".
- It's more flexible and more convenient for online reading than PDF — doubly so by comparison

[3] Madore ends his posting as follows: "And, to remain fully on topic, I ask: has this remarkable statement by Ramanujan ever been proven rigorously? And, if so, how complicated is it?"



The following identity may be regarded as a formulation of the Weierstrass product for the Gamma function.

$$\int_0^\infty t^x e^{-t}\,\frac{dt}{t} = \frac{1}{x}\prod_{k=1}^{\infty}\frac{\left(1+\frac{1}{k}\right)^x}{\left(1+\frac{x}{k}\right)}$$

Understanding the derivation of this identity is reasonable for a bright student of first year undergraduate calculus in the United States.

Figure 3

with double-column online PDF.

- In many cases print journals are disappearing as librarians strive to conserve money and shelf space.

- In an electronic library browsing HTML is the analogue of browsing in the stacks, while printing a PDF document is the analogue of making a copy of an article.

4.2 Special needs

- It's great for proof reading. (Enlarge and shorten the lines.)

- **Large print editions** at no cost.

  The small gamma bit presented earlier may easily be made to look like figure 3 in *Firefox* (a screenshot).

- Articles presented in XHTML + MathML comply with web accessibility guidelines. PDF documents normally do not. (In the GELLMU production stream (see figure 5) an intermediate stage file with suffix `.zml` generated during the final stage of translation to XHTML + MathML may be of more interest than the XHTML + MathML form to those wishing to generate specific output formats for various accessibility-related purposes.)

5 Compiling an article

5.1 Acquiring the software

GELLMU is based on cross-platform free software licensed under the GNU GPL. Its package is available from CTAN [2] and from the GELLMU web site [6]. The package requires several other free cross-platform programs: GNU *Emacs*, *perl*, and two standard libraries of SGML/XML software, Open SP (for *onsgmls*) and Expat (for *xmlwf*). Whatever image manipulation software is used for handling graphic inclusions in TEX on one's platform should suffice except one should note that neither PDF nor encapsulated PostScript (usually `.eps` files) may be included as an image within an HTML page. Therefore, if one is using incorporated images, one will want to have the ability to generate copies in, for example, the PNG and JPEG formats.

Linux: The required packages are generally part of a full GNU/Linux distribution. GELLMU should be installed in `/usr/local/gellmu` and symlinks to driver scripts should be made from a suitable place in one's command path.

Mac OS X and other Unix variants: The only difference from GNU/Linux is that some of the supporting packages may need to be acquired and installed.

MS Windows: The best strategy is to install a *full* Cygwin (`http://www.cygwin.com/`) distribu-

```
\documenttype{article}
\title{}
\begin{document}

The following identity may be regarded as a formulation of the
Weierstrass product for the Gamma function.
\[ \int_{0}^{\infty} t^x e^{-t} \frac{dt}{t} \int:
   = \frac{1}{x}
     \prod_{k=1}^{\infty}
       \frac{\bal{1 + \frac{1}{k}}^x}{\bal{1 + \frac{x}{k}}}
     \prod: \]
Understanding the derivation of this identity is reasonable for
a bright student of first year undergraduate calculus in the
United States.

\end{document}
```

Figure 4

tion. Then proceed as with Linux. (It is possible to operate natively, but the author has not done so since 2002, and no native MS-Windows user has offered to update the native MS-Windows batch files.)

5.2 Procedure

Let's package the preceding Weierstrass product markup segment as a tiny article. Because it is GELLMU, not LaTeX, it begins with

`\documenttype{article}`

rather than with

`\documentclass{article}` .

See figure 4.

Beyond early-stage syntactic processing the system requires that there be a *title* in the preamble of every *article*. An empty *title* is allowed.

Text normally must be in paragraphs. (There are exceptions.) Therefore, the blank line after `\begin{document}` is essential.

It is sometimes said about LaTeX that a blank line *ends* a paragraph. However, in GELLMU a blank line *begins* a paragraph.

We place the tiny article text in a file named `gammabit.glm`, with `.glm` the canonical suffix for a GELLMU source file, enter the command

```
mmkg gammabit
```

and prepare to read the scroll. At the end when all goes well there are the following outputs:

- XHTML + MathML — the best online version
- PDF — for widely distributable print
- DVI — for TeXies
- classical HTML — for challenged browsing

Additionally one might note that some level of rendering based on *cascading style sheets* (CSS) is possible for the author-level XML.

In order to understand the scroll one needs to understand the system design.

6 System components

Regular GELLMU is a system assembled from modular components. Each step along the way produces an intermediate stage output that has its own sense and that, when things go wrong, provides opportunity both for diagnosis and intervention. A flow chart for regular GELLMU is found in figure 5.

In figure 5 what I call the "side door" is the second row entry showing the possibility of translation from source languages other than the markup of regular GELLMU into the author-level XML document type corresponding to the source markup of regular GELLMU.

One will note from the scroll that SGML/XML validation is done at several stages. This validation can be important for catching the author's mistakes. When there are error messages, it is possible and important to consult the scroll's last message regarding the stage of processing. Note in this regard that the translation from elaborated XML to XHTML + MathML takes place in three stages that are not shown on the chart but that may be seen in the example scroll.

7 Further information

Much more information may be found in the *User Guide* [3], the *GELLMU Manual* [4], and the web site for GELLMU,

`http://www.albany.edu/~hammond/gellmu/` .

A link to an online version of this document with live links should be available at the GELLMU web site for several years.

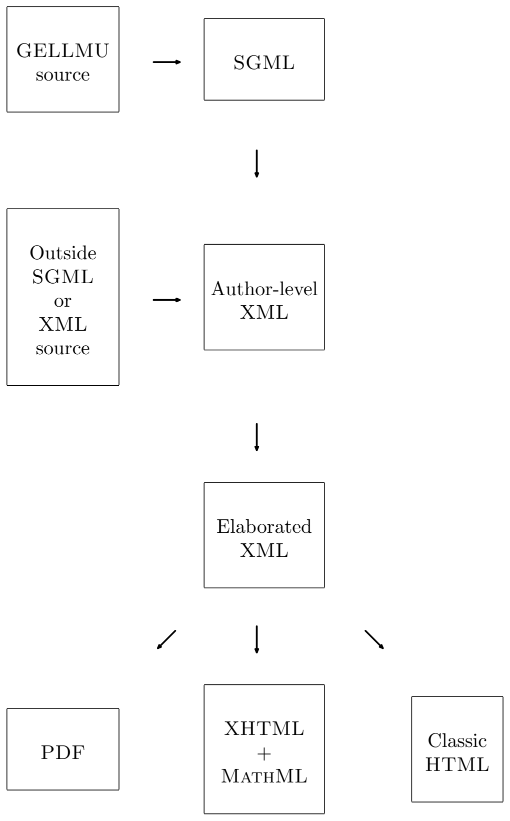


Figure 5

References

[1] William F. Hammond, "GELLMU: A Bridge for Authors from LaTeX to XML", *TUGboat: The Communications of the TeX Users Group*, vol. 22 (2001), pp. 204–207; also available online at `http://www.tug.org/TUGboat/Contents/contents22-3.html`.

[2] GELLMU at CTAN: `http://www.tex.ac.uk/tex-archive/help/Catalogue/entries/gellmu.html`

[3] William F. Hammond, "Introductory User's Guide to Regular GELLMU", `http://www.albany.edu/~hammond/gellmu/igl/userdoc.xhtml` (parallel PDF).

[4] William F. Hammond, "The GELLMU Manual", `http://www.albany.edu/~hammond/gellmu/glman/glman.xhtml` (parallel PDF).

[5] "New York Journal of Mathematics Articles in Mathematically-Capable HTML"; demonstration versions of past articles from *The New York Journal of Mathematics* ported from classical LaTeX using GELLMU, `http://math.albany.edu/demos/nyj/`.

[6] The GELLMU web site: `http://www.albany.edu/~hammond/gellmu/`

LuaTEX

Taco Hoekwater
`http://luatex.org`

Abstract

LuaTEX is an extended version of TEX that uses Lua as an embedded scripting language. The main objective of the LuaTEX project is to provide an open and configurable variant of TEX while at the same time offering downward compatibility. This paper gives a broad overview of the project.

1 Introduction

The LuaTEX source code of course includes Lua and the latest versions of pdfTEX and Aleph, but it also contains a few other more or less distinguishable parts:

- Some specialized TEX extensions
- A set of Lua module libraries
- The font reading code from FontForge
- Some of the font writing code from xdvipdfmx
- C source code to glue all of this together

If `\pdfoutput` is not set, LuaTEX is a lot like Aleph with additional support for the microtypography pdfTEX is known for. And if `\pdfoutput` is set, then it is like pdfTEX with the much better directionality support provided by Aleph.

If needed, Lua code is used to apply input reencoding, instead of I/O translation OTPs (Aleph) or tcx files (pdfTEX). Also, some experimental features of both programs were removed since the original problems can be better dealt with using Lua.

2 Time-line

The first informal start of LuaTEX was around TUG 2005. After an initial period of playing around and experimenting, the project gained momentum in the spring of 2006, when funding from Colorado State University and TUG (via the Oriental TEX Project) was acquired.

Soon after that, a public repository was set up and a mailing list and website were started. After a year of continuous work, the first beta was released at the TUG 2007 conference in San Diego, USA. The offered functionality is not completely finalized yet, so the interfaces are likely to change a bit still. A stable release is planned for the next TUG conference, in Cork, Ireland, 2008.

3 Features

The new functionality of LuaTEX falls into a few broad categories that are explained briefly in the next paragraphs.

3.1 Unicode support

LuaTEX uses UTF-8 encoded Unicode throughout the system. That means input and output files are Unicode, and also that the hyphenation patterns are expected to express hyphenation points of Unicode characters (instead of the traditional font glyphs).

Commands like `\char` and `\catcode` are extended to accept the full Unicode range, and used fonts can (but do not have to) be Unicode encoded.

3.2 TEX extensions

In the process of extending TEX for Unicode support and the cleanup required for interfacing to Lua code, some other extensions were also added. Here we briefly describe the most interesting of these.

The startup processing is altered to allow the document (via a Lua script) to have access to the command line.

A new feature called '`\catcode` tables' allows switching of all category codes in a single statement.

A new set of registers called `attribute` is added. Attributes can be used as extra counter values, but their usefulness comes mostly from the fact that all the 'set' attributes are automatically attached to all typesetting nodes created within their active scope. These node attributes can then be queried from any Lua code that deals with node processing.

The single internal memory heap that traditional TEX uses for tokens and nodes is split into two separate arrays, and each of these will grow dynamically when needed. The same is true for the input line buffer and the string pool size. All font memory is allocated on a per-font basis. Some less important arrays are still statically allocated, but eventually all memory allocation will become dynamic.

There is no separate pool file any more; all strings from that file are embedded during the final phase of the compilation of the `luatex` executable.

The format files are passed through zlib, allowing them to shrink to roughly half of the size they would have had in uncompressed form.

3.3 Extended font subsystem

The font system of LuaTEX is totally configurable through optional Lua code. If you do nothing, LuaTEX handles both TEX (TFM) and Omega (OFM) fonts as well as the related virtual font formats.

With some user-supplied Lua code, LuaTEX can also happily use OpenType and TrueType fonts. In addition, it is possible to build up encodings and virtual fonts totally in-memory.

The handling of 'virtualness' takes place at a different level in LuaTEX, meaning every single character can be either virtual or real, instead of this being handled at the font level. Inside a virtual character, it is possible to use arbitrary typeset data as 'character contents'.

3.4 Lua execution

Execution of Lua code within a document is handled by two new primitives: the expandable `\directlua` command, and the non-expandable `\latelua` command. The latter creates a node with Lua code that will be executed inside the `\output` routine, just like the traditional `\write`.

There can be more than one Lua interpreter state active at the same time. Some modules that are normally external to Lua are statically linked in with LuaTEX, because they offer useful functionality. This is one of the areas where future change is likely, but at the moment the list comprises:

- slnunicode, from the Selene libraries, `luaforge.net/projects/sln` (v1.1)
- luazip, from the kepler project, `www.keplerproject.org/luazip/` (v1.2.1)
- luafilesystem, also from the kepler project, `www.keplerproject.org/luafilesystem/` (v1.2)
- lpeg, by Roberto Ierusalimschy, `www.inf.puc-rio.br/~roberto/lpeg.html` (v0.6)
- lzlib, by Tiago Dionizio, `mega.ist.utl.pt/~tngd/lua/` (v0.2)
- md5, by Roberto Ierusalimschy, `www.inf.puc-rio.br/~roberto/md5/md5-5/md5.html`
- fontforge, a partial binding to the FontForge font editor by George Williams, `fontforge.sf.net`

It is also possible to make LuaTEX behave like the standalone Lua interpreter or the Lua bytecode compiler.

3.5 Lua interface libraries

LuaTEX would not be very useful if the Lua code did not have a way to communicate with the TEX internals. For this purpose, a set of Lua modules is defined:

- tex (general TEX access)
- pdf (routines related to pdf output)
- lua (lua bytecode registers)
- texio (writing to the log and terminal)
- font (accessing font internals)
- status (LuaTEX status information)
- kpse (file searching)
- callback (setting up callback hooks)
- token (handling TEX tokens)
- node (handling typeset nodes)

3.6 Callbacks

A callback is a hook into the internal processing of LuaTEX. Using callbacks, you can make LuaTEX run a Lua function you have defined instead of (or on top of) a bit of the core functionality.

It is easiest to think of it this way: callbacks offer a way to define something equivalent to compiled executable code. They have no connection to the TEX input language at all. Because of this they are very different from the argument to `\directlua`.

There are a few dozen callback hooks already defined, with many more to come later. There are callbacks for a wide variety of tasks, for instance: finding files, reading and preprocessing textual input, defining fonts, token creation, node list handling, and information display.

Here is a short example of defining a callback:

```
\directlua0{
function read_tfm (name)
 archive = zip.open('texmf-fonts.zip')
 if archive then
   tfmfile = archive:open(name .. '.tfm')
   if tfmfile then
     data = tfmfile:read('*all')
     return true, data, \string#data
   end
 end
 return false, nil, 0
end
callback.register('read_font_file',read_tfm)
}
```

4 Contact

The LuaTEX project is currently run by:

- Hans Hagen (general overview and website)
- Hartmut Henkel (pdf backend)
- Taco Hoekwater (coding and manual)

With help from:

- Arthur Reutenauer (binaries and testing)
- Martin Schröder (release support)

ConTEXt MkIV: Going UTF

Hans Hagen
`http://pragma-ade.com`
`http://luatex.org`

1 Introduction

In this document I will keep track of the transition of ConTEXt from MkII to MkIV, the latter being the Lua aware version.

The development of LuaTEX started with a few email exchanges between me and Hartmut Henkel. I had played a bit with Lua in Scite and somehow felt that it would fit into TEX quite well. Hartmut made me a version of pdfTEX which provided a `\lua` command. After exploring this road a bit Taco Hoekwater took over and we quickly reached a point where the pdfTEX development team could agree on following this road to the future.

The development was boosted by a substantial grant from Colorado State University in the context of the Oriental TEX Project of Idris Samawi Hamid. This project aims at bringing features into TEX that will permit ConTEXt to do high quality Arabic typesetting. Due to this grant Taco could spent substantial time on development, which in turn meant that I could start playing with more advanced features.

The full MkIV document is not so much a users manual as a history of the development. Consider it a collection of articles, and some chapters — like this one — have indeed ended up in the journals of user groups. Things may evolve and the way things are done may change, but it felt right to keep track of the process this way. Keep in mind that some features may have changed while LuaTEX matured.

Just for the record: development in the LuaTEX project is done by Taco Hoekwater, Hartmut Henkel and Hans Hagen. Eventually, the stable versions will become pdfTEX version 2 and other members of the pdfTEX team will be involved in development and maintenance. In order to prevent problems due to new and maybe even slightly incompatible features, pdfTEX version 1 will be kept around as well, but no fundamentally new features will be added to it. For practical reasons we use LuaTEX as the name of the development version but also for pdfTEX 2. That way we can use both engines side by side.

Editor's note: This is the introduction and one chapter of the full *ConTEXt MkIV* document. See the 'Random comments' column on p. 377 for more publication information.

This document is also one of our test cases. Here we use traditional TEX fonts (for math), Type 1 and OpenType fonts. We use color and include test code. Taco and I always test new versions of LuaTEX (the program) and MkIV (the macros and Lua code) with this document before a new version is released. Keep tuned . . .

2 Going UTF

LuaTEX only understands input codes in the Universal Character Set Transformation Format, aka UCS Transformation Format, better known as: UTF. There is a good reason for this universal view on characters: whatever support gets hard coded into the programs, it's never enough, as 25 years of TEX history have clearly demonstrated. Macro packages often support more or less standard input encodings, as well as local standards, user adapted ones, etc.

There is enough information on the Internet and in books about what exactly is UTF. If you don't know the details yet: UTF is a multi-byte encoding. The characters with a bytecode up to 127 map onto their normal ASCII representation. A larger number indicates that the following bytes are part of the character code. Up to 4 bytes make an UTF-8 code, while UTF-16 always uses two pairs of bytes.

byte1	byte2	byte3	byte4	Unicode
192–223	128–191			0x80–0x7FF
224–239	128–191	128–191		0x800–0xFFFF
240–247	128–191	128–191	128–191	0x10000–0x1FFFF

In UTF-8 the characters in the range 128–191 are illegal as first characters. The characters 254 and 255 are completely illegal and should not appear at all since they are related to UTF-16.

Instead of providing a never-complete truckload of other input formats, LuaTEX sticks to one input encoding but at the same time provides hooks that permits users to write filters that preprocess their input into UTF.

While writing the LuaTEX code as well as the ConTEXt input handling, we experimented a lot. Right from the beginning we had a pretty clear picture of what we wanted to achieve and how it could be done, but in the end arrived at solutions that permitted fast and efficient Lua scripting as well as a simple interface.

What is involved in handling any input encoding and especially UTF? First of all, we wanted to support UTF-8 as well as UTF-16. LuaTEX implements UTF-8 rather straightforwardly: it just assumes that the input is usable UTF. This means that it does not combine characters. There is a good reason for this: any automation needs to be configurable (on/off) and the more is done in the core, the slower it gets.

In Unicode, when a character is followed by an 'accent', the standard may prescribe that these two characters are replaced by one. Of course, when characters turn into glyphs, and when no matching glyph is present, we may need to decompose any character into components and paste them together from glyphs in fonts. Therefore, as a first step, a collapser was written. In the (pre)loaded Lua tables we have stored information about what combination of characters need to be combined into another character.

So, an `a` followed by an `‘` becomes `à` and an `e` followed by `"` becomes `ë`. This process is repeated till no more sequences combine. After a few alternatives we arrived at a solution that is acceptably fast: mere milliseconds per average page. Experiments demonstrated that we can not gain much by implementing this in pure C, but we did gain some speed by using a dedicated loop-over-utf-string function.

A second UTF related issue is UTF-16. This coding scheme comes in two endian variants. We wanted to do the conversion in Lua, but decided to play a bit with a multi-byte file read function. After some experiments we quickly learned that hard coding such methods in TEX was doomed to be complex, and the whole idea behind LuaTEX is to make things less complex. The complexity has to do with the fact that we need some control over the different linebreak triggers, that is, (combinations of) character 10 and/or 13. In the end, the multi-byte readers were removed from the code and we ended up with a pure Lua solution, which could be sped up by using a multi-byte loop-over-string function.

Instead of hard coding solutions in LuaTEX a couple of fast loop-over-string functions were added to the Lua string function repertoire and the solutions were coded in Lua. We did extensive timing with huge UTF-16 encoded files, and are confident that fast solutions can be found. Keep in mind that reading files is never the bottleneck anyway. The only drawback of an efficient UTF-16 reader is that the file is loaded into memory, but this is hardly a problem.

Concerning arbitrary input encodings, we can be brief. It's rather easy to loop over a string and replace characters in the 0–255 range by their UTF counterparts. All one needs is to maintain conversion tables and TEX macro packages have always done that.

Yet another (more obscure) kind of remapping concerns those special TEX characters. If we use a traditional TEX auxiliary file, then we must make sure that for instance percent signs, hashes, dollars and other characters are handled right. If we set the catcode of the percent sign to 'letter', then we get into trouble when such a percent sign ends up in the table of contents and is read in under a different catcode regime (and becomes for instance a comment symbol). One way to deal with such situations is to temporarily move the problematic characters into a private Unicode area and deal with them accordingly. In that case they no longer can interfere.

Where do we handle such conversions? There are two places where we can hook converters into the input.

1. each time when we read a line from a file, i.e. we can hook conversion code into the read callbacks
2. using the special `process_input_buffer` callback which is called whenever TEX needs a new line of input

Because we can overload the standard file open and read functions, we can easily hook the UTF collapse function into the readers. The same is true for the UTF-16 handler. In ConTEXt, for performance reasons we load such files into memory, which means that we also need to provide a special reader to TEX. When handling UTF-16, we don't need to combine characters so that stage is skipped then.

So, to summarize this, here is what we do in ConTEXt. Keep in mind that we overload the standard input methods and therefore have complete control over how LuaTEX locates and opens files.

1. When we have a UTF file, we will read from that file line by line, and combine characters when collapsing is enabled.
2. When LuaTEX wants to open a file, we look into the first bytes to see if it is a UTF-16 file, in either big or little endian format. When this is the case, we load the file into memory, convert the data to UTF-8, identify lines, and provide a reader that will give back the file linewise.
3. When we have been told to recode the input (i.e. when we have enabled an input regime) we

use the normal line-by-line reader and convert those lines on the fly into valid UTF. No collapsing is needed.

Because we conduct our experiments in ConTEXt MkIV the code that we provide may look a bit messy and more complex than the previous description may suggest. But keep in mind that a mature macro package needs to adapt to what users are accustomed to. The fact that LuaTEX moved on to UTF input does not mean that all the tools that users use and the files that they have produced over decades automagically convert as well.

Because we are now living in a UTF world, we need to keep that in mind when we do tricky things with sequences of characters, for instance in processing verbatim. When we implement verbatim in pure TEX we can do as before, but when we let Lua kick in, we need to use string methods that are UTF-aware. In addition to the linked-in Unicode library, there are dedicated iterator functions added to the `string` namespace; think of:

```
for c in string.utfcharacters(str) do
    something_with(c)
end
```

Occasionally we need to output raw 8-bit code, for instance to DVI or PDF backends (specials and literals). Of course we could have cooked up a truckload of conversion functions for this, but during one of our travels to a TEX conference, we came up with the following trick.

We reserve the top 256 values of the Unicode range, starting at hexadecimal value 0x110000, for byte output. When writing to an output stream, that offset will be subtracted. So, 0x1100A9 is written out as hexadecimal byte value A9, which is the decimal value 169, which in the Latin 1 encoding is the slot for the copyright sign.

MPlib: MetaPost as a reusable component

Taco Hoekwater, Hans Hagen
http://tug.org/metapost

Abstract

This short article introduces MPlib, a project that will be started in the autumn of 2007. The goal of this project is to turn MetaPost into a modern, re-entrant system library that can be used by many different applications programs at the same time.

1 The MetaPost workflow

Probably the most common use of MetaPost today is as a batch drawing program to create graphics that are then included inside a pdfTEX document. It is even quite normal for those graphics to be included inside the TEX source, with the MetaPost input file created on the fly by macro processing: for LATEX, this functionality is provided by the packages `emp`, `feynmf`, and `mfpic`; in ConTEXt, extensive in-line MetaPost support is built into the core engine.) In that case, MetaPost is often run on the fly by means of Web2C's `\write18` TEX extension.

In figure 1 you will see a typical flowchart of that process, and you will notice that it is a fairly complex affair.

The left column is the process that is immediately visible to the user: you run pdfTEX on a `.tex` file, and it generates a `.pdf` file.

The next column shows the execution of MetaPost, along with the required pre-processing (the TEX macro package has to create a temporary input file for MetaPost) and the post-processing (converting MetaPost's output from EPS to PDF format). In the figure, `mptopdf` is represented as a single program for the sake of simplicity. Various solutions exist for this, and the most common one is based on a set of TEX macros that ship with the ConTEXt distribution. These macros can be executed via a standalone program, or (most often) as a macro package that is included by the main TEX document.

The whole right-hand side of the flowchart is taken up by `makempx`, the program that handles TEX-based labels inside images. The MetaPost executable calls `makempx` automatically when it discovers that there are TEX-based labels in the document. `makempx` itself is just a dispatcher: it runs the separate program `mpto` to extract those labels from the MetaPost input file and place them in a TEX file, then it runs TEX on that file, and finally it runs `dvitomp` to convert the DVI file back into low-level drawing routines that MetaPost understands.

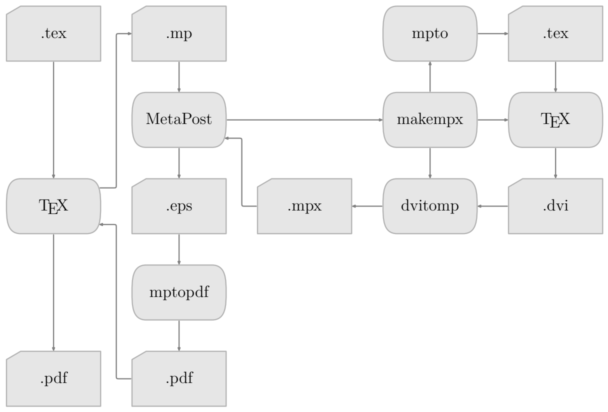


Figure 1: A typical workflow for MetaPost images inside a PDF document.

In this workflow, there are half a dozen programs called and the same number of intermediate files created.

2 Rationale

From looking at the workflow figure, it should be clear that all of this is not very efficient. In particular, the whole `makempx` block is wasteful of system resources, especially when MetaPost is executed on-the-fly.

It would be much nicer if MetaPost behaved like other system library components such as XML parsers and OpenGL engines: just link your application to the library, and there should be no more need for all those intermediate files and external programs.

Unfortunately, updating the label handling and creating system integration requires massive changes to the source code as well as the build system, and therefore it was very unlikely that this would ever get done without extra incentives. A significant amount of time and effort has to be invested to fix those particular problems.

It was clear to us that, to get these tasks done within a reasonable time frame, at least some of the work would have to be done under an organized project umbrella, and that is why we started the MPlib project.

3 Goals

What we want is to convert MetaPost into a reusable component library that is fully re-entrant and whose functionality can be easily embedded into other programs. To reach this primary goal, MetaPost not only has to be converted to a form suitable for library use, but also a set of new components needs to be added: an indirection layer for input and output, a configurable system for strategies regarding error handling, and a re-engineered labeling system.

4 Implementation

Work will start in the autumn of this year, and it is our current estimate that the project will be complete by the summer of 2008. The actual programming will be carried out by Taco. Hans Hagen will lead the project, and Bogusław Jackowski will be in charge of quality control.

The current version of MetaPost is a mix of WEB (Pascal) and C code, that is compiled using a complex build system based on the Web2C Pascal converter. One of the implementation tasks for MPlib is to convert MetaPost into a more mainstream distribution package. For that, all of the source code will be converted into C, using either CWEB or NOWEB to retain the literate programming quality of MetaPost.

Some parts of the internals of MetaPost will be opened up and a documented application interface will be offered. Besides a MetaPost-compatible standalone executable based on MPlib, a Lua language binding to the library will be provided. This binding will allow the immediate use of MPlib within LuaTEX, as well as function as an example for other language bindings.

5 Acknowledgments

This project is supported by the worldwide TEX User Groups. In particular, we want thank the following user groups that have already promised financial and other support: DANTE e.V., TUGIndia, TUG, NTG, CSTUG, and GUST.

Extending TEX and METAFONT with floating-point arithmetic

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
WWW URL: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 581 4148
Internet: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org

Abstract

The article surveys the state of arithmetic in TEX and METAFONT, suggests that they could usefully be extended to support floating-point arithmetic, and shows how this could be done with a relatively small effort, *without* loss of the important feature of platform-independent results from those programs, and *without* invalidating any existing documents, or software written for those programs, including output drivers.

Contents

1 Dedication

This article is dedicated to Professors Donald Knuth (Stanford University) and William Kahan (University of California, Berkeley), with thanks for their many scientific and technical contributions, and for their writing.

2 Introduction

Arithmetic is a fundamental feature of computer programming languages, and for some of us, the more we use computers, the more inadequate we find their computational facilities. The arithmetic in TEX and METAFONT is particularly limiting, and this article explains why this is so, why it was not otherwise, and what can be done about it now that these two important programs are in their thirtieth year of use.

3 Arithmetic in TEX and METAFONT

Before we look at issues of arithmetic in general, it is useful to summarize what kinds of numbers TEX and METAFONT can handle, and how they do so.

TEX provides binary integer and fixed-point arithmetic. Integer arithmetic is used to count things, such as with TEX's `\count` registers. Fixed-point arithmetic is needed for values that have fractional parts, such as the `\dimen` dimension registers, the `\muskip` and `\skip` glue registers, and scale factors, as in `0.6\hsize`.

For portability reasons, TEX requires that the host computer support an integer data type of at least 32 bits. It uses that type for the integer arithmetic available to TEX programs. For fixed-point numbers, it reserves the lower 16 bits for the fractional part, and all but two of the remaining bits for the integer part. Thus, on the 32-bit processors that are commonly found in personal computers, 14 bits are available for the integer part. One of the remaining two bits is chosen as a sign bit, and the other is used to detect *overflow*, that is, generation of a number that is too large to represent.

When fractional numbers represent TeX dimensions, the low-order fraction bit represents the value 2^{-16} pt. While printer's points have been a common unit of measurement since well before the advent of computer-based typesetting, this tiny value is new with TeX, and has the special name *scaled point*. The value 1 sp is so small that approximately 100 sp is about the wavelength of visible light. This ensures that differences of a few scaled points in the positioning of objects on the printed page are completely invisible to human eyes.

The problem with fixed-point numbers in TeX is at the other end: 14 integer bits can only represent numbers up to 16383. As a dimension, that many points is about 5.75 m, which is probably adequate for printed documents, but is marginal if you are typesetting a billboard. The PDP-10 computers on which TeX and METAFONT were developed had 36-bit words: the four extra bits raised the maximum dimension by a factor of 16. Nevertheless, if TeX's fixed-point numbers are used for purposes other than page dimensions, then it is easy to exceed their limits.

TeX is a macro-extensible language for typesetting, and arithmetic is expected to be relatively rare. TeX has little support for numerical expressions, just verbose low-level operators, forcing the TeX programmer to write code such as this fragment from `layout.tex` to accomplish the multiply-add operation noted in the comment:

```
% MRGNOTEYA = 0.75*TEXTHEIGHT + FOOTSKIP
\T = \TEXTHEIGHT
\multiply \T by 75  % possible overflow!
\divide \T by 100
\advance \T by \FOOTSKIP
\xdef \MRGNOTEYA {\the \T}
```

Notice that the scale factor 0.75 could have been reduced from 75/100 to 3/4 in this example, but that is not in general possible. Similarly, here we could have written `\T = 0.75 \TEXTHEIGHT`, but that is not possible if the constant 0.75 is replaced by a variable in a register. The multiplication by 75 can easily provoke an overflow if `\T` is even as big as a finger length:

```
*\dimen1 = 220pt

*\dimen2 = 75\dimen1
! Dimension too large.

*\multiply \dimen1 by 75
! Arithmetic overflow.
```

See the LaTeX `calc` package for more horrors of fixed-point arithmetic.

TeX has, however, also seen use a scripting language, chosen primarily because of its superb quality, stability, reliability, and platform independence. TeX distributions now contain macro packages and utilities written in TeX for generating complex font tables, for packing and unpacking document archives, for scanning PostScript graphics files, and even for parsing SGML and XML.

