

TUGBOAT Volume 38 (2017), Number 2
BIULETYN GUST Zeszyt 25
TUG@BachTeX 2017 Conference Proceedings

TUG 2017	110	Conference sponsors, participants, program
	114	Jean-Michel Hufflen / <i>TUG@BachTeX 2017</i>
General Delivery	116	Janusz Nowacki / <i>Calligraphy by Barbara Galińska</i>
	118	Maciej Rychły / <i>Released sounds</i>
	125	Boris Veytsman / <i>The state of TeX</i>
Futures	126	Hans Hagen / <i>Children of TeX</i>
	141	Luigi Scarso / <i>MFLua 0.8 — Prologue</i>
Survey	145	Michał Gasewicz / <i>Off topic (completely): Many faces (and types) of beer</i>
	147	Jean-Michel Hufflen / <i>History of accidentals in music</i>
Methods	157	Willi Egger / <i>Bookbinding workshop: Making a portfolio</i>
	159	Barbara Beeton / <i>Debugging L^AT_EX files — Illegitimi non carborundum</i>
Typography	165	Kumaran Sathasivam, S.K. Venkatesan, Yakov Chandy / <i>Revealing semantics using subtle typography and punctuation</i>
	171	Boris Veytsman and Leila Akhmadeeva / <i>To justify or not to justify? Why bad typography may be harmful for your readers</i>
	173	Boris Veytsman / <i>Making ltxsparklines: The journey of a CTAN contributor into the world of CRAN</i>
Education	175	Petr Sojka and Vít Novotný / <i>TeX in Schools? Just Say Yes: The use of TeX at the Faculty of Informatics, Masaryk University</i>
Graphics	185	Takuto Asakura / <i>Implementing bioinformatics algorithms in TeX — the Gotoh package, a case study</i>
Software & Tools	188	Norbert Preining / <i>updmap and fntutil — past and future changes (or: cleaning up the mess)</i>
	193	Siep Kroonenberg / <i>TLaunch, the TeX Live Launcher for Windows</i>
	197	Sigitas Tolušis, Arūnas Povilaitis, and Valentinas Kriaučiukas / <i>Xdvipsk: Dvips ready for OpenType fonts and more image formats</i>
Fonts	202	Jerzy Ludwichowski / <i>GUST e-foundry current font projects</i>
	203	Hans Hagen / <i>Variable fonts</i>
	208	Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski / <i>Parametric math symbol fonts</i>
L^AT_EX	212	L ^A T _E X Project Team / <i>L^AT_EX news, issue 27, April 2017</i>
	213	Frank Mittelbach / <i>L^AT_EX table columns with fixed widths</i>
Macros	214	Vít Novotný / <i>Using Markdown inside TeX documents</i>
	218	Grzegorz Murzynowski / <i>GMS, the “General Meta-Scenarios”: A proper extension to the l3expan package of the expl3 bundle and language, two years later</i>
Bibliographies	238	Dávid Lupták / <i>Typesetting bibliographies compliant with the ISO 690 standard in L^AT_EX</i>
	245	Jean-Michel Hufflen / <i>MIBIB_{TeX} now handles Unicode</i>
Publishing	249	Lolita Tolenè / <i>TeX user habits versus publisher requirements</i>
	255	Marcin Borkowski / <i>Ten years of work in Wiadomości Matematyczne — an adventure with L^AT_EX and Emacs hacking</i>
Production Notes	263	Karl Berry / <i>Production notes</i>
Abstracts	264	Streszczenia
	270	TUG@BachTeX 2017 abstracts (de Souza, Egger, Hagen, Hoekwater, Izaola, Kwiatkowska, Ludwichowski, Miklavec, Mittelbach, Reutenauer, Scherwentke, Thiriet, Tomaszewski, Twardoch, Vieth)
	273	<i>Die TeXnische Komödie: Contents of issue 2/2017</i>
Book Reviews	274	Charles Bigelow / <i>Review and summaries: The History of Typographic Writing — The 20th century Volume 2 (ch. 1–5), from 1950 to 2000</i>
	280	Boris Veytsman / <i>Book reviews: What Is Reading For? and Some Reflections on Reading and Writing, Culture and Nature, & Sorting Things Out by Robert Bringhurst</i>
	282	David Walden / <i>Book review: Paper: Paging Through History by Mark Kurlansky</i>
Hints & Tricks	284	Karl Berry / <i>The treasure chest</i>
Cartoon	285	John Atkinson / <i>Word on the street</i>
News	286	Calendar
Advertisements	287	TeX consulting and production services
TUG Business	288	TUG institutional members

Polska Grupa Użytkowników systemu T_EX (gust.org.pl)

Biuletyn Polskiej Grupy Użytkowników Systemu T_EX (ISSN 1230-5630) is published by GUST.
Zeszyt 25.

T_EX Users Group (tug.org)

TUGboat (ISSN 0896-3207) is published by TUG.
Volume 38, Number 2.

TUG individual memberships

2017 dues for individual members are as follows:

- Regular members: \$105.
- Special rate: \$75.

The special rate is available to students, seniors, and citizens of countries with modest economies, as detailed on our web site. Also, anyone joining or renewing before March 31 receives a \$20 discount:

- Regular members (early bird): \$85.
- Special rate (early bird): \$55.

Members also have the option to receive TUGboat and other benefits electronically, for an additional discount.

Membership in the T_EX 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 carries with it such rights and responsibilities as voting in TUG elections. All the details are on the TUG web site.

TUGboat subscriptions

TUGboat subscriptions (non-voting) are available to organizations and others wishing to receive TUGboat in a name other than that of an individual. The subscription rate for 2017 is \$110.

TUG institutional memberships

Institutional membership is primarily a means of showing continuing interest in and support for T_EX and the T_EX Users Group. It also provides a discounted membership rate, site-wide electronic access, and other benefits. For more information, see tug.org/instmem.html or contact the TUG office.

Trademarks

Many trademarked names appear in these pages. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is.

[printing date: August 2017]

Printed in U.S.A.

TUG Board of Directors

Donald Knuth, *Grand Wizard of T_EX-arcana*[†]
Boris Veytsman, *President**
Arthur Reutenauer*, *Vice President*
Karl Berry*, *Treasurer*
Susan DeMeritt*, *Secretary*
Barbara Beeton
Johannes Braams
Kaja Christiansen
Taco Hoekwater
Klaus Höppner
Frank Mittelbach
Ross Moore
Cheryl Ponchin
Norbert Preining
Will Robertson
Herbert Voß
Raymond Goucher, *Founding Executive Director*[†]
Hermann Zapf (1918–2015), *Wizard of Fonts*

*member of executive committee

[†]honorary

See tug.org/board.html for a roster of all past and present board members, and other official positions.

Addresses

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

Telephone

+1 503 223-9994

Fax

+1 815 301-3568

Web

tug.org
tug.org/TUGboat

Electronic Mail

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

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

Technical support for
T_EX users:
support@tug.org

Contact the
Board of Directors:
board@tug.org

Copyright © 2017 Polska Grupa Użytkowników systemu T_EX and T_EX Users Group.

Copyright to individual articles within this publication remains with their authors, so the articles may not be reproduced, distributed or translated without the authors' permission.

For the editorial and other material not ascribed to a particular author, permission is granted to make and distribute verbatim copies without royalty, in any medium, provided the copyright notice and this permission notice are preserved.

Permission is also granted to make, copy and distribute translations of such editorial material into another language, except that the T_EX Users Group must approve translations of this permission notice itself. Lacking such approval, the original English permission notice must be included.

2017 TUG@BachTeX Conference Proceedings

TeX Users Group • Polska Grupa Użytkowników systemu TeX

Thirty-seventh annual TUG meeting • XXV GUST conference

Bachotek, Poland

April 29–May 3, 2017

**Biuletyn Polskiej Grupy Użytkowników
Systemu TeX**

ZESZYT 25

TUGBOAT

COMMUNICATIONS OF THE TeX USERS GROUP

TUGBOAT EDITOR BARBARA BEETON

VOLUME 38, NUMBER 2, 2017

PORTLAND, OREGON, U.S.A.

PROCEEDINGS EDITORS TOMASZ PRZECHLEWSKI

KARL BERRY

BOGUSŁAW JACKOWSKI

JERZY LUDWICHOWSKI

TUG@BachTeX 2017 — Premises, predilections, predictions

38th Annual TUG meeting • XXV GUST conference
April 29–May 3, 2017 • Bachotek, Poland



Organizers

Polska Grupa Użytkowników systemu TeX (GUST), gust.org.pl • TeX Users Group (TUG), tug.org

Organizing committee (GUST)

Jolanta Szelatyńska, Chair • Marek Czubenko • Janusz Gumkowski
• Bogusław Jackowski • Jerzy Ludwiczowski

Program committee

Bogusław Jackowski, Chair • Jerzy Ludwiczowski, Co-chair • Ryszard Kubiak, Secretary
• Karl Berry • Hans Hagen • Mojca Miklavc • Boris Veytsman

Sponsors

- Die Deutschsprachige Anwendervereinigung TeX e.V. (DANTE), www.dante.de
- 7bulls, 7bulls.com
- Information & Communication Technology Centre, Nicolaus Copernicus University, www.uci.umk.pl
- PARCAT Product Information Management, parcat.eu
- Frans Goddijn • Siep Kroonenberg • Sebastian Krüger • Frank Mittelbach
- Volker RW Schaa • Martin Schröder • Boris Veytsman • Alan Wetmore

Participants

Leila Akhmadeeva, Bashkir State Medical University,
Ufa, Russia

Takuto Asakura, Department of Bioinformatics, and
Systems Biology, Faculty of Science, The University
of Tokyo

Nelson H. F. Beebe, Department of Mathematics,
University of Utah, USA

Barbara Beeton, American Mathematical Society, USA

Piotr Bolek, 7bulls.com sp. z o.o., Warszawa, Poland

Marcin Borkowski, Wydział Matematyki i Informatyki,
Poznań, Poland

Andrzej Borzyszkowski, Instytut Informatyki, Gdańsk,
Poland

Gyöngyi Bujdosó, Faculty of Informatics,
University of Debrecen, Hungary

Katarzyna Burakowska, Gdański Archipelag Kultury,
Gdańsk, Poland

Yakov Chandy, TNQ Books & Journals Limited, India

Marek Czubenko, Uniwersytet Mikołaja Kopernika,
w Toruniu, Uczelniane Centrum Informatyczne,
Toruń, Poland

Paulo Ney de Souza, Books in Bytes, USA

Wiktor Dziubiński, tme.eu, parcat.eu, Łódź, Poland

Willi Egger, BOEDE, Sambeek, The Netherlands

Yukitoshi Fujimura, Japan

Deimantas Galčius, VTeX, Vilnius, Lithuania

Michał Gasewicz, Uniwersytet Mikołaja Kopernika,
w Toruniu, Uczelniane Centrum Informatyczne,
Toruń, Poland

Frans Goddijn, The Netherlands

Eimantas Gumbakis, VTeX, Vilnius, Lithuania

Janusz Gumkowski, GUST, Toruń/Warszawa, Poland

Hans Hagen, PRAGMA ADE, Hasselt, The Netherlands

Taco Hoekwater, Bittext, The Netherlands

Karel Horák, Academy of Sciences,
Institute of Mathematics, Praha, Czech Republic

Jean-Michel Hufflen, FEMTO-ST DISC, France

Andrzej Icha, Akademia Pomorska w Słupsku,
Instytut Matematyki, Słupsk, Poland

Zunbeltz Izaola, Books in Bytes, Spain

Bogusław Jackowski, BOP s.c., Gdańsk, Poland

Paweł Jackowski, GUST, Kraków, Poland

Piotr Kielanowski, Departamento de Física,
Centro de Investigación y de Estudios Avanzados,
Mexico City, Mexico

Ewa Kmieciak, GUST, Kraków, Poland

Jacek Kmieciak, GUST, Kraków, Poland

Adam Kolany, GUST, Katowice, Poland

Dorota Kolany, Pałac Młodzieży w Katowicach,
Katowice, Poland

Reinhard Kotucha, DANTE e.V., Hannover, Germany

Harald König, DANTE e.V., Balingen, Germany

Valentinas Kriauciukas, VTeX, Vilnius, Lithuania
Siep Kroonenberg, Rijksuniversiteit Groningen,
The Netherlands

Sebastian Krüger, DANTE e.V., GUST, Berlin, Germany
Ryszard Kubiak, Biuro Informatyki Ubezpieczeniowej,
Pachocki i Ziajka s.c., Gdańsk, Poland

Ján Kula, Praha, Czech Republic

Manfred Lotz, CTAN, DANTE e.V., Frankfurt, Germany

Jerzy Ludwchowski, Uniwersytet Mikołaja Kopernika
w Toruniu, Toruń, Poland

Mojca Miklavec, Sežana, Slovenia

Frank Mittelbach, L^AT_EX3 Project, Mainz, Germany

Grzegorz Murzynowski, Transfer Multisort Elektronik,
Sulejówek, Poland

Kristian Nordestgaard, DK-TUG, Denmark

Vít Novotný, The Faculty of Informatics,
Masaryk University, Brno, Czech Republic

Janusz Marian Nowacki, GUST, Grudziądz, Poland

Andrzej Odyniec, Macrolog S.A., Warszawa, Poland

Karel Píška, Institute of Physics, Academy of Sciences,
Praha, Czech Republic

Kaja Podlaska Christiansen, DK-TUG, Denmark

Norbert Preining, T_EX Live, TUG, Accelia Inc., Tokyo,
Japan

Tomasz Przechlewski, GUST, Sopot, Poland

Krzysztof Pszczoła, Instytut Matematyki i Kryptologii,
Wojskowa Akademia Techniczna, Warszawa, Poland

Arthur Reutenauer, Enköping, Sweden

Jan Ryćko, LPP S.A., Gdańsk, Poland

Marek Ryćko, GUST, Gdańsk, Poland

Luigi Scarso, NTG, GUST, GuIT, TUG, Padova, Italy

Volker RW Schaa, DANTE e.V., Darmstadt, Germany
Przemysław Scherwentke, Politechnika Wrocławska,
Instytut Matematyki i Informatyki, Wrocław, Poland

Martin Schröder, DANTE e.V., Duisburg, Germany

Agnieszka Sekuła, LPP S.A., Gdańsk, Poland

Petr Sojka, Faculty of Informatics, Masaryk University,
Brno, Czech Republic

Piotr Strzelczyk, BOP s.c., Gdańsk, Poland

Ewa Szelańska, Scan-System, Warszawa, Poland

Jolanta Szelańska, Uniwersytet Mikołaja Kopernika
w Toruniu, Uczelniane Centrum Informatyczne,
Toruń, Poland

Joanna Szyller, Gdańskie Wydawnictwo Oświatowe,
Gdańsk, Poland

Damien Thiriet, Lycée français René Goscinnny,
Warszawa, Poland

Lolita Tolenė, VTeX, Vilnius, Lithuania

Sigitas Tolousis, VTeX, Vilnius, Lithuania

Andrzej Tomaszewski, Acta Poligraphica, Warszawa,
Poland

Adam Twardoch, FontLab Ltd., MyFonts, Berlin,
Germany

Boris Veytsman, George Mason University, USA

Ulrik Vieth, Stuttgart, Germany

Stanisław Wawrykiewicz, GUST, Sopot, Poland

Alan Wetmore, US Army Research Laboratory, USA

Lutz Wirsig, Germany

Marcin Woliński, Instytut Podstaw Informatyki PAN,
Warszawa, Poland

Katarzyna Wójcik, GUST, Warszawa, Poland

Krzysztof Zubik, Gdańsk, Poland

11th ConT_EXt Meeting
Butzbach-Maibach
Germany
September 11-17, 2017
meeting.contextgarden.net/2017

Practical T_EX 2018
Troy, New York
USA
June 25–27, 2018
tug.org/practicaltex2018

BachoT_EX 2018
Bachotek
Poland
April 29-May 3, 2018
gust.org.pl/bachotex

TUG 2018
Rio de Janeiro
Brazil
July 20–22, 2018
tug.org/tug2018

Saturday

08:00	<i>Breakfast</i>	
09:00	<i>Conference opening</i>	
09:15	Hans Hagen	<i>Children of T_EX</i>
10:00	Kumaran Sathasivam, S. K. Venkatesan, Yakov Chandy	<i>Revealing semantics using subtle typography and punctuation</i>
11:00	<i>Coffee break</i>	
11:30	Frank Mittelbach	<i>Through The Looking Glass — and what Alice found there ...</i>
12:30	Leila Akhmadeeva, Boris Veytsman	<i>To justify or not to justify?</i>
13:00	Przemysław Scherwentke	<i>L^AT_EX Restaurant</i>
13:30	<i>Lunch</i>	
15:00	Barbara Beeton	<i>Debugging L^AT_EX files — Illegitimi non carborundum</i>
15:30	Boris Veytsman	<i>Making the ltxsparklines package: A journey of a CTAN contributor into the world of CRAN</i>
16:00	<i>Coffee break</i>	
16:30	Grzegorz Murzynowski	<i>The GM scenarios two years later. A complete madness. But — Turing-complete or not? Or: how the spirit of l3expan made me conceive and bear a monster</i>
18:00	Jean-Michel Huppen	<i>MLBIB_TE_X now deals with Unicode</i>
18:30	Dávid Lupták	<i>Typesetting bibliographies compliant with the international standard ISO 690 in L^AT_EX</i>
19:00	<i>Dinner</i>	
21:00	<i>Bonfire</i>	

Sunday

00:00	<i>Bonfire (continued)</i>	
08:00	<i>Breakfast</i>	
09:00	Willi Egger	<i>ConT_EXt: Tutorial/workshop (for ConT_EXt beginners)</i>
11:00	<i>Coffee break</i>	
11:30	<i>Work group meetings</i>	
13:30	<i>Lunch</i>	
14:45	<i>Conference photo</i>	
15:00	Willi Egger	<i>Bookbinding workshop: Portfolio</i>
15:00	Damien Thiriet	<i>Hackaton: Documenting L^AT_EX packages</i>
16:00	<i>Coffee break</i>	
16:30	Maciej Rychły, Piotr Bolek	<i>Released sounds</i>
17:30	Maciej Rychły, Mateusz Rychły	<i>Concert: Released sounds</i>
19:00	<i>Dinner</i>	
20:00	<i>TUG & GUST Annual General Meetings</i>	
21:30	Michał Gasewicz	<i>Off topic (completely): Many faces (and types) of beer</i>