TeX's arithmetic does not go beyond the four basic operations of *add*, *subtract*, *multiply*, and *divide*. In particular, no elementary functions (square root, exponential, logarithm, trigonometric and hyperbolic functions, and so on) are provided in TeX itself, even though, in principle, they can be provided with macro packages.

In TeX, overflow is detected in division and multiplication but not in addition and subtraction, as I described in my *TUG 2003* keynote address [4].

Input numbers in METAFONT are restricted to 12 integer bits, and the result of even trivial expressions can be quite surprising to users:

```
% mf expr
gimme an expr: 4095       >> 4095
gimme an expr: 4096
! Enormous number has been reduced.
>> 4095.99998
gimme an expr: infinity   >> 4095.99998
gimme an expr: epsilon    >> 0.00002
gimme an expr: 1/epsilon
! Arithmetic overflow.
>> 32767.99998
gimme an expr: 1/3        >> 0.33333
gimme an expr: 3*(1/3)    >> 0.99998
gimme an expr: 1.2 - 2.3  >> -1.1
gimme an expr: 1.2 - 2.4  >> -1.2
gimme an expr: 1.3 - 2.4  >> -1.09999
```

Notice that although 4096 is considered an overflow, internally METAFONT can generate a number almost eight times as large. Binary-to-decimal conversion issues produce the anomaly in 3 × (1/3). The last line shows that even apparently simple operations are not so simple after all.

Overflows in METAFONT can also produce a report like this:

```
Uh, oh.  A little while ago one of the
quantities that I was computing got
too large, so I'm afraid your answers
will be somewhat askew.  You'll
probably have to adopt different
tactics next time.  But I shall try to
carry on anyway.
```

METAFONT provides a few elementary functions: `++` (Pythagoras), `abs`, `angle`, `ceiling`, `cosd`, `dir`, `floor`, `length`, `mexp`, `mlog`, `normaldeviate`, `round`, `sind`, `sqrt`, and `uniformdeviate`. They prove useful in the geometric operations required in font design.

4 Historical remarks

TeX and METAFONT are not the only systems that suffer from the limitations of fixed-point arithmetic. Most early computers were inadequate as well:

> It is difficult today to appreciate that probably the biggest problem facing programmers in the early 1950s was scaling numbers so as to achieve acceptable precision from a fixed-point machine.
>
> Martin Campbell-Kelly
> *Programming the Mark I: Early Programming Activity at the University of Manchester*
> Annals of the History of Computing,
> **2**(2) 130–168 (1980)

Scaling problems can be made much less severe if numbers carry an exponent as well as integer and fractional parts. We then have:

> Floating Point Arithmetic ... The subject is not at all as trivial as most people think, and it involves a surprising amount of interesting information.
>
> Donald E. Knuth
> *The Art of Computer Programming: Seminumerical Algorithms* (1998)

However, more than just an exponent is needed; the arithmetic system also has to be predictable:

> Computer hardware designers can make their machines much more pleasant to use, for example by providing *floating-point arithmetic* which satisfies simple mathematical laws.
>
> The facilities presently available on most machines make the job of rigorous error analysis *hopelessly difficult*, but properly designed operations would encourage numerical analysts to provide better subroutines which have certified accuracy.
>
> Donald E. Knuth
> *Computer Programming as an Art*
> *ACM Turing Award Lecture* (1973)

5 Why no floating-point arithmetic?

Neither TeX nor METAFONT have floating-point arithmetic natively available, and as I discussed in my *Practical TeX 2005* keynote address [3], there is a very good reason why this is the case. Their output needs to be identical on all platforms, and when they were developed, there were many different computer vendors, some of which had several incompatible product lines. This diversity causes several problems, some of which still exist:

- There is system dependence in *precision*, *range*, *rounding*, *underflow*, and *overflow*.
- The number base varies from 2 on most, to 3 (Setun), 4 (Illiac II), 8 (Burroughs), 10, 16 (IBM S/360), 256 (Illiac III), and 10000 (Maple).
- Floating-point arithmetic exhibits bizarre behavior on some systems:
  - $x \times y \neq y \times x$ (early Crays);
  - $x \neq 1.0 \times x$ (Pr1me);
  - $x + x \neq 2 \times x$ (Pr1me);
  - $x \neq y$ but $1.0/(x - y)$ gets zero-divide error;
  - wrap between underflow and overflow (e.g., C on PDP-10);
  - job termination on overflow or zero-divide (most).
- No standardization: almost every vendor had one or more distinct floating-point systems.
- Programming language dependence on available precisions:
  - Algol, Pascal, and SAIL (only `real`): recall that SAIL was the implementation language for the 1977–78 prototypes of TeX and METAFONT;
  - Fortran (`REAL`, `DOUBLE PRECISION`, and on some systems, `REAL*10` or `REAL*16`);
  - C/C++ (originally only `double`, but `float` added in 1989, and `long double` in 1999);
  - C# and Java have only `float` and `double` data types, but their arithmetic is badly botched: see Kahan and Darcy's *How Java's Floating-Point Hurts Everyone Everywhere* [28].
- Compiler dependence: multiple precisions can be mapped to just one, without warning.
- BSD compilers on IA-32 still provide no 80-bit format after 27 years in hardware.
- Input/output problem requires base conversion, and is *hard* (e.g., conversion from 128-bit binary format can require more than 11500 decimal digits).
- Most languages do not guarantee exact base conversion.

Donald Knuth wrote an interesting article with the intriguing title *A simple program whose proof isn't* [29] about how TeX handles conversions between fixed-point binary and decimal. The restriction to fixed-point arithmetic with 16-bit fractional parts simplifies the base-conversion problem, and allows TeX to guarantee exact round-trip conversions of such numbers.

TeX produces the same line- and page-breaking across all platforms, because floating-point arithmetic is used only for interword glue calculations that could change the horizontal position of a letter by at most a few scaled points, but as we noted earlier, that is invisible.

METAFONT has no floating-point at all, and generates identical fonts on all systems.

6 IEEE 754 binary floating-point standard

With the leadership of William Kahan, a group of researchers in academic, government, and industry began a collaborative effort in the mid-1970s to design a new and much improved floating-point architecture. The history of this project is chronicled in an interview with Kahan [37, 38].

A preliminary version of this design was first implemented in the Intel 8087 chip in 1980, although the design was not finalized until its publication as IEEE Standard 754 in 1985 [23].

Entire books have beeen written about floating-point arithmetic: see, for example, Sterbenz [41] for historical systems, Overton [35] for modern ones, Omondi [34] and Parhami [36] for hardware, Goldberg [15, 17] for an excellent tutorial, and Knuth [30, Chap. 4] for theory. I am hard at work on writing two more books in this area. However, here we need only summarize important features of the IEEE 754 system:

- Three formats are defined: 32-bit, 64-bit, and 80-bit. A 128-bit format was subsequently provided on some Alpha, IA-64, PA-RISC, and SPARC systems.
- Nonzero normal numbers are *rational*: $x = (-1)^s f \times 2^p$, where the *significand*, f, lies in $[1,2)$.
- Signed zero allows recording the direction from which an underflow occurred, and is particularly useful for arithmetic with complex (real + imaginary) numbers. The IEEE Standard requires that $\sqrt{-0}$ evaluate to -0.
- The largest stored exponent represents Infinity if $f = 0$, and either quiet or signaling NaN (Not-a-Number) if $f \neq 0$. A vendor-chosen significand bit distinguishes between the two kinds of NaN.
- The smallest stored exponent allows leading zeros in f for *gradual underflow* to *subnormal* values.
- The arithmetic supports a model of fast *nonstop* computing. Sticky flags record exceptions, and Infinity, NaN, and zero values automatically replace out-of-range values, without the need to invoke an exception handler, although that capability may also be available.
- Four rounding modes are provided:
  - *to nearest with ties to even* (default);
  - *to* $+\infty$;
  - *to* $-\infty$;
  - *to zero* (historical chopping).
- Values of $\pm\infty$ are generated from huge/tiny and finite/0.
- NaN values are generated from $0/0$, $\infty - \infty$, ∞/∞, and any operation with a NaN operand.
- A NaN is returned from functions when the result is undefined in real arithmetic (e.g., $\sqrt{-1}$), or when an argument is a NaN.
- NaNs have the property that they are unequal to anything, even themselves. Thus, the C-language inequality test `x != x` is true *if*, and *only if*, `x` is a NaN, and should be readily expressible in *any* programming language. Sadly, several compilers botch this, and get the wrong answer.

7 IEEE 754R precision and range

In any computer arithmetic system, it is essential to know the available range and precision. The precisions of the four IEEE 754 binary formats are equivalent to approximately 7, 15, 19, and 34 decimal digits. The approximate ranges as powers of ten, including subnormal numbers, are $[-45,38]$, $[-324,308]$, $[-4951,4932]$, and $[-4966,4932]$. A future 256-bit binary format will supply about 70 decimal digits, and powers-of-ten in $[-315\,723, 315\,652]$.

A forthcoming revision of the IEEE Standard will include decimal arithmetic as well, in 32-, 64-, and 128-bit storage sizes, and we can imagine a future 256-bit size. Their precisions are 7, 16, 34, and 70 decimal digits, where each doubling in size moves from n digits to $2n+2$ digits. Their ranges are wider than the binary formats, with powers of ten in $[-101,96]$, $[-398,384]$, $[-6176,6144]$, and $[-1\,572\,932, 1\,572\,864]$.

In each case, the range and precision are determined by the number of bits allocated for the sign and the significand, and for the decimal formats, by restrictions imposed by the compact encodings chosen for packing decimal digits into strings of bits.

It is highly desirable that each larger storage size increase the exponent range (many older designs did not), and at least double the significand length, since that guarantees that products evaluated in the next higher precision *cannot overflow*, and are *exact*. For example, the Euclidean distance $\sqrt{x^2 + y^2}$ is then trivial to compute; otherwise, its computation requires careful rescaling to avoid premature underflow and overflow.

8 Remarks on floating-point arithmetic

Contrary to popular misconception, even present in some books and compilers, floating-point arithmetic is *not fuzzy*:

- Results are *exact* if they are representable.
- Multiplication by a power of base is always exact, in the absence of underflow and overflow.
- Subtracting numbers of like signs and exponents is *exact*.

Bases other than 2 or 10 suffer from *wobbling precision* caused by the requirement that significands be normalized. For example, in hexadecimal arithmetic, $\pi/2 \approx 1.571 \approx 1.922_{16}$ has three fewer bits (almost one decimal digit) than $\pi/4 \approx 0.7854 \approx \text{c.910}_{16}$. Careful coders on such systems account for this in their programs by writing

```
y = (x + quarter_pi) + quarter_pi;
```

instead of

```
y = x + half_pi;
```

Because computer arithmetic systems have finite range and precision, they are not *associative*, so commonly-assumed mathematical transformations do not hold. In particular, it is often necessary to control evaluation order, and this may be at odds with what the compiler, or even a high-performance CPU with dynamic instruction reordering, does with the code.

The presence of multiple rounding modes also invalidates common assumptions. For example, the Taylor series for the sine function begins

$$\sin(x) = x - (1/3!)x^3 + (1/5!)x^5 - \cdots.$$

If x is small enough, because of finite precision, one might expect that $\sin(x)$ could be computed simply as x. However, that is only true for the default rounding mode; in other modes, the correct answer could be one *ulp* (unit in the last place) higher or lower, so at least two terms must be summed. Similarly, the mathematical equivalence $-(xy + z) \equiv (-xy - z)$ does not hold in some rounding modes. Except for some special numbers, it is not in general permissible to replace slow division with fast multiplication by the reciprocal, even though many optimizing compilers do that.

Some of the common elementary functions are *odd* ones: they satisfy $f(x) = -f(-x)$. This relation does not in general hold computationally if a rounding direction of other than *round-to-nearest* is in effect. Software designers are then forced to decide whether obeying computer rounding modes is more important than preserving fundamental mathematical symmetries: in well-designed software, symmetry wins. Nevertheless, in some applications, like *interval arithmetic*, which computes upper and lower bounds for every numeric operation, precise control of rounding is imperative, and overrides symmetry.

See Monniaux [33] for a recent discussion of some of the many problems of floating-point evaluation. A good part of the difficulties described there arise because of higher intermediate precision in the Intel IA-32 architecture, the most common desktop CPU family today. Other problems come from unexpected instruction reordering or multiple threads of execution, and the incidence of these issues increases with each new generation of modern processors.

9 Binary versus decimal

Why should we care whether a computer uses binary or decimal arithmetic? Here are some reasons why a switch to decimal arithmetic has advantages:

- Humans are less uncomfortable with decimal arithmetic.
- In some case, binary arithmetic always gets the wrong answer. Consider this sales tax computation: 5% of 0.70 = 0.0349999... in *all* binary precisions, instead of the exact decimal 0.035. Thus, there can be significant cumulative rounding errors in businesses with many small transactions (food, music downloading, telephone, ...).
- Financial computations need fixed-point decimal arithmetic.
- Hand calculators use decimal arithmetic.
- Additional decimal rounding rules (eight instead of four) handle the financial and legal requirements of some jurisdictions.
- Decimal arithmetic eliminates most base-conversion problems.
- There is a specification of decimal arithmetic subsumed in the *IEEE 854-1987 Standard for Radix-Independent Floating-Point Arithmetic* [21].
- Older Cobol standards require 18D fixed-point.
- Cobol 2002 requires 32D fixed-point *and* floating-point.
- Proposals to add decimal arithmetic to C and C++ were submitted to the ISO language committees in 2005 and 2006.
- Twenty-five years of Rexx and NetRexx scripting languages give valuable experience in arbitrary-precision decimal arithmetic.

- The excellent IBM `decNumber` library provides *open source* decimal floating-point arithmetic with a billion (10^9) digits of precision and exponent magnitudes up to 999 999 999.
- Preliminary support in `gcc` for +, −, ×, and / became available in late 2006, based on a subset of the IBM `decNumber` library.
- The author's `mathcw` package [5] provides a C99-compliant run-time library for binary, and also for decimal, arithmetic (2005–2008), with hundreds of additional functions, and important and useful extensions of the I/O functions.
- IBM zSeries mainframes got IEEE 754 binary floating-point arithmetic in 1999, and decimal floating-point arithmetic in firmware in 2006.
- The IBM PowerPC version 6 chips announced on 21 May 2007 add hardware decimal arithmetic, probably the first mainstream processor to do so in more than four decades.
- Hardware support seems likely in future Intel IA-32 and EM64T (x86_64) processors, and the current family members are among the most widely-used in the world for general-purpose computing. Other chip vendors will have to offer similar facilities to remain competitive.

10 Problems with IEEE 754 arithmetic

Despite the many benefits of IEEE 754 floating-point arithmetic, there are many impediments to its effective use:

- Language access to features has been slow: more than 27 years have passed since the Intel 8087, and we are still waiting!
- Programmer unfamiliarity, ignorance, and inexperience.
- A deficient educational system, both in academia, and in textbooks, leaves most programmers with little or no training in floating-point arithmetic.
- Partial implementations by some vendors deny access to important features (e.g., subnormals may flush to zero, IA-32 has only one NaN, IA-32 and IA-64 have imperfect rounding, Java and C# lack rounding modes and higher precisions).
- Long internal registers are generally beneficial, but also produce many computational surprises and double rounding [33], compromising portability.
- Rounding behavior at underflow and overflow limits is unspecified by the IEEE standards, and thus, is vendor dependent.
- Overeager, or incorrect, optimizations by compilers may produce wrong results, and prevent obtaining similar results across different platforms, or between different compilers on the same system, or even from the same compiler with different options.
- Despite decades of availability of IEEE 754 arithmetic, some compilers still mishandle signed zeros and NaNs, and it can be difficult to convince compiler vendors of the significance of such errors (I know, because I've tried, and failed).

11 How decimal arithmetic is different

Programmers in science and engineering have usually only had experience with binary floating-point arithmetic, and some relearning is needed for the move to decimal arithmetic:

- Nonzero normal floating-point numbers take the form $x = (-1)^s f \times 10^p$, where f is an *integer*, allowing simulation of fixed-point arithmetic.
- Lack of normalization means multiple storage forms, but 1., 1.0, 1.00, 1.000, … compare equal, as long as floating-point instructions, rather than bitwise integer comparisons, are used.
- *Quantization* is detectable (e.g., for financial computations, 1.00 differs from 1.000).
- Signed zero and infinity, plus quiet and signaling NaNs, are detectable from the first byte, whereas binary formats require examination of all bits.
- There are *eight* rounding modes because of legal and tax mandates.
- Compact storage formats — Densely-Packed Decimal (DPD) [IBM] and Binary-Integer Decimal (BID) [Intel] — need fewer than BCD's four bits per decimal digit.

12 Software floating-point arithmetic

It may be better in some applications to have floating-point arithmetic entirely in software, as Apple once did with the no-longer-supported SANE (Standard Apple Numerics Environment) system. Here are some reasons why:

- TeX and METAFONT must continue to guarantee identical results across platforms.
- Unspecified behavior of low-level arithmetic guarantees *platform dependence*.
- Floating-point arithmetic is not associative, so instruction ordering (e.g., compiler optimization) affects results.

- Long internal registers on some platforms, and not on others, alter precision, and results.
- Multiply-add computes $x \times y + z$ with *exact* product and single rounding, getting different result from separate operations.
- Conclusion: only a single *software* floating-point arithmetic system in TEX and METAFONT can guarantee *platform-independent results*.

Software is often best enhanced by connecting two or more systems with a clean and simple interface:

> What if you could provide a seamlessly integrated, fully dynamic language with a conventional syntax while increasing your application's size by less than 200K on an x86? You can do it with *Lua*!
>
> Keith Fieldhouse

If we want to have floating-point arithmetic in TEX and METAFONT, then rather than modify those stable and reliable programs, including adding convenient expression syntax, and a substantial function library, there is a cleaner, and easier, approach:

- There is no need to modify TEX beyond what has already been done: LuaTEX interfaces TEX to a clean and well-designed scripting language — we just need to change the arithmetic and library inside `lua`.
- Scripting languages usually offer a single floating-point datatype, typically equivalent to IEEE 754 64-bit `double` (that is all that the C language used to have).
- `qawk` and `dnawk` are existing extensions by the author of `awk` for 128-bit binary and decimal arithmetic, respectively.
- Modern machines are fast and memories are big. We could adopt a 34D 128-bit format, or better, a 70D 256-bit format, instead as default numeric type.
- The author's `mathcw` package [5] is a highly-portable open-source library with support for *ten* floating-point precisions, including 256-bit binary and decimal.

Two more quotes from the father of the IEEE 754 design lead into our next points:

> The convenient accessibility of double-precision in many Fortran and some Algol compilers indicates that double-precision will soon be universally acceptable as a substitute for ingenuity in the solution of numerical problems.
>
> W. Kahan
> *Further Remarks on Reducing Truncation Errors*
> Comm. ACM **8**(1) 40, January (1965)

> Nobody knows how much it would cost to compute y^w correctly rounded for every two floating-point arguments at which it does not over/underflow. Instead, reputable math libraries compute elementary transcendental functions mostly within slightly more than half an ulp and almost always well within one ulp. Why can't y^w be rounded within half an ulp like SQRT? Because nobody knows how much computation it would cost.... No general way exists to predict how many extra digits will have to be carried to compute a transcendental expression and round it correctly to some preassigned number of digits. Even the fact (if true) that a finite number of extra digits will ultimately suffice may be a deep theorem.
>
> W. Kahan
> *Wikipedia entry*

We need more than just the basic four operations of arithmetic: several dozen elementary functions, and I/O support, are essential.

- The *Table Maker's Dilemma* (Kahan) is the problem of always getting exactly-rounded results when computing the elementary functions. Here is an example of a hard case:
  `log(+0x1.ac50b409c8aeep+8) =`
  `0x60f52f37aecfcffffffffffffffeb...p-200`
  There are 62 consecutive 1-bits in that number, and at least $4 \times 13 + 62 + 1 = 115$ bits must be computed correctly in order to determine the correctly-rounded 53-bit result.

- Higher-than-needed-precision arithmetic provides a practical solution to the dilemma, as the Kahan quote observes.

- Random-number generation is a common portability problem, since algorithms for that computation are platform-dependent and vary in quality. Fortunately, several good ones are now known, and can be supplied in libraries, although careful attention still needs to be given to computer wordsize.

- The `mathcw` library gives *platform-independent* results for decimal floating-point arithmetic, since evaluation order is completely under programmer control, and identical everywhere, and the underlying decimal arithmetic is too.

13 How much work is needed?

I argue that decimal floating-point arithmetic in software, isolated in a separate scripting language, is an effective and reasonable way to extend TEX and METAFONT so that they can have access to floating-point arithmetic, and remove the limitations and nuisance of fixed-point arithmetic that they currently suffer.

It is therefore appropriate to ask what kind of effort would be needed to do this. In four separate experiments with three implementations of awk, and one of lua, I took two to four hours each, with less than 3% of the code requiring changes:

Program	Lines	Deleted	Added
dgawk	40 717	109	165
dlua	16 882	25	94
dmawk	16 275	73	386
dnawk	9 478	182	296
METAFONT in C	30 190	0	0
TEX in C	25 215	0	0

14 Summary

Had the IEEE 754 design been developed before TEX and METAFONT, it is possible that Donald Knuth would have chosen a software implementation of binary floating-point arithmetic, as he later provided for the MMIX virtual machine [2, 31, 32] that underlies the software analyses in newer editions of his famous book series, *The Art of Computer Programming*.

That did not happen, so in this article, I have shown how a different approach might introduce *decimal* floating-point arithmetic to TEX and METAFONT through a suitable scripting language, for which Lua [26, 25, 27] seems eminently suited, and has already been interfaced to TEX and is now in limited use for production commercial typesetting, and also for document style-file design. By selecting high working precision, at least 34 decimal digits and preferably 70, many numerical issues that otherwise compromise portability and reproducibility of typeset documents simply disappear, or at least, become highly improbable.

To make this workable, the compilers, the basic software arithmetic library, the elementary function library, and the I/O library need to be highly portable. The combination of the GNU gcc compiler family with the IBM decNumber library and the author's mathcw library satisfy all of these requirements. Within a year or two, we may therefore expect that decimal floating-point arithmetic in C could be available on all of the common platforms, allowing future TEX Live releases to build upon that foundation, and LuaTEX could become the TEX version of choice in many environments. LuaMETAFONT and LuaMETAPOST could soon follow.

References

[1] P. H. Abbott, D. G. Brush, C. W. Clark III, C. J. Crone, J. R. Ehrman, G. W. Ewart, C. A. Goodrich, M. Hack, J. S. Kapernick, B. J. Minchau, W. C. Shepard, R. M. Smith, Sr., R. Tallman, S. Walkowiak, A. Watanabe, and W. R. White. Architecture and software support in IBM S/390 Parallel Enterprise Servers for IEEE floating-point arithmetic. *IBM Journal of Research and Development*, 43(5/6):723–760, 1999. ISSN 0018-8646. URL http://www.research.ibm.com/journal/rd/435/abbott.html. Besides important history of the development of the S/360 floating-point architecture, this paper has a good description of IBM's algorithm for exact decimal-to-binary conversion, complementing earlier ones [39, 7, 29, 6, 40].

[2] Heidi Anlauff, Axel Böttcher, and Martin Ruckert. *Das MMIX-Buch: ein praxisnaher Zugang zur Informatik. (German) [The MMIX Book: A practical introduction to computer science]*. Springer-Lehrbuch. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2002. ISBN 3-540-42408-3. xiv + 327 pp. EUR 24.95. URL http://www.informatik.fh-muenchen.de/~mmix/MMIXBuch/.

[3] Nelson Beebe. The design of TEX and METAFONT: A retrospective. *TUGboat*, 26(1):33–41, 2005. ISSN 0896-3207.

[4] Nelson H. F. Beebe. 25 Years of TEX and METAFONT: Looking back and looking forward — TUG 2003 keynote address. *TUGboat*, 25(1):7–30, 2004. ISSN 0896-3207.

[5] Nelson H. F. Beebe. *The mathcw Portable Elementary Function Library*. 2008. In preparation.

[6] Robert G. Burger and R. Kent Dybvig. Printing floating-point numbers quickly and accurately. *ACM SIGPLAN Notices*, 31(5):108–116, May 1996. ISSN 0362-1340. URL http://www.acm.org:80/pubs/citations/proceedings/pldi/231379/p108-burger/. This paper offers a significantly faster algorithm than that of [39], together with a correctness proof and an implementation in Scheme. See also [7, 1, 40, 8].

[7] William D. Clinger. How to read floating point numbers accurately. *ACM SIGPLAN Notices*, 25(6):92–101, June 1990. ISBN 0-89791-364-7. ISSN 0362-1340. URL http://www.acm.org:80/pubs/citations/proceedings/pldi/93542/p92-clinger/. See also output algorithms in [29, 39, 6, 1, 40].

[8] William D. Clinger. Retrospective: How to read floating point numbers accurately. *ACM SIGPLAN Notices*, 39(4):360–371, April 2004. ISSN 0362-1340. Best of PLDI 1979–1999. Reprint of, and retrospective on, [7].

[9] William J. Cody, Jr. Analysis of proposals for the floating-point standard. *Computer*, 14(3):63–69, March 1981. ISSN 0018-9162. See [23, 24].

[10] Jerome T. Coonen. An implementation guide to a proposed standard for floating-point arithmetic. *Computer*, 13(1):68–79, January 1980. ISSN 0018-9162. See errata in [11]. See [23, 24].

[11] Jerome T. Coonen. Errata: An implementation guide to a proposed standard for floating point arithmetic. *Computer*, 14(3):62, March 1981. ISSN 0018-9162. See [10, 23, 24].

[12] Jerome T. Coonen. Underflow and the denormalized numbers. *Computer*, 14(3):75–87, March 1981. ISSN 0018-9162. See [23, 24].

[13] Charles B. Dunham. Surveyor's Forum: "What every computer scientist should know about floating-point arithmetic". *ACM Computing Surveys*, 24(3):319, September 1992. ISSN 0360-0300. See [15, 16, 45].

[14] W. H. J. Feijen, A. J. M. van Gasteren, D. Gries, and J. Misra, editors. *Beauty is our business: a birthday salute to Edsger W. Dijkstra*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1990. ISBN 0-387-97299-4. xix + 453 pp. LCCN QA76 .B326 1990.

[15] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991. ISSN 0360-0300. URL `http://www.acm.org/pubs/toc/Abstracts/0360-0300/103163.html`. See also [16, 13, 45].

[16] David Goldberg. Corrigendum: "What every computer scientist should know about floating-point arithmetic". *ACM Computing Surveys*, 23(3):413, September 1991. ISSN 0360-0300. See [15, 13, 45].

[17] David Goldberg. Computer arithmetic. In *Computer Architecture—A Quantitative Approach*, chapter H, pages H–1–H–74. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, third edition, 2002. ISBN 1-55860-596-7. LCCN QA76.9.A73 P377 2003. US$89.95. URL `http://books.elsevier.com/companions/1558605967/appendices/1558605967-appendix-h.pdf`. The complete Appendix H is not in the printed book; it is available only at the book's Web site: `http://www.mkp.com/CA3`.

[18] David Gries. Binary to decimal, one more time. In Feijen et al. [14], chapter 16, pages 141–148. ISBN 0-387-97299-4. LCCN QA76 .B326 1990. This paper presents an alternate proof of Knuth's algorithm [29] for conversion between decimal and fixed-point binary numbers.

[19] David Hough. Applications of the proposed IEEE-754 standard for floating point arithmetic. *Computer*, 14 (3):70–74, March 1981. ISSN 0018-9162. See [23, 24].

[20] IEEE. IEEE standard for binary floating-point arithmetic. *ACM SIGPLAN Notices*, 22(2):9–25, February 1985. ISSN 0362-1340. See [23].

[21] IEEE. *854-1987 (R1994) IEEE Standard for Radix-Independent Floating-Point Arithmetic*. IEEE, New York, NY, USA, 1987. ISBN 1-55937-859-X. 16 pp. US$44.00. URL `http://standards.ieee.org/reading/ieee/std_public/description/busarch/854-1987_desc.html`. Revised 1994.

[22] IEEE Computer Society Standards Committee. Working group of the Microprocessor Standards Subcommittee and American National Standards Institute. *IEEE standard for binary floating-point arithmetic*. ANSI/IEEE Std 754-1985. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1985. 18 pp. See [23].

[23] IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. IEEE, New York, NY, USA, August 12, 1985. ISBN 1-55937-653-8. 20 pp. US$35.00. URL `http://standards.ieee.org/reading/ieee/std_public/description/busarch/754-1985_desc.html; http://standards.ieee.org/reading/ieee/std/busarch/754-1985.pdf; http://www.iec.ch/cgi-bin/procgi.pl/www/iecwww.p?wwwlang=E&wwwprog=cat-det.p&wartnum=019113; http://ieeexplore.ieee.org/iel1/2355/1316/00030711.pdf`. Revised 1990. A preliminary draft was published in the January 1980 issue of IEEE Computer, together with several companion articles [9, 12, 10, 11, 19, 42, 43]. The final version was republished in [20, 22]. See also [44]. Also standardized as *IEC 60559 (1989-01) Binary floating-point arithmetic for microprocessor systems*.

[24] IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. IEEE, New York, August 12 1985. A preliminary draft was published in the January 1980 issue of IEEE Computer, together with several companion articles [9, 12, 10, 11, 19, 42, 43]. Available from the IEEE Service Center, Piscataway, NJ, USA.

[25] Roberto Ierusalimschy. *Programming in Lua*. Lua.Org, Rio de Janeiro, Brazil, 2006. ISBN 85-903798-2-5. 328 (est.) pp.

[26] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. *Lua 5.1 Reference Manual*. Lua.Org, Rio de Janeiro, Brazil, 2006. ISBN 85-903798-3-3. 112 (est.) pp.

[27] Kurt Jung and Aaron Brown. *Beginning Lua programming*. Wiley, New York, NY, USA, 2007. ISBN (paperback), 0-470-06917-1 (paperback). 644 (est.) pp. LCCN QA76.73.L82 J96 2007. URL `http://www.loc.gov/catdir/toc/ecip074/2006036460.html`.

[28] W. Kahan and Joseph D. Darcy. How Java's floating-point hurts everyone everywhere. Technical report, Department of Mathematics and Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA, USA, June 18, 1998. 80 pp. URL `http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf; http://www.cs.berkeley.edu/~wkahan/JAVAhurt.ps`.

[29] Donald E. Knuth. A simple program whose proof isn't. In Feijen et al. [14], chapter 27, pages 233–242. ISBN 0-387-97299-4. LCCN QA76 .B326 1990.

This paper discusses the algorithm used in TEX for converting between decimal and scaled fixed-point binary values, and for guaranteeing a minimum number of digits in the decimal representation. See also [7, 8] for decimal to binary conversion, [39, 40] for binary to decimal conversion, and [18] for an alternate proof of Knuth's algorithm.

[30] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, third edition, 1997. ISBN 0-201-89684-2. xiii + 762 pp. LCCN QA76.6 .K64 1997. US$52.75.

[31] Donald Ervin Knuth. *MMIXware: A RISC computer for the third millennium*, volume 1750 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1999. ISBN 3-540-66938-8 (softcover). ISSN 0302-9743. viii + 550 pp. LCCN QA76.9.A73 K62 1999.

[32] Donald Ervin Knuth. *The art of computer programming: Volume 1, Fascicle 1. MMIX, a RISC computer for the new millennium*. Addison-Wesley, Reading, MA, USA, 2005. ISBN 0-201-85392-2. 134 pp. LCCN QA76.6 .K64 2005.

[33] David Monniaux. The pitfalls of verifying floating-point computations. Technical report HAL-00128124, CNRS/École Normale Supérieure, 45, rue d'Ulm 75230 Paris cedex 5, France, June 29, 2007. 44 pp. URL `http://hal.archives-ouvertes.fr/docs/00/15/88/63/PDF/floating-point.pdf`.