Monday

08:00	<i>Breakfast</i>	
09:00	Ulrik Vieth	<i>10 years of OpenType math font development</i>
10:00	Jerzy Ludwichowski	<i>GUST's e-foundry current font projects</i>
10:30	Sigitas Tolušis, Arūnas Povilaitis, Valentinas Kriaučiukas	<i>Xdvi_{psk}: dvips ready for OpenType fonts and more image types</i>
11:00	<i>Coffee break</i>	
11:30	Adam Twardoch	<i>Variable and color OpenType fonts: Chances and challenges</i>
12:30	Hans Hagen, Taco Hoekwater	<i>Colorful fonts, an update and peek into the future</i>
13:30	<i>Lunch</i>	
15:00	Hans Hagen	<i>Variable fonts</i>
15:45	Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski	<i>Parametric math symbol font</i>
16:15	<i>Coffee break</i>	
17:00	Adam Twardoch	<i>STIX, Fira, Noto and friends: Beautiful new opensource fonts</i>
17:45	Mojca Miklavec, Arthur Reutenauer	<i>One rule to break them all</i>
18:30	Mojca Miklavec	<i>Automating binary building for T_EX Live</i>
20:00	<i>Conference dinner</i>	
21:30	Katarzyna Jackowska	<i>Concert: "Kapela Hałasów" and "Kapela Jazgódki"</i>

Tuesday

08:00	<i>Breakfast</i>	
09:00	Siep Kroonenberg	<i>The T_EX Live Launcher</i>
09:30	Norbert Preining	<i>fntutil and updmap — past and future changes (or: cleaning up the mess)</i>
10:00	Luigi Scarso	<i>MFLua 0.8</i>
10:30	<i>Coffee break</i>	
11:00	Takuto Asakura	<i>Implementing bioinformatics algorithms in T_EX — the Gotoh package, a case study</i>
11:15	Lolita Tolén	<i>T_EX users habits versus publishers requirements</i>
12:00	Petr Sojka, Vít Novotný	<i>T_EX in Schools? Just Say Yes: The use case of T_EX usage at the Faculty of Informatics, Masaryk University</i>
12:45	Vít Novotný	<i>Using Markdown inside T_EX documents</i>
13:30	<i>Lunch</i>	
15:00	Jean-Michel Hufflen	<i>History of accidentals in music</i>
15:30	Janusz M. Nowacki	<i>Calligraphy by Barbara Galińska</i>
15:45	Andrzej Tomaszewski	<i>An example of a humanist scholarly book</i>
16:15	<i>Coffee break</i>	
16:45	Adam Twardoch	<i>CORDIDA! Collaborative Opensource Rapid Digital Internet Documentation Authoring</i>
17:15	Paulo de Ney Souza	<i>T_EX Annoyances — what is on the way to a full production environment</i>
18:00	Paulo de Ney Souza	<i>T_EX Production — ePub, the new target</i>
18:50	Zunbeltz Izaola, Paulo de Ney Souza	<i>DocVar: Manage document variables</i>
19:10	<i>Dinner</i>	

Wednesday

08:00	<i>Breakfast</i>	
09:00	Anna Beata Kwiatkowska, Jerzy Ludwichowski	<i>T_EX in secondary schools — an idea to be taken up by GUST</i>
10:00	Marcin Woliński	<i>bredzenie.sty — Polish language version of lipsum.sty</i>
10:20	Marcin Borkowski	<i>What a T_EXnician can learn from ten years' editorial work</i>
11:00	<i>Conference closing</i>	
11:15	<i>Coffee break</i>	
13:00	<i>Lunch</i>	

TUG@BachTeX 2017*

Jean-Michel HUFFLEN

It is always with great pleasure that I go to the annual conference of the Polish-speaking group GUST.¹ We are in the countryside, bordered by the Bachotek² lake, near to Brodnica. The small houses that accommodate us look like chalets in the Alps. The only drawback is that going there by train from Warsaw is rather lengthy, but the journey through the Polish countryside is scenic. BachTeX conferences are very open and welcome many foreign visitors. If a presentation is given in Polish, either slides include an English version, or you will easily find someone who will translate it by word of mouth. BachTeX has already welcomed EuroTeX conferences several times³ and this year the TUG conference took place there. So there were more foreign visitors — especially from the USA — than usual.

Following the conference theme of *Premises, predictions, predictions*, there was a rich variety of talks.⁴ *Children of TeX* was undoubtedly the talk most related to the theme: Hans Hagen gave us a personal vision of TeX's evolution and future. How to solve tenacious errors and other vexations while processing L^ATeX files was brilliantly addressed by Barbara Beeton. Using L^ATeX within publishing activities was the subject of the talks of Paulo Ney de Souza, Marcin Borkowski and Lolita Tolené. In particular, some pearls of end-users were displayed during this last talk. The introduction of TeX into academic curricula was addressed by Petr Sokja and Vít Novotný as part of experiments conducted at the Masaryk University.⁵ In another talk, Jerzy Ludwichowski announced a project that aims to provide resources to students of a grammar school. Last, a very positive review of an introductory book for L^ATeX in Polish was given by Przemysław Scherwentke.

The installation procedures of TeX Live's programs are improved continually; this is what we could learn from the talks of Siep Kroonenberg, Mojca Miklavec and Norbert Preining. Various pack-

ages were shown: for managing dynamic documents, by Boris Veytsman; for structuring documents, by Vít Novotný; for managing documents' metadata by means of variables, by Paulo Ney de Souza, for bioinformatics' algorithms, by Takuto Asakura;⁶ for generating Polish texts for testing purposes, by Marcin Woliński; and for typesetting bibliographies compliant with the ISO 690, by Dávid Lupták. Last, Grzegorz Murzynowski gave a very thorough and very didactic description of some choices related to macro expansion within the L^ATeX 3 project.

The tools that revolve around (L^A)TeX were not put away. I personally demonstrated the new version of my MLBIBTeX⁷ bibliography processor, which can now take advantage of Unicode's features. Luigi Scarso showed new functions of his MFLua program: let us recall that it allows functions written using the Lua programming language to be called within METAFONT programs.

Many talks dealt with fonts, especially OpenType ones. This topic encompasses *variable fonts*, shown by Hans Hagen, as well as two talks by Adam Twardoch. We also include the extension of the x_dvi program to process such fonts, demonstrated by Valentinas Kriaučiukas, and the very good synthesis of the progress concerning mathematical OpenType fonts, by Ulrik Vieth. Bogusław Jackowski spoke about parameterisation of mathematical fonts, whereas Hans Hagen and Taco Hoekwater brought a little relaxation with a demonstration of a font for child play. Talking of the future, the current projects of the GUST e-foundry were presented by the GUST president, Jerzy Ludwichowski.

Let us now go to typography. I personally greatly enjoyed Yakov Chandy's talk about subtleties of punctuation and disambiguation allowed by it. Frank Mittelbach's quest for an algorithm finding optimal pagination gave us a great performance. Experiments conducted by Boris Veytsman and Leila Akhmadeeva about text justification, especially within narrow columns, deserve attention. Mojca Miklavec and Arthur Reutenauer shared with us their experience about managing hyphenation patterns for TeX. Andrzej Tomaszewski showed interesting typographic effects within a poetry book by Ovid. Concerning my talk about the history of accidentals in music — flat, sharp, natural signs, . . . — and associated typographic rules, I link it to this rubric.

This TUG@BachTeX conference included *workshops*, too. A tutorial for ConTeXt beginners was

* This text is a translation (by the author) of a report commissioned by the French-speaking group GUTenberg. The original French version will appear in *La Lettre GUTenberg*.

¹ *Grupa Użytkowników Systemu TeX*, that is, *group of TeX system users*. The display of this group name at its Web site (<http://www.gust.org.pl>) reveals the boxes used by TeX to handle non-space characters' dimensions and kerning applied to TeX's logo.

² So this BachTeX conference's name is a play on words.

³ In 2002, 2007, 2011 and 2013.

⁴ In addition to the articles and abstracts in this *TUGboat* issue, you can get the slides of most talks at the conference's Web site: <http://www.gust.org.pl/bachotex/2017-en>.

⁵ Located in Brno, Czech Republic.

⁶ For several years, a prize for the best talk has been awarded at the end of a BachTeX conference, on the basis of a vote of all participants. This year, Asakura won this prize.

⁷ *MultiLingual BibTeX*.

organised by Willi Egger; a second workshop, more specialised, was connected to packages' documentation by Damien Thiriet. Farther from $\text{T}_{\text{E}}\text{X}$ & Co., closer to manual work, Willi Egger's second workshop prompted participants to make an artist's *portfolio*. Finally, a fourth workshop, hosted by Michał Gasewicz, was labelled *off topic* from the start: it was devoted to beer tasting!

I cannot forget the relaxation moments that punctuated this conference. Discussions at coffee breaks were very rich. Unfortunately, the traditional bonfire of Bacho $\text{T}_{\text{E}}\text{X}$ conferences occurred during a rainy evening, so most participants took refuge under a shelter, except for those grilling sausages. However we sang *Frère Jacques* (*Brother John*) in many languages, as well as traditional Polish songs, accompanied on the guitar and accordion. Going on with musical attractions, we attended a talk by Piotr Bolek and Maciej Rychły about relationship between painting and music, followed by a concert with flutes and guitars. The second concert was supervised by Katarzyna Jackowska: four musicians — singing, playing the violin, the accordion, the clarinet — involved us in dances at the end of the conference's dinner. To be exact, I should mention 'at the end of the first part', because after a menu composed of traditional Polish meals and ending with some imposing pastry showing the conference's name, we again saw the arrival of salted meats, and it set out again for another turn.

To sum up, this was a vintage year for Bacho $\text{T}_{\text{E}}\text{X}$ and TUG. Let me close with thanks to the French-speaking group GUTenberg once again, for their support making my participation possible.

◇ Jean-Michel HUFFLEN
FEMTO-ST (UMR CNRS 6174)
& University of Bourgogne
Franche-Comté
jmhuffle (at) femto-st dot fr
[http://members.femto-st.fr/
jean-michel-hufflen](http://members.femto-st.fr/jean-michel-hufflen)

The accompanying photos were taken by Gyöngyi Bujdosó (first two) and Andrzej Odynec (second two). Thanks to them for their permission to include their pictures in this article.



Calligraphy by Barbara Galińska

Janusz Nowacki

Again we had the opportunity to meet calligraphy in Bachotek—that is, creating letters by hand. The creator of the slides shown is Barbara Galińska, a calligrapher and typographer from Warsaw. All her works are of course handmade, with no computer employed.

Galińska is an architect by training, and works as a book illustrator, graphic artist, designer and creator of animated movies. Photography used to be her great passion; now it is calligraphy.

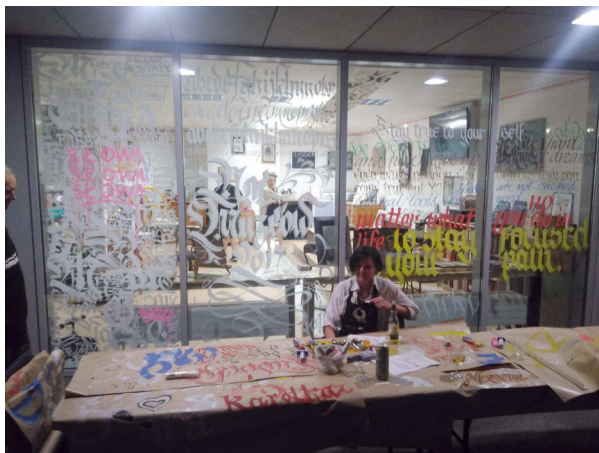
Her work encompasses all types of script—from uncial, to many kinds of blackletter, to copperplate, to modern calligraphy.

She writes, draws, and paints on everything everywhere—invitations, diplomas, love letters, labels, advertisements, and book covers. Her works are on paper, window panes, walls, furniture, people, and even in sand.

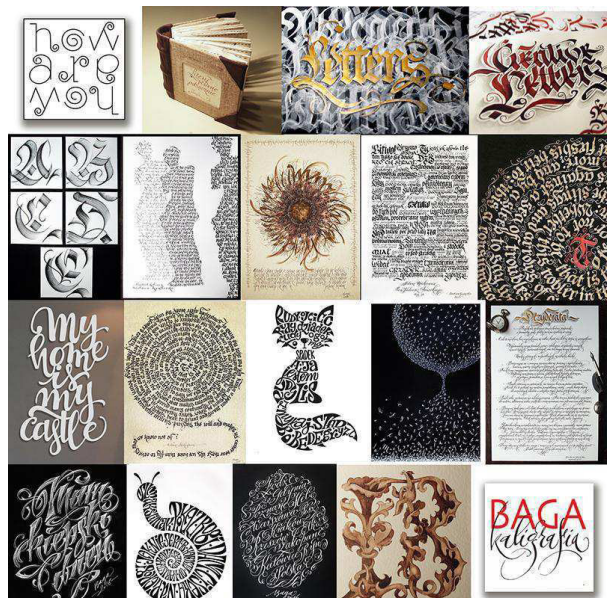
She leads workshops and demonstrations of various forms of calligraphy. She is most interested in the variety of ways letters can be written, drawn, or painted, and she uses a great variety of tools.

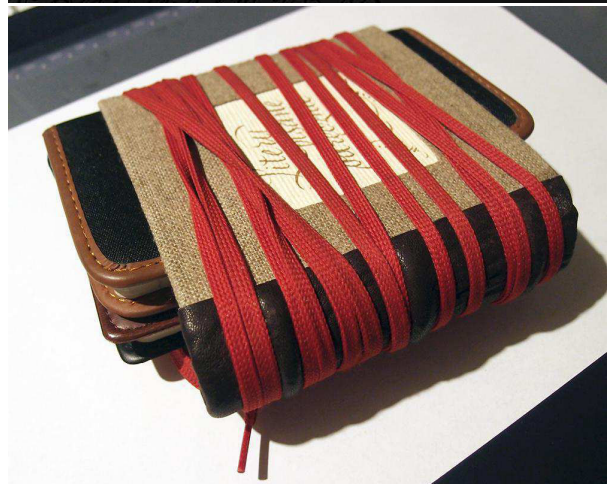
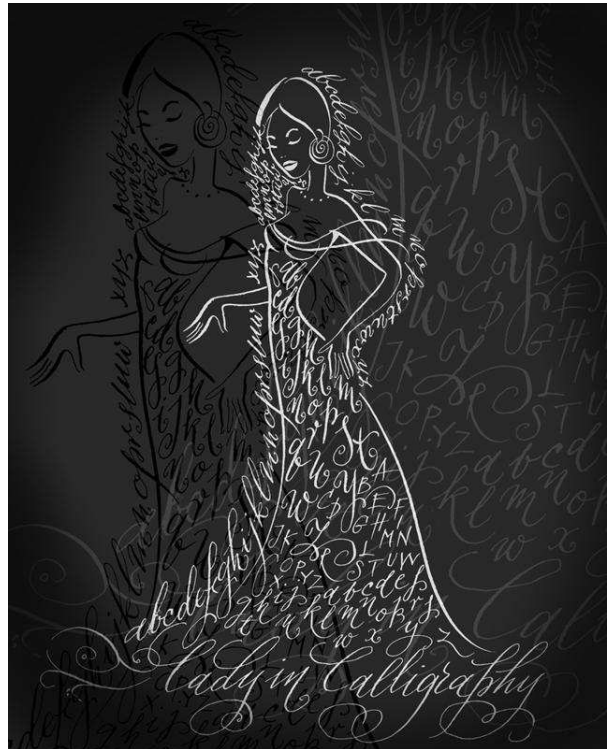
These works were presented in April and May 2017 at her solo exhibit in Galeria Pięknych Książek in Warsaw. Several hundred works by her can be seen at <https://www.behance.net/barbara-galinska>. A few selections are shown here.

◇ Janusz Nowacki
janusz (at) jmn dot pl



Barbara Galińska in front of the windows to Galeria Pięknych Książek on the Night of the Museums, Warsaw, May 20–21, 2017.





Released sounds (Uwolnione dźwięki)

Maciej Rychły

Abstract

People see the world and become painters. Or they hear the world, and become musicians. Painters are fortunate, or unfortunate, in that once painted, their work will endure. The achievements of their predecessors become “historical painting”, while present-day artists seek new solutions in their own work. Musicians are unfortunate, or fortunate, in that their work, the sound, evaporates after each live performance, regardless of when it was composed, and has to be reanimated anew.

Streszczenie

Ludzie widzą świat. Dlatego mogą być malarzami. Ludzie słyszą świat. Dlatego mogą być muzykami. Malarze są w tej szczęśliwej/nieszczęśliwej sytuacji, że obraz kiedyś namalowany trwa. Dzieła poprzedników stają się „malarstwem historycznym”, a współcześni artyści szukają nowych rozwiązań dla własnej twórczości. Muzycy są w tej szczęśliwej/nieszczęśliwej sytuacji że, niezależnie od czasu powstania, dzieło muzyczne po każdym wykonaniu „rozpada się”. Muzyka musi być ciągle przywoływana na nowo.

Pomysł „odkodowania” zapisów nutowych z wybranych obrazów malarstwa polskiego i europejskiego oraz przygotowania koncertu zapisanej przez artystów malarzy muzyki, powstawał kilka lat. Studia nad rozczytaniem zapisów nutowych z obrazów rozpocząłem badając dawne obrazy ze scenami muzycznymi, na których grupa muzyków skupiona jest wokół niewidzialnego zjawiska muzyki. Może wydać się absurdalne to, że artystów tworzących obrazy – znaki wizualne, fascynowała taka sytuacja. Sceny muzyczne są bowiem bardzo częstym malarskim tematem.