[34] Amos R. Omondi. *Computer Arithmetic Systems: Algorithms, Architecture, and Implementation*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1994. ISBN 0-13-334301-4. xvi + 520 pp. LCCN QA76.9.C62 O46 1994. US$40.00.

[35] Michael Overton. *Numerical Computing with IEEE Floating Point Arithmetic, Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-482-6. xiv + 104 pp. LCCN QA76.9.M35 O94 2001. US$40.00. URL `http://www.cs.nyu.edu/cs/faculty/overton/book/`; `http://www.siam.org/catalog/mcc07/ot76.htm`.

[36] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 2000. ISBN 0-19-512583-5. xx + 490 pp. LCCN QA76.9.C62P37 1999. US$85.00.

[37] C. Severance. An interview with the old man of floating-point: Reminiscences elicited from William Kahan. World-Wide Web document., 1998. URL `http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html`. A shortened version appears in [38].

[38] Charles Severance. Standards: IEEE 754: An interview with William Kahan. *Computer*, 31(3): 114–115, March 1998. ISSN 0018-9162. URL `http://pdf.computer.org/co/books/co1998/pdf/r3114.pdf`.

[39] Guy L. Steele Jr. and Jon L. White. How to print floating-point numbers accurately. *ACM SIGPLAN Notices*, 25(6):112–126, June 1990. ISSN 0362-1340. See also input algorithm in [7, 8], and a faster output algorithm in [6] and [29], IBM S/360 algorithms in [1] for both IEEE 754 and S/360 formats, and a twenty-year retrospective [40]. In electronic mail dated Wed, 27 Jun 1990 11:55:36 EDT, Guy Steele reported that an intrepid pre-SIGPLAN 90 conference implementation of what is stated in the paper revealed 3 mistakes:

1. Table 5 (page 124):
   insert `k <-- 0` after assertion, and also delete `k <-- 0` from Table 6.
2. Table 9 (page 125):
   for `-1:USER!("");`
   substitute `-1:USER!("0");`
   and delete the comment.
3. Table 10 (page 125):
   for `fill(-k, "0")`
   substitute `fill(-k-1, "0")`

[40] Guy L. Steele Jr. and Jon L. White. Retrospective: How to print floating-point numbers accurately. *ACM SIGPLAN Notices*, 39(4):372–389, April 2004. ISSN 0362-1340. Best of PLDI 1979–1999. Reprint of, and retrospective on, [39].

[41] Pat H. Sterbenz. *Floating-point computation*. Prentice-Hall series in automatic computation. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1973. ISBN 0-13-322495-3. xiv + 316 pp. LCCN QA76.8.I12 S77 1974.

[42] David Stevenson. A proposed standard for binary floating-point arithmetic. *Computer*, 14(3):51–62, March 1981. ISSN 0018-9162. See [23, 24].

[43] David Stevenson. *A proposed standard for binary floating-point arithmetic: draft 8.0 of IEEE Task P754*. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1981. 36 pp. See [23, 24].

[44] Shlomo Waser and Michael J. Flynn. *Introduction to Arithmetic for Digital Systems Designers*. Holt, Reinhart, and Winston, New York, NY, USA, 1982. ISBN 0-03-060571-7. xvii + 308 pp. LCCN TK7895 A65 W37 1982. Master copy output on Alphatype CRS high-resolution phototypesetter. This book went to press while the IEEE 754 Floating-Point Standard was still in development; consequently, some of the material on that system was invalidated by the final Standard (1985) [23].

[45] Brian A. Wichmann. Surveyor's Forum: "What every computer scientist should know about floating-point arithmetic". *ACM Computing Surveys*, 24(3):319, September 1992. ISSN 0360-0300. See [15, 16, 13].

Support for multiple TEX distributions in i-Installer and MacTEX

Richard Koch
2740 Washington St.
Eugene, Oregon, USA
`koch (at) math dot uoregon dot edu`
`http://uoregon.edu/~koch/`

Abstract

We discuss a data structure by Gerben Wierda and Jérôme Laurens which makes it easy to use multiple TEX distributions on Mac OS X.

1 A confession

The wonderful data structure described below was designed by Gerben Wierda and Jérôme Laurens and implemented first in Gerben's i-Installer package for gwTEX and then later in the various MacTEX install packages I maintain. When I learned the details of the design, I was repulsed by its complexity, and opposed it with increasingly vitriolic emails. Then one day, an email from Jérôme led to a religious conversion on my part. The data structure now seems natural, and the Gods have condemned me to write this article as punishment for opposition.

2 The problem

TEX GUI applications on the Macintosh, like (my program) TeXShop, LATEXiT, iTEXMac, BibDesk, and others, call command line programs to typeset. A year ago, the standard Mac TEX distribution was teTEX, as packaged by Gerben Wierda. Therefore these GUI applications were configured to use that distribution, and so the applications *worked right out of the box without any configuration.*

But in May of 2006, Thomas Esser announced the end of his support for teTEX. This led to a mad scramble in the TEX world — some users switched to the full unmodified TEX Live and others continued to rely on teTEX. At the international TUG meeting in Marrakesh held in November 2006, Gerben introduced a new distribution named gwTEX based on TEX Live. But alarmingly, he also announced the end of email support for his distribution, and modified i-Installer so the first dialog which appears says **Unsupported Software** in bold letters, followed by a paragraph of text which begins "I regret having to inform you that i-Installer is unsupported software as of Jan 1, 2007." Gerben continues to maintain gwTEX, with a substantial following, and "unsupported" seems to mean merely that he has adopted Donald Knuth's policy of not reading email. But his message puts new users into panic mode.

Meanwhile, TUG's Mac OS X Working Group produced three one-button install packages for TEX, all based on TEX Live and differing mainly in size. These packages are available at `http://tug.org/mactex`. The first of these, BasicTEX, is a 39.7 MB package for users with slow download connections; it installs a surprisingly useful subset of TEX Live 2007. The second, gwTEX, is a 321 MB package which installs gwTEX; users can use i-Installer to maintain this distribution. The third, TEX Live 2007, is a 619 MB package which installs the complete 2007 version of TEX Live.

There are also independent distributions based on teTEX in Fink and in MacPorts.

All of these distributions install in different locations, so installing one does not overwrite the others. For example, BasicTEX is a subset of TEX Live 2007, but installing TEX Live 2007 creates a completely separate installation rather than upgrading the BasicTEX installation.

This proliferation of distributions confuses new users. In early June, a physics graduate student at the University of Oregon called me after switching from Windows to a Mac. He set aside a Saturday to install software. From the web he learned about the MacTEX full TEX Live distribution and installed it. Then his lab partners told him to get scientific applications with Fink, so he installed Fink. Fink asked him which programs to install, so he said "give me everything" and unknowingly got a second TEX distribution. After that, friends suggested learning TEX by starting with LyX, so he installed that. The LyX installer told him that it needed to put style files in /usr/local/gwTeX, so he searched the internet and found gwTEX. In the course of a single afternoon, he had managed to install three complete TEX distributions on his portable. Everything went smoothly until a Fink web page explained how to reconfigure TeXShop for teTEX, and he couldn't figure out what teTEX was.

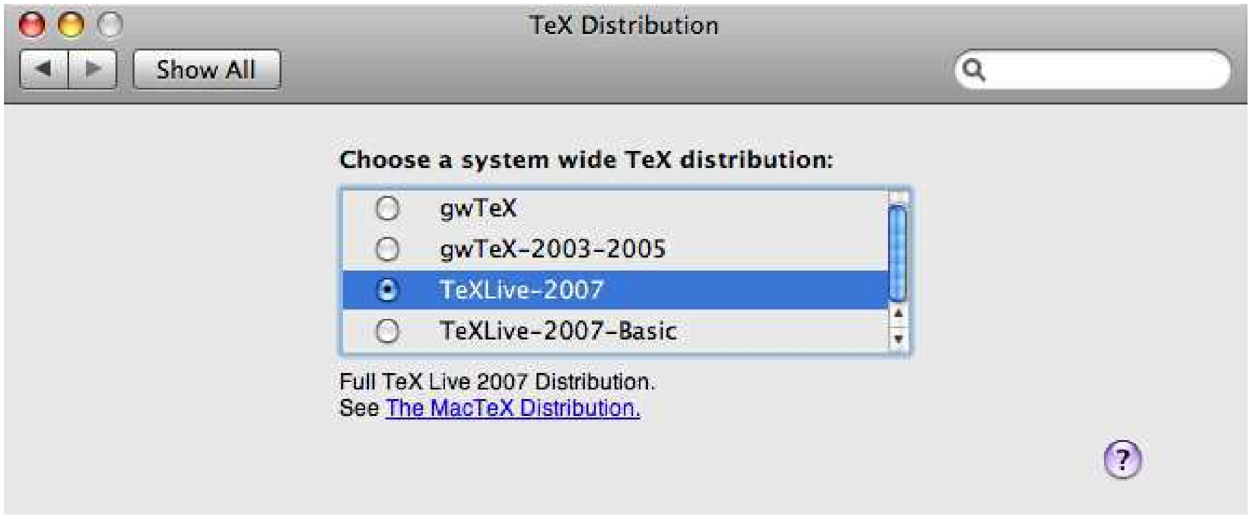
3 The solution

I'll get to the new data structure in a minute, but first let me tell you what Jérôme Laurens wrote in email which led to my religious conversion. Jérôme told me he had a preference pane for Apple's System Preferences which would list available TeX distributions and allow users to switch from one to another.

Jérôme's pane lists distributions installed on a machine. The active distribution is marked in this list. Click on another distribution to make it active. This action automatically switches the PATH and MANPATH variables so Terminal and other shells use the correct binaries and man pages. It automatically reconfigures *all* GUI programs to use the appropriate distribution. For example, TeXShop, LaTeXiT, iTeXMac, and BibDesk switch instantly to the new distribution. You can open a source file, typeset, switch distributions with the pane, and typeset again with the new distribution, all without restarting the GUI program or even reloading the source file.

All of the GUI programs I've listed are configured by default to use the new data structure, so they all *work right out of the box without any configuration.*

Here is the preference pane:



Choose a system wide TeX distribution:

- gwTeX
- gwTeX-2003-2005
- [✓ TeXLive-2007]
- TeXLive-2007-Basic

Full TeX Live 2007 Distribution.
See The MacTeX Distribution.

Do you remember our physics student? I told him "TeXShop is already configured; don't do anything. Go to System Preferences, open the TeX Distribution pane, and you'll see a list of your three distributions. Switch to the one you want to use." Magical, huh?

Jérôme's panel is automatically installed by i-Installer when it installs gwTeX, and by all three MacTeXdistributions. So of course the preference pane recognizes these distributions. But it also recognizes legacy distributions written before the pane existed: Fink's teTeX, MacPort's teTeX, Gerben's old teTeX, TeX Live 2004, TeX Live 2005, and — well, there wasn't any TeX Live 2006.

4 Another problem solved

Developers want the newest versions of computer software. But many TeX users don't think that way. I've often heard variants of "I'm in the middle of two book projects and five papers with coauthors; don't talk to me about upgrading TeX." To some extent, interest in upgrading depends on the component of TeX being used; ConTeXt and XeTeX are undergoing rapid development and users of these programs upgrade regularly, while users of LaTeX and pdfTeX are more likely to desire stability over change.

The TeX Live distribution is upgraded once a year and each upgrade is installed in a different location. So TeX Live users can safely upgrade, knowing that it is easy to return to their older distribution. Nevertheless, I'm willing to bet that a very large percentage of TUG members put the TeX Collection DVD aside when it arrives because they are in the middle of a project. I also bet that most of these folks never retrieve the disk and upgrade.

The problem is that retreating to the old distribution is not foolproof. It is all too easy to reconfigure the GUI front end and leave the PATH in Terminal unchanged. Months later such a user might reconfigure TeX with a Terminal command, only to discover that the change didn't "take" in the GUI program. This is a difficult bug to diagnose.

But Jérôme's panel completely solves the problem. Users can retreat to the old distribution with a single panel click and be assured that all programs are configured correctly. Consequently, I'm hoping that Mac users get in the habit of upgrading as soon as the new TUG DVD arrives.

5 Some history

When Thomas Esser made his announcement, there was barely a ripple in the Mac world because "Gerben will take care of it." But when Gerben announced end of support, there was an explosion of concern; several messages on the TEX on Mac OS X mailing list asked "is this the end of TEX on the Mac?"

At that point, several people began experimenting with TEX Live. I was one of them. It is common knowledge that TEX Live is difficult to install; its install script asks obscure questions like "does your compiler use BSD calling conventions." Mind you, nobody ever told me about this difficulty, but I somehow knew. And as confirmation, I often talk to Karl Berry, who currently maintains the install script, at meetings and he never once asked "why don't you Mac guys use TEX Live?"

But anyway, I dug up an old TUG DVD and ran the script. Gosh. No obscure questions. Very few questions at all, and then poof, TEX Live was installed and it ran like a charm.

So I have a complaint about Karl: excessive modesty. Let me push this complaint by quoting a similar complaint about the inventor of the theory of electricity and magnetism. Freeman Dyson's great essay *Missed Opportunities*, Bulletin of the AMS, 1972, is about situations where mathematicians would have made faster progress is they had paid attention to the physicists. The first case he discusses is Maxwell's theory of electricity and magnetism. Dyson begins by quoting Maxwell himself, who said in a lecture "According to a theory of electricity which is making great progress in Germany, two electrical particles act on one another directly at a distance, but with a force which, according to Weber, depends on their relative velocity, and according to a theory developed by Riemann, Lorenz, and Neumann, acts not instantaneously, but after a time depending on the distance. The power with which this theory explains every kind of electrical phenomena must be studied in order to be appreciated. Another theory of electricity which I prefer ..." and then described his own theory.

Dyson writes "It is difficult to read Maxwell's address without being infuriated by his excessive modesty, which led him to refer to his epoch-making discovery of nine years earlier as only 'Another theory of electricity which I prefer.' How different is his style from that of Newton, who wrote at the beginning of the third book of his *Principia*: 'It remains that, from the same principles, I now demonstrate the frame of the System of the World.' "

That's my complaint about Karl. If only we had known earlier that TEX Live is easy to install! But to get back to the story —

The three MacTEX install packages mentioned earlier are based on TEX Live. In particular, the MacTEX package TEX Live-2007 package the full TEX Live exactly as it would appear if installed directly with the TEX Live script. However, there are several reasons that Mac users should install using the MacTEX packages. First, the packages modify PATH and MANPATH, whereas the TEX Live install script asks users to do this themselves. MacTEX modifies these variables exactly as Gerben does from i-Installer and as required by the TEX distribution data structure, so the MacTEX method is compatible with both gwTEX and this structure.

In addition, the MacTEX packages automatically configure paper size, while the TEX Live scripts ask users to run a post-installation script.

Finally, the MacTEX packages install Jérôme's preference pane and the associated data structures, while the TEX Live script knows nothing about these features.

6 How the data structure came about: A small suggestion from me ...

When I made MacTEX packages for TEX Live, I ran into a small problem. TeXShop and the other GUI programs no longer worked out of the box; they had to be configured first. To make this easy, I stole an idea from TEX Live and created a symbolic link `/usr/local/texprograms` pointing to the binary directory of the installed distribution. The idea was that GUI programs would have to be configured just once to use this link, and all installation packages on the Mac would set the link during installation. I wrote Gerben and casually asked him to support the link in i-Installer.

7 ... and Gerben runs with it

Instead of making this change, Gerben began to look at the problem from a larger perspective. There are many things that a GUI application might like to know about the active TEX distribution. Certainly it will want to know where the binaries are. But each TEX distribution contains a lot of documentation, and the GUI app might well like to find and display that documentation. Perhaps the GUI app wants to display man pages too. And recall the LyX installer, which wanted to install style files in the distribution;

it needed to know the location of the distribution's main `texmf` tree.

Gerben proposed a data structure which would organize and locate all of this information and more.

Let's return to the simple issue of binary location. The trouble is that each distribution handles this in a slightly different way. In gwTEX, there are two binary directories, one for Intel binaries and one for PowerPC binaries. The Intel directory is `i386-apple-darwin-current` and the PowerPC directory is `powerpc-apple-darwin-current`. TEX Live works similarly, but the binary directories are named `i386-darwin` and `powerpc-darwin`. Fink contains universal binaries which work with both processors, located in `/sw/bin`. A GUI app might want to find the binary directory for the processor of the machine on which it is running; this GUI app shouldn't have to know these various naming conventions to do its job.

Once we get this far, the basic idea of Gerben and Jérôme's data structure will be clear. For each distribution, they create a small folder of symbolic links pointing to actual locations in the distribution. The names chosen for these links are common for all distributions rather than names used by a particular distribution. Thus a GUI app only needs to know the common names assigned by Gerben and Jérôme rather than the actual names chosen by a distribution. In addition, there is a link named `DefaultTeX` pointing to the folder of symbolic links for the active distribution.

Here is how this works in practice. The folder for each distribution has a subfolder `Programs` and inside that folder are two symbolic links `i386` and `powerpc`. These point to the actual directories containing the Intel and PowerPC binaries for that distribution. Putting this together, a GUI program running on an Intel processor will find the binaries for the currently active distribution in `DefaultTeX/Programs/i386`. If the active TEX happens to be TEX Live 2007, this will yield `/usr/local/texlive/2007/bin/i386-darwin`. If the active TEX is Fink's teTEX, it will be `/sw/bin`. That is the simple idea in a nutshell. (In this explanation I have left out a few details, so the paths to the links aren't exactly these.)

Tthe path to binaries for the active TEX using the architecture of the current machine is of such importance that Gerben and Jérôme provide a shortcut: a link `/usr/texbin` points to `DefaultTeX/Programs/i386` on Intel machines and `DefaultTeX/Programs/powerpc` on PowerPC machines. Thus GUI apps can use `/usr/texbin` as a path to TEX binaries. (Thus `/usr/texbin` replaces my earlier `/usr/local/texprograms`.) All the standard GUI apps on Mac OS X now use this as their default path. That is why they run right out of the box.

When the structure is installed by i-Installer or MacTEX, a technical description of the structure is installed in `/Library/TeX/Distributions`. It isn't necessary for me to give full details here, but let me say just a little more to indicate the range of possibilities opened up by the structure. The folder of links for a particular distribution contains subfolders named `Doc`, `Info`, `Man`, `Root`, `AllTexmf`, and others. `Doc` contains symbolic links to actual folders in the distribution containing documentation; the number of such links varies with distribution. For instance, for TEX Live 2007 the links are called `texmf-dist-doc`, `texmf-doc`, `texmf-doc-doc`, and `texmf-var-doc`. Thus a GUI application can use `DefaultTeX/Doc` to discover the entire documentation tree for the active distribution.

`Info` contains links to the various GNU Info files; `Man` contains links to the various man pages, and `Root` contains a link to the folder containing the entire distribution. `AllTexmf` contains links to the various `texmf` trees of the distribution; for example, the TEX Live 2007 `AllTexmf` contains links named `texmf`, `texmf-dist`, `texmf-doc`, `texmf-local`, and `texmf-var`.

The Macintosh file system has a folder named `/Library` containing various Apple and Third Party data files which apply system wide. The TEX distribution data structure is installed in `/Library/TeX`, where `TeX` is a subfolder created by i-Installer or MacTEX. At the moment this folder only contains information about the TEX distribution data, but I understand that some developers are eyeing it for other uses. So I don't recommend throwing it away cavalierly. The actual distribution symbolic links are in folders in `/Library/TeX/Distributions`. There will be one such folder for each distribution installed on the machine, together with folders for legacy distributions even if those aren't installed.

It is not necessary to clean up the symbolic links in the data structure if the corresponding distribution is thrown away, because Jérôme Laurens' preference pane is smart and only shows distributions when the data structure points to something concrete. Thus, for instance, it is perfectly ok to entirely remove TEX Live 2007 using the simple command

```
sudo rm -R /usr/local/texlive/2007
```

even though this will leave a stranded data structure behind. The data structure itself is tiny.

I have now explained all of the key ideas. The structure and preference pane are quite simple, but I use them virtually every day. Because of this structure,

- GUI apps work right out of the box;
- TeX can safely be upgraded because it's trivial to revert if necessary;
- GUI TeX applications may display documentation in the future.

8 But give us the dirt

You may be saying "Look, I really don't care about the data structure. I don't even use a Mac. I'm here because I want to hear about the vitriolic email." If so, this is the section for you.

Let me say from the start that the disagreements I'm going to describe are in the past — there was a battle, Gerben and Jérôme were right and I was wrong. But for your amusement —

When Gerben first designed the data structure, there was no hint of a preference pane. I didn't exactly know how users would switch the default distribution, but it looked like they were expected to directly open `/Library/TeX/Distributions` and manipulate the folders and data inside it. This made it urgent that this location contain straightforward data.

Initially, Gerben's design was simple, but then it took an unexpected turn. Gerben and Jérôme added an extension `.texdist` to the names of the folders which contain symbolic links describing actual distributions; for example, `TeXLive-2007` became `TeXLive-2007.texdist`. Inside this folder they placed a subfolder named `Contents`. Further investigation revealed that most of the `.texdist` folders were associated with mirror images in a hidden folder; for example, the folder `TeXLive-2007.texdist` had a mirror named `TeXLive-2007` in a hidden folder inside `/Library/TeX/Distributions`. This mirror folder also contained a `Contents` folder, and the `Contents` in the original `TeXLive-2007.texdist` folder proved to be a symbolic link to this `Contents` folder in the mirror `TeXLive-2007`. That is, the actual symbolic links which described the distribution were inside the Contents folder in the mirror folder living in a hidden directory.

Around this time, I realized that I could no longer explain the data structure in a few sentences and became alarmed. I have never met Gerben or Jérôme (or most of my other collaborators), but Gerben and I have been in email contact for years — really since the start of OS X — and all of this email has been pleasant or better. So as I realized the baroque nature of the developing data, I began complaining.

In growing frustration, I wrote a few people who weren't involved in the design and must have wondered what the heck I was talking about. One of those folks was Jonathan Kew, and here's what I wrote him:

```
I shouldn't be writing you, but I need
someone to "kvetch" to and I don't want
to pollute the mailing lists. I have just
seen Gerben's final "link" design. I think
it is a mess. Am I supposed to support
the design in MacTeX?

Gerben will end support of his packages
in January, so we face the possibility of
dealing with multiple TeX distributions on
the Mac. With multiple packages, we need
an easy way to configure GUI applications.
A couple of weeks ago I introduced MacTeX
packages for three distributions and had
each package set a symbolic link
/usr/local/texprograms.  When I asked
Gerben to set /usr/local/texprograms in
i-Installer packages, he decided to design
a more complete solution. Fair enough.

KVETCH:

But when it came time to implement
these changes, Gerben lost all sense
of proportion. Am I expected to support
his new baroque design in MacTeX?
Should I use Gerben's data structure
or just give up and make my
own. Am I taking this too personally? Do I
need to get a life?
```

Jonathan is a very intelligent fellow. He didn't answer this email.

9 A Macintosh secret

Why was the new data structure so complicated?

On the Macintosh, it is possible to trick the Finder so that it displays a folder as just another file. The program XCode, which developers use to write applications, saves projects in this way; each saved project appears to be a simple file, but if you hold down the control key while clicking on the file name, a menu will appear offering to open up the file and display its contents.

What makes a folder magical in this way? Well, its name must have an extension, and it must contain a subfolder named `Contents`.

If such a magical folder lives on a computer, it will look like any other folder *until* it is claimed by an application which understands its extension. When such an application is installed, the magical folder will suddenly obtain an icon, and clicking on it will open the associated application instead of opening the folder. Thus magical folders are like the pods growing in the basements of houses in *Invasion of the Body Snatchers*. They are faceless and asleep until a governing alien application is installed, when they suddenly obtain a face, wake up, and do the application's bidding.

Gerben and Jérôme had decided that the folders in `/Library/TeX/Distributions` would be such magic folders. This explained every complication in their data structure. But immediately it led to a key question: *what was the alien application they intended to claim these magical folders?* I asked Gerben this question in an email, and he replied "I'm just designing the data structure; it is up to other people to use it."

10 A suspicion

If you don't come from the Macintosh world, there is a key fact in this story that you may not know. I am the author of a front end for TeX named TeXShop. Jérôme Laurens is the author of a competing front end named iTeXMac. Jérôme is my rival. Suddenly I knew what the alien application was intended to be! Gerben and Jérôme are both Europeans; clearly the Europeans were planning to pull the TeX rug right out from under me!

11 I fight

In a series of increasingly angry emails, I argued that no front end should interfere with the underlying TeX distribution. Gerben was so caught up in the data structure design that he found my email distracting and irritating, and Jérôme ignored me completely. But suddenly I had an idea. I modified two lines of code in TeXShop, and suddenly TeXShop became the alien application claiming the magical folders. I wrote a final email explaining what I had done, ending with, "Why can't you see that no front end should do what I have done?"

12 Paranoia doesn't pay

At that point, a magical thing happened. I had been getting email from Gerben, but nothing from Jérôme. Now Jérôme managed to get in contact by an indirect route; and he wrote "Did you know that the University of Oregon email system has been rejecting my mail?"

Next Jérôme told me that he always believed front ends shouldn't claim `.texdist`.

After that, Jérôme he told me for the first time about his TeXDist Preference Pane, which in my opinion is the key ingredient which makes the data structure usable by everyone. Finally he said that he had written a small application which claimed the magical folders, but all it did was to open the TeXDist Preference Pane when the user clicked on a magical folder.

So in a single email, I learned that users could avoid learning of `/Library/TeX/Distributions` altogether, that the magic folders were benign, and that a preference pane existed which met Apple's high standards for simplicity and usability. It was an exciting day.

First moral: there are fewer evil people in the world than you may think.

Second moral: Your rivals also have good ideas.

Design decisions for a structured front end to LaTeX documents

Barry MacKichan
MacKichan Software, Inc.
`barry dot mackichan at mackichan dot com`

1 Logical design

Scientific WorkPlace and *Scientific Word* are word processors that have been designed from the start to handle mathematics gracefully. Their design philosophy is descended from Brian Reid's *Scribe*,[1] which emphasized the separation of content from form and was also an inspiration for LaTeX.[2] This *logical design* philosophy holds that the author of a document should concern him- or herself with the content of the document, and with identifying the *role* that each bit of text plays, such as a header, a footnote, or a quote. The details of formatting should be ignored by the author, and handled instead by a predefined (or custom) style specification.

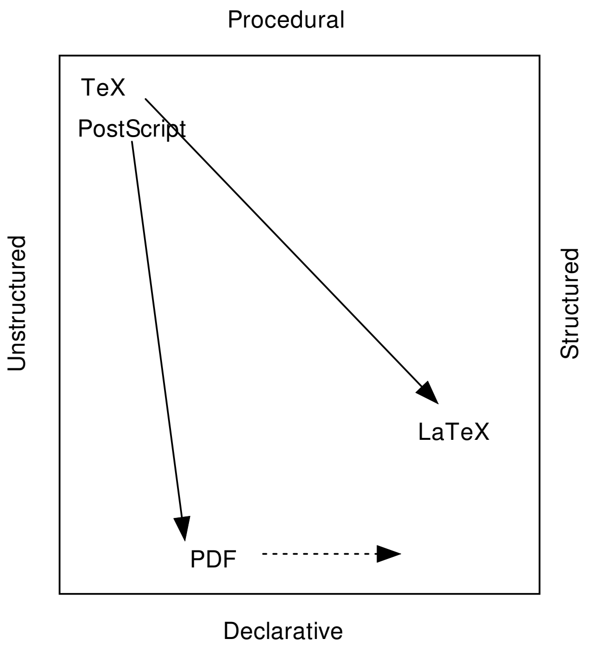
There are several very compelling reasons for the separation of content from form.

- The expertise of the author is in the content; the expertise of the publisher is in the presentation.
- Worrying and fussing about the presentation is wasted effort when done by the author, since the publisher will impose its own formatting on the paper.
- Applying formatting algorithmically is the easiest way to assure consistency of presentation.
- When a document is re-purposed it can be reformatted automatically for its new purpose. This can happen when a document is put on the Web in addition to being published, or even when the author sends the document to a new publisher.

The most powerful typesetting programs tend to be programming languages themselves. The two most prominent examples are PostScript and TeX. Although these are extremely powerful, they are not always simple, and they do not separate content from form. Consequently, there is a migration on the following plot from the top to the bottom, and from the left to the right.

[1] Brian K. Reid, "Scribe: A Document Specification Language and its Compiler," Ph.D. Dissertation, Carnegie-Mellon University, Pittsburgh, PA, Oct. 1980.

[2] Leslie Lamport, LaTeX: A Document Preparation System, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, Second Edition, 1994.



Thus, PostScript is a powerful programming language, but it was later supplemented by PDF, which is not a programming language, but instead contains declarations of where individual characters are placed. PDF is not structured, but Adobe has been adding a structural overlay. LaTeX is quite structured, but it still contains visible signs of the underlying programmability of TeX, so I haven't quite placed it at the bottom of the plot. The pattern is that power and flexibility generally get supplemented or replaced in some circumstances with structured and declarative alternatives.

The original design philosophy for *Scientific WorkPlace* and *Scientific Word* was to make visual word processors that live at the bottom right of the diagram, and produce their output by generating LaTeX using one of over a hundred typesetting styles. This is the optimal solution for publishing, at least when we support a publisher's style, or when a publisher's style uses the same tags as one of the standard LaTeX document types.

2 Enter the customer

Although this philosophy works very well for publishing, many of our customers want to have greater control over the appearance of their documents. The truth is that not all mathematical documents are written for publication in a journal. The author might want to post a document on the Web or to send out preprints, or to prepare reports that will not be published, or to prepare handouts for students. The cold hard truth is that programs like Microsoft Word — despite its intellectual roots being also in *Scribe* — have over the years encouraged users to fiddle and futz with formatting. The experts may all agree that the result is ugly, but the customer is the one who pays our salaries.

In the past, *Scientific Word* users had a hard time if they wanted to change or add to a style. The advice of our tech support staff has been:

- You don't want to do that
- You shouldn't do that
- You can use package X to do that
- You can rewrite the style file

We no longer give the first two responses, and our users are not going to be able to use the fourth bit of advice. Due to the large number of useful packages, we now encourage users to start with a standard LaTeX document type and to use packages. This works, but it is not the most elegant way to solve the problem, since you shouldn't have to write options for the `geometry` package in order to change a margin.

We also allow the user to enter snippets of raw TeX or LaTeX code in what we call a "TeX button" (which is how we enter "TeX" and "LaTeX") but this runs counter to the design philosophy, and can't address problems when a user wants to change, for example, how list items are generated (since the code to be added would be in the middle of code we have generated).

3 A statement of the problem

This discussion now allows a statement of the problem we are solving.

1. We want an internal form for our documents that is both rich and extensible, and a rendering engine that is rich enough to render a LaTeX document and which is extensible.

2. We want to convert a LaTeX document to our internal form in a way that is extensible and preferably uses standard, well-documented tools, and in particular does not require access to our source code.

3. We want to convert our internal form to LaTeX in a way that is extensible and uses standard tools, and does not require access to our source code.

Part of the motivation for not needing access to our source code is that extending these operations will be easier for us if it is not necessary to change and re-build C++ code in order to support a new tag or to change the behavior of a standard tag. The other part of the motivation is that if the tools are standard and well-documented, then advanced users can make their own changes.

3.1 Internal form of a document

Scientific Word has an internal form that is not LaTeX but looks superficially like LaTeX, and we have an adequate rendering engine for it. However, it is not extensible — that is, to extend it means rewriting C++ code and extending the rendering engine. To avoid this problem and get the extensibility we need, we choose an internal form that is rich enough and extensible (and it must also be declarative and structured). The obvious candidate (at least in this century) is XML. We are basing future versions of our software on the Mozilla Gecko rendering engine for HTML and XML. Tags can be introduced at will, and CSS (Cascading Style Sheets) are used to determine how these tags appear on the screen.

Some of the features of Gecko that are very useful to us are:

- The rendering engine is open-source under a license that allows us to extend it if necessary.
- The rendering engine is rich and powerful (the program user-interface is in fact a Gecko document).
- XML is a standard that is easily converted to and from LaTeX.
- A powerful scripting language is integrated into Gecko.
- A technology (XBL–XML Binding Language) allows attaching behavior to (new) XML tags.
- A system of broadcasters and observers simplifies coordinating the behaviors of objects.
- Support for infinite undo and redo is built into the document-modifying functions.

3.2 Conversion from LaTeX

Scientific Word does not process TeX or LaTeX files with TeX. It simply determines the structure of the file by recognizing tags such as `\section` and `\subsection`. In the past, it has caused problems when users defined their own macros: we did not recognize them and loaded the macro invocation as a TeX button. Beginning with version 5.5 (two years ago) we now run a version of the TeX macro processor, and we evaluate macros defined by the user, but we do not evaluate macros defined in LaTeX or any of the standard packages. The result should be a document that contains only the standard macros, and which can be read by *Scientific Word*.

We continue with this same approach in our new architecture, except that the definitions of the standard LaTeX macros converts them to XML. The resulting files are complicated, but most of the complication is in some utility macros that make the final macros quite easy to understand. Some sample code from one of these files is:

```
\def\out@begin@abstract{%
 \msitag{^^0a}%
 \msiopentag{abstract}{<abstract>}
}
\def\out@end@abstract{%
 \msitag{^^0a}%
 \msiclosetag{abstract}{</abstract>}
}
```

This is all that is required to convert the `abstract` environment to XML.

3.3 Conversion to LaTeX

The conversion to LaTeX is done using XSLT (XML Stylesheet Language Transformations). As the name implies, XSLT was designed as part of a method of applying styles to XML objects, which sometimes requires making some transformations or re-ordering the XML elements. It has evolved into a powerful standalone transformation language for XML documents. It can be used to transform XML into XML, or XML into text, which includes TeX.

For instance, here is the XSLT rule that generates the `abstract` environment:

```
<xsl:template match="abstract">
  \begin{abstract}
    <xsl:apply-templates/>
  \end{abstract}
</xsl:template>
```

When XSLT finds the `<abstract>` tag, it first generates `\begin{abstract}`, then applies any rules needed for the content of the tag, and finally generates `\end{abstract}` when it reaches the end of the `abstract` node. The tag may have attributes, which might affect the TeX generated, and the rules can depend on the context of the tag.

The point here is that it is relatively easy to add support for new tags, or to change the TeX that gets produced by a tag. In older versions of our products, these operations took place in compiled code, but now they are controlled by text files that can be replaced or modified without rewriting or recompiling C++ code. It is now feasible to support different flavors of TeX for *Math Reviews*, or to support something like ConTeXt.

The next section addresses the question of how you can tailor the on-screen presentation of a tag.

4 Some examples

4.1 Displaying 'LaTeX' on screen

This is a brief discussion of how you can display a new tag, such as `<latex/>`, on the screen. This is done by using XBL. We'll skip lightly over the details.

In a CSS file there is a line that tells Gecko that special rules apply to this tag:

```
latex {
  -moz-binding: url(
    "resource://app/res/xbl/latex.xml#latex");
}
```

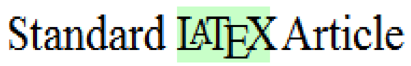
In the file `latex.xml`, there is a section that says how to display the tag:

```
<xbl:content>
  <sw:invis><xbl:children/></sw:invis>
  <sw:latex2>L<sw:latexa>A</sw:latexa>
  <sw:latext>T</sw:latext>
  <sw:latexe>E</sw:latexe>X</sw:latex2>
</xbl:content>
```

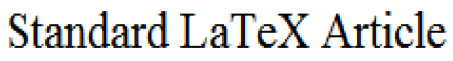
Each letter in LaTeX (almost) is in a separate tag, which allows us to change the style for each letter. Here is the style rule for the 'A' (the tag `latexa`):

```
latexa {
  font-size: smaller;
  position: relative;
  bottom: .15em;
  left: -0.20em;
}
```

This rule shrinks the 'A' and moves it up and left. A style rule for the `latex2` rule changes the letter spacing to squeeze them together a bit. The final result on the screen is:


Standard LaTeX Article

Actually, what appears in the internal format is `<latex>LaTeX</latex>`. The content of the tag ('`LaTeX`') is thrown away, except when the XML document is viewed by some other browser, such as Internet Explorer, or even Firefox. Internet Explorer, when it sees the `-moz-binding` statement in the CSS file will ignore it completely. Firefox will understand it, but will be unable to find the `latex.xml` file, which is internal to our program. As a result, they will ignore the `latex` tag and will simply display the contents. Thus, the above displayed on Firefox will appear as:


Standard LaTeX Article

Of course, the LaTeX generated by this tag, no matter what its content, will be `\LaTeX`.

4.2 Spaces

LaTeX provides a wide choice of spaces, both horizontal and vertical. It is possible to make them visible by selecting a menu item "Show Invisibles". This is accomplished in the same way as the above example, with special CSS rules to apply in the case when "Show Invisibles" is on.

5 User interface enhancements

The next two items are not particularly related to the new architecture for *Scientific Word*; rather, they can be looked on as one solution to the problem of converting a rich keyword-value interface to a friendlier (to the novice) dialog-based interface. The result is marginally less powerful, but still allows the advanced user to get access to almost all the features of the keyword-value interface.

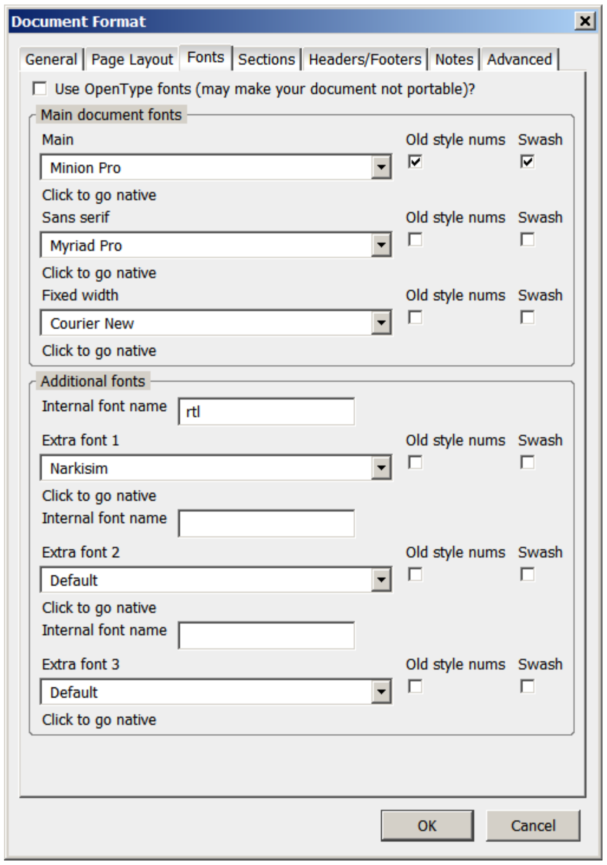
The dialog shown in figure 1 is for selecting OpenType fonts. Before the user gets to this point, he will be warned that if he proceeds, his document will have to be compiled with XeTeX and therefore will not be completely portable.

This allows the user to pick the three main fonts: the main (roman) font, the sans serif font, and the monospaced font. He can also choose other fonts and give them names. We have a `rtlpara` tag for right-to-left text, and this uses the `rtl` font, for which the user has chosen Narkisim.

There are many font attributes, and many are not widely supported in available fonts, so we have chosen only two for access by checkboxes: old-style numerals and swash italics. Other attributes are accessible, but only by falling back on the keyword-value interface and clicking on the "Go native" link; see figure 2.

The first line in the "Go native" box was provided automatically since the user had clicked on "Old style nums" and "Swash". The user added the next line to use the MinionPro-Bold font as his bold Roman font rather than the default Semibold. This interface allows almost complete access to the power of the `fontspec` package but gives more casual users the ability to choose basic fonts easily.

Another dialog interface to package options is the page layout dialog, as shown in figure 3. Here the user is adjusting the left margin by pressing or holding the up or down arrow key in the left margin width field.



☐ Use OpenType fonts (may make your document not portable)?

Main document fonts

	Old style nums	Swash
Main: Minion Pro	✓	✓
Click to go native		
Sans serif: Myriad Pro	☐	☐
Click to go native		
Fixed width: Courier New	☐	☐
Click to go native		

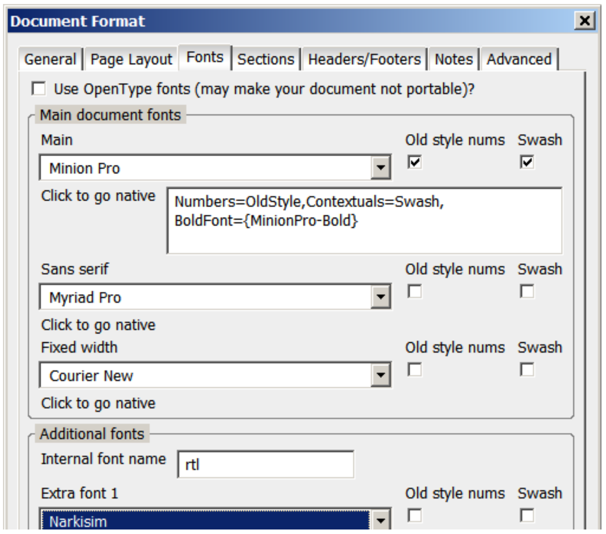
Additional fonts

	Old style nums	Swash
Internal font name: rtl		
Extra font 1: Narkisim	☐	☐
Click to go native		
Internal font name:		
Extra font 2: Default	☐	☐
Click to go native		
Internal font name:		
Extra font 3: Default	☐	☐
Click to go native		

Figure 1: Dialog for selecting OpenType fonts.

6 Conclusion

Scientific WorkPlace and *Scientific Word* are designed to make it easy for authors to write a beautiful LaTeX document with skills they already have. To keep its simplicity from becoming a limitation, we have to provide ways for more advanced users to override the default decisions that *Scientific Word* makes. This paper has covered a few of the new technologies we are using to make a more modular system, with the interconnections provided by stable and well-documented standards in a way that we, or a knowledgeable user, can easily customize. We expect this new platform to allow us to be more nimble than before in responding to the changing needs of our customers, and to serve as a solid base for the next ten years of development.



Document Format

General | Page Layout | Fonts | Sections | Headers/Footers | Notes | Advanced

☐ Use OpenType fonts (may make your document not portable)?

Main document fonts

Main: Minion Pro — Old style nums ☑ — Swash ☑

Click to go native: Numbers=OldStyle,Contextuals=Swash, BoldFont={MinionPro-Bold}

Sans serif: Myriad Pro — Old style nums ☐ — Swash ☐

Click to go native

Fixed width: Courier New — Old style nums ☐ — Swash ☐

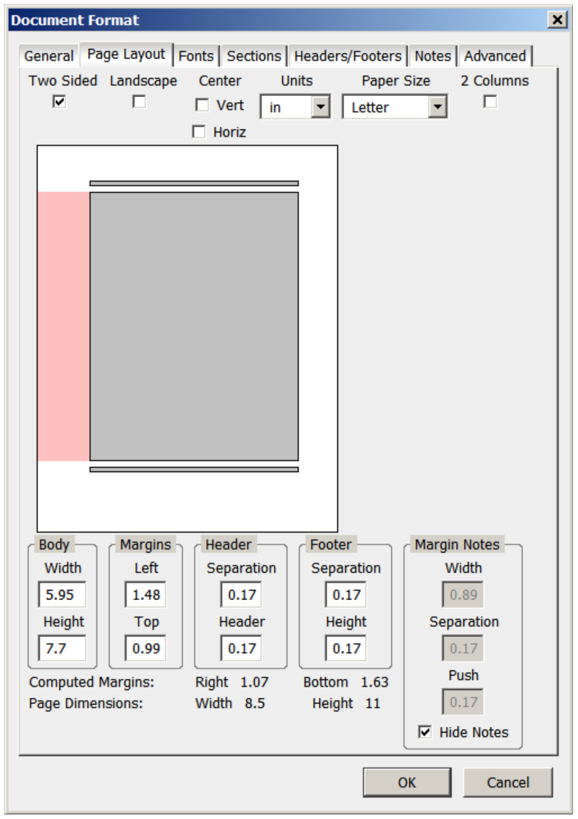
Click to go native

Additional fonts

Internal font name: rtl

Extra font 1: Narkisim — Old style nums ☐ — Swash ☐

Figure 2: Selecting OpenType font attributes.



Document Format

General | Page Layout | Fonts | Sections | Headers/Footers | Notes | Advanced

Two Sided ☑ — Landscape ☐ — Center: ☐ Vert ☐ Horiz — Units: in — Paper Size: Letter — 2 Columns ☐

Body		Margins		Header		Footer		Margin Notes	
Width	5.95	Left	1.48	Separation	0.17	Separation	0.17	Width	0.89
Height	7.7	Top	0.99	Header	0.17	Height	0.17	Separation	0.17
								Push	0.17

☑ Hide Notes

Computed Margins: Right 1.07 — Bottom 1.63

Page Dimensions: Width 8.5 — Height 11

Figure 3: Page layout dialog.

MathType 6.0's TEX input for MS Word and Wikipedia

Paul Topping
Design Science, Inc.
140 Pine Ave.
Long Beach, CA
USA
pault (at) dessci dot com
http://www.dessci.com

Abstract

MathType is well-known for its point-and-click user interface for editing math. However, some users feel more comfortable typing math using TEX, so in MathType 6.0 we have added a TEX input mode. This provides the user with the best of both worlds: TEX for initial entry, point-and-click and drag and drop for easy editing and manipulation. Since MathType can save equations in several graphics formats and objects, it provides a direct path from TEX to Microsoft Word, PowerPoint, and virtually any document or application. Since many blogs and wikis accept a variant of TEX math syntax and expose it in their web pages, we are now able to support both authoring and reuse of equations in these environments. In particular, MathType users can now copy equations out of the thousands of Wikipedia pages containing equations for use in educational and research authoring. In addition, MathType users can create equations and paste them directly into new Wikipedia content.

1 Introduction

Throughout MathType's 20-year history, it has been firmly in the point-and-click camp of equation editing. Because a trimmed-down version has shipped with Microsoft Office since 1991, it has been used to type a lot of math. Of course, TEX remains popular and is heavily used in some scientific communities. Once a person's hands "know" TEX, it is hard for them to imagine typing math any other way. And, because TEX is free and easy to integrate into web servers as an equation image generator, many blog and wiki applications support it. Unfortunately, many people who don't know TEX struggle with authoring math in these environments. With MathType 6.0, we tried to bridge both of these gaps, bringing TEX input to MathType and allowing people who don't know TEX to more easily work with wikis and blogs. Since Wikipedia is so popular and contains many equations authored in TEX, we have made working with its equations especially easy.

2 Typing TEX in a MathType window

Typing equations using TEX and LATEX math syntax in MathType is very easy. At any point in building up an equation, if the user enters a run of text starting with one of the characters `$ ^ _ \`, that run will be treated as TEX input to be converted into math notation (see Figure 1). The resulting TEX text will appear in dark grey as opposed to the normal black of converted math. Hit Enter and the TEX input is converted into normal MathType equation content (see Figure 2). Any errors appear in red. Corrections can be made using MathType's normal point-and-click editing facilities, or the conversion can be undone to allow corrections to be made in the original TEX. TEX can also be pasted into MathType via the clipboard.

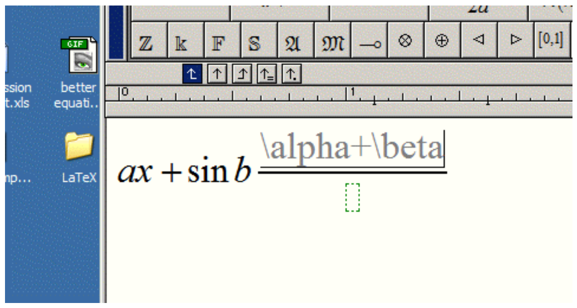
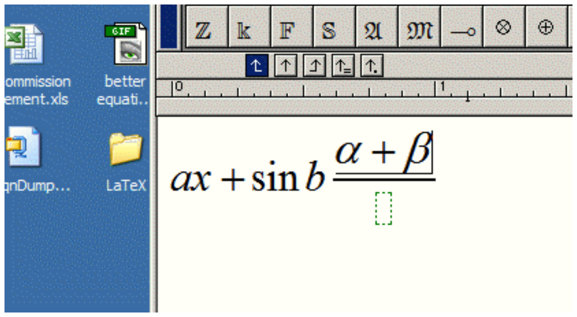


Figure 1: User types some TEX.

3 Copying equations out of Wikipedia

Wikipedia, the popular online encyclopedia, contains thousands of pages with equations represented



$$ax + \sin b\,\frac{\alpha+\beta}{\square}$$

Figure 2: After typing ENTER.

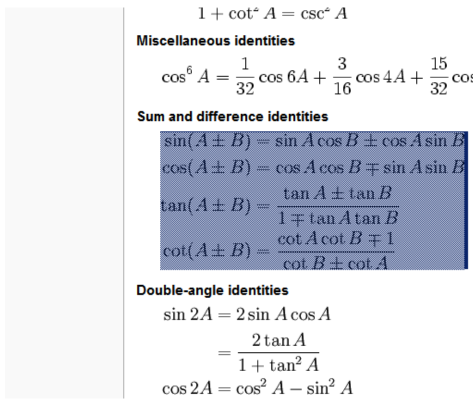
in both HTML and as images. The math is authored using the Texvc subset of TEX markup with some extensions for LATEX and AMS-LATEX. If it is simple enough, TEX input is converted to HTML markup.

Otherwise, an image is generated with the TEX input stored in the image's ALT attribute. When an equation is copied from the web page onto the Windows clipboard, the math notation is made available to MathType where it is converted to MathType's own equation representation. This works with Internet Explorer, Firefox, and any browser that supports Microsoft's HTML clipboard format (see Figures 3 and 4). A stack of HTML and/or TEX equations can be copied in a single operation.

We have also added a Texvc output translator allowing equations to be created in MathType, or copied from equations in Microsoft Word documents, and pasted into Wikipedia pages.

4 Implementation

MathType's translation process involves two separate translators working together, one for HTML math and one for TEX. The translators are written in a home-grown, rule-based language named Sevilla after the Spanish restaurant on the lower floors of Design Science's Long Beach, California, offices. Since the translation is defined using rules in text files, the existing translators can be customized and entire new translators may be created.



$$1 + \cot^2 A = \csc^2 A$$

Miscellaneous identities

$$\cos^6 A = \frac{1}{32}\cos 6A + \frac{3}{16}\cos 4A + \frac{15}{32}\cos 2A + \frac{5}{16}$$

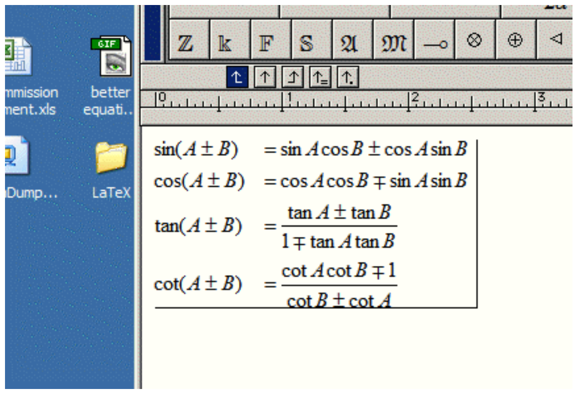
Sum and difference identities

$$\sin(A \pm B) = \sin A \cos B \pm \cos A \sin B$$
$$\cos(A \pm B) = \cos A \cos B \mp \sin A \sin B$$
$$\tan(A \pm B) = \frac{\tan A \pm \tan B}{1 \mp \tan A \tan B}$$
$$\cot(A \pm B) = \frac{\cot A \cot B \mp 1}{\cot B \pm \cot A}$$

Double-angle identities

$$\sin 2A = 2\sin A \cos A$$
$$= \frac{2\tan A}{1+\tan^2 A}$$
$$\cos 2A = \cos^2 A - \sin^2 A$$

Figure 3: User copies from Wikipedia's Trigonometry page, `http://en.wikipedia.org/wiki/Trigonometry`.



$$\begin{aligned}
\sin(A \pm B) &= \sin A \cos B \pm \cos A \sin B\\
\cos(A \pm B) &= \cos A \cos B \mp \sin A \sin B\\
\tan(A \pm B) &= \frac{\tan A \pm \tan B}{1 \mp \tan A \tan B}\\
\cot(A \pm B) &= \frac{\cot A \cot B \mp 1}{\cot B \pm \cot A}
\end{aligned}$$

Figure 4: After pasting into MathType.

5 Current status

MathType 6.0 for Windows was released in July 2007. The same technology will also be brought into a future Macintosh version. In the future, we expect to be looking at creating smooth interfaces between MathType and other web-based math environments.

CrossTEX: A modern bibliography management tool

Robert Burgess and Emin Gün Sirer
Cornell University, Ithaca, NY
http://www.cs.cornell.edu/People/egs/crosstex

Abstract

CrossTEX is a new bibliography management tool that eases database maintenance, style customization, and citation. It is based on an object-oriented data model that minimizes redundant information in bibliographic databases. It enables a work to be cited not only through arbitrarily-assigned object keys but through semantic information that uniquely identifies the work. It automates common tasks in order to avoid human errors and inconsistencies in bibliographies, while providing users with fine-grained control. CrossTEX's other features include support for modern reference objects such as URLs and patents, direct generation of HTML documents, the ability to write styles in a modern programming language, and extensive databases of published works included in the distribution. It is backwards-compatible with existing BIBTEX databases, and, overall, builds on BIBTEX's strengths while fundamentally fixing the design restrictions that lead to errors in BIBTEX-formatted documents.

1 Introduction

Bibliography management and typesetting play a critical role in publishing. Since its introduction in 1985, BIBTEX has become the dominant tool for professional bibliography management with LATEX. It has achieved this dominance due to several good design decisions, such as tight integration with LATEX, a human-readable format for bibliographic databases, and overall ease of use. However, two decades of experience with BIBTEX have revealed several fundamental weaknesses that require re-evaluating what features a modern bibliography management system must provide. CrossTEX is such a tool, which learns from the example of BIBTEX and provides backwards compatibility while moving forward with new models of bibliographic data and stylistic control.

1.1 What's wrong with BIBTEX?

First, BIBTEX interprets databases with a single-table *relational* model in which every bibliographic entry contains all of the information that can appear in it. This redundancy is a challenge to those who maintain bibliographic databases because they must correctly look up, enter, and maintain all that information for every reference; it is easy to enter or modify entries separately and use slightly different versions of the name of a journal, conference, or even author.

BIBTEX's `crossref` field provides a simple, specialized form of "inheritance", allowing information to be factored out into a single other object, but is not a generalized feature — it is insufficient, for example, for an author who wants to create a bibliography of all of his or her own works, with a common note or URL associate with each entry. Although `@string` objects could help prevent spelling mistakes, there is no practical way around adding the fields explicitly to each entry.

Furthermore, meeting publication requirements in a professional setting that requires consistent abbreviations, names, and formatting guidelines requires users to edit the database. To abbreviate a journal name, the user must edit a copy of the database, find each occurrence of the name, and change it to the desired value. Even if the database takes advantage of `@string` objects, a feature included in BIBTEX to attempt to circumvent the restrictive relational model, the values of the strings must be changed, because information appears in a BIBTEX-formatted bibliography as it appears in the database. This fact alone prevents large, common databases from being useful save for looking up citation information to copy-and-paste to smaller, per-document bibliographies that authors must manage.

BIBTEX citations within documents are based on arbitrary keys attached to each entry in the database. In the best case, authors establish site-wide rules for creating keys from entries so they can correctly guess the required key based on the publication information of the article to be cited. This is still quite fragile, as a single typo may cause the wrong work to be cited; when databases have been assembled by cutting and pasting entries from different sources, the keys are unlikely to follow a convention, and the author must instead look up the appropriate key for each citation.

BIBTEX style files are written in their own, arcane programming language. Few authors or database maintainers have the skill to create or accurately modify a bibliography style to meet requirements handed to them by publishers — they must count on an appropriate style file being provided or already existing, or they must blindly change one that is "close" until it seems to produce the correct typeset appearance.

In addition, BIBTEX does not easily permit the introduction of new kinds of objects. Database maintainers represent newly emerging referenced works, such as URLs, and unsupported objects, such as patents, with the all-purpose `@misc` object whose appearance is difficult to control because it is too general.

Finally, BIBTEX provides very little automation besides formatting the references as specified in the database. For example, it does not check capitalization in titles, ensure that an author's name appears consistently, or abbreviate journals. This can lead to inconsistent spelling, capitalization, or accents for authors' names or titles of papers. A modern tool should be able to automate the tasks of checking and fixing these common errors by enforcing consistency by default.

1.2 A modern bibliography tool

We have developed a new bibliographic management tool, CrossTEX, that addresses these problems. It unifies features from modern programming languages, databases, and bibliography management tools to solve problems authors and database maintainers have struggled with for two decades, and also adds convenient new features while being completely backwards-compatible to allow authors to use old BIBTEX databases unmodified.

Authors and maintainers have very different needs from a tool such as CrossTEX. Authors must conform to a variety of requirements on appearance, fields, and formatting, and so must have great flexibility with their bibliographic data. Database maintainers, on the other hand, should be able to specify everything just once, so they need look up and correctly enter conference locations, author names, and book editors just once, and can find them in one place to update and verify. Separating their jobs is also very important. Authors should need to do very little searching through the database for keys, because their concern is writing. Maintainers should be able to manage databases without concern for document styles, because independence from individual documents allows databases to be shared widely.

CrossTEX is designed not only to enable, but to encourage large, common databases so that authors can get on with their writing while the maintainers have clean databases to manage. As a start, it comes distributed with many large databases of the papers published at major computer science conferences, converted from the DBLP [13] project.

The most important aspect of CrossTEX is that an *object-oriented model* replaces the underlying relational model of BIBTEX. Every entry is an object, which contains fields and ultimately has a value itself; in BIBTEX, entries have fields, and `@string` objects have values, but this notion is not taken to the level of principle. One object can use the value of another by assigning its key to a field, by extension of the syntax for using `@string`s in BIBTEX. In addition, if the containing object leaves any fields unspecified, it will inherit them, if possible, from the other objects it refers to. For example, an object representing a conference could specify not only a value corresponding to the name of the conference, but include fields such as editor or location. Any articles that appear in the conference will simply use that value as a book title and then automatically inherit an editor and location without specifying anything extra. Those very familiar with BIBTEX will note that this is similar to the behavior of the special 'crossref' field, but has become a part of the way data is interpreted across the board rather than a special-purpose feature.

The object-oriented model enables CrossTEX databases to be concise and easily customized. For instance, most authors refer to conferences by their full names in formal journal papers, but abbreviate them otherwise; in CrossTEX, objects are flexible and have both long and short values. For example, "OSDI" and "Symposium on Operating System Design and Implementation" are two very different strings, but they are both names for the same conference. In BIBTEX, the database maintainer would have to choose just one value — but in CrossTEX, both can appear in the same `@conference` object, and the choice can wait until the author chooses stylistic options for each document.

Objects can also specify fields conditionally. Some information depends on context, especially for often-reused objects such as conferences and authors. If the location of a conference changes every year, it can be specified one way for 2006, and another for 2007. Conditional fields are inherited along with their conditions, allowing richly specified objects to adapt to the contexts where they are used. This enables powerful new idioms such as the ability to express everything about a conference over time in a single object.

CrossTEX supports new kinds of objects, including `@url`, `@patent`, and `@rfc`, that help modernize databases and allow them to be more precise, rather than depending on overly permissive, non-specific `@misc` objects. Adding other entirely new objects is easy as well. Due to the more precise taxonomy authors can apply stylistic options with better granularity because each kind of object represents one specific kind of reference.

CrossTEX provides fine-grained control over every aspect of the bibliography's presentation. From choosing between short and long conference names, states or journals to abbreviating author names, capitalizing titles or even pulling out hyperlinks in fields, the author has control with command-line switches. The LATEX files need not change at all and there is no need to write new style files for each combination of options.

CrossTEX provides new ways to keep track of citation keys to make them more meaningful and easier to find and remember. Objects are not restricted to just one key, but can be assigned any number of shorter or more descriptive keys, and even extended with new aliases after their definition. An even more powerful technique inspired by the `NbibTeX` [15] project is constrained citation: An author can cite a paper by Sirer and Walsh in 2006 as `\cite{!sirer-walsh:2006}` without worrying at all about what its key is in the database.

In addition to making consistency of data easy through object inheritance, CrossTEX automatically enforces consistency in other ways to avoid common errors in databases, such as inconsistent capitalization in titles. By default, it processes each title and re-capitalizes them to a consistent style that can be controlled by the author. CrossTEX carefully handles accents in author names, math in titles, and other complications to maintain consistency and a professional appearance even if the database is not perfect.

Maintainers will find many small but potent features for managing large databases, such as the ability to piece together many files with `@include` statements or specify the same field for groups of objects with `@default`. CrossTEX also allows objects to be updated far from their original definition, to permit anyone to correct and extend shared databases entirely without copying and pasting, even if they do not have permission to change the database itself.

CrossTEX is structured as a drop-in replacement for BIBTEX. Simply replace invocations of `bibtex` in the typesetting process with `crosstex`, and then incrementally update your databases to allow more and more consistency and reusability over time using CrossTEX's new features. The rest of this introductory paper will summarize some examples of these features and how they fit together to simplify the typesetting process.

2 Objects, inheritance, and conditions

CrossTEX enables a new idiom for managing conference information that exemplifies the usefulness of its object-oriented model, inheritance, and conditional fields. Information that is always true is specified first; fields specified in square brackets introduce conditions, such as a particular year for which the 'location' or 'month' fields have a particular value. The result is readable and intuitive, but more importantly collects information into a single object that can adapt to its context:

```
@conference{nsdi,
  shortname = "NSDI",
  longname = "Symposium on Networked System
              Design and Implementation",
  [year=2007] address=CambridgeMA, month=apr,
  [year=2006] address=SanJose, month=may,
  [year=2005] address=Boston, month=may,
  [year=2004] address=SF, month=mar,
}
```

The example defines the object `nsdi`, which typically has the value "Symposium on Networked System Design and Implementation" when assigned to a field. Additionally, however, this object is aware that if the referring context includes, for example, the year 2006 and does not already specify an address, the address will be assigned `SanJose`, a reference to a `@location` object representing the city in California. Thus, the papers in the proceedings of that conference need not specify that information at all, but simply pull in all redundant information from `nsdi`:

```
@inproceedings{credence,
  title = "Experience with an Object Reputation
           System for Peer-to-Peer Filesharing",
  author = "Kevin Walsh and Emin {G\"un} Sirer",
  booktitle = nsdi,
  year = 2006,
}
```

The assignment `booktitle = nsdi` is like specifying either `booktitle = "NSDI"` or `booktitle =`

"Symposium ...", depending on whether the author has chosen long or short names to be used for `@conference` objects. Additionally, precisely because the `credence` object does not specify an address or month, those fields will appear exactly as they do in the conference. This concision and consistency make conferences a perfect example of CrossTEX's flexibility from both the perspective of the database and the author.