Od czasu renesansu artyści byli wszechstronnie wykształceni. Tworząc obraz traktowany jako dzieło, zawierali w nim swoją wiedzę na temat kultury. Postacie na obrazach umieszczane są w kontekście przedmiotów, które określają status i profesję portretowanych. Muzycy umieszczani są wśród instrumentów, często trzymają kartki z zapisami utworów muzycznych. Niektórzy artyści malując swój

autoportret chętnie przedstawiali siebie wśród instrumentów muzycznych. Czy znali również sztukę dźwięku? Czy zapisy nutowe, które umieszczali na swoich obrazach noszą ślad muzycznych kompetencji? Jeżeli tak, to obraz taki można traktować jako „multimedialną pigułkę” która, jeśli poświęcimy jej swój czas, zabrzmi muzyką, sprowokuje opowieść, pozwoli wejść nam w utrwaloną przez malarza chwilę. Czasami kartka z nutami skierowana jest wprost do nas, tak jakby artysta czynił to z wiarą że ktoś, na przykład my dzisiaj, odczyta ten zapis. Warto też pamiętać że obrazy, nawet te najbardziej realistycznie przedstawiające rzeczywistość, nie są fotografiami. Żaden z elementów obrazu nie pojawił się przypadkiem. Za kreacją stoi intencja twórcy. Istnieje możliwość, że zapisana na obrazach muzyka przetrwała do nas tylko w tej formie. Obraz można więc traktować jako rękopis. „Uwolnione Dźwięki” to pomysł na jedną z wielu rozmów z przeszłością, czytanie jej znaków.



The idea of decoding musical notation from selected pieces of Polish and European painting, and of organising a concert of the music inscribed by painters in their work, has been taking shape over several years. I began my forays into reading the musical notation in paintings by studying old works with music-related scenes. It might seem odd for artists working in the visual medium of painting to

be fascinated by the invisible medium of music, but scenes with a musical theme are a frequent painterly motif.

Since the Renaissance, artists have often been people of wide knowledge. A painting they created and considered a work of art would comprise and convey much of what they knew about culture. The figures in their paintings are portrayed in the context

of objects which determine their status and profession. Even in their self-portraits some artists readily included instruments. Were they conversant with the art of sound? Does the musical notation found in their paintings bear the traces of musical competence? If so, then the painting may be treated as a “multimedia pill” which, given our time and dedication, will resound with music, provoke a tale, allow us to slip into the moment captured by the painter. Sometimes the note-filled sheet is turned towards us, as if the artist had done this in the hope that somebody will read the notation.

It should also be remembered that even the ones which present reality most realistically are not in fact photographs; none of the elements in a painting found its way there by accident. Behind the creation, there is the intention of the artist. It is possible that the music recorded in the painting survived only in that singular form. The picture may therefore be approached as a manuscript. Released Sounds are just one of the many possible dialogues with the past and readings of its signs.



Wybrane opowieści/Selected tales



„Letter Rack” Everta Colliera

Na jednym ze swoich obrazów Evert Collier namalował w roku 1693 w Londynie wszystkie najpotrzebniejsze przedmioty codziennego użytku – nożyczki, gazetę, grzebień, pióro. Najwyraźniej zafascynowało go „wieszadło” – „letter rack”. Wśród rzeczy najpotrzebniejszych znalazł się też flet i zeszyt, w którym zapisane są menuety. Być może te menuety przetrwały do naszych czasów wyłącznie dzięki Collierowi.

“Letter Rack” by Evert Collier

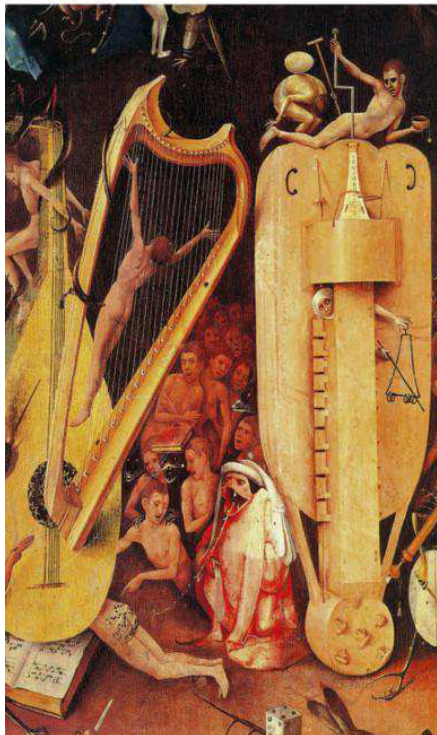
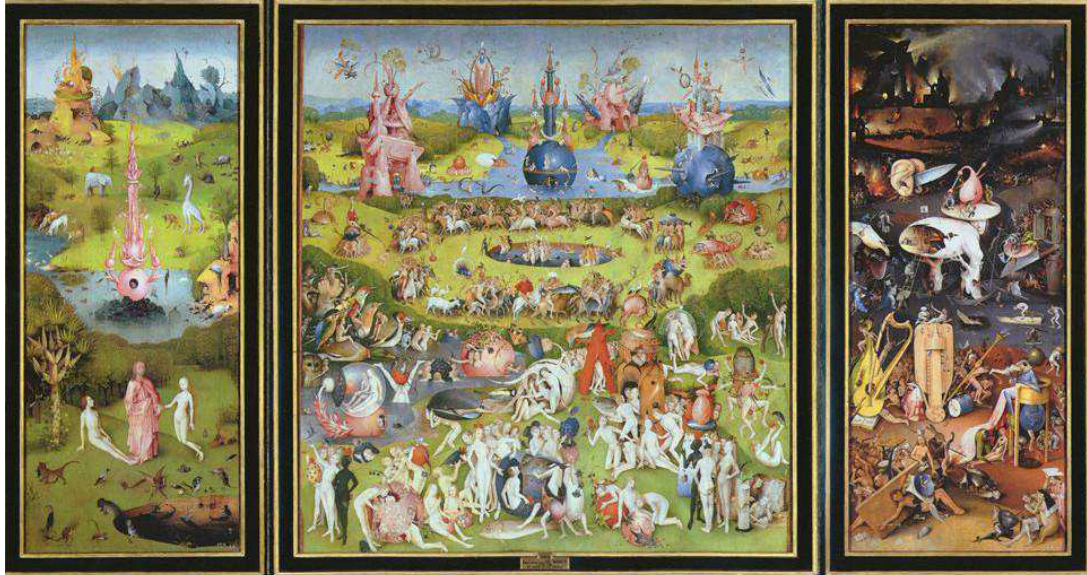
In a 1693 painting, Londoner Evert Collier depicted the mundane yet necessary objects of everyday use — a pair of scissors, a newspaper, two combs, a feather. He was evidently fascinated by the letter rack, on which various items could be hung. Among the utilitarian paraphernalia, there is also a flute and a book of minuets. Perhaps it is only thanks to Collier that the minuets have survived until today.



„Piekło muzykantów” Hieronima Boscha

Spójrzmy na sekwencję zapisaną w księdze umieszczonej przez Hieronima Boscha w „Piekle Muzykantów”. Księga znajduje się w lewym skrzydle tryptyku „Tysiącletnie Królestwo”. Widać tam nagich ludzi przygniecionych nieludzkiej wielkości muzycznymi instrumentami. Zapis ujawnia melodię nasyconą trytonami. Tryton to współbrzmienie, na któ-

rym nie można zbudować nic sensownego. W Średniowieczu nazywano go diabelskim krokiem. Czy Bosch celowo umieścił trytony w zapisanej przez siebie melodii? Czy jest to melodia szczególnie „niepoprawna”, „zepsuta”? Czy Bosch wierzył, że ktoś kiedyś odczyta ten zapis i głębiej odczuje piekło muzycznej niemocy?



“Musicians’ Hell” by Hieronymus Bosch

Let’s take a look at the sequence written in the book which Hieronymus Bosch placed in the “Musicians’ Hell”. This book is found in the right-hand panel of *The Garden of Earthly Delights*. In this area of the panel naked people are crushed by instruments of extraordinary, inhuman dimensions. The notation reveals a melody full of tritones. A tritone represents a futile consonance: one which cannot serve to

construct anything sensible. Since the Middle Ages, the tritone has been called ‘the devil’s interval’, ‘the chord of evil’ or ‘diabolus in musica’. Did Bosch deliberately put those tritones in his melody? Is this melody particularly “indecent” or “corrupt”? Did Bosch believe that one day someone would read the notation and have a more profound experience of the infernal torment of musical impotence?



Pieśń nad pieśniami Caravaggia

Na obrazie Caravaggia „Odpoczynek w czasie ucieczki do Egiptu” anioł gra na skrzypcach melodię z nut, które trzyma Józef. Nuty widać tak wyraźnie, że można ustalić wydawcę i odczytać melodię napisaną przez niderlandzkiego kompozytora Noela Bauldewejna do słów „Pieśni nad pieśniami”.

Caravaggio’s Song of Songs

In Caravaggio’s *Rest on the Flight to Egypt*, an angel plays a melody on violin, reading the music from the sheets held by Joseph. The note-filled staves are depicted so clearly that it is possible to identify the publisher and read the melody written by a Dutch composer, Noel Bauldeweyn, to the verses of *The Song of Songs*.



J.-B. Oudry 1734-1613 Paris

138

Bukoliki Jeana Baptisty Oudry’ego

Istnieje wiele obrazów, na których tematem są instrumenty muzyczne i porzucane kartki z nutami. Wybrałem obraz z muzyką burdonową zapisaną w 1730 roku przez Baptiste Oudry’ego. Arkadia. Bukoliki. Sielskość.

The Pastorals of Jean-Baptiste Oudry

There are numerous other paintings in which musical instruments and scattered sheets of music are a prominent theme. I have chosen a 1730 still-life by Jean-Baptiste Oudry featuring a musette bagpipe and drone music. *Arcadia. Pastorals. Idyll*.



Salon Księżnej Czartoryskiej

W podlubelskim pałacu w Kozłówce znajduje się olejny obraz będący kopią sztychu, który w 1777 roku wykonał Giuseppe Marchi. Izabela

Czartoryska siedzi przy fortepianie. Właśnie przestała grać. Nuty na pulpicie widać wyraźnie. Jest to zapis polskiego tańca – poloneza. Podobną muzykę możemy odczytać z obrazu nieznanego francuskiego malarza. Obraz znajduje się w poznańskim Muzeum Instrumentów Muzycznych. Grupa ubranych w orientalne stroje muzyków i reszta zgromadzonych wsłuchuje się w słodkie kaskady tercji, w muzykę salonu. Słuchacze i muzycy w teatralnej pozie, w teatralnych szatach, igrają z historią.

The salon of Duchess Czartoryska

In the palace of Kozłówka near Lublin, there is an oil painting, a copy of an etching made in 1777 by Giuseppe Marchi. Izabela Czartoryska is sitting at the piano. She has just stopped playing. The sheets of music on the stand are clearly legible: it is a Polish dance — the polonaise. A similar piece may be read from the painting by an unknown French artist, found at the Museum of Musical Instruments in Poznań. A group of musicians, dressed after an oriental fashion, as well as the rest of the people gathered there, listen enraptured to the sweet cascades of the third, to the music of the salon. In theatrical poses and theatrical guises, the listeners and the musicians play with history.



„Pieśń miłosna” Antoin’a Watteau

Obrazy Watteau, malarza łamiącego akademickie kanony, często przedstawiają muzyków i ludzi teatru. Magnetyzują widza nastrojową aurą. Na obrazie „The love song” gitarzysta odkrywa nowe harmonie w bezpiecznej scenarii ogrodu. Dziewczyna trzyma nuty. Płynie pieśń miłosna.

“Love Song” by Antoine Watteau

The paintings by Watteau, an artist who went against the academic canons, often feature musicians and people of the theatre. They mesmerise the beholder with an atmospheric aura. In The Love Song the guitar player discovers new harmonies in the safe haven of a garden. The girl is holding the music book. A love song floats through the air.





„Enchiridon” Holbaina

„Ambasadorzy”. U stóp dwóch młodzieńców widać rozmytą tajemniczą plamę. Gdy popatrzymy na obraz „z ukosa”, to w szalonej perspektywie wielkiego skrótu rozmyta plama staje się ludzką czaszką. Czy tym akcentem malarz przypomina nam o przemijaniu? Gdy patrzymy na obraz „na wprost”, nad plamą/czaszką widać księgę z nutami, lutnię, flet, cyrkiel, globus – model Ziemi. Namalowani młodzieńcy mają pewność, że opanowali Ziemię, że czas „omija ich bokiem”, że stali się nieśmiertelni, że posiadają przyrządy do panowania nad czasem i przestrzenią. Księga leżąca pod lutnią to Enchiridon Martina Lutera z 1524 roku. Powstała dziewięć lat wcześniej niż obraz Holbaina. Jest to księga pieśni służących nowej liturgii, w której niezrozumiałe już łacińskie teksty Biblii przełożono na język współczesny. Melodie pieśni zostały zaczerpnięte z muzyki ludowej. Jo-

han Walter, muzyk współpracujący z Lutrem, dopisał do wybranych melodii trzy harmonizujące głosy. Czy dawna niemiecka muzyka ludowa przetrwała do naszych czasów złączona ze świętymi tekstami nowej liturgii? Melodia którą można odczytać na obrazie Holbaina, to jeden z głosów dopisany przez Waltera.

“Enchiridion” by Holbein

The Ambassadors. There is a mysterious, vague shape at the feet of the two young men. When we look at the picture obliquely, the bizarre perspective of great foreshortening reveals a human skull. Does the painter use this device to remind us of transience and impermanence? Looking straight, a number of items can be seen above the shape/skull: a book of music, a lute, a flute, a compass and a globe — the model of Earth. The young men in the painting are certain that they have the ascendancy, that time “passes them by”, that they have become immortal and have the instruments to dominate time and space. The volume lying under the lute is Martin Luther’s Enchiridion of 1524. A book nine years younger than Holbein’s painting. It is a book of songs for the new liturgy, in which the already incomprehensible Latin texts of the Bible have been translated into a contemporary language. The melodies of the songs originate from folk music. Johann Walter, a musician collaborating with Luther, added three harmonised voices to the melodies. Has the old German folk music survived until today, woven into the holy verses of new liturgy? The melody that can be read from Holbein’s painting is one of the voices composed by Walter. We choose to play the basic theme, which undoubtedly possesses the charm and grace of a folk song.





Taniecny krąg barokowej śmierci

Nieznany barokowy malarz namalował w Krakowie moralitet. Obraz można zobaczyć w kościele bernardynów. Spójrzmy na muzykanta, który siedzi przy keyboardzie z dawnych lat. Śmierć podsuwa mu nuty. Ludzie różnych stanów złączeni w kręgu tańczą tak, jak tego chce kostucha. Zapis muzyczny ukazuje konstelację rozbitych kropek o molowym trybie. Muzyka zmierza ku ciszy.

The dancing circle of baroque death

In Kraków, an unknown Baroque painter depicted a morality scene. The painting can be found in the church of St. Bernardine. Take a look at the musician who sits at the keyboard of the olden days. Death passes him the music. People of various estates, joined in a circle, dance to the whim of the Grim Reaper. The notation shows a constellation of broken and crumbling dots in a minor key. The music descends towards silence.



◇ Maciej Rychły
Poznań
rychly (at) poczta dot onet dot pl

The state of T_EX

Boris Veytsman

This article is based on my address at the Annual General Meeting at TUG 2017. I am grateful to the TUG Board & AGM participants for the discussion, and, of course, to GUST for the hosting of BachoT_EX!

When we talk about the state of T_EX, three things come to mind: T_EX software, T_EX community and the T_EX Users Group as part of this community.

While the original Knuthian `tex` is frozen, **T_EX software** as a whole is not. Nor should it be. The world around us changes, so our software must change if we want to stay relevant. There have been several important shifts in digital typesetting, which presented challenges for us, such as the switch to the PDF formats, and the advent of first 8-bit encodings and then Unicode. In response T_EX software changed: modern engines and macro packages natively work with PDF, Unicode, TTF and OTF fonts.

We have now new challenges, which need to be addressed. There are new PDF features, which are not adequately addressed by the free viewers. Another important problem is the emergence of PDF archiving and accessibility standards. Governments increasingly demand standards compliance from their contractors. If T_EX does not produce compliant output “out of the box”, it will become not relevant first for government-related publications, then for big publishers, and then for the rest of the world. To ensure compliance we need both volunteer development effort *and* some funding. I think we can reach publishers and other stakeholders about this. I urge those interested in this issue to join the TUG Accessibility & Standards Working Group (<https://tug.org/twg/accessibility>).

Speaking of **T_EX community**, I must say that at almost any T_EX conference there are concerns about the decreased popularity of T_EX. I do not necessarily share this feeling. As a T_EX consultant I see among my clients, besides the usual bunch of publishers, professionals and students of mathematics & “hard” sciences, also rather unexpected users: lawyers, physicians, biologists, philosophers, and many others. Still for the future of T_EX, we need to ensure that new generations are exposed to it. I do not say that *everybody* should use T_EX. Rather I think those who might like it must be given the opportunity to learn and use it. We should show the

possibilities of T_EX and friends to the students of schools and universities. We should encourage teachers to share their experience, lesson plans and ideas. Perhaps we can help to organize competitions among students and teachers, olympiads, T_EX camps and courses. Again I would like to use this opportunity to remind about the TUG Education Working Group (<https://tug.org/twg/edutex>) and urge those interested in T_EX in schools and universities to join.

The efforts described above require coordination, organization, and, let us be practical, money. This leads us to the state of the **T_EX Users Group**. In the old days TUG membership was almost the only way to get those T_EX tapes or the answers to one’s T_EXnical questions. Now with the advent of the Internet, easy access to distributions and online forums, the situation has changed. Thus it is not unexpected that the membership in TUG has considerably dropped. Is TUG itself still relevant or needed?

I think it is. We need a center to coordinate development, advocacy and fundraising efforts. TUG seems to be a natural entity for this—especially being a tax exempt (in the United States) organization. We already serve several T_EX-related projects by collecting money for them, and we can increase this activity.

However, I think we need to reconsider the role of TUG membership for TUG. While we always will serve our members, we need to realize that we also serve the T_EX community in general. We should accept that many T_EX users do not want the commitment of full membership. We should tell these users that any donation of time or money is welcome too. We also need to do fundraising among organizational T_EX users: publishers, universities, etc. In other words, we might want to consider behaving as a charity organization serving the wide community of T_EX users, including, but not limited to, our core membership.

In conclusion, there are many interesting opportunities and challenges for us as T_EX developers, T_EX users and TUG members. I hope we all will rise to meet them.

◇ Boris Veytsman
 President, T_EX Users Group
 borisv (at) lk (dot) net
<http://borisv.lk.net>

Children of T_EX

Hans Hagen

1 The theme

Nearly always T_EX conferences carry a theme. As there have been many conferences the organizers have run out of themes involving fonts, macros and typesetting and are now cooking up more fuzzy ones. Take the BachoTUG 2017 theme:

Premises The starting point, what we have, what do we use, what has been achieved?