Because CrossTEX encourages centralized objects with a lot of information, it can be very important to group them into meaningful databases and include them wherever they are needed with the `@include` statement. CrossTEX includes files from a standard path, including such databases as `dates`, `locations`, `conferences-cs` (which contains entries such as the one above for many of the important computer science conferences), `journals-cs`, and so forth. The `standard` database is always included when CrossTEX begins, which allows the administrator to include system-wide databases that must always be available. By default, `standard` includes `dates` (for backwards compatibility with BIBTEX and also because dates are so universally useful) and the databases containing the default rules for capitalizing titles.

Databases can also be found in the same directory as the document being processed. Because that directory is searched before the system directories, users can easily supply their own databases, even a `standard` database to control their own defaults. This flexibility allows maintainers and authors to work together to easily find, re-use, and adapt helpful databases.

3 Consistency and automation

Title case is one of the most common inconsistencies when using BIBTEX. Often, some papers appear with lower-case titles, some with all upper-case, and some with mixed title-case. Entries haphazardly capitalize key acronyms such as "BGP", and proper nouns such as "Internet".

CrossTEX ensures that all titles follow the same uniform capitalization standard, even if they appear in a wild variety of styles in the database. By default, the first letter of each word will become capitalized, the rest lower; this is the system known as "titlecase". CrossTEX is very careful to ensure the titles come out looking "good": It retains as-is words in StudlyCaps or CAPITALS, LATEX commands, and anything in math mode; compound words such as "Peer-to-Peer" are split into words, capitalized correctly, and re-assembled; and finally a list of known phrases are found and formatted. For example, any appearance of a string that is, regardless of case, equivalent to "Internet" will be capitalized as "Internet". CrossTEX determines these phrases at run-time in using `@titlephrase` commands:

```
@titlephrase "USENIX"
@titlephrase "Internet"
```

The standard include files define certain common computer science phrases such as these, but they can appear in any database. Small words, such as "a", "an", "the", etc. are also handled specially: They are made lower-case except at the beginning of the title or after certain punctuation, such as long dashes or colons. These, too, are determined at run-time by `@titlesmall` commands:

```
@titlesmall "a"
@titlesmall "the"
```

Again, the standard include files define common English small words.

Thus an example title with the default capitalization might appear as "Aardvark: A System for Peer-to-Peer BGP Routing on the Internet", despite messy or inconsistent capitalization in the database.

CrossTEX provides other capitalization options: With `--titlecase lower`, only the first letter of the title and those following punctuation are capitalized, the rest put into lower-case. All of the special cases for the default title-case still apply. Thus, the example title would appear as "Aardvark: A system for peer-to-peer BGP routing on the Internet".

With `--titlecase upper`, everything, even known phrases and small words, are put into upper-case, thus: "AARDVARK: A SYSTEM FOR PEER-TO-PEER BGP ROUTING ON THE INTERNET". Commands and math mode are still parsed and protected.

Finally, `--titlecase as-is` tells `crosstex` to allow titles to appear as they are specified in the database.

This philosophy of enforcing consistency by default makes it easier to achieve clean, professional appearance in bibliographies without any interference. At the same time, CrossTEX grants the user control with styles and run-time options and even fine-grained control over how specific phrases should appear throughout the bibliography.

4 Default fields

When databases get large and many elements have very similar fields — if, for example, they are all in the same conference or have the same informative `category` field — the CrossTEX command `@default` can help make them more concise and prevent typos by allowing the maintainer to specify that field just once for the whole group. For example, in the `nsdi.xtx` database, which contains the entries for papers published in NSDI, every entry will obviously have the same booktitle: `nsdi`. Because all of the entries are from the DBLP [13] project, they also share bibsource fields. Finally, they are grouped in the database by year. With default values for fields, maintainers can significantly shorten the database and save effort and typos with the following (line break is editorial):

```
@default booktitle = nsdi
@default bibsource
         = "DBLP, http://dblp.uni-trier.de"
@default year = 2004
```

From this point in the file until the end, entries will by default contain these default fields where they are relevant. Because of this factorization and the inheritance from `nsdi`, most papers now simply state author, title, and pages fields. Later in the file, however, are entries with different years. A new `@default` command takes precedence over the first:

```
@default year = 2003
```

The booktitle and bibsource defaults are unchanged, but now the year defaults to 2003. As with field values inherited from referenced objects, field values inherited from `default` definitions have lower precedence than those specified in the object. If it is desirable to allow an object to appear with no year but not use the default, either put it before the first default specifying the year field, or remove the default by explicitly assigning `@default year = ""`. Defaults are easily overridden and serve only to simplify and shorten the database.

5 Extending objects

Occasionally it is useful to add information to an object that already exists. For example, an author must cite a paper that appeared in NSDI 2005, but the system database only has information about the NSDI conference up to 2004. Obviously, the best solution is to add the following line to fill in the entry in the system conferences database:

```
[year=2005] address=Boston, month=may,
```

However, the author might not have permission to edit the database. Two options come to mind: Cut-and-paste the `nsdi` object into some local database with a new name so there is no conflict, or put the address and month directly into the paper's entry rather than relying on inheritance. Neither is a good solution. More desirable is to be able to add information to the `nsdi` object, even though it has already been defined in another database, using the CrossTEX `@extend` feature:

```
@extend{nsdi, [year=2005] address=Boston,
        month=may}
```

An `@extend` entry looks just like an object definition. However, rather than defining a new object, CrossTEX will find the specified object and re-build it with the information provided, retaining its old fields where they are not changed.

Just as object definitions can specify multiple keys to alias the same object, so can `@extend` statements. If any keys specified do not yet refer to an object, they will be created; however, CrossTEX will report an error rather than change a key that already points to a different object than the one being extended, so it is not possible to accidentally break extant keys. Authors can take advantage of this to define shorter, easier-to-remember names for database objects even when no fields need changing.

6 Constrained citations

CrossTEX borrows from `nbibTeX` the very useful notion of *constrained citation*. Constrained citations enable an author user to cite a work by specifying pieces of information that uniquely identify it. For instance, consider a reference to a paper written by Emin Gün Sirer in 1999 on how to split up virtual machines, which appeared at SOSP. The author could search the database for some terms that will appear in the entry (e.g. 1999, sosp, sirer), copy the key for the entry, and issue a plain citation using that precise key. This is what many BIBTEX users do without thinking; however, with constrained citation, CrossTEX will search on your behalf.

A constrained citation begins with an exclamation point, and specifies a series of colon-separated terms that identify the reference being cited. Some examples of constrained citations (line break is editorial):

```
\cite{!author=sirer:title=virtual:year=1999}
\cite{!author=sirer:title=virtual
      :title=machines:year=1999}
\cite{!author=sirer:author=walsh:year=2006}
```

Colons separate constraints. Each constraint identifies a field that the reference must have, as well as a string that should appear somewhere within that named field. So `author=smith` will match both "Smith" and "Smithson".

Sometimes, multiple constraints apply to the same field. Specifying the same field multiple times, as in the second and third examples above, is perfectly acceptable, but tedious. Instead, CrossTEX provides a way to specify multiple constraints for the same field: Every word separated by a "-" sign is treated as a separate constraint. Thus the examples above can appear as:

```
\cite{!author=sirer:title=virtual:year=1999}
\cite{!author=sirer:title=virtual-machines
      :year=1999}
\cite{!author=sirer-walsh:year=2006}
```

Multiple constraints within a given field are not ordered and can appear anywhere in the string, so "virtual-machines" will match "virtual machines", as well as "machines virtual", and even "building a machineshop virtually".

Several shorthands make constrained citations even easier to specify by providing defaults for fieldnames. If the fieldnames are missing, the first constraint defaults to "author". The second constraint defaults to "title" if the value is not numeric; if it is, it defaults to "year". Finally, the last constraint defaults to "year". So the examples above can be even shorter:

```
\cite{!sirer:virtual:1999}
\cite{!sirer:virtual-machines:1999}
\cite{!sirer-walsh:2006}
```

Two caveats are worth remembering about constrained citations. First, the citation needs to be uniquely identifiable. If the constraints match more than one object, CrossTEX prints an error and identify the matching objects. The author can then specify more constraints until the reference is correct or switch to a plain citation based on the search CrossTEX performed. Second, due to a limitation in LATEX, referring to the same work through different constraints (e.g. `!sirer:virtual:1999` and `!sirer:virtual-machines:1999`) will cause CrossTEX to flag an error so the citation does not appear twice in the references section, because LATEX would think it was two different works. For each work, one must decide on a set of constraints and use them consistently throughout a document.

Overall, constrained citations are a convenient way to cite papers without having to look anything up; they fit naturally to the way people recall citations.

7 Extending CrossTEX

CrossTEX is designed to be easy to extend with only trivial knowledge of Python. New objects or fields are defined by editing the standard objects module `crosstex.objects`, which will be found wherever CrossTEX's library files are installed as `crosstex/objects.py`.

To create a new field for a particular object type, find its definition (e.g. the section defining the `@string` object begins `class string`). Most objects already define some fields; simply copy that syntax for your own field. To create an entirely new class `@foo` which is identical to a current one named `@bar`, add the following to the end of the list of objects:

```
class foo(bar):
    pass # 'pass' is only necessary
         # if no fields are defined.
```

Fields are defined as optional or required by assigning them the values `OPTIONAL` and `REQUIRED`, respectively. To make an optional field required or a required field optional, simply assign it the new value in the class where you want the change. To allow a field to inherit its value from another field in the same object if left blank, assign a string containing the name of the other field. A list containing `OPTIONAL`, `REQUIRED`, and one or more string field names will be processed and define several sibling fields and the given requirement level. For example:

```
class foo(bar):
    baz = REQUIRED
    blah = OPTIONAL
    quux = [REQUIRED, 'baz', 'blah']
```

This defines a new kind of object `@foo`, which behaves the same as `@bar`; additionally, the 'baz' field is required, the 'blah' field is optional, and the 'quux' field is required but if unspecified will try to take its value from 'baz' or 'blah' in that order.

Styles are defined in small Python modules in the `style` directory in the same place you found `objects.py`. There you will find the default styles, `plain.py`, `full.py`, etc. Styles are built up with small filter functions, many of which are provided in `crosstex.objects`. Each field is filtered through four phases:

- Production, in which an initial value is generated from the object itself;
- List filtering, if the value is a list;
- List formatting, to turn the list into a string for the final step; and
- Filtering, in which zero or more filters modify the value into its final form.

As a starter, consider the following statements taken from existing styles:

```
misc._addproducer(emptyproducer, 'label')
conference._addfilter(proceedingsfilter,'value')
misc._addfilter(emphfilter, 'fullpublication',
                'booktitle')
```

The first line states that the label attribute of any `@misc` object (or any object of a type derived from `@misc`) can be produced by `emptyproducer` if that function returns anything other than `None`. (`emptyproducer` is defined in `crosstex.objects` and always returns an empty string, which in the case of labels causes LaTeX to default to numeric citation.)

The second line causes the value of objects derived from `@conference` to be filtered through a function that prepends 'Proceedings of the' to the value.

The last line filters the 'booktitle' field of objects derived from `@misc`, but only when used within the 'fullpublication' field (which happens to be a virtual field defined solely by attaching producers to it).

It is important to note that filters and producers are applied starting from the most recent, so later producers will take precedence and later filters will be nested inside earlier filters. The standard styles are well-commented and should provide a good start towards extending CrossTeX with new stylistic features.

Finally, because CrossTeX also searches for styles in the same directory as the file being processed, one can develop styles without editing anything installed system-wide. This can be useful for personal, per-paper, or experimental styles, or even extending CrossTeX without the privileges of the system administrator.

8 Related work

BibTeX is the dominant tool in the TeX world to automatically format bibliographies. The corresponding tool in `roff` typesetting is `refer`, which uses a similar relational model and a more concise but slightly less user-readable database format. `refer` also introduced the notion of semantic (constrained) citation to computer typesetting. `nbibTeX` [15] is a true drop-in replacement for BibTeX that supports the same database language and style files, but adds support for semantic citations in order to assist multiple authors working together from a large database. EndNote [3] is a commercial product that manages databases for Microsoft Word; it uses a GUI database editor and its own database format, adapted from that of `refer`.

Because of BibTeX's dominance but lack of easy automation, the community has developed numerous database editors and other tools to support it. Database editors such as Pybliographer [7], KBibTeX [6], and JabRef [5] are relevant to CrossTeX because they attempt to solve some of the same problems without changing BibTeX and also because, since CrossTeX is backwards-compatible with their BibTeX output, one can take advantage of both.

Other tools for formatting bibliographies for publication on the web exist, such as the BibTeX-XML-HTML [9] project, which provides a tool for converting bibliographies to HTML documents by first converting them into XML. `xtx2html` has the advantage of integrated styling using the same methods as styling documents themselves.

Many projects address the need for large central databases, including the DBLP [13] project, which has so far assembled bibliography entries in BibTeX format for more than 870,000 publications in computer science, some of which have been converted to be distributed with CrossTeX. Other large databases and integrated search engines include CiteSeer [2], Google Scholar [4], and arXiv [10]. RefDB [8] is an approach to the actual sharing of databases, which allows users to share bibliographies over the network using SQL databases and the RIS bibliography format. However, all of these efforts are focused on providing authors with the ability to find a reference in order to copy-and-paste it into their own local database — in short, assisting with looking up the information but not solving the problem of centralizing information because of the limitations of BibTeX. CrossTeX is designed to address this problem and allow vast databases such as these projects to be easily incorporated directly into a document, as well as assisting those who must keep the databases up-to-date.

9 Conclusions

We have presented CrossTeX, a modern bibliography management tool based on and replacing BibTeX. CrossTeX solves a number of problems in BibTeX, including its relational model that requires duplication of information, the dependence of presentation details such as abbreviation on choices made in the database, arbitrary object keys, and an impenetrable style language.

The primary contribution of CrossTeX is its object-oriented database language, which brings the power of inheritance to bear on the goal of specifying information in only one place, to be used and adapted everywhere it is needed. The many kinds of CrossTeX objects allow useful fields to be bundled together, assigned both long and short names, and inherited in different forms throughout the database. New kinds of objects allow precise semantics and self-explanatory databases. Providing full author names and both long and short versions of strings allow typesetting-time decisions about style without modifying the database.

Conditional fields allow data that depend on context, such as locations of a conference by year, to be collected together into one place so they are visible and useful together. Because such conditional fields are also inherited, objects automatically adapt to the contexts in which they are used.

CrossTEX enforces consistency by using a sophisticated and flexible algorithm to guarantee consistent capitalization in paper titles and applying abbreviation decisions across every object. Automation of tedious tasks such as abbreviation and capitalization prevents human error while being easy to customize.

Backwards compatibility with BibTEX combined with fundamentally new idioms and powerful semantics make CrossTEX easier to use and less error-prone than its predecessor. Overall, CrossTEX makes consistency and professional appearance easy to achieve, both in databases and typeset documents.

References

[1] CS Bib. `http://liinwww.ira.uka.de/bibliography/index.html`. Accessed June 15, 2007. The Collection of Computer Science Bibliographies.

[2] CiteSeer. `http://citeseer.ist.psu.edu/`. Accessed June 15, 2007. Scientific literature digital library.

[3] EndNote. `http://www.endnote.com/`. Accessed June 15, 2007. A PC product that does what BibTEX does, for Word on Windows.

[4] Google Scholar. `http://scholar.google.com/`. Accessed June 15, 2007. Search engine dedicated to scientific publications on the web.

[5] JabRef. `http://jabref.sourceforge.net/`. Accessed June 15, 2007. A graphical database editor for BibTEX based on Java.

[6] KBibTEX. `http://www.unix-ag.uni-kl.de/~fischer/kbibtex/screenshots.html`. Accessed June 15, 2007. A graphical database editor for BibTEX.

[7] Pybliographer. `http://www.pybliographer.org/`. Accessed June 15, 2007. A Python tool for managing bibliographic databases.

[8] RefDB. `http://refdb.sourceforge.net/`. Accessed June 15, 2007. RefDB is a reference database and bibliography tool for SGML, XML, and LATEX/BibTEX documents that allows users to share databases over a network.

[9] BibTEX-XML-HTML Project. `http://www.authopilot.com/xml/home.htm`. Accessed June 15, 2007. Transforms BibTEX databases into HTML by way of XML.

[10] arXiv. `http://arxiv.org`. Accessed June 15, 2007. E-prints in physics, mathematics, computer science and quantitative biology.

[11] Donald E. Knuth. *The TEXbook.* Addison-Wesley, Reading, Massachusetts, 1984.

[12] Leslie Lamport. *LATEX: A Document Preparation System.* 2nd Edition, Addison-Wesley, 1994.

[13] Michael Ley. DBLP. `http://www.informatik.uni-trier.de/~ley/db/`. Accessed June 15, 2007. DBLP is a huge effort by a dedicated team that has so far assembled bibliographic entries for 830,000 publications in computer science. The databases shipped with CrossTEX are derived from DBLP.

[14] Oren Patashnik. BibTEXing. February 1988.

[15] Norman Ramsey. NbibTEX. `http://www.eecs.harvard.edu/~nr/nbibtex/`. Accessed June 15, 2007. The origin of constrained citation.

Typesetting tables with LaTeX

Klaus Höppner
Haardtring 230 a
64295 Darmstadt
Germany
`klaus.hoeppner (at) gmx dot de`

Abstract

From a LaTeXoligist's point of view, LaTeX is a perfect tool to typeset nearly everything in a beautiful manner. Without any doubt, LaTeX can typeset tables, but it is easy to produce bad tables with ugly lines and text touching the lines. This talk is intended to introduce how to typeset tables with LaTeX on a beginners' level, mentioning some typographic aspects, showing some packages that help the author in formatting tables and concluding with how to typeset tables with page breaks.

1 Basic tables

LaTeX already has built-in support to typeset tables. For beginners it may be a bit confusing, since LaTeX provides two environments: `tabular` and `table`. To typeset material in rows and columns, `tabular` is needed, while the `table` environment is a container for floating material similar to `figure`, into which a `tabular` environment may be included.

So, let's have a look how to typeset a simple table:

```
\begin{tabular}{lcr}
a   & b   & c\\
aa  & ab  & ac\\
aaa & aab & aac
\end{tabular}
```

will result in

a	b	c
aa	ab	ac
aaa	aab	aac

The rows of the table are divided by LaTeX's usual `\\` command (in some cases, it may be needed to use `\tabularnewline` instead, as we will see later in this article). Columns are separated by `&`, the ampersand character.

The required argument of `\begin{tabular}` defines the basic layout of the table, especially the alignment of the columns:

l left aligned column

c centered column

r right aligned column

p{⟨*width*⟩**}** paragraph-like column of a predefined width (with the baseline of the paragraph's first line aligned relative to the other cells in the table row)

The normal space between columns, which is also added before the first and after the last column, may be overridden by `@{`⟨*sep*⟩`}`, where ⟨*sep*⟩ is any LaTeX code, inserted as the separator. For illustration, let's typeset some flight data:

flight no.	route	
LH 402	Frankfurt–Newark	
KL 3171	Amsterdam–Cork	
US 1152	San Diego–Philadelphia	

Here, the `@` command is used twice: The space that normally would have been inserted left of the first column is replaced by nothing, thus the table is left aligned with the surrounding text (compare it with the first tabular example in this article, you will see the difference). Additionally, the inter-column space between the points of departure and destination is replaced by a dash. So the code used to produce this table looks as follows (silently introducing `\multicolumn` to combine cells):

```
\begin{tabular}{@{}lr@{--}l}
flight no. & \multicolumn{2}{c}{route}\\
LH\,402 & Frankfurt & Newark\\
KL\,3171 & Amsterdam & Cork\\
US\,1152 & San Diego & Philadelphia
\end{tabular}
```

2 Extra packages for typesetting tables

Beyond LaTeX's built-in ability to typeset tables, several extra packages exist. Some of them add new effects in typography and layout, others simplify the task of writing the document's source code. The packages that I will introduce in this article (and more that I won't) are covered in detail in the *LaTeX Companion* [8].

Here are some important packages for authors who want to typeset tables:

array adds paragraph-like columns `m{⟨width⟩}` and `b{⟨width⟩}` similar to the `p`-column, but vertically aligned to the center or bottom. Additionally, the package allows defining command sequences to be executed before or after the contents of a column.

tabularx typesets a table with fixed widths, introducing the column type `X` that works like a `p`-column with automatically calculated width.

booktabs provides fancy commands for horizontal lines with appropriate spacing above and below.

ctable we won't discuss this one, but have a look on CTAN, it's a modern table package with many nice features.

2.1 Using array

From my point of view, the most important feature of `array` [7] is the new possibility of defining commands that are added automatically before or after a column's cell. It saves a lot of typing, making the source code more concise and more flexible. `array` is one of the required LaTeX packages, so it must be part of any LaTeX installation.

For a simple example, have a look at the following table:

Command	Symbol
`\alpha`	α
`\beta`	β
`\gamma`	γ

The left column displays LaTeX commands, beginning with a backslash and using a typewriter font, while the right columns displays the corresponding math symbol. Using the `array` package, the source code is pretty straightforward:

```
\begin{tabular}%
    {>{\ttfamily\char'\\}c>{$}c<{$}}
\multicolumn{1}{c}{Command} &
    \multicolumn{1}{c}{Symbol}\\
\hline
alpha & \alpha\\
beta & \beta\\
gamma & \gamma
\end{tabular}
```

As shown in this code, we can now define a command sequence inside the `>{...}` preceding the column type definition. These commands are executed before each cell of that column. Similarly, using `<{...}` after the column type defines which commands to be executed after typesetting the column's cells.

In the example above, the first row is different from the others, since it contains the column titles that obviously should not be typeset in typewriter font or math mode. This is handled by 'abusing' the `\multicolumn` command, which prevents the `>` and `<` command hooks from being applied for these cells.

Another use of these command hooks is typesetting paragraphs in narrow columns. LaTeX typesets these paragraphs left and right justified by default, but in narrow columns it is often more appropriate to typeset them using `\raggedright`. So we might think of trying the following code:

```
\begin{tabular}{l>{\raggedright}p{3in}}
```

Unfortunately this fails when ending the table rows with the `\\` command, with rather weird error messages about misplaced aligns. The problem is that `\raggedright` redefines `\\`, so it can't be recognized as the end of table rows. There are three solutions for this problem:

1. Use `\tabularnewline` instead of `\\`. In fact, it does no harm to always use this, even when you don't have problems with `\raggedright`.
2. Restore the original definition of `\\` by using the command `\arraybackslash`, as follows:

   ```
   \begin{tabular}%
       {l>{\raggedright\arraybackslash}p{3in}}
   ```
3. Use the `ragged2e` [9] package. It redefines the command `\raggedright` to prevent it from redefining `\\`, so the problem disappears without any further change to the original code. Additionally, `ragged2e` provides the new command `\RaggedRight` that typesets the paragraph left aligned, but doesn't disable hyphenation.

2.2 Using tabularx

Besides the normal `tabular` environment, a rarely used environment `tabular*` exists. In addition to the column definition, it takes a table width as argument. The resulting table is typeset to this width, but often — surprisingly — it expands the space between columns.

A more convenient implementation is done by the `tabularx` [5] package (another required LaTeX package present in every LaTeX installation). This introduces a column type `X` for paragraph-like columns whose width is automatically calculated in order to achieve a table of a desired total width. For example, let's look at the following:

```
\begin{tabularx}{\linewidth}{lX}
Label & Text\\
\hline
One & This is some text without meaning,
  just using up some space. It is not
  intended for reading.\\
...
\end{tabularx}
```

This produces a table across the full line width, where the right column just uses the space remaining after typesetting the left column:

Label	Text
One	This is some text without meaning, just using up some space. It is not intended for reading.
Two	This is another text without meaning, just using up some space. It's not intended for reading either.
Three	This is yet another text without meaning. Guess what? It's not intended for reading. It is just there.
Four	How often did I mention that you should not read this text?

It is possible to use more than one `X`-column. By default, all of them are typeset to the same width, but it is possible to manually adjust how the available space is divided. Here's our next example:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space. It is not meant for reading either.

This table was produced with the following code:

```
\begin{tabularx}{\linewidth}%
 {l>{\setlength\hsize{0.6\hsize}\raggedright}X%
   >{\setlength\hsize{1.4\hsize}\raggedright}X}
Label & Text & More text\tabularnewline
\hline
...
\end{tabularx}
```

When balancing the column widths manually, it is important that the `\hsize` fractions add up to the number of `X`-columns, as in the example above, where $0.6+1.4=2$. To achieve *automatic* balancing of columns, take a look at the `tabulary` package.

Be aware that the way `tabularx` parses the contents of a table limits the possibility of defining new environments based on the `tabularx`. If you consider doing this, first look at the documentation.

3 Using lines in tables

LaTeX provides the possibility of using lines in tables: vertical lines are added by placing a `|` at the appropriate position in the definition of the column layout, and horizontal lines are added by using the command `\hline`.

While using lines in tables can help the reader in understanding the contents of a table, it is quite easy to produce really ugly tables like the following:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Though nobody would typeset this particular table in real life, it illustrates a general and common problem — the column titles and the word "another" in the rightmost column touch the horizontal lines above them.

As a first step to improve the spacing between the table rows and the horizontal lines in such cases, set `\extrarowheight` to a non-zero length, e. g. to 4 pt. If this isn't enough, additional adjustment may be done by adding invisible rules. Here is revised source code for the above example illustrating both these points:

```
\setlength{\extrarowheight}{4pt}
\begin{tabularx}{\linewidth}%
    {|l|>{\setlength\hsize{0.67\hsize}}X%
       |>{\setlength\hsize{1.33\hsize}}X|}
\hline
\Large Label & \Large Text
    & \Large More text\tabularnewline
\hline
\hline
One & This is some text without meaning.
    & \rule{0pt}{18pt}%
      This is {\huge another} text without meaning,
      just using up some space.\\
\hline
\end{tabularx}
```

we get a somewhat better result:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

Please notice that the `\rule` used as an additional spacer was typeset with a horizontal width of 0.4 pt instead of 0 pt (as shown in the code) in order to make its effect and location visible.

Even after this, the layout of the table still looks quite poor, e. g. the broken vertical lines between the double horizonal line. This might be solved with the package `hhline` [2], but for typesetting tables with pretty lines, have a look at the `booktabs` [6] package. It starts by giving users a basic piece of advice, namely to avoid vertical lines, and introduces commands to typeset horizontal lines with appropriate thickness and spacing. Using `booktabs`, the source code for our weird example now looks as follows:

```
\begin{tabularx}{\linewidth}%
    {l>{\setlength\hsize{0.67\hsize}}X%
      >{\setlength\hsize{1.33\hsize}}X}
\toprule
\Large Label & \Large Text
    & \Large More text\tabularnewline
\midrule
One & This is some text without meaning.
    & This is {\huge another} text without meaning,
      just using up some space.\\
\bottomrule
\end{tabularx}
```

Using this, the result becomes:

Label	Text	More text
One	This is some text without meaning.	This is another text without meaning, just using up some space.

At last, we've improved the layout of the table quite a bit. The content with arbitrary changes of font size still looks weird, but that's something for which the author and not LaTeX must be blamed.

For a more realistic example of using rules, here I present an example from the booktabs manual:

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

4 Typesetting tables across multiple pages

The usual `tabular(x)` environment is restricted to single-page tables, i. e. no page breaks are allowed within tables.

However, two extension packages provide support for typesetting tables across multiple pages, namely `longtable` [3] and `supertabular` [1]. The main difference between them is that `longtable` keeps the column widths the same throughout the entire table, while `supertabular` recalculates the column widths on each page. According to the documentation, `longtable` doesn't work in two- or multi-column mode, and I didn't try `supertabular` on this case. The syntax of the two packages is different, so one has to decide which one to use.

Let's have a look at the general structure of a `longtable`:

```
\begin{longtable}{ll}
Label (cont.) & Text (cont.)\\
\endhead
\multicolumn{2}{l}{This is the first head}\\
Label & Text\\
\endfirsthead
\multicolumn{2}{l}{to be cont'd on next page}
\endfoot
\multicolumn{2}{l}{this is the end (finally!)}
\endlastfoot
One & Some content\\
Two & Another content\\
Three & Yet another content\\
[...]
\end{longtable}
```

As shown in this source code, the `longtable` environment may contain definitions for headers and footers before the normal content of the table:

- `\endfirsthead` defines what to typeset at the very first head of the table,
- `\endhead` defines a table head that is used on continuation pages,
- `\endfoot` defines what to typeset on the foot of the table if a continuation page follows, and
- `\endlastfoot` defines the foot at the very end of the table.

I personally prefer `longtable` simply because there exists yet another package `ltxtable` [4], which combines `longtable` and `tabularx`. It provides the command `\LTXtable{`⟨*width*⟩`}{`⟨*file*⟩`}`, which reads a given file containing a `longtable` environment using `X`-columns and typesets it to the requested width.

References

[1] Johannes Braams and Theo Jurriens. The `supertabular` package. `CTAN:macros/latex/contrib/supertabular/`.

[2] David Carlisle. The `hhline` package.

[3] David Carlisle. The `longtable` package.

[4] David Carlisle. The `ltxtable` package. `CTAN:macros/latex/contrib/carlisle/`.

[5] David Carlisle. The `tabularx` package.

[6] Simon Fear and Danie Els. The `booktabs` package. `CTAN:macros/latex/contrib/booktabs/`.

[7] Frank Mittelbach and David Carlisle. The `array` package.

[8] Frank Mittelbach, Michel Goossens, et al. *The LaTeX Companion.* Addison-Wesley, 2nd edition, 2004.

[9] Martin Schröder. The `ragged2e` package. `CTAN:macros/latex/contrib/ms/`.

Packages [2, 3, 5] are part of the required LaTeX tools, available from `CTAN:macros/latex/required/tools`.

Programming with PerlTEX

Andrew Mertz, William Slough
Department of Mathematics and Computer Science
Eastern Illinois University
Charleston, IL 61920
aemertz (at) eiu dot edu, waslough (at) eiu dot edu

Abstract

PerlTEX couples two well-known worlds — the Perl programming language and the LATEX typesetting system. The resulting system provides users with a way to augment LATEX macros with Perl code, thereby adding programming capabilities to LATEX that would otherwise be difficult to express. In this paper, we illustrate the use of PerlTEX with a variety of examples and explain the associated Perl code. Although Perl may perhaps be best known for its string manipulation capabilities, we demonstrate how PerlTEX indirectly provides support for "programming" graphics through the use of additional packages such as Ti*k*Z.

1 Introduction

The typesetting capabilities of TEX and LATEX are well known. Each has the ability to define macros, adding significant flexibility and convenience. However, to achieve some effects in TEX requires a level of expertise many users lack.

Perl [8] is a programming language with particular strengths in string processing and scripting. Since it borrows concepts from other languages such as C and SED, its syntax is likely to be reasonably familiar to many.

PerlTEX [4] provides a way to incorporate the expressiveness of Perl directly within a LATEX document. In doing so, the computing capabilities of Perl are coupled with the document preparation abilities present in LATEX. Combining these two systems has an important outcome: for those who already know or are willing to learn some of Perl's rudiments, a number of typesetting tasks become more convenient to express.

2 A first example

In Chapter 20 of *The TEXbook* [3], a TEX macro which generates the first N prime numbers is described, where N is a specified parameter of the macro. This discussion and macro earn Knuth's double dangerous-bend symbols, a warning to readers that esoteric topics are under discussion.

It is probably fair to say that the design of such a macro using the primitives of TEX is a task best left to experts. A simpler approach uses PerlTEX. Figure 1 shows the details.