Predilections How do we act now, how do we want to act, what is important to us, what do we miss?

Predictions What is the future of T_EX, what we'd like to achieve and can we influence it?

My first impression with these three P words was: what do they mean? Followed by the thought: this is no longer a place to take kids to. But the Internet gives access to the Cambridge Dictionary, so instead of running to the dusty meter of dictionaries somewhere else in my place, I made sure that I googled the most recent definitions:

premise an idea or theory on which a statement or action is based

predilection if someone has a predilection for something, they like it a lot

prediction a statement about what you think will happen in the future

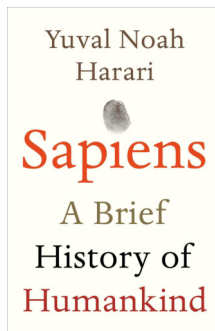
I won't try to relate these two sets of definitions but several words stand out in the second set: idea, theory, action, like, statement and future. Now, as a preparation for the usual sobering thoughts that Jerzy, Volker and I have when we staring into a BachoT_EX campfire I decided to wrap up some ideas around these themes and words. The books that I will mention are just a selection of what you can find distributed around my place. This is not some systematic research but just the result of a few weeks making a couple of notes while pondering about this conference.

2 Introduction

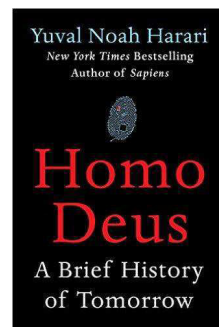
One cannot write the amount of T_EX macros that I've written without also liking books. If you look at my bookshelves the topics are somewhat spread over the possible spectrum of topics: history, biology, astronomy, paleontology, general science but surprisingly little math. There are a bunch of typography-related books but only some have been read: it's the visuals that matter most and as there are no real developments I haven't bought new ones in over a decade, although I do buy books that look nice for our office

display but the content should be interesting too. Of course I do have a couple of books about computer (related) science and technology but only a few are worth a second look. Sometimes I bought computer books expecting to use them (in some project) but I must admit that most have not been read and many will soon end up in the paper bin (some already went that way). I'll make an exception for Knuth, Wirth and a few other fundamental ones that I (want to) read. And, I need to catch up on deep learning, so that might need a book.

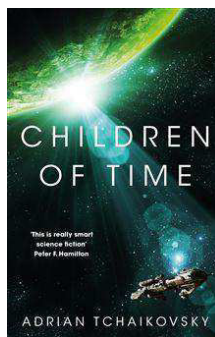
My colleagues and I have many discussions, especially about what we read, and after a few decades one starts seeing patterns. Therefore the last few years it was a pleasant surprise for me to run into books and lectures that nicely summarize what one has noticed and discussed in a consistent way. My memory is not that good, but good enough to let some bells ring.



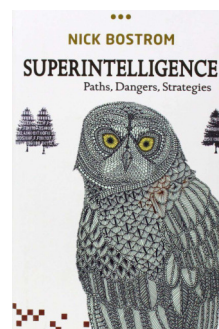
history



futurology



science fiction



informatics

The first book that gave me this “finally a perfect summary of historic developments” feeling is “Sapiens” by Yuval Noah Harari. The author summarizes human history from a broad perspective where modern views on psychology, anthropology and technical developments are integrated. It's a follow up on a history writing trend started by Jared Diamond. The follow up “Homo Deus” looks ahead and is just as well written. It also integrates ideas from other fields, for instance those related to development of artificial intelligence (Dennett, Bostrom, etc.).

Another inspiration for this talk and article is the 50 hour lecture series on behavioral biology by Robert Sapolsky of Stanford University, brought to my attention by my nephew Bram who visited a few \TeX conferences with me and who is now also forced to use \TeX for assignments and reports. (How come self-published books used at universities often look so bad?)

The title of this talk is inspired by the book “Children of Time” by Adrian Tchaikovsky that I read recently. There are science fiction writers who focus on long term science and technology, such as some of Alastair Reynolds, while others follow up on recent development in all kind of sciences. One can recognize aspects of “Superintelligence” by Bostrom in Neal Asher’s books, insights in psychology in the older Greg Bear books, while in the mentioned “Children of Time” (socio)biological insights dominate. The main thread in that book is the development of intelligence, social behaviour, language, script and cooperation in a species quite different from us: spiders. It definitely avoids the anthropocentric focus that we normally have.

So how does this relate to the themes of the Bacho \TeX conference? I will pick out some ways to approach them using ideas from the kind of resources mentioned above. I could probably go on and on for pages because once you start relating what you read and hear to this \TeX ecosystem and community, there is no end. So, consider this a snapshot, that somehow relates to the themes:

premise Let’s look at what the live sciences have to say about \TeX and friends and let’s hope that I don’t offend the reader and the field.

predilection Let’s figure out what brings us here to this place deeply hidden in the woods, a secret gathering of the \TeX sect.

prediction Let’s see if the brains present here can predict the future because after all, according to Dennett, that is what brains are for.

At school I was already intrigued by patterns in history: a cyclic, spiral and sinusoid social evolution instead of a pure linear sequence of events. It became my first typeset-by-typewriter document: Is history an exact science? Next I will use and abuse patterns and ideas to describe the \TeX world, not wearing a layman’s mathematical glasses, but more from the perspective of live sciences, where chaos dominates.

3 The larger picture

History of mankind can be roughly summarized as follows. For a really long time we were hunters but at some point (10K years ago) became farmers. As

a result we could live in larger groups and still feed them. The growing complexity of society triggered rules and religion as instruments for stability and organization (I use the term religion in its broadest sense here). For quite a while cultures came and went, and climate changes are among the reasons.

After the industrial revolution new religions were invented (social, economic and national liberalism) and we’re now getting dataism (search for Harari on youtube for a better summary). Some pretty great minds seem to agree that we’re heading to a time when humans as we are will be outdated. Massive automation, interaction between the self and computer driven ecosystems, lack of jobs and purpose, messing around with our genome. Some countries and cultures still have to catch up on the industrial revolution, if they manage at all, and maybe we ourselves will be just as behind reality soon. Just ask yourself: did you manage to catch up? Is \TeX a stone age tool or a revolutionary turning point?

A few decades ago a trip to Bacho \TeX took more than a day. Now you drive there in just over half a day. There was a time that it took weeks: preparation, changing horses, avoiding bad roads. Not only your own man-hours were involved. It became easier later (my first trip took only 24 hours) and recently it turned into a piece of cake: you don’t pick up maps but start your device; you don’t need a travel agent but use the Internet; there are no border patrols, you can just drive on. (Okay, maybe some day soon border patrols at the Polish border show up again, just like road tax police in Germany, but that might be a temporary glitch.)

Life gets easier and jobs get lost. Taxi and truck drivers, travel agents, and cashiers become as obsolete as agricultural workers before. Next in line are doctors, lawyers, typesetters, printers, and all those who think they’re safe. Well, how many people were needed 400 years ago to produce the proceedings of a conference like this in a few days’ time span? Why read the introduction of a book or a review when you can just listen to the author’s summary on the web? How many conferences still make proceedings (or go for videos instead), will we actually need editors and typesetters in the future? How much easier has it become to design a font, including variants? What stories can designers tell in the future when programs do the lot? The narrower your speciality is, the worse are your changes; hopefully the people present at this conference operate on a broader spectrum. It’s a snapshot. I will show some book covers as reference but am aware that years ago or ahead the selection could have been different.

4 Words

Words (whatever they represent) found a perfect spot to survive: our minds. Then they made it from speech (and imagination) into writing: carved in stone, wood, lead. At some point they managed to travel over wires but no matter what happened, they are still around. Typesetting as visualization is also still surrounding us so that might give us a starting point for ensuring a future for \TeX to work on, because \TeX is all about words. There is a lot we don't see; imagine if our eyes had microscopic qualities. What if we could hear beyond 20KHz. Imagine we could see infrared. How is that with words. What tools, similar in impact as \TeX , can evolve once we figure that out. What if we get access to the areas of our brain that hold information? We went from print to screen and \TeX could cope with that. Can it cope with what comes next?

The first printing press replaced literal copying by hand. Later we got these linotype-like machines but apart from a few left, these are already thrown out of windows (as we saw in a movie a few Bachtex's ago). Phototypesetting has been replaced too and because a traditional centuries old printing press is a nice to see item, these probably ring more bells than that gray metal closed box typesetters. Organizers of \TeX conferences love to bring the audience to old printing workshops and museums. At some point computers got used for typesetting and in that arena \TeX found its place. These gray closed boxes are way less interesting than something mechanical that at least invites us to touch it. How excited can one be about a stack of \TeX Live DVDs?

5 Remembering

Two times I visited the part of the science museum in London with young family members: distracted by constantly swiping their small powerful devices, they didn't have the least interest in the exhibited computer related items, let alone the fact that the couch they were sitting on was a Cray mainframe. Later on, climbing on some old monument or an old cannon seemed more fun. So, in a few decades folks will still look at wooden printing presses but quickly walk through the part of an exhibition where the tools that we use are shown. We need to find ways to look interesting. But don't think we're unique: how many kids find graphical trend-setting games like *Myst* and *Riven* still interesting? On the other hand a couple of month ago a bunch of nieces and nephews had a lot of fun with an old Atari console running low-res bitmap games. Maybe there is hope for good old \TeX .