To use this command, we request the prime numbers within a specified interval. For example,

```
\perlnewcommand{\listPrimes}[2] {
  use Math::Prime::XS "primes";
  return join(" ", primes($_[0], $_[1]));
}
```

Figure 1: Definition of a command, `\listPrimes`, to generate prime numbers. The two arguments of this command specify the desired range of values to be generated.

```
\listPrimes{10}{30}
```

generates the typeset sequence

11 13 17 19 23 29

consisting of the prime numbers between 10 and 30.

Let's take a closer look at how this is accomplished. To begin, we use `\perlnewcommand`, the PerlTEX analog of the `\newcommand` of LATEX. We thus define a new command, `\listPrimes`, with two arguments.

In contrast to the `\newcommand` of LATEX, Perl code is placed within the definition portion of a `\perlnewcommand`. For the current example,

```
use Math::Prime::XS "primes";
```

imports a Perl module which contains a function, named `primes`, which is perfectly suited to the task at hand. Given a pair of values which specify the desired range, this function returns a list of the primes in that range.

The arguments of `\listPrimes` are accessed with the Perl notation `$_[0]` and `$_[1]`. Thus,

```
primes($_[0], $_[1])
```

yields a *list* of the desired primes. To complete the definition of `\listPrimes`, a single *string* must be created from the collection of primes just obtained.

This is easily achieved with Perl's join. Here, we use a single space to separate adjacent primes.

In this example, the use of an existing Perl function, primes, avoids "reinventing the wheel". Since there are many such functions available, this is one direct benefit of PerlTEX. A wealth of Perl functions can be located by consulting CPAN, the comprehensive Perl archive network, found at www.cpan.org.

3 Variations on a theme

Rather than produce one sequence of primes, as in our first example, now suppose a tabular array of primes is desired. We will define a new command, \tablePrimes, with three arguments: the first two specify the desired range of primes, as before, and the third argument indicates the number of columns to be used. For example, the command

```
\tablePrimes{1}{20}{3}
```

will produce a table consisting of the primes between 1 and 20, typeset in three columns.

We will show two different definitions for this command. The first solution uses a direct approach, illustrating how the looping and conditional control structures of Perl can be used to generate the required LATEX code. In the second solution the power of regular expressions is used to achieve the same result, avoiding the need for explicit looping and testing.

Before taking up the Perl details, let's consider the desired LATEX code to be generated. For the three-column example above, the following needs to be generated:

```
\begin{tabular}{*{3}{r}}
2 & 3 & 5\\
7 & 11 & 13\\
17 & 19 &
\end{tabular}
```

Of course, this is just a tabular environment consisting of the primes to appear within the table. It is helpful to think of this as one string, subdivided into three parts: the beginning of the environment, the environment content, and the end of the environment. The first definition of \tablePrimes, shown in Figure 2, reflects this view.

Consider the final return statement in this definition. Using Perl's concatenation or dot operator, three string components are formed to yield the desired tabular environment. In the first component, $_[2] is used to obtain the value of the third parameter, the number of columns. Each backslash which is to appear in the generated LATEX code must also be escaped in Perl to avoid its usual meaning. So, for example, \\begin appears in order to obtain \begin. Without escaping the backslash, Perl would interpret \b as a backspace. The use of \n in this return statement ensures that each component begins on a new line.

```
\perlnewcommand{\tablePrimes}[3] {
 use Math::Prime::XS "primes";
 my $count = 0;
 my $primes = "";
 foreach my $item (primes($_[0], $_[1])) {
   $primes .= $item;
   $count++;
   if ($count == $_[2]) {
     $primes .= "\\\\ \n";
     $count = 0;
   }
   else {
     $primes .= " & ";
   }
 }
 return "\\begin{tabular}{*{$_[2]}{r}}\n" .
        $primes . "\n" .
        "\\end{tabular} \n";
}
```

Figure 2: Definition of a command, \tablePrimes, to generate a table of prime numbers.

At this point in the definition, the Perl variable $primes contains the string of all primes needed for the table, with & column separators and \\ row separators inserted as appropriate. Everything in the definition prior to the return statement is present to generate this string.

This portion of the definition is straightforward, though a few comments might be helpful. The keyword my is used when Perl variables are introduced to indicate they are *local*, which is generally a good idea to prevent unintended interactions.

The variable $primes begins as an empty string and grows to include all of the needed values to appear in the table. Perl's compound operator .= is used to append a new value to this string. We use the foreach construct to iterate over all of the primes generated, appending each in turn to the $primes string. Column or row separators are appended to this string by keeping count of which column has just been added to the string. As before, some care is needed regarding the escape character. For example, \\\\ is used to generate \\.

The second definition for \tablePrimes takes a different viewpoint of the generation of $strings. A two-step process is used to generate the value for the tabular environment. As a first step, a &-separated string of primes is constructed:

```
my $primes = join("&",
                  primes($_[0], $_[1]));
```

This generates all primes, incorrectly assuming that the tabular will consist of one very long row. The second step corrects this assumption by replacing every kth column separator with a row separator. This can be achieved using regular expressions:

```
$primes =~ s/((\d+&){$k}\d+)&/
           $1\\\\ \n/g;
```

Though this might be viewed as somewhat cryptic, the use of regular expressions is one of the widely quoted strengths of Perl and can be used, as here, to concisely describe a pattern substitution. Putting these ideas together yields the definition shown in Figure 3.

```
\perlnewcommand{\tablePrimes}[3] {
 use Math::Prime::XS "primes";

 # Number of ampersands needed per line
 my $k = $_[2] - 1;

 # Build a string of &-separated primes
 my $primes = join("&",
                   primes($_[0], $_[1]));

 # Insert newlines for each row
 $primes =~ s/((\d+&){$k}\d+)&/$1\\\\ \n/g;

 # Put the pieces together
 return "\\begin{tabular}{*{$_[2]}{r}}\n" .
        $primes . "\n" .
        "\\end{tabular} \n";
}
```

Figure 3: Alternate definition of `\tablePrimes`. A regular expression is used to subdivide the primes into rows.

4 Layout and processing

The layout of a LaTeX document which uses PerlTeX is straightforward. Within the preamble

```
\usepackage{perltex}
```

loads the PerlTeX package. Also in the preamble, one or more PerlTeX commands are defined with `\perlnewcommand`. Within the document environment, the PerlTeX commands which were defined can be utilized. Figure 4 shows an example.

Processing a PerlTeX source file requires the services of both Perl and TeX. This is accomplished using a script provided with PerlTeX. For example, the command

```
perltex foo.tex
```

```
\documentclass{article}
...
\usepackage{perltex}
...

\perlnewcommand{\tablePrimes}[3]{
    definition
}

\begin{document}
   ...
   \tablePrimes{1}{20}{3}
   ...
   \tablePrimes{1}{20}{4}
   ...
\end{document}
```

Figure 4: Sample layout of a LaTeX document intended for PerlTeX. The definition of `\tablePrimes` is omitted here.

processes the source file `foo.tex`. As shown in Figure 5, this initiates the processing by creating a pair of communicating processes, one each for TeX and Perl, ultimately creating the output file.

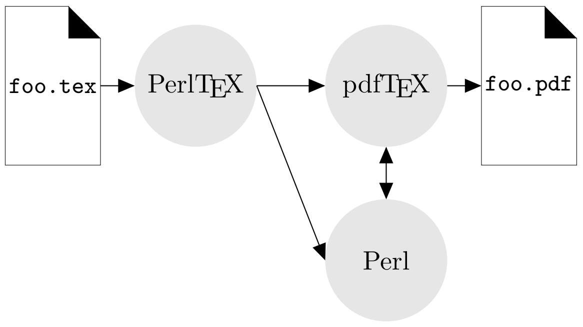


Figure 5: Processing a source file with PerlTeX.

By default, PerlTeX causes the Perl processing to use a secure sandbox, insulating the user from potentially dangerous actions, such as removal of directories or other undesirable system-related actions. If this is not desired, the command

```
perltex --nosafe foo.tex
```

disables the sandbox. Disabling the sandbox is helpful when importing Perl modules, accessing files or the network, and in many other cases.

Introducing Perl code provides new opportunities for errors. To assist with debugging, PerlTeX creates a log file with the suffix `lgpl` which contains all of the Perl and LaTeX code generated during processing. Any error messages returned by the Perl interpreter also appear in this file.

5 Graphical output

One appealing feature of PerlTEX is its ability to interact with other packages. To see an example of this, consider the triangular array of binomial coefficients — more popularly known as Pascal's triangle. If a fixed modulus m is selected, each entry of the triangle can be reduced, modulo m. Each of the m possible remainder values can be assigned a color, providing a way to visualize the coefficients as a multi-colored graphic. A number of very attractive diagrams of this sort with varying moduli appear in *Chaos and Fractals* [5]. In this section, we use Ti*k*Z [6] to take care of the graphical aspects, while using PerlTEX to generate the numerical values of Pascal's triangle.

As a starting point, Figure 6 defines a PerlTEX command, `\pascalMatrix`, which generates a tabular array of binomial coefficients. Its single argument specifies the size of the triangle. (For an argument n, the rows of the array are numbered 0 through n.) For example, `\pascalMatrix{5}` yields the triangular array:

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

As in previous examples, the `return` statement in this definition is responsible for creating the entire tabular environment. In this case, the variable `$tabularRows` contains all of the rows needed for Pascal's triangle.

Each iteration of the outer loop appends one complete row to `tabularRows`. The inner loop is responsible for generating row $r+1$ given the contents of row r, using a well-known identity for binomial coefficients: $\binom{r+1}{c} = \binom{r}{c-1} + \binom{r}{c}$.

Perl supports the syntax of the familiar `for` loop of the C programming language, thus allowing a common looping mechanism to be used within TEX.

Figure 7 gives a graphical view of Pascal's triangle. In this diagram, a modulus of two has been used, giving two possible remainders, shown as white and black. This diagram can be specified using Ti*k*Z as a sequence of `\fill` statements, as shown in Figure 8. Each `\fill` statement is responsible for producing one small square of the diagram and specifies its color, position, and size.

Figure 9 is a revision of `\pascalMatrix`. Using this definition, the necessary `\fill` statements to generate a graphical form of Pascal's triangle can

```
\perlnewcommand{\pascalMatrix}[1] {
  my $tabularRows = "";
  my @row = (1);
  for (my $r = 0; $r <= $_[0]; $r++) {
    # Output the current row
    $tabularRows .= join ("&", @row) .
                    " \\\\\n";
    # Generate the next row
    my @nextRow = (1);
    for (my $c = 1; $c <= $r; $c++) {
      push @nextRow,
           @row[$c - 1] + @row[$c];
    }
    push @nextRow, 1;
    @row = @nextRow;
  }
  return
    "\\begin{tabular}{*{$_[0]}{c}c}\n" .
    $tabularRows .
    "\\end{tabular}\n";
}
```

Figure 6: Generating entries of Pascal's triangle.

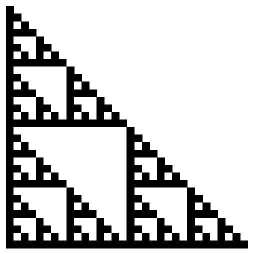


Figure 7: Graphical view of Pascal's triangle.

```
\begin{tikzpicture}[scale=0.5]
  \fill[black] (0,0) rectangle +(1,1);
  \fill[black] (0,-1) rectangle +(1,1);
  \fill[black] (1,-1) rectangle +(1,1);
  \fill[black] (0,-2) rectangle +(1,1);
  \fill[white] (1,-2) rectangle +(1,1);
  \fill[black] (2,-2) rectangle +(1,1);
  \fill[black] (0,-3) rectangle +(1,1);
  \fill[black] (1,-3) rectangle +(1,1);
  \fill[black] (2,-3) rectangle +(1,1);
  \fill[black] (3,-3) rectangle +(1,1);
\end{tikzpicture}
```

Figure 8: Ti*k*Z code for four rows of Pascal's triangle.

```
\perlnewcommand{\pascalGraphic}[1]{
  my $result = "";
  my @row = (1);
  my @colors = ("white", "black");
  for (my $r = 0; $r <= $_[0]; $r++) {
    # Output the current row
    for (my $c = 0; $c <= $r; $c++) {
      $result .= sprintf("\\fill[%s] (%d, %d) rectangle +(1, 1);\n",
                         $colors[$row[$c]], $c, -$r);
    }
    # Generate the next row
    my @nextRow = (1);
    for (my $c = 1; $c <= $r; $c++) {
      push @nextRow, (@row[$c - 1] + @row[$c]) % 2;
    }
    push @nextRow, 1;
    @row = @nextRow;
  }
  return $result;
}
```

Figure 9: Generating a graphical view of Pascal's triangle, modulo two.

```
TimeCDT,TemperatureF,Dew PointF,Humidity,Sea Level PressureIn,VisibilityMPH,
Wind Direction,Wind SpeedMPH<BR>
12:53 AM,73.0,70.0,90,30.05,10.0,SSW,4.6<BR>
1:53 AM,73.0,69.1,87,30.04,9.0,SW,3.5<BR>
2:53 AM,72.0,68.0,87,30.04,10.0,West,3.5<BR>
 additional lines omitted
<!-- 0.122:1 -->
```

Figure 10: A brief excerpt of weather information obtained from the Weather Underground.

be obtained. For example,

```
\begin{tikzpicture}[scale=0.1]
  \pascalGraphic{31}
\end{tikzpicture}
```

generates the 32-row graphic of Figure 7.

As might be expected, the definitions for the two Pascal triangle commands are very similar. In the graphical version, a single string consisting of the sequence of `\fill` statements is built up in `$result`. Each of these `\fill` statements is obtained by a formatted print statement, appended to `$result`. Also, since values within any one row of the triangle are stored modulo two, Perl's `%` operator is used. Both `sprintf` and the `%` operator are language features shared with C.

6 LaTeX documents and the Internet

The LWP* Perl library [1] can be used to access data on the web. With the assistance of PerlTeX, this allows information from web sites to be retrieved and incorporated within a LaTeX document.

For example, suppose we wish to access weather data and display it in either tabular or graphical form within a LaTeX document. This type of processing is made possible by LWP and Perl's support for regular expressions.

The Weather Underground† is one of many sites which provides access to historical weather data. Given an airport code, year, month, and day, it is possible to retrieve many details about the weather recorded at the requested location and date. Figure 10 shows an excerpt of the results of a web request for June 28, 2007, at the Coles County Memorial Airport, with airport code KMTO. What is important to know about this request is that

```
KMTO/2007/6/28
```

appears as a substring of a lengthy URL. Figure 11 shows how this raw weather data is to be formatted as a tabular.

* LWP is an acronym for 'Library for WWW in Perl'.

† `www.wunderground.com`

Time	Temp	Dew Point	Humidity	SL Pressure	Vis	Wind Dir	Wind Speed
12:53 AM	73.0	70.0	90	30.05	10.0	SSW	4.6
1:53 AM	73.0	69.1	87	30.04	9.0	SW	3.5
2:53 AM	72.0	68.0	87	30.04	10.0	West	3.5

Figure 11: An excerpt of formatted weather data.

```
\perlnewcommand{\getWeather}[4] {
  use LWP::Simple;

  # Form the URL from the airport code, year, month, and day
  $id = join"/", @_;
  $URL = A URL incorporating $id;

  # If we have already looked up this day, do nothing
  return "" if exists $data{$id};

  # Otherwise fetch and store the data
  $data{$id} = get$URL;

  # Return nothing as this command only fetches data but
  # does not cause anything to appear in the document.
  return "";
}
```

Figure 12: A command to fetch the weather data from a web site. The four parameters specify airport code, year, month, and day.

The data is retrieved and formatted with two PerlTEX commands: one retrieves the data from the Web site, the other one performs some text manipulations and formats the data as a tabular. A variety of text substitutions are needed, for example, to account for HTML tags which are present in the retrieved data.

The first of these commands, `\getWeather`, is shown in Figure 12. This command introduces variables `$id` and `$data`, but does not declare them to be local, thus making them accessible from the second command, `\formatWeather`. The command

```
\getWeather{KMTO}{2007}{6}{28}
```

creates the appropriate URL, accesses the web site to retrieve the data, and saves the result in the Perl hash variable `$data`. This command differs from the others presented in that the return value is of no interest: rather, it is the side-effect of storing the weather data in `$data` that is desired.

Figure 13 shows the details of `\formatWeather`. Weather data is obtained from the web site by invoking `latex_getWeather`, the Perl function created from the definition of `\getWeather`. At this point, various textual substitutions are made to the data. For example, the comma-separated values are adjusted to become ampersand-separated values, in anticipation of inclusion in a tabular environment. This command illustrates some of the string-processing conveniences of Perl.

In this example, information was retrieved from the Internet. However, data can be obtained from files, databases, and other sources just as easily.

7 Graphical animations

The `animate` package [2] provides a convenient way to create PDF files with animated content consisting of a sequence of frames, optionally controlled with VCR-style buttons. We provide a brief example of the use of this package with a special focus on the role PerlTEX can play.

One of the environments provided by this package, `animateinline`, creates animations from the typeset material within its body. This might consist of a sequence of Ti*k*Z commands which generate frames of the animation.

In many cases, each frame of an animation can be viewed under the control of some parameter t. For example, an animation of a Bézier curve can be constructed as a sequence of frames controlled by t, where t varies from 0 to 1.

For the present example, assume there is a command, `\bcurve`, with a single parameter t which yields a graphical image of a single frame. To achieve

```
\perlnewcommand{\formatWeather}[4] {
  # Ensure the appropriate weather data has been retrieved
  latex_getWeather($_[0], $_[1], $_[2], $_[3]);
  $rows = $data{$id};

  $rows =~ s/(.*\n){2}//; # Strip the first two lines

  # Add the headings
  $rows = "Time,Temp,Dew Point,Humidity,SL Pressure,Vis," .
          "Wind Dir,Wind Speed\n" . $rows;
  $rows =~ tr/,/&/;       # Commas become alignment tabs
  $rows =~ s/\n/\\\\\n/g; # Newlines become the end of a row
  $rows =~ s/<.*>//g;     # Remove any HTML tags

  # Return the table
  return "\\begin{tabular}{...}\n $rows \\end{tabular}\n";
}
```

Figure 13: A command to tabulate the weather data obtained from a web site. Notice how the PerlTEX command `\getWeather` is invoked.

a smooth animation, what is needed is a sequence of these frames, with closely spaced values of t. For the `animateinline` environment, a sequence such as:

```
\bcurve{0}%
\newframe%
\bcurve{0.02}%
\newframe%
```
etc.

is needed to generate the animation.

This sequence can be generated easily with a PerlTEX command. In doing so, we cover a wide class of animations, namely, those that can be described as a sequence of frames controlled by a single parameter.

Figure 14 shows the details of a command which generates a sequence of frames. This command takes four arguments: the name of the command which generates a single frame, the starting and ending values of t, and the total number of frames desired. For example,

```
\animationLoop{bcurve}{0}{1}{51}
```

generates the sequence of 51 frames `\bcurve{0.0}`, `\bcurve{0.02}`, ..., `\bcurve{1.0}`.

8 Demonstrating algorithms

One way to understand an algorithm is to trace its actions, maintaining a history of the values being computed and stored. For some algorithms, this history can be compactly displayed with a tabular environment.

Using the `beamer` package [7], this type of tabular information can be incrementally revealed by inserting `\pause` commands at selected locations. In the context of demonstrating an algorithm, such pauses can be used to show the effect of one iteration of a loop.

To illustrate, consider Euclid's algorithm for computing the greatest common divisor. At the heart of this algorithm is a loop which repeats the following three actions:

```
r = x % y;  x = y;  y = r;
```

The history for this algorithm is a tabular array with three columns, one for each of the three variables. A partial history might reveal the following table, "at rest" at a `\pause`.

x	y	r
120	70	50
70	50	20
50	20	

Picking up after the `\pause`, the history grows by an amount equal to one iteration of the loop:

x	y	r
120	70	50
70	50	20
50	20	10
20	10	

As shown in Figure 15, these types of tabular environments can be generated with a PerlTEX command. Since the values in the tabular are being computed by Euclid's algorithm, it is easy to generate a wide variety of example histories — and to have confidence that the values in the table are correct!

```
\perlnewcommand{\animationLoop}[4]{
  my $result = "";
  my $delta = ($_[2] - $_[1]) / ($_[3] - 1.0);
  my $x = $_[1];
  for (my $count = 1; $count < $_[3]; $count++) {
    $result .= "\\" . $_[0] . "{$x}%\n" .
               "\\newframe%\n";
    $x += $delta;
  }
  return $result . "\\" . $_[0] . "{$_[2]}%\n";
}
```

Figure 14: A command to generate a sequence of frames in a graphical animation.

```
\perlnewcommand{\euclidAlgorithm}[2]{
  my $x = $_[0];
  my $y = $_[1];
  my $result = "$x & $y & \\pause ";
  while ($y != 0) {
    my $r = $x % $y;
    $result .= "$r \\\\ \\hline \\pause \n";
    $x = $y;
    $y = $r;
    $result .= "$x & $y & ";
    $result .= "\\pause " if $y != 0;
  }
  return
    "\\begin{tabular}{c|c|c} \\hline \n" .
    "\\$x\\$ & \\$y\\$ & \\$x \\bmod y\\$\\\\  \\hline \\pause \n" .
    $result . "\\\\ \\hline \n" .
    "\\end{tabular}";
}
```

Figure 15: A PerlTEX command to generate a tabular history for Euclid's algorithm.

9 Shuffling an enumerated list

So far, all of the examples presented have focused on the ability to define new commands with PerlTEX. However, PerlTEX also provides a mechanism to define a new environment: `\perlnewenvironment`. A command of the form

`\perlnewenvironment{foo}{`*start*`}{`*finish*`}`

defines a new environment, named `foo`. PerlTEX replaces a subsequent `\begin{foo}` with the *start* text and an `\end{foo}` with the *finish* text.

To illustrate, suppose it is desired to typeset a shuffled enumerated list. In principle, each time the source text is processed, a different ordering of the list items could result. To accomplish this, we introduce a new environment, `shuffle`, and a new command, `\shuffleItem`. For example, Figure 16 illustrates how this environment can be used to typeset an enumerated list of the four given items, arranged in an arbitrary order.

```
\begin{shuffle}
  \shuffleItem{TUG 2007}
  \shuffleItem{SDSU}
  \shuffleItem{San Diego}
  \shuffleItem{California}
\end{shuffle}
```

Figure 16: An example of the use of the `shuffle` environment.

To achieve this behavior, consider the action required for each item in the `shuffle` environment. As shown in Figure 17, each item which appears gets appended to a variable, `@items`, which maintains a list of all items encountered thus far.

The bulk of the work takes place at the conclusion of the `shuffle` environment. The items that have been accumulating are now rearranged and an enumerated list is constructed from this permuted list. Figure 18 provides the Perl details. The variable `@items` is undefined as a final step, since a

```
\perlnewcommand{\shuffleItem}[1]{
  push @items, $_[0];
  return "";
}
```

Figure 17: A PerlTEX command which stores one item of a `shuffle` environment.

subsequent `shuffle` environment must start with a "clean slate" of items. Permuting the list of items is a simple operation, since there is a Perl library module well-suited to this task.

Although this implementation of shuffled enumerated lists does not allow for nested shuffled lists, it does nevertheless provide an illustration of the ability to define new environments within PerlTEX.

```
\perlnewenvironment{shuffle}
{return "\\begin{enumerate}\n"}
{
  use List::Util "shuffle";

  @items = shuffle(@items);

  my $result = " \\item ".
               join("\n \\item ", @items).
               "\n\\end{enumerate}";
  undef @items;
  return $result;
}
```

Figure 18: A PerlTEX command which stores an item of a `shuffle` environment.

10 Summary

TEX provides powerful ways to format text and to perform general-purpose computations. For many users, however, the techniques required to access the computational features of TEX are cumbersome. By providing a bridge between TEX and Perl, PerlTEX makes these computations more accessible.

Perl's widely acknowledged strengths, including extensive libraries, support for regular expressions, a rich collection of string primitives, and familiar control structures, make PerlTEX a natural candidate for the LATEX user seeking finer control over typesetting tasks.

References

[1] Sean Burke. *Perl & LWP*. O'Reilly, 2002.

[2] Alexander Grahn. *The animate package*. `http://www.ctan.org/tex-archive/macros/latex/contrib/animate`.

[3] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1986.

[4] Scott Pakin. PerlTEX: Defining LATEX macros using Perl. *TUGboat*, 25(2):150–159, 2004.

[5] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals*. Springer-Verlag, 2004.

[6] Till Tantau. *TikZ and PGF manual*. `http://sourceforge.net/projects/pgf/`.

[7] Till Tantau. *User's Guide to the Beamer Class*. `http://latex-beamer.sourceforge.net`.

[8] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl, Third Edition*. O'Reilly, 2000.

LaTeX conversion into normalized forms and speech

Eitan M. Gurari
Ohio State University
gurari (at) cse dot ohio-state dot edu
http://www.cse.ohio-state.edu/~gurari

Abstract

LaTeX is an authoring language designed for producing documents through native TeX compilers. Over the years many other applications have been developed to accept LaTeX inputs via alternative engines programmed from scratch. These engines are restricted in power to subsets of LaTeX features.

The first part of this report shows how TeX4ht can translate general LaTeX constructs into the restricted dialects recognizable by such engines. The jsMath dialect for rendering LaTeX through JavaScript is employed as an example.

An especially significant use of LaTeX input was T. V. Raman's 1994 pioneering AsTeR program for automatically rendering technical documents into audio. Newer audio browsers are expected to address XML documents that adhere to the SSML and ACSS specifications. The second part of this report extends Raman's work by showing how TeX4ht can translate LaTeX to XML-based representations that support speech.

1 Applications of LaTeX dialects

The LaTeX system offers a rich set of high-level features for authoring manuscripts, and a powerful engine for typesetting documents. The human friendly design of the language, in particular within its mathematical component, promoted different programs to choose variants of LaTeX as their input languages. Similarly, the superior typesetting capabilities encouraged different tools to offer LaTeX for exported document formats.

For instance, the *jsMath* utility [2] is dedicated to rendering restricted LaTeX mathematical expressions embedded within HTML files. In doing so it offers a friendly medium for on-line content management. Specifically, information is easy to enter and edit, a single document file provides for both content rendering and editing, document files can be accessible throughout the web as is the case for Wiki pages, and viewers need not install new software. The program is written in JavaScript. Figure 1(a) shows the jsMath source code for obtaining the output in Figure 1(b).

As another example, the source mathematical code of *MediaWiki* [10] is expressed in LaTeX. The code is channeled to the *texvc* program [17] for converting the expressions into images.

This material is based upon work supported by the National Science Foundation under Award No. IIS-0312487. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

On the other hand, the *Scientific Notebook* document processing system [15] is an example of a utility capable of exporting LaTeX documents. The LaTeX mathematical code emitted by this program can be imported into the *Duxbury Braille Translator* for embossing the expressions into Nemeth braille [3].

In all of the above examples, only subsets of the LaTeX features are supported. In the first two examples, minor non-LaTeX features are added.

2 A TeX4ht mode for jsMath

The jsMath system supports only a few core features of LaTeX, and its vocabulary is quite restricted due to the very limited macro capabilities of the system. TeX4ht, on the other hand, is a highly configurable converter for TeX-based sources [4]. Hence, TeX4ht can assume the bulk of the work of processing given files into forms jsMath can handle. To translate a LaTeX file named `file.tex` into HTML, with the mathematical expressions converted into jsMath, one can issue the following command:

```
htlatex file "html,jsmath" " -cmozhtf"
```

The jsMath engine recognizes a limited set of symbol names, but it fully supports Unicode representations. The flag '`-cmozhtf`' requests Unicode encodings for the majority of the symbols usually contributed from the (La)TeX fonts, ignoring the possibility of using names for the symbols recognized by the jsMath engine.

Figure 2(a) shows the jsMath output of TeX4ht for the source of Figure 2(b), and Figure 4(a) ex-

```
<p> A quadratic equation
     <span class="math">
        ax^2 + bx + c = 0
     </span>
  with
     <span class="math">a \neq 0</span>
  has the following solution.
</p>
<div class="math">
   x = \frac {-b \pm \sqrt{b^2 - 4ac} }
             {2a}
</div>
```

(a)

A quadratic equation $ax^2 + bx + c = 0$ with $a \neq 0$ has the following solution.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(b)

Figure 1: HTML code with embedded jsMath expressions and its rendering.

```
<p class="noindent">
  A quadratic equation
    <span class="math">
       a{x}^{2} + bx + c = 0
    </span>
  with
    <span class="math">
       a\mathrel{&#x2260;}0
    </span>
  has the following solution.
</p>
<div class="math">
    x ={ &#x2212;b &#x00B1;\sqrt{{b}^{2 }
         &#x2212; 4ac} \over 2a}
</div>
```

(a)

```
A quadratic equation $ax^2 + bx + c = 0$
with $a \ne 0$ has the following solution.

$$x={-b \pm \sqrt{b^2 - 4ac} \over 2 a}$$
```

(b)

Figure 2: TEX4ht jsMath output for a LATEX source.

```
<p class="noindent">
  A quadratic equation
     <span class="math"> a{x}^{2} + bx + c = 0 </span>
  with
     <span class="math"> a\ne 0 </span>
  has the following solution.
</p>
<div class="math">
     x =\frac{ -b\pm \sqrt{{b}^{2 } - 4ac} }
              {2a}
</div>
```

Figure 3: Reconfigured TEX4ht output.

hibits the jsMath code created for the source of Figure 4(b).

3 A taste of the TEX4ht configurations

The default jsMath configurations of TEX4ht do not take advantage of the full range of the LATEX features permitted by the jsMath utility. As a result, the jsMath code created by TEX4ht has room for improvements with respect to making the code more friendly for handling by human beings. This section demonstrates how TEX4ht can be reconfigured to produce from the input of Figure 2(b) the output of Figure 3, as an alternative to the default output shown in Figure 2(a).

3.1 Using literal characters instead of Unicode values

When LATEX encounters the minus character '-' in

```
<div class="math">
W(&#x03A6;) = \left \Vert \array{
{    &#x03C6;
\over ({&#x03C6;}_{1},{&#x025B;}_{1})}
      &amp;     0
      &amp;\mathop{\mathop{&#x2026;}}
           \kern 1.66702pt
      &amp;     0
\cr
{  &#x03C6;{k}_{n2}
\over ({&#x03C6;}_{2},{&#x025B;}_{1})}
     &amp;{     &#x03C6;
\over ({&#x03C6;}_{2},{&#x025B;}_{2})}
     &amp;\mathop{\mathop{&#x2026;}}
          \kern 1.66702pt
     &amp;     0
\cr
  .&amp;.&amp;.&amp;.&amp;.
\cr
{  &#x03C6;{k}_{n1}
\over ({&#x03C6;}_{n},{&#x025B;}_{1})}
     &amp;{   &#x03C6;{k}_{n2}
\over ({&#x03C6;}_{n},{&#x025B;}_{2})}
     &amp;\mathop{\mathop{&#x2026;}}
          \kern 1.66702pt
     &amp;{   &#x03C6;{k}_{n\kern
                 1.66702pt n&#x2212;1}
\over ({&#x03C6;}_{n},
             {&#x025B;}_{n&#x2212;1})}
     &amp;{      &#x03C6;
\over ({&#x03C6;}_{n},{&#x025B;}_{n})} }
\right \Vert
</div>
```

(a)

```
\documentclass{article}
  \usepackage{amsmath}
\begin{document}
  \[W(\Phi)= \begin{Vmatrix}
  \dfrac\varphi
        {(\varphi_1,\varepsilon_1)}&
0&\dots&0\\
  \dfrac{\varphi k_{n2}}
        {(\varphi_2,\varepsilon_1)}&
  \dfrac\varphi
        {(\varphi_2,\varepsilon_2)}
        &\dots&0\\
  \hdotsfor{5}\\
  \dfrac{\varphi k_{n1}}
        {(\varphi_n,\varepsilon_1)}&
  \dfrac{\varphi k_{n2}}
        {(\varphi_n,\varepsilon_2)}&\dots&
  \dfrac{\varphi k_{n\,n-1}}
        {(\varphi_n,\varepsilon_{n-1})}&
  \dfrac{\varphi}
        {(\varphi_n,\varepsilon_n)}
  \end{Vmatrix}\]
\end{document}
```

(b)

$$W(\Phi)=\left\Vert\begin{array}{ccccc} \frac{\varphi}{(\varphi_1,\varepsilon_1)} & 0 & \dots & 0 & \\ \frac{\varphi k_{n2}}{(\varphi_2,\varepsilon_1)} & \frac{\varphi}{(\varphi_2,\varepsilon_2)} & \dots & 0 & \\ \multicolumn{5}{c}{\dotfill} \\ \frac{\varphi k_{n1}}{(\varphi_n,\varepsilon_1)} & \frac{\varphi k_{n2}}{(\varphi_n,\varepsilon_2)} & \dots & \frac{\varphi k_{n\,n-1}}{(\varphi_n,\varepsilon_{n-1})} & \frac{\varphi}{(\varphi_n,\varepsilon_n)} \end{array}\right\Vert$$

(c)

Figure 4: TeX4ht jsMath output for a LaTeX input.

the input, it places in the `dvi` file a request that the character will be typeset by the first symbol of the `cmsy` font. When TeX4ht encounters the request while processing the `dvi` file, it opens an alternative hypertext font file of its own, named `cmsy.htf`, and retrieves the first entry in that file. This entry gives the Unicode value `&#x2212;`.

Given this Unicode value `&#x2212;`, the TeX4ht utility searches for the value in the active encoding file `unicode.4hf`. If the value `&#x2212;` is not found in the encoding file, it is inserted as is into the output. If the value is found in the encoding file, the replacement from the encoding file is instead placed in the output.

The flag '`-cmozhtf`' of the command line requests an encoding file that does not include an entry for `&#x2212;`. Consequently, in the default setting, the Unicode value is placed in the output. The following steps show how the simple '`-`' character can be output instead:

- Make a copy of the following font encoding file,

  `ht-fonts/mozilla/charset/unicode.4hf`
- Add the following record to the new file:

  ```
  '&#x2212;' '' '-' ''
  ```
- Specify the location of the new encoding file on the command line. If the file is in the current directory, the command line can take the following form:

  ```
  htlatex file "xhtml,jsmath"
  ```

3.2 Preventing the expansion of symbol macros

A redefinition of the control sequences `\pm` and `\ne` in the following manner prevents the expansion of the symbol macros, respectively, into '`&#x00B1;`' and '`\mathrel{&#x2260;}`':

```
\edef\pm{\HCode{\string\pm\space}}
\edef\ne{\HCode{\string\ne\space}}
```

3.3 Transformations involving rewriting

A translation of the `\over` operator into the `\frac` function can be achieved by introducing the following TEX4ht configuration.

```
\Configure{over}
  {\Send{GROUP}{0}{\string\frac\l:brace}}
  {\HCode{\r:brace\l:brace}%
   \Send{EndGROUP}{0}{\r:brace}}
```

The configuration relies on capabilities to inversely process DVI code into source code. The next observations provide some insight into how the configuration works:

1. The arguments of a configuration

   ```
   \Configure{over} {...} {...}
   ```

   are inserted by TEX4ht immediately before and after the `\over` operator.
2. The `\Send{GROUP}{0}{...}` instruction sends its argument backward to the start of the current group.
3. The `\Send{EndGROUP}{0}{...}` code delivers its argument forward to the end of the current group.
4. The contribution of `\Configure{over}` to the code fragment `{...\over...}` provides the following initial outcome.

   ```
   {...
   \Send{GROUP}{0}{\string\frac\l:brace}
   \over
   \HCode{\r:brace\l:brace}
   \Send{EndGROUP}{0}{\r:brace}
   ...}
   ```
5. After applying the `\Send` instructions, the code takes the following form.

   ```
   {\frac\l:brace ...
   \over
   \r:brace\l:brace ... \r:brace}
   ```
6. The braces '`{`' and '`}`' and the `\over` operator do not introduce content to the output file. Consequently, the net contribution is as follows, where `\l:brace` and `\r:brace` produce left and right braces, respectively.

   ```
   \frac\l:brace ...\r:brace\l:brace
           ...\r:brace
   ```

4 Speech markup and synthesis

XML and Cascading Style Sheets (CSS) conventions are commonly and increasingly being used for describing the desirable rendering of documents into visual forms. Similar attention is also being given to the development of analogous standards for rendering documents in audio forms.

The aural conventions are concerned with properties such as pitches, volumes, rates, and pauses to be associated with the different parts of the documents. The conventions are in particular valuable for introducing annotations that highlight the structural characteristics of documents.

Of particular interest in this regard are the draft proposals from the W3C consortium of the Speech Synthesis Markup Language (SSML) [16] and Aural Cascading Style Sheets (ACSS) [1]. Figures 5(a) and 5(b), respectively, illustrate the notations in those proposals. The SSML specifications are based on, and are similar to, the Java Speech Markup Language (JSML) specifications [7].

Emacspeak [13] supports a restricted variant of ACSS, recognizing the properties of voice-family, stress, richness, pitch, and pitch-range. The C++ program of Figure 5(c) renders, on the Microsoft Windows Vista platform, files in SSML format. The Java program of Figure 5(d) renders JSML files, on platforms offering a JSML-based implementation to the Java Speech APIs [6].

5 From LATEX to speech

Audio representations of documents are of great importance for people with print disabilities [14] as in many cases they have no alternative ways to access the content of the documents. Yet, documents in audio formats can be also useful for the general public. For instance, that might be the case for people who want to listen to a document while driving, or for

```
<speak> <p>Take a deep breath <break strength="weak"/> then
        <prosody rate="-10%">speak slower</prosody>.</p>
        <p>Also <prosody volume="loud">raise your voice</prosody>
        so everyone will hear you.</p>                    </speak>
```

(a)

```
h1 { voice-stress: strong; voice-rate:-10%; pause-after: 20ms; }
msqrt.before { content: "Square root: " }
msqrt.after  { content: "End root. " }
```

(b)

```
#include <sapi.h>
int main(int argc, char* argv[]){
  ISpVoice * synth = NULL;
  if (FAILED(::CoInitialize(NULL))){ return 0; }
  HRESULT hr = CoCreateInstance(CLSID_SpVoice, NULL,
                  CLSCTX_ALL, IID_ISpVoice, (void **)&synth);
  if( SUCCEEDED( hr ) ){
    int n = strlen(argv[1]);
    wchar_t *s = (wchar_t *)  malloc(n+1); s[n] = '\0';
    while( n-- > 0 ){ s[n] = argv[1][n]; }
    hr = synth->Speak(s, SPF_IS_FILENAME | SPF_PARSE_SSML, NULL);
    synth->Release();
    synth = NULL;
  }
  ::CoUninitialize(); return 0;
}
```

(c)

```
import javax.speech.*;
import javax.speech.synthesis.*;
import java.net.*;
import java.io.*;
public class Speaker{
  public static void main(String args[]) {
    try {
      Synthesizer synth = Central.createSynthesizer(new SynthesizerModeDesc());
      synth.allocate();
      synth.resume();
      synth.speak(new File(args[0]).toURI().toURL(), null);
      synth.waitEngineState(Synthesizer.QUEUE_EMPTY);
      synth.deallocate();
    }catch( Exception e ){
      System.err.print("--- ERROR --- "); e.printStackTrace();
} } }
```

(d)

Figure 5: (a) SSML. (b) ACSS. (c) SSML file speaker. (d) JSML file speaker.

authors wishing to "proof listen" their writings in addition to (or instead of) proof reading.

LaTeX documents can be translated by TeX4ht into files annotated for speech [5]. For the ACSS speech variant of Emacspeak the requests can be made with commands similar to the following:

```
eslatex file
```

For output in JSML format the calling commands can be as follows:

```
jslatex file
```

TeX4ht configurations similar to those provided for the JSML and the Emacspeak variant of ACSS can, and in time will, be also tailored for output modes in SSML and the W3C version of ACSS.

It should be noted that currently no browser is available for effectively inspecting and navigating highly structural content in audio mode. In fact, it is even not clear what features audio browsers should offer to tackle this issue. Such a deficiency makes it very difficult to use audio resources to study technical topics, including those relying heavily on mathematical notations. In addition, it makes it difficult to decide what added information TEX4ht should provide in the translated material to enhance its accessibility.

Much of the approach for audio rendering of mathematics is motivated by Nemeth braille and expressed in MathSpeak [9]. The audio cues for different logical elements of data might also be specified within browsers instead of being provided to them within the data. MathPlayer [8], for instance, behaves so in rendering MathML expressions into audio. LATEX files can be transformed by TEX4ht to satisfy MathPlayer requirements with commands of the following form:

```
mzlatex file "xhtml,mathplayer"
```

The pioneering work of automatically putting technical content into audio format is due to T.V. Raman and assumed the LATEX language for the input data [11, 12].

Acknowledgement

I am grateful to Barbara Beeton, Karl Berry, and Susan Jolly for their valuable input.

References

[1] *Aural style sheets*, W3C Working Drafts, `http://www.w3.org/TR/CSS21/aural.html`, `http://www.w3.org/TR/css3-speech/`.

[2] D. Cervone, *jsMath: A Method of Including Mathematics in Web Pages*, `http://www.math.union.edu/~dpvc/jsMath/` and `http://sourceforge.net/projects/jsmath/`.

[3] *Duxbury Braille Translator (DBT)*, Duxbury Systems, `http://www.duxburysystems.com/`.

[4] E. Gurari, *TEX4ht*, `http://www.cse.ohio-state.edu/~gurari/TeX4ht`.

[5] E. Gurari, *LaSpeak: LATEX and Speech*, `http://www.cse.ohio-state.edu/~gurari/laspeak`.

[6] *Java Speech API*, Sun Microsystems, `http://java.sun.com/products/java-media/speech/` (version 1) and `http://jcp.org/en/jsr/detail?id=113` (version 2, proposed draft, 11 June 2007).

[7] *Java Speech Markup Language (JSML)*, Sun Microsystems, 1999, `http://java.sun.com/products/java-media/speech/forDevelopers/JSML/index.html`.

[8] *MathPlayer*, Design Science, `http://www.dessci.com/en/products/mathplayer/`.

[9] *MathSpeak Core Specification Grammar Rules*, `http://www.gh-mathspeak.com/examples/grammar-rules/`.

[10] *MediaWiki*, `http://www.mediawiki.org/wiki/MediaWiki`.

[11] T. V. Raman, *An audio view of (LA)TEX documents*, *TUGboat* 13:3, October 1992, 372–379, `http://www.tug.org/TUGboat/Articles/tb13-3/raman.pdf`.

[12] T. V. Raman, *Audio System for Technical Readings (AsTeR)*, Ph.D. Dissertation, Cornell University, May 1994, `http://emacspeak.sourceforge.net/raman/publications/web-aster/root-thesis.html`, Examples: `http://www.cs.cornell.edu/home/raman/aster/aster-toplevel.html`.

[13] T. V. Raman, *Emacspeak — The Complete Audio Desktop*, `http://emacspeak.sourceforge.net/`.

[14] Recording for the Blind and Dyslexic, `http://www.rfbd.org/`.

[15] *Scientific Notebook*, MacKichan Software, `http://www.mackichan.com/`.

[16] *Speech Synthesis Markup Language (SSML)*, W3C Working Draft, 11 June 2007, `http://www.w3.org/TR/speech-synthesis11/`.

[17] T. Wegrzanowski, *Texvc: TEX Validator and Converter*, `http://en.wikipedia.org/wiki/Texvc` and `http://meta.wikimedia.org/wiki/Help:Formula`.

Abstracts

(See the 'Random comments' column on p. 377 for publication information on items marked with ''. Ed.)*

Three dimensional graphics with Sketch
David M. Allen

Sketch is a 3D scene description translator by Eugene K. Ressler. Its web page is at `http://www.frontiernet.net/~eugene.ressler`.

Sketch is a small, simple system for producing line drawings of two- or three-dimensional solid objects and scenes. Sketch produces finely wrought, mathematically based illustrations with no extraneous detail. It does not do photo-realistic scenes. The input language is reminiscent of PSTricks, so will be easy to learn for current PSTricks users.

Sketch output was PSTricks code until recently. In addition to PSTricks, Sketch now understands Ti*k*Z/PGF options (key/value pairs) and generates Ti*k*Z output. Some advantages are that Ti*k*Z works directly with pdfLaTeX and supports transparency. Happily, the depth sort hidden surface algorithm used by Sketch is perfectly compatible with transparent polygons.

This presentation gives a brief overview of Sketch from the perspective of a beginning user. It gives some detailed examples that introduce a substantial set of commands. There are also examples from the Sketch distribution.

From TeX to XML: The legacy of techexplorer and the future of math on the Web
Don DeLand

In 1997, IBM Research first released techexplorer, a web browser plugin for rendering TeX markup directly within browsers. Since Integre took over techexplorer development in 2003 there have been relatively few advances in browser technology, but tremendous developments in collaboration tools and other web-based applications. This talk gives a brief history of the techexplorer project and explains why its development has shifted away from TeX to its current focus on native XML/MathML authoring. Although delivering math in web browsers continues to be a frustrating process, "Web 2.0" holds substantial promise for a new generation of web-based applications that support mathematics.

Long-time preservation strategies for TeX-sourced content
Paulo Ney de Souza

The amount of published material in the world has grown exponentially since Gutenberg's invention, with a rate of doubling every 7 1/2 years right now. Electronic publishing will only increase this rate, posing new challenges for the long-time preservation of records and usability for the future.

TeX has changed us into our own typists and even graphics designers sometimes, but at the same time has provided the best strategy for preservation of scientific content we have. This talk will examine some of these strategies and how MSP — a non-profit scientific publisher — has used it to improve usability of journals over time.

LuaTeX Attributes: The new kid on the block*
Hans Hagen

When you switch fonts in TeX, normal grouping keeps changes to another font local to the group. But there is more than fonts. Most macro packages provide color support and in ConTeXt we also support some PDF related features like outlines and hidden invisible ink. These features all share a common problem: we need to keep track of their state in the page stream and across pages and splitting content also takes some care. A maybe less obvious example is hyperlinks, which are natively supported by pdfTeX (although ConTeXt does it slightly differently).

In order to make such features easier (and more robust) to implement, LuaTeX provides attributes. These behave like fonts but are by design agnostic as to what they represent. They travel with the nodes (each node can have attributes) and it's up to the macro package to make sure that the intended behaviour takes place. For this, Lua code is used in combination with processing node lists, either by using callbacks or by postprocessing boxes.

In this talk I will explain what attributes are, and how they can be of use to macro writers.

Zapfino: Hermann's torture test for TeX*
Hans Hagen

Over the next couple of years TeXies have to explore the new landscape of OpenType fonts. Most of the implementation details will be hidden beyond user interfaces of macro packages. However this does not hide the potential mess that users can invoke when they start enabling or disabling features related to fonts.

Thanks to the Oriental TeX project Taco can spend substantial time on coding LuaTeX which in turn means that I have lots of testing and protyping on my plate. We also spend much time on discussing the interfaces and extensions to the program and due to this as well as realistic testing LuaTeX develops rapidly. However, in order to fulfill the requirements of the Oriental TeX project we need to be able to typeset high quality Arabic. Since I'm more familiar with Latin and since I had the Zapfino Pro handwriting font waiting for me in OpenType format I decided to use that font as benchmark for advanced node processing in LuaTeX. It proved to be a worthy contender. In the process we were able to optimize node support in LuaTEX and it also triggered reimplementing the OpenType tables (from FontForge format 1 to format 2).

In this presentation I will discuss how we deal with advanced features that are part of OpenType fonts like Zapfino. I will also explain how such features are implemented in Lua and TEX code.

CritTEXt: The critical-edition module for ConTEXt*

Idris Hamid

TEX has become a very important tool in the preparation of critical editions of texts. In particular, EDMAC for plain TEX, by Lavagnino and Wujastyk, has come to the rescue of many an editor of texts (myself included). In the last two years the LATEX world has produced at least two general-purpose packages for the preparation of critical editions: LEDMAC (a port of EDMAC with even additional features) and ednotes. With its extensive and high-level configurability, typographic flexibility, and database capabilities — not to mention its user-friendly and *consistent* interface — ConTEXt is perhaps the most natural typesetting platform for a full-featured and easy-to-use user interface for authors and typesetters who need to produce a sophisticated camera-ready critical edition. With the maturation of projects like LuaTEX and Oriental TEX well underway, the capacity of TEX for even more efficient production and configuration of the highest quality of critical texts in multiple languages is almost at hand.

This document outlines the high-level interface to the critical-text editing module for ConTEXt, called *CritTEXt*. The module is in pre-alpha status and the entire interface can be expected to undergo extensive changes in the coming weeks, especially as LuaTEX matures.

On the technical level, preparation of the critical edition involves three things:

- the main text;
- the apparatus;
- the rest of the book.

The critical edition module we are developing is concerned primarily with preparing and typesetting the apparatus. This module is meant to be consistent with the rest of ConTEXt so that the integration of the apparatus with the main text and the rest of the book is seamless.

In preparation of the apparatus we distinguish two things, *elemental structure* and *typographic structure*.

1. *Elemental Structure*:
   Each building block of the apparatus, up to and including the apparatus itself and relevant parts of the main text, constitutes an *element*. Given an element, it performs a distinct function. For example, a textual *variant* of a passage is one element of the apparatus, the source for that variant is another.
2. *Typographic Structure*:
   Given an element of the apparatus, the editor and/or typesetter must decide where (layout) and how (style) to place that element on the page. Will we use footnotes, endnotes, marginal notes, or a separate volume entirely for the presentation of the apparatus? What symbol will we use to separate the *lemma* of a given *entry class* from its associated *comment*?

EDMAC and other critical edition packages for TEX mix elemental structure and typographic structure. In what follows we try to precisely identify the various components of the elemental structure of the critical edition. Then, when we consider typesetting in ConTEXt, one may choose a plethora of options for the typographic structure. This allows for much more flexibility as we will see. Another advantage of this approach is that it gives us a framework for easy translation of apparatus data to and from XML.

The Text Encoding Initiative (TEI) has defined a special XML schema for critical editions (`tei-c.org/release/doc/tei-p5-doc/html/TC.html`). Some parts of our structure roughly correspond to aspects of the TEI schema for critical editions. That schema is more detailed and less precise than we need; for us at this juncture it is more useful to have a clearer and more precisely defined elemental structure. Then, once the typographic structure interface is in place, a user can define those elements of the TEI schema that are needed.

Towards an ontology of Arabic-script typography: An implementation strategy for Oriental TEX*

Idris Hamid

At the core of the Oriental TEX project is the implementation of a scheme for typesetting culturally authentic Arabic-script. Such cultural authenticity has, in general, been on the decline since the onset of digital typography in the Arabic-script world. The coming-of-age of Unicode as well as the new OpenType font standard is playing some part in alleviating this situation at the font level. Yet, although digital typography carries within it by far much the greater potential for high-quality Arabic typesetting, Arabic digital typography has yet to meet the standards set by lead-press typesetting in the Arabic-script world.

In this document we present the outlines of a new approach to the *ontology* or *typology* of Arabic-script typography. Ontology is the philosophical study of the categories of being. In this case, our universe of discourse is the Arabic script, particularly its typeset instantiation. An ontology of Arabic-script typesetting, then, identifies and organizes the categories of the manifestation of Arabic script in the context of typography.

Our analysis is organized along the following lines:

- The three-fold challenge of Arabic-script digital typography: First-, second-, and third-order analysis of Arabic script
- A critical examination of Arabic script and Unicode
- First-order analysis of the Naskh script

- Historical overview of Naskh typography
- Second-order analysis of the Naskh script: Macro-typography
- Third-order analysis of the Naskh script: Micro-typography

An experimental CTAN upload process
Jim Hefferon

Some improvements to the package upload facility at CTAN have the potential to make the process smoother. We'll talk about some of the features being developed and tested, and also have a demo.
(This paper will appear in the EuroBachoTEX 2007 proceedings. Ed.)

The breqn package: revised and revived
Morten Høgholm

The breqn package was originally developed by Michael Downes of the American Mathematical Society to facilitate automatic line-breaking of displayed math expressions. It has recently undergone some restructuring as part of a thesis project and is now actively maintained again. This presentation gives an overview of the current status of breqn, what it can do and what it can't (yet) do, and finally what the immediate, mid-term and long-term goals are.

About Lua
Roberto Ierusalimschy

Lua is an embeddable scripting language that aims for simplicity, small size, portability, and performance. Unlike most other scripting languages, Lua has a strong focus on embeddability, favoring a development style where parts of an application are written in a "hard" language (such as C or C++) and parts are written in Lua. Currently Lua is used in a vast range of applications, being regarded as the leading scripting language in the game industry.

In this talk I will give an overview of the language, covering not only the technical aspects of the language but also its origins back in 1993, its evolution, and its current status.

XeTeX Live
Jonathan Kew

With the release of TeX Live 2007, the XeTeX engine has "come of age" and entered the mainstream of the TeX world. XeTeX, which provides built-in Unicode and OpenType support, is now a standard part of a TeX Live installation, and thus is readily available to any user who installs this distribution.

This presentation will show how users can take advantage of XeTeX to easily use additional fonts in TeX or LaTeX documents, with no complex installation or setup procedures. It will also show how non-Latin scripts such as Chinese, Arabic, Devanagari, and many others can be typeset just as easily as English, thanks to full Unicode support throughout the system.

In addition, some of the newest developments in XeTeX (beyond the TeX Live 2007 release) will be discussed and demonstrated. These include support for the Graphite rendering technology for complex scripts; extensions that can simplify Chinese/Japanese character spacing and mixed-script typesetting; and more complete Unicode math support.
(This paper will appear in the EuroBachoTEX 2007 proceedings. Ed.)

Everything you wanted to know about PDF but were afraid to ask
Leonard Rosenthol

If you ever wondered just what makes PDF tick, come to this presentation by Adobe's PDF Standards Evangelist. You'll learn about the many features of PDF, from page content to interactive features to 3D, and how it all fits together. PDF is on track to becoming an ISO standard.

Vistas for TeX
Chris Rowley

This is a polemic in favour of liberating the core typesetting structures and algorithms around which TeX is built from the monolithic superstructure of the program called tex and its derivatives such as xetex, luatex etc.

Although the aims of the programme of activity advocated here are have a lot in common with those behind the very exciting and active luaTeX project, the route I support seems to me to be to be very different from embedding the whole of the TeX system within such a vastly more complex monolith (sic), along with its many intrusions (sic) into but a single instance of the ancient base rock of TeX! Of course, this is by no means all that luatex promises to give us, hence the importance and fascination for me of the approach taken by the luaTeX project.

Pursuing the paleontological metaphor well beyond its total collapse, my plan can be thought of as providing many tools for influencing the evolution of automated tpesetting without the need to fossilize the perectly preserved skeleton of the whole ancestral dinosaur.

Incorporating LaTeX text into graphics and presentations with LaTeXiT
Ari Stern

It is often a challenge to combine text created in LaTeX with content from other software, such as graphics and presentation software, while maintaining a consistent typographical style; for instance, matching text labels in mathematical figures to the main text. This demo will show how this can be done easily using LaTeXiT, a free utility for Mac OS X included with the MacTeX distribution. Other examples will include 2D and 3D mathematical figures in Adobe Illustrator, as well as presentations using Apple Keynote.

ConTEXt basics for users: Table macros

Aditya Mahajan

Abstract

ConTEXt has four different table building macros. In the author's view the `table` macros, which are the oldest of the four, are the easiest to use for simple tables. This article gives a gentle introduction to these macros.

1 Introduction

Tables provide visual representation of information; understanding how to build them in a markup language can be difficult and frustrating. To make matters especially confusing for a new user, ConTEXt provides four different mechanisms to build tables — tables, tabulations, line tables, and natural tables— and the user must decide which one of the four to use. This can be a difficult decision, because there is no "one size fits all" solution. Each mechanism has its own strengths and weaknesses, and the best match depends on the application.

Tables can get fairly complicated. They can have fancy horizontal and vertical lines, colored rows, sophisticated MetaPost backgrounds for cells, and/or may need to be split across multiple pages. Some mechanisms are better than others at handling such advanced features.

However, most users rarely need these features; for the most part, they just want a simple, publication quality table. I think that out of the four table building macros, the `table` macros, which are based on Michael Wichura's TABLE package [1], are the easiest to use for simple tables.

I should note that this mechanism is no longer actively developed, and the more modern natural table macros are the unofficially recommended choice. However, I find natural tables to be a bit too verbose. The `table` macros take care of the *simple* tables. If you need to typeset *complicated* tables, you need to look into all the mechanisms and decide which one suits your needs best.

This article explains the basics of the `table` macros. It is cumbersome to separate out the features of Michael Wichura's TABLE package, and those added or adapted by Hans Hagen in the course of integrating with ConTEXt. For simplicity of exposition and with sincere apologies to Michael, I am going to refer to all these features as features of ConTEXt's `table` macros.

2 How to build tables

The basic structure of a table built with the `table` macros looks like this:

```
\starttable[preamble]
 \NC ... \NC \AR % first row
     ...         % middle rows
 \NC ... \NC \AR % last row
\stoptable
```

The *preamble* indicates the number and formatting of the columns. The column formatting is defined by "column specifiers", separated by vertical bars `|`; we also put bars at the beginning and end of the preamble. These `|` characters do *not* indicate vertical lines, as they do in LATEX; they simply separate columns.

Each column specifier consists of keys, which are single characters, and the key combination indicates the formatting for that column. The number of columns is simply the number of column specifiers (that is, the number of `|`'s minus one).

The body of the table can contain an arbitrary number of rows. Each row has the form

```
 \NC cell 1 \NC ... \NC cell n \NC \AR
```

The `\NC` (mnemonic: new column) control sequence is the usual column separator, and `\AR` (mnemonic: automatic row) is the usual row separator. Each column separator corresponds to one `|` in the preamble.

Let's consider an example. Suppose we want to produce the following table:

left	center	right
23	45	67

Each row consists of three columns, with the first one left aligned, the second center aligned, and the last right aligned. The preamble for this table is `[|l|c|r|]`. Here `l` is an abbreviation for left,[1] `c` for center, and `r` for right. Each column is as wide as necessary to accommodate the width of its contents. The input for the above table is:

```
\starttable[|l|c|r|]
  \NC left \NC center \NC right \NC \AR
  \NC 23   \NC 45     \NC 67    \NC \AR
\stoptable
```

The "extra" `\NC`'s at the end of each row are required. Omitting them can lead to hard-to-detect problems.

3 Specifying column widths

By default, each column is wide enough to accommodate the largest entry in that column. This may not

[1] `l` is actually *ell*. In the fonts used for this article, it is difficult to distinguish `l` (ell) from `1` (one). To make things easier for the reader, I will not use the digit `1` (one) in any example.

always be desired. If you want to ensure that each column has a certain *minimum* width, use `w(width)` in the column specifier. For example, if we want the middle column in the above table to be at least 4 cm wide, we can specify the preamble as

```
  \starttable[|l|cw(4cm)|r|]
```

which gives

left	center	right
23	45	67

On the other hand, if you want to specify a certain *maximum* width of a column, use the `p(width)` in the column specifier. Such columns are called paragraph columns. For example, suppose we want a column to be 5 cm wide:

TUGboat	The TUGboat journal is a unique benefit of joining TUG. It is currently published three times a year ...

We can use the following:

```
\starttable[|l|lp(5cm)|]
  \NC TUGboat
  \NC The TUGboat journal ...
  \NC \AR
\stoptable
```

This `lp(5cm)` key combination gives us left-justified text, i.e., ragged right. You can also use `rp(5cm)` to get right-justified (ragged left) text, and `cp(5cm)` to get centered text.

However, in most cases we want such paragraph columns to be justified. To achieve this we can use `xp(5cm)`, which gives:

TUGboat	The TUGboat journal is a unique benefit of joining TUG. It is currently published three times a year ...

4 Horizontal and vertical lines

To get horizontal lines that span the width of the entire table, you can use `\HL` between the rows where a line is desired.

By default, the width of the line (a.k.a. "rule") is 0.4 pt — TeX's default for lines. The linewidth can be controlled by the `rulethickness` option of `\setuptables`. For example, if you want 2 pt thick lines, use `\setuptables[rulethickness=2pt]`.

The width of a specific line can be increased using an optional argument to `\HL`. This argument must be an integer, and it increases the width of the current line by that factor. For example:

```
\setuptables[rulethickness=0.03em]
\starttable[|l|l|r|]
  \HL[3]
  \NC Animal \NC Desc \NC Cost (\$) \NC \AR
  \HL
  \NC Gnat \NC per gram \NC 13.65 \NC \AR
  \NC Emu  \NC stuffed  \NC 33.33 \NC \AR
  \NC Gnu  \NC stuffed  \NC 92.50 \NC \AR
  \HL[3]
\stoptable
```

gives

Animal	Desc	Cost ($)
Gnat	per gram	13.65
Emu	stuffed	33.33
Gnu	stuffed	92.50

Here the first and last lines are $3 * 0.03\,\text{em} = 0.09\,\text{em}$ thick, while the middle line is the default 0.03 em. Notice that the height of the rows around the horizontal lines is automatically adjusted — that's the `A` in `\AR`.

Other row specifiers allow manual adjustment of space: `\NR` (new row), `\SR` (single row), `\FR` (first row), `\MR` (middle row), and `\LR` (last row). Depending on the surrounding rows, `\AR` is converted into one of these row specifiers.

One can use `\tracetablestrue` to see what ConTeXt is doing behind the scenes. If we add `\tracetablestrue` before calling the above table, we get:

Animal	Desc	Cost ($)	\SR
Gnat	per gram	13.65	\FR
Emu	stuffed	33.33	\MR
Gnu	stuffed	92.50	\LR

The first row was surrounded by two horizontal lines, so the `\AR` in the first row was changed into `\SR` (single row). This introduces some space above and below the row, so that the row is not too close to the horizontal lines.

The second row has a horizontal line above it, so the `\AR` was changed into `\FR` (first row). This adjusts the space above the row to provide some distance from the horizontal line.

The third row does not have a horizontal line above or below it, so the `\AR` is changed into `\MR` (middle row). This adds normal inter-row space above and below the row.

The last row had a horizontal line below it, so the `\AR` was changed into `\LR` (last row). This adjusts the space below the row to provide some distance from the horizontal line.

The \NR command sequence does not adjust space at all. It should be used only when such tight spacing is required.

If you feel that ConTEXt has made a wrong choice, you can use the desired row separator instead of \AR to end that particular row in the table.

Vertical lines are rarely considered good typography. However, if you insist, use \VL as the column separator instead of \NC. For example, this input:

```
\starttable[|l|cw(4cm)|r|]
  \HL
  \VL left \VL center \VL right \VL \AR
  \HL
  \VL 23   \VL 45     \VL 67    \VL \AR
  \HL
\stoptable
```

gives

left	center	right
23	45	67

This also shows that with the w key, the width of the middle column was increased, in this case with centered text.

5 Documentation

So far I have shown some very basic features of the table macros. A few more details of these macros are documented in the beginner's manual [2]. However, the manual does not list all features of these macros. Documentation of the features inherited from the TABLE package is available in its own manual, which is sold by PCTEX [3].

Unfortunately, there is no real documentation of the enhancements, especially with regards to color, done by ConTEXt. There are a few examples in the source file core-tab.tex [4], and a few more on the ConTEXt wiki [5].

So, in the next issue of this series, I will explain some additional features of table macros, such as specifying the font style and color of each column, spanning multiple rows and columns, controlling the space between the columns, and splitting tables across pages.

6 Other mechanisms

To conclude this installment, here is a brief overview of the other table-building mechanisms mentioned earlier.

The tabulate macros were added for building running text aligned blocks like formula legends and facts. They are capable of automatic width calculation of paragraph columns (similar to the tabularx package in LATEX) and splitting across pages. They are built on the same underlying principle as the table macros, and in due time will be backward compatible with them. However, at present they do not support vertical rules and colors. Documentation and examples of tabulations are given in a *MAPS* article [6] by Hans Hagen.

Line tables are experimental macros for building large tables that can be split horizontally and vertically. It is possible to repeat table header lines and entry columns. Unfortunately, these macros are largely undocumented.

Natural tables are the most configurable table macros. Their syntax is inspired from HTML tables. It is possible to change the color, background, and style of a single cell, row, or column, and of odd or even rows and columns. It is also easy to use MetaPost backgrounds. Again, there is no detailed documentation, but many examples are present in Hans's example document [7]; another interesting example is Willi Egger's MyWay [8].

Finally, the pros and cons of each of these mechanisms is summarized in an article in the ConTEXt Garden wiki [9].

Bibliography

[1] Michael Wichura: The TABLE Macro Package. http://www.pctex.com/kb/47.html.

[2] Hans Hagen: ConTEXt an excursion. http://www.pragma-ade.com/show-man-1.htm.

[3] Michael Wichura: PICTEX and TABLE manual. http://store.pctexstore.com/maclimprom.html.

[4] Hans Hagen: ConTEXt core macros — TABLE embedding. http://www.logosrl.it/context/modules/current/singles/core-tab_ebook.pdf.

[5] ConTEXt garden — Table. http://wiki.contextgarden.net/Table.

[6] Hans Hagen: Tabulating in ConTEXt — text flow tables. *NTG MAPS*, Spring 1999. http://www.ntg.nl/maps/pdf/22_28.pdf.

[7] Hans Hagen: Natural tables in ConTEXt. http://www.pragma-ade.com/general/manuals/enattab.pdf.

[8] Willi Egger: My Way — Use of natural table environment. http://dl.contextgarden.net/myway/NaturalTables.pdf.

[9] ConTEXt garden — Tables Overview. http://wiki.contextgarden.net/Tables_Overview.

◇ Aditya Mahajan
University of Michigan
adityam (at) umich dot edu

A roadmap for TEX development

TUG's TEX Development Fund committee

A foundation has generously provided a substantial outright grant and an equal matching grant to the TEX Users Group (TUG) to be used for TEX development. Along with these grants, there was a strong suggestion, with which we fully concur, to create a "roadmap" of TEX development for allocating these and future funds.

We do not contemplate a prescriptive roadmap. No central authority can dictate TEX directions, as many independent institutions and individuals contribute to TEX's development. In addition, the available money is not sufficient to fully fund (and thus perhaps more strongly direct) major projects.

The TEX user groups do cooperate in many ways, including coordination of development efforts. For instance, one major community project is the Comprehensive TEX Archive Network (CTAN), supported by the user groups and other institutions. CTAN amounts to a giant library of user-developed shared routines, and is the central release point for TEX updates; the TEX Live and MiKTEX distributions take the bulk of their material from CTAN.

Overall, we feel our best general strategy for allocating funding is to select existing efforts that look especially significant and promising, and which already involve people who have the necessary skill, motivation, availability, and a track record of major TEX accomplishments. We can then give those people a financial boost, as well as a nudge of approval, that we can hope will help them stay motivated and working, perhaps finishing a little faster and with greater certainty.

When we look around the TEX world for such ongoing efforts and people, three activities look especially exciting to us:

- LuaTEX, `http://www.luatex.org/`
- TEX Gyre fonts, `http://www.gust.org.pl/projects/e-foundry/tex-gyre/`
- XƎTEX, `http://scripts.sil.org/xetex/`

LuaTEX

The LuaTEX effort aims to fulfill the long-held desire for a general purpose programming language embedded in TEX: the difficulty of coding everything in TEX macros is well known.

The project is being led by Hans Hagen (developer of the ConTEXt macro package et al., as well as president of the Dutch language TEX user group), Taco Hoekwater (current maintainer of MetaPost and long-time TEX developer), and Hartmut Henkel (another long-time developer on pdfTEX and other projects), with assistance from several others. Generally speaking, Hans is handling the Lua side of things, Taco is handling the TEX side of things, and Hartmut is handling the PDF side of things. The project is guided to some extent by the successful development of pdfTEX, originated by Hàn Thế Thành (who is consulting to LuaTEX) and subsequently joined by essentially this same group of people.

The LuaTEX FAQ gives the following criteria which resulted in Lua being the language chosen for the project: freely available, portable, straightforward to embed within pdfTEX, small footprint, easy to extend with pdfTEX-specific functionality, and fun to work with.[1] In addition to Lua and some of its libraries, the LuaTEX project intends to include bidirectional typesetting functionality from Aleph, have flexible and diverse font support (including Unicode and OpenType), and integrate MetaPost as a native graphics capability in the system.

The project has been underway for about two years, and the first public beta demonstration was in July 2007 at the TUG 2007 conference: the third day of the conference included several presentations relating to LuaTEX.[2] A public release is expected in the summer of 2008, and some early users are already working with and testing it.

A related project is MPlib: modernizing the MetaPost implementation to greatly improve graphics support in LuaTEX, among other benefits.

LuaTEX has already received a grant from Colorado State University (for support of typesetting of Oriental languages, which relates to the features above) and support from the TEX user groups. Continued funding is critical to maintain progress.

TEX Gyre fonts

The TEX Gyre project aims at extending the 33 base PostScript text fonts (the 35 fonts minus the two symbol fonts) by adding glyphs and accents to support all languages using the Latin character set, and also making them available in OpenType format. Furthermore, the existing free versions of these fonts never had enough math capability for the needs of many TEX users, and so Gyre also plans to add Unicode-based math — a gargantuan effort.

The people doing this work are Bogusław Jackowski and Janusz Nowacki of the Polish lan-

[1] The FAQ continues, "Lua was the first language to match all these criteria. The 'known' scripting languages tended to be much too large for our use. Specifically, we have rejected Java, Perl, Python, Ruby, Scheme on one or more of those criteria."

[2] `http://river-valley.tv/conferences/tex/tug2007`

guage TeX user group, who developed and continue to maintain the Latin Modern font family (`http://www.gust.org.pl/projects/e-foundry/latin-modern`), which extend Knuth's Computer Modern in just the same way. They have also reconstituted several other historical typefaces. The infrastructure for these prior projects is now being brought to bear on Gyre.

An interesting connection is that the LuaTeX developers have stated that the Gyre fonts will be a basic component of the LuaTeX distribution.

To date, the Gyre project has been funded by the TeX user groups. Again, continued funding is critical to maintain progress.

XeTeX

XeTeX has been developed by Jonathan Kew of SIL. It is a modification of Knuth's TeX engine enabling use of Unicode and modern font technologies. XeTeX thus allows users to ignore the complexities that typically frustrate a new TeX user (and many long-time TeX users) when they try to configure their systems to use fonts not originally built for use with TeX, i.e., most of the fonts in the world.

XeTeX is substantially a one-man show, and the product of Jonathan's efforts is eliciting great excitement and interest in the TeX world. While XeTeX was originally developed for Mac OS X systems, the 2007 release of TeX Live includes XeTeX binaries for Windows, GNU/Linux, and many other Unix variants. Jonathan's presentation on XeTeX at the TUG 2007 conference is available from the same web site cited earlier.

While the XeTeX and LuaTeX project are basically independent, Hans et al. and Jonathan maintain contact and see no severe incompatibilities.

Jonathan's work with XeTeX shows his motivation and capabilities. However, we have caught the XeTeX effort at a point where its core functionality is already complete, and therefore is not a good candidate for our limited funding — which brings us to the next project.

A new front-end

Our donor strongly requested consideration of a new or updated front end, based especially on experiences working with students. As it turns out, Jonathan Kew is seeking additional work, and he is also personally interested in working on such a front-end project.

Thus, he, Karl Berry, and Richard Koch (who created and maintains the successful TeXShop front end for Mac OS X, and is also a new TUG director) have sketched some approaches for a new front end and how it could improve on the many extant programs in this area. Examples: use an existing cross-platform toolkit (the goal being to appear "native" to users on any platform); ease importing, conversion, and placement of graphics; ease handling of errors; automate typeset output refresh to minimize pain from the edit-compile-preview cycle.

Of course we cannot know precisely how a new program will turn out, but we are confident that there is a niche for it, and that Jonathan is the right person to get the project off the ground.

Other worthy work

There is plenty of other worthy work going on, and it is not our intention to slight it. However, we feel it is better to select a few important projects where funding is known both to be needed and to make a significant difference, rather than to try to fund many projects in a smaller way.

Our recommendations

Based on the above, we propose to divide the grant funds into three parts, allocating one part to LuaTeX, one part to TeX Gyre, and one part to initiating a suitable front-end project, making use of Jonathan Kew's availability. The proportions will be determined as the need arises; for instance, if a particular project receives significant support from other sources, clearly that could have an impact.

We cannot emphasize enough that we believe the best way to make use of these funds is to find motivated and capable people who we can expect to work in sensible and pragmatic directions — not to try to guess the best directions and then struggle to find people to do work not of their own choosing.

TeX Development Fund committee

TUG president Karl Berry is deeply involved in many areas of the TeX world, including aspects of core TeX development. Kaja Christiansen is the vice-president of TUG, has a development background, maintains TeX (among other things) at the University of Århus in Denmark, and supports the TUG web site there. Jim Hefferon is a TUG director and on the mathematics faculty at Saint Michael's College in Vermont, where he runs one of the three backbone CTAN nodes. Dave Walden is treasurer of TUG, is an intermediately skilled user of TeX, and spent his pre-retirement working life contributing to and leading significant and innovative software development projects.

◇ TUG's TeX Development Fund committee
Info: `http://tug.org/tc/devfund`
Donations: `http://tug.org/donate`

Random comments

TUGboat Editors

Errata: Lars Hellström, "Writing ETX format font encoding specifications", *TUGboat* 28:2, pp. 186–197

Owing to a production error, corrections intended for this article were omitted from the printed version; the on-line version incorporates all fixes. We list below the most egregious omissions.

- p. 191, footnote: Ulrik Vieth's name should *not* be hyphenated; our abject apologies to Ulrik.
- p. 193, col. 2, line 9: for `\endcoding` read `\encoding`.
- p. 197, ref. [5]: the file reference should be `fontinst.pdf`.
- p. 197, ref. [9]: the URL should be `http://omega.enstb.org/roadmap/doc-1.12.ps`.
- p. 197, ref. [10]: the URL should be `http://www.ntg.nl/maps/pdf/26_27.pdf`.

DEK's periodic bug review

Don Knuth has announced on his TEX web page (`http://www-cs-faculty.stanford.edu/~knuth/abcde.html`) that 2007 will be a "bug review" year. The last review took place in 2002, and the interval between reviews is increasing by a year with each cycle, so the next review won't occur until 2013.

If you believe you have found a bug, please do the following:

- check the bug listings and *all* the errata lists at CTAN, in the area `tex-archive/systems/knuth/errata`;
- create a minimal test file that demonstrates the error;
- send the test file and a clear description of the problem to `bnb@ams.org` as soon as possible, to get included in the current cycle;
- include a postal address valid through the first few months of 2008, for delivery of any reward checks.

Don requires that every bug be vetted by a competent TEXnician (approved by him) before it is forwarded to him for review; this process weeds out quite a few non-bugs or items that have already been addressed. Then, the collection is organized into logical categories (*The TEXbook*, TEX the program, etc.) so that it will take a minimum of Don's time. However, the preparatory steps do take (sometimes considerable) time, and all the individuals involved in the process are very busy, so please be thoughtful and timely in your reports.

When Don reviews the collection, he does so from a printout of the reports, writes his comments in pencil on this copy, and returns it to BNB for transcription and distribution. Because transcription can take a very long time, for this cycle Don will be sending out checks directly. The updated material will be posted to CTAN as soon as it is ready.

"Off-site" complement to TUG 2007 proceedings

Rather than write a paper specifically for these proceedings, Hans Hagen has chosen instead to create an on-line document describing the current state of ConTEXt with LuaTEX — a rapidly moving target at this moment. Two sections are included in this issue: the introduction, and a report on using the Unicode UTF encodings as the basis for the next generation. (The encoding question has been examined and experimented with enough that it is reasonably stable.)

The complete document, "ConTEXt: MkII & MkIV", can be found via a link at `http://www.pragma-ade.com/overview.htm`.

Other comments on the TUG 2007 proceedings

The work by Idris Hamid, on critical editions and Arabic script typography, which provided an important impetus as well as much of the funding for the development of LuaTEX, will appear in a separate monograph, expected to be published by TUG later this fall.

In lieu of the ConTEXt articles, we are including Aditya Mahajan's next installment of his column on ConTEXt basics, this one on tables. (In a nice coincidence, Klaus Höppner's article in these proceedings is an introduction to tables in LATEX: enjoy the comparison.) Aditya's first installment was on fonts in ConTEXt, published in *TUGboat* 28:2 and available online at `http://tug.org/TUGboat/Articles/tb28-2/tb89mahajan.pdf`. We are very pleased that Aditya has undertaken this series, and expect that future issues will contain more of his columns on other ConTEXt topics, directed to new and intermediate users.

TEX Development Fund

A generous contribution from an anonymous foundation has provided a welcome infusion to the TEX Development Fund as well as a matching grant to encourage donations from other sources. (The Development Fund is listed on the membership form among various projects for which contributions are accepted.) The committee overseeing the fund has taken this opportunity to develop a "roadmap" to guide the allocation of grants from the fund. The full text of the roadmap appears in this issue. Please read it carefully, and let the Board know if you have comments or suggestions. And please consider a contribution to help meet the matching grant: `http://tug.org/donate`.

EuroBachoTEX proceedings to be the first 2008 *TUGboat* issue

The organizers and program committee of EuroTEX 2007 (held in Bachotek, Poland, April 28–May 2, 2007) approached the boards of TUG and all the European user groups to see if it would be possible to distribute the proceedings to all their members. Such distribution would not only reach a wider audience than a proceedings published only for GUST, it would also greatly reduce the unit production cost.

The *TUGboat* editors offered their redactory services, and TUG will join the other groups in distributing these proceedings to their members. Work on the collection is expected to be complete by the end of the year.

Institutional Members

Aalborg University, Department of Mathematical Sciences, *Aalborg*, *Denmark*

American Mathematical Society, *Providence*, *Rhode Island*

Banca d'Italia, *Roma*, *Italy*

Center for Computing Sciences, *Bowie*, *Maryland*

Certicom Corp., *Mississauga*, *Ontario Canada*

CNRS - IDRIS, *Orsay*, *France*

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, *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*

New York University, Academic Computing Facility, *New York*, *New York*

Princeton University, Department of Mathematics, *Princeton*, *New Jersey*

Springer-Verlag Heidelberg, *Heidelberg*, *Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford*, *California*

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*

Universiti Tun Hussein Onn Malaysia, Pusat Teknologi Maklumat, *Batu Pahat*, *Johor*, *Malaysia*

University of Oslo, Institute of Informatics, *Blindern*, *Oslo*, *Norway*

Vanderbilt University, *Nashville*, *Tennessee*

Calendar

2007

Sep 1 – 2	Transylvania TeX Conference, “Babeş-Bolyai” University Cluj-Napoca, Romania. `math.ubbcluj.ro/~aga_team/translatex`
Sep 12 – 16	Association Typographique Internationale (ATypI) annual conference, Brighton, UK. `www.atypi.org`
Sep 15	DANTE TeX-Tagung, 37th meeting, Universität Ulm, Germany. `www.dante.de/dante/events/mv37`
Sep 18 – 19	Conference on “Non-Latin typeface Design”, St Bride Library, London, and the Department of Typography, University of Reading, UK. `stbride.org/events_education/events`
Sep 24 – Nov 22	Guild of Book Workers 100th Anniversary Exhibition: A traveling juried exhibition of books by members of the Guild of Book Workers. Dartmouth College Library, Hanover, New Hampshire. Sites and dates are listed at `palimpsest.stanford.edu/byorg/gbw`
Oct 8	GUTenberg Workshop on Unicode & LaTeX, Paris, France. `www.gutenberg.eu.org`
Oct 11 – 13	American Printing History Association 2007 annual conference, “Transformations: The persistence of Aldus Manutius”, University of California at Los Angeles and the Getty Research Institute. `www.printinghistory.org`
Oct 13	GuIT meeting 2007 (Gruppo utilizzatori Italiani di TeX), Pisa, Italy. `www.guit.sssup.it/GuITmeeting/2007`
Oct 20 – 22	The Fifth International Conference on the Book, “Save, Change or Discard: Tradition and Innovation in the World of Books”, Madrid, Spain. `b07.cgpublisher.com`
Dec 3 – 5	XML 2007 Conference,, “XML in Practice”, Boston, Massachusetts. `2007.xmlconference.org`

2008

Mar 5 – 7	DANTE 2008, 38th meeting, Friedrich-Schiller-Universität, Jena, Germany. `www.dante.de/dante2007`
May 15 – 16	Seventh annual Friends of St Bride Library Conference, “Seeking inspiration: Creative thinking around the design process”, London, England. `stbride.org/events_education/events`
Jul 15 – 19	TypoCon 2008, 10th anniversary, Buffalo, New York. `www.typecon.com`

TUG 2008 — TeX’s 30th birthday
University College Cork, Ireland.

Jul 21 – 24	The 29th annual meeting of the TeX Users Group. `www.tug.org/tug2008`.

Aug 11 – 15	SIGGRAPH 2008, Los Angeles, California. `www.siggraph.org/events/s2008`
Aug 20 – 25	Second International ConTeXt User Meeting, Kranjska Gora, Slovenia. `meeting.contextgarden.net`
Jun 24 – 28	SHARP 2008, “Teaching and Text”, Society for tne History of Authorship, Reading and Publishing, Oxford Brookes University, Oxford. `ah.brookes.ac.uk/conference/sharp2008`
Jnn 25 – 29	Digital Humanities 2008, Association of Literary and Linguistic Computing / Association for Computers and the Humanities, Oulu, Finland. `www.ekl.oulu.fi/dh2008`
Sep 16 – 19	ACM Symposium on Document Engineering, São Paolo, Brazil. `www.documentengineering.org`

Status as of 1 September 2007

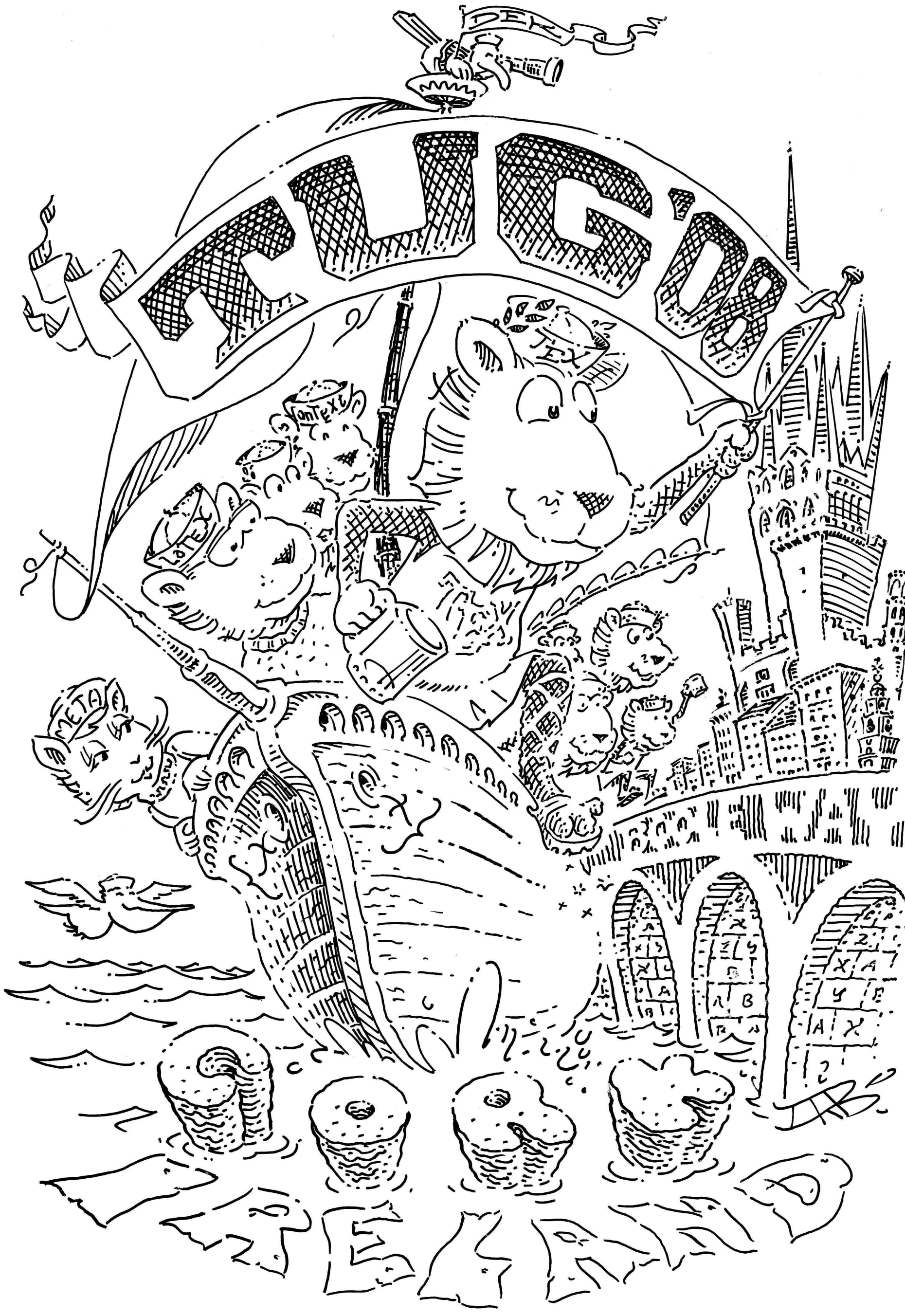
For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 206 203-3960, e-mail: `office@tug.org`). For events sponsored by other organizations, please use the contact address provided.

An updated version of this calendar is online at `www.tug.org/calendar`

TeX Users Group 2008 Conference

University College Cork
Cork, Ireland
21–24 July 2008
`http://tug2008.ucc.ie/`

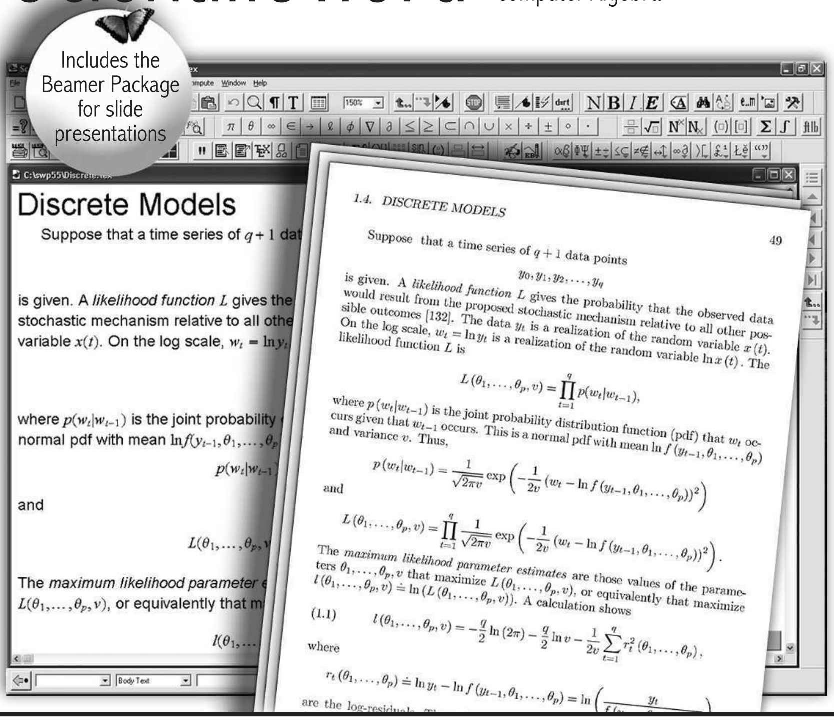
TeX's 30th birthday
Interfaces to TeX
Workshops
Presentations



Hosted by the Human Factors Research Group (`http://hfrg.ucc.ie`)

Scientific WorkPlace®
Scientific Word®

Mathematical Word Processing
LaTeX Typesetting
Computer Algebra



C:\swp55\Discrete.tex

Discrete Models

Suppose that a time series of $q+1$ dat

is given. A *likelihood function* L gives the
stochastic mechanism relative to all othe
variable $x(t)$. On the log scale, $w_t = \ln y_t$

where $p(w_t|w_{t-1})$ is the joint probability
normal pdf with mean $\ln f(y_{t-1}, \theta_1, \ldots, \theta_p$

$p(w_t|w_{t-1}$

and

$L(\theta_1, \ldots, \theta_p,$

The *maximum likelihood parameter*
$L(\theta_1, \ldots, \theta_p, v)$, or equivalently that m

$l(\theta_1, \ldots$

Body Text

1.4. DISCRETE MODELS 49

Suppose that a time series of $q+1$ data points

$$y_0, y_1, y_2, \ldots, y_q$$

is given. A *likelihood function* L gives the probability that the observed data would result from the proposed stochastic mechanism relative to all other possible outcomes [132]. The data y_t is a realization of the random variable $x(t)$. On the log scale, $w_t = \ln y_t$ is a realization of the random variable $\ln x(t)$. The likelihood function L is

$$L(\theta_1, \ldots, \theta_p, v) = \prod_{t=1}^{q} p(w_t|w_{t-1}),$$

where $p(w_t|w_{t-1})$ is the joint probability distribution function (pdf) that w_t occurs given that w_{t-1} occurs. This is a normal pdf with mean $\ln f(y_{t-1}, \theta_1, \ldots, \theta_p)$ and variance v. Thus,

$$p(w_t|w_{t-1}) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{1}{2v}(w_t - \ln f(y_{t-1}, \theta_1, \ldots, \theta_p))^2\right)$$

and

$$L(\theta_1, \ldots, \theta_p, v) = \prod_{t=1}^{q} \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{1}{2v}(w_t - \ln f(y_{t-1}, \theta_1, \ldots, \theta_p))^2\right).$$

The *maximum likelihood parameter estimates* are those values of the parameters $\theta_1, \ldots, \theta_p, v$ that maximize $L(\theta_1, \ldots, \theta_p, v)$, or equivalently that maximize $l(\theta_1, \ldots, \theta_p, v) \doteq \ln(L(\theta_1, \ldots, \theta_p, v))$. A calculation shows

$$l(\theta_1, \ldots, \theta_p, v) = -\frac{q}{2}\ln(2\pi) - \frac{q}{2}\ln v - \frac{1}{2v}\sum_{t=1}^{q} r_t^2(\theta_1, \ldots, \theta_p), \tag{1.1}$$

where

$$r_t(\theta_1, \ldots, \theta_p) \doteq \ln y_t - \ln f(y_{t-1}, \theta_1, \ldots, \theta_p) = \ln\left(\frac{y_t}{}\right.$$

are the log-residual

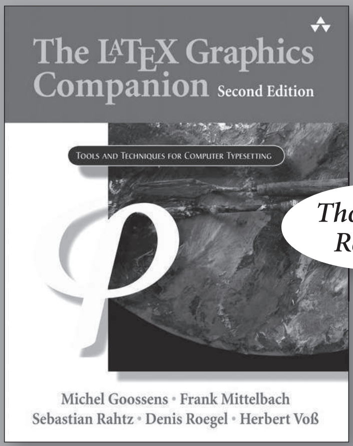
Scientific WorkPlace and ***Scientific Word Version 5.5*** make writing, sharing, and doing mathematics easier. You compose and edit your documents directly on the screen, without having to think in a programming language. A click of a button allows you to typeset your documents in LaTeX. You choose to print with or without LaTeX typesetting, or publish on the web. ***Scientific WorkPlace*** and ***Scientific Word*** enable both professionals and support staff to produce stunning books and articles. Also, the integrated computer algebra system in ***Scientific WorkPlace*** enables you to solve and plot equations, animate 2D and 3D plots, rotate, move, and fly through 3D plots, create 3D implicit plots, and more.

MacKichan SOFTWARE, INC.

www.mackichan.com/tug
Email: info@mackichan.com
Toll-free: 877-724-9673 • Fax: 360-394-6039

The LaTeX Graphics Companion

Second Edition



The LaTeX Graphics Companion Second Edition

Tools and Techniques for Computer Typesetting

Michel Goossens • Frank Mittelbach
Sebastian Rahtz • Denis Roegel • Herbert Voß

Thoroughly Revised!

Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voss

ISBN: 0-321-50892-0

The LaTeX *Graphics Companion* has long been the most comprehensive guide to making illustrations in LaTeX documents. This completely revised and expanded edition includes the latest developments in LaTeX graphics. The authors describe the most widely used packages and provide hundreds of solutions to the most commonly encountered LaTeX illustration problems.

Addison-Wesley

Available at fine bookstores everywhere.

For more information, visit:
www.awprofessional.com/title/0321508920

Do you need on-site training for LaTeX?

Contact Cheryl Ponchin at

`cponchin@comcast.net`

Training will be customized for your company needs.

Any level, from Beginning to Advanced.

Carleton Production Centre

HUMANITIES TYPESETTING
Specialising in Linguistics
Since 1991

613-823-3630 • 15 Wiltshire Circle
Nepean, Ont., Canada • K2J 4K9

New MathType^6 for Windows

- Create equations by typing TeX and insert them into Microsoft Word, PowerPoint, Wikipedia pages, and 1000's of other applications.
- Save equations as GIF images for blogs and wikis.
- Wikipedia and other wikis contain many equations that can be copied into MathType, and then into other applications and document types.

Download a free, 30-day evaluation — www.dessci.com

MathType™
The *best* thing for writing equations since chalk!™

Design Science, Inc. 140 Pine Avenue, Long Beach, CA 90802, USA Toll-free: 800-827-0685 or 562-432-2920, Fax: 562-432-2857, Email: sales@dessci.com

Design Science
How Science Communicates™

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 fill out the form at `https://www.tug.org/consultants/listing.html` or email us at `consult-admin@tug.org`. To place a larger ad in *TUGboat*, please see `http://tug.org/TUGboat/advertising.html`.

Kinch, Richard J.
7890 Pebble Beach Ct
Lake Worth, FL 33467
+1 561-966-8400
Email: `kinch (at) truetex.com`
Publishes TrueTEX, a commercial implementation of TEX and LATEX. Custom development for TEX-related software and fonts.

Martinez, Mercè Aicart
Tarragona 102 4^o 2^a
08015 Barcelona, Spain
+34 932267827
Email: `m.aicart (at) menta.net`
Web: `www.edilatex.com/`
We provide, at reasonable low cost, TEX and LATEX typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

Peter, Steve
310 Hana Road
Edison, NJ 08817
+1 (732) 287-5392
Email: `speter (at) dandy.net`
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.

Veytsman, Boris
2239 Double Eagle Ct.
Reston, VA 20191
+1 (703) 860-0013
Email: `borisv (at) lk.net`
Web: `http://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 twelve years of experience in TEX and twenty-five years of experience in teaching & training. I have authored several packages on CTAN and published papers in TEX related journals.

Table of Contents (ordered by difficulty)

TUGboat

Volume 28, Number 3 / 2007
TUG 2007 Conference Proceedings

TUGBOAT Volume 28 (2007), No. 3 TUG 2007 Conference Proceedings

Table of Contents (ordered by difficulty)