Hans Hagen

If indeed we're heading to a radically different society one can argue if this whole discussion makes sense. When the steam engine showed up, the metaphor for what went on in our heads was that technology, It's a popular example of speakers on this topic: "venting off steam". When electricity and radio came around metaphors like "being on the same wavelength" showed up. A few decades ago the computer replaced that model although in the meantime the model is more neurobiological: we're a hormone and neurotransmitter driven computer. We don't have memory the way computers do.

How relevant will page breaks, paragraph and line breaks be in the future? Just like "venting off steam" may make no sense to the youth, asking a typesetter to "give me a break" might not make much sense soon. However, when discussing automated typesetting the question "are we on the same page" still has relevance.

Typesetting with a computer might seem like the ultimate solution but it's actually rather dumb when we consider truly intelligent systems. On the large scale of history and developments what we do might get quite unnoticed. Say that mankind survives the next few hundred years one way or the other. Science fiction novels by Jack McDevitt have an interesting perspective of rather normal humans millennia ahead of us who look back on these times in the same way as we look back now. Nothing fundamental changed in the way we run society. Nearly nothing from the past is left over and apart from being ruled by AIs people still do sort of what they do now. \TeX ? What is that? Well, there once was this great computer scientist Knuth (in the remembered row of names like Aristotle — I just started reading "The Lagoon" by Armand Leroi — Newton, Einstein, his will show up) who had a group of followers that used a program that he seems to have written. And even that is unlikely to be remembered, unless maybe user groups manage to organize an archive and pass that on. Maybe the fact that \TeX was one of the first large scale open source programs, of which someone can study the history, makes it a survivor. The first program that was properly documented in detail! But then we need to make sure that it gets known and persists.

6 Automation

In a recent interview Daniel Dennett explains that his view of the mind as a big neural network, one that can be simulated in software on silicon, is a bit too simplistic. He wonders if we shouldn't more tend to think of a network of (selfish) neurons that group together in tasks and then compete with each other, if only because they want to have something to do.

Maybe attempts to catch the creative mindset and working of a typesetter in algorithms is futile. What actually is great typography or good typesetting? Recently I took a look at my bookshelf wondering what to get rid of—better do that now than when I’m too old to carry the crap down (crap being defined as uninteresting content or bad looking). I was surprised about the on-the-average bad quality of the typesetting and print. It’s also not really getting better. One just gets accustomed to what is the norm at a certain point. Whenever they change the layout and look and feel of the newspaper I read the arguments are readability and ease of access. Well, I never had such a hard time reading my paper as today (with my old eyes).

Are we, like Dennett, willing to discard old views on our tools and models? When my first computer was an RCA 1802 based kit, that had 256 bytes of memory. My current laptop (from 2013) is a Dell Precision workstation with an extreme quad core processor and 16 GB of memory and ssd storage. Before I arrived there I worked with DEC-10, VAX and the whole range of Intel CPUs. So if you really want to compare a brain with a computer, take your choice.

I started with $\text{T}_{\text{E}}\text{X}$ on a 4 MHz desk top with 640 MB memory and a 10 MB hard disk. Running $\text{ConT}_{\text{E}}\text{Xt}$ MkIV with $\text{LuaT}_{\text{E}}\text{X}$ on such a machine is no option at all, but I still carry the burden of trying to write efficient code (which is still somewhat reflected in the code that makes up $\text{ConT}_{\text{E}}\text{Xt}$). In the decades that we have been using $\text{T}_{\text{E}}\text{X}$ we had to adapt! Demands changed, possibilities changed, technologies changed. And they keep changing. How many successive changes can a $\text{T}_{\text{E}}\text{X}$ user handle? Sometimes, when I look and listen I wonder.

If you look back, that is, if you read about the tens of thousands of years that it took humans to evolve (“The mind in the cave” by Lewis-Williams is a good exercise) you realize even more in what a fast-paced time we live and that we’re witnessing transitions of another magnitude.

In the evolution of species some tools were invented multiple times, like eyes. You see the same in our $\text{T}_{\text{E}}\text{X}$ world: multiple (sub)macro packages, different font technologies, the same solutions but with an alternative approach. Some disappear, some stay around. Just like different circumstances demand different solutions in nature, so do different situations in typesetting, for instance different table rendering solutions. Sometime I get the feeling that we focus too much on getting rid of all but one solution while more natural would be to accept diversity, like bio-diversity is accepted. Transitions nowadays happen faster but the question is if, like aeons before,

we (have to) let them fade away. When evolution is discussed the terms ‘random’, ‘selection’, ‘fit’, and so on are used. This probably also applies to typography: at some point a font can be used a lot, but in the end the best readable and most attractive one will survive. Newspapers are printed in many copies, but rare beautiful books hold value. Of course, just like in nature some developments force the further path of development, we don’t suddenly grow more legs or digits on our hands. The same happens with $\text{T}_{\text{E}}\text{X}$ on a smaller timescale: successors still have the same core technology, also because if we’d drop it, it would be something different and then give a reason to reconsider using such technology (which likely would result in going by another path).

7 Quality

Richard Dawkins “The Ancestor’s Tale” is a non-stop read. In a discussion with Jared Diamond about religion and evolution they ponder this thread: you holding the hand of your mother who is handing her mother’s hand and so on till at some point fish get into the picture. The question then is, when do we start calling something human? And a related question is, when does what we call morality creeps in? Is 50% neanderthaler human or not?

So, in the history of putting thoughts on paper: where does $\text{T}_{\text{E}}\text{X}$ fit in? When do we start calling something automated typesetting? When do we decide that we have quality? Is $\text{T}_{\text{E}}\text{X}$ so much different from its predecessors? And when we see aspects of $\text{T}_{\text{E}}\text{X}$ (or related font technology) in more modern programs, do we see points where we cross qualitative or other boundaries? Is a program doing a better job than a human? Where do we stand? There are fields where there is no doubt that machines outperform humans. It’s probably a bit more difficult in aesthetic fields except perhaps when we lower the conditions and expectations (something that happens a lot).

For sure $\text{T}_{\text{E}}\text{X}$ will become obsolete, maybe even faster than we think, but so will other typesetting technologies. Just look back and have no illusions. Till then we can have our fun and eventually, when we have more free time than we need, we might use it out of hobbyism. Maybe $\text{T}_{\text{E}}\text{X}$ will be remembered by probably its most important side effect: the first large scale open source, the time when users met over programs, Knuth’s disciples gathered in user groups, etc. The tools that we use are just a step in an evolution. And, as with evolution, most branches are pruned. So, when in the far future one looks back, will they even notice $\text{T}_{\text{E}}\text{X}$? The ancestor’s tail turns the tree upside down: at the end of the successful branch one doesn’t see the dead ends.

Just a thought: CDs and media servers are recently being replaced (or at least accompanied) by Long Play records. In the shop where I buy my CDs the space allocated to records grows at the cost of more modern media. So, maybe at some point retypesetting will pop up. Of course it might skip \TeX and end up at woodcutting or printing with lead.

8 What mission

We rely on search engines instead of asking around or browsing libraries. Do students really still read books and manuals or do they just search and listen to lectures. Harari claims that instead of teaching kids facts in school we should just take for granted that they can get all the data they want and that we should learn them how to deal with data and adapt to what is coming. We take for granted that small devices with human voices show us the route to drive to \BachTeX , for instance, although by now I can drive it without help. In fact, kids can surprise you by asking if we're driving in Germany when we are already in Poland.

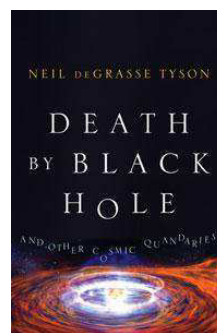
We accept that computer programs help physicians in analyzing pictures. Some wear watches that warn them about health issues, and I know a few people who monitor their sugar levels electronically instead of relying on their own measurements. We seem to believe and trust the programs. And indeed, we also believe that \TeX does the job in the best way possible. How many people really understand the way \TeX works?

We still have mailing lists where we help each other. There are also wikis and forums like stack exchange. But who says that even a moderate bit of artificial intelligence doesn't answer questions better. Of course there needs to be input (manuals, previous answers, etc.) but just like we need fewer people as workforce soon, the number of experts needed also can be smaller. And we're still talking about a traditional system like \TeX . Maybe the social experience that we have on these media will survive somehow, although: how many people are members of societies, participate in demonstrations, meet weekly in places where ideas get exchanged, compared to a few decades ago? That being said, I love to watch posts with beautiful \ConTeXt solutions or listen to talks by enthusiastic users who do things I hadn't expected. I really hope that this property survives, just like I hope that we will be able to see the difference between a real user's response and one from an intelligent machine (an unrealistic hope I fear). Satisfaction wins and just like our neurological subsystems at some point permanently adapt to thresholds (given that you trigger things often

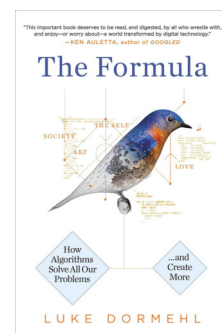
enough), we get accustomed to what \TeX provides and so we stick to it.

9 Intelligence versus consciousness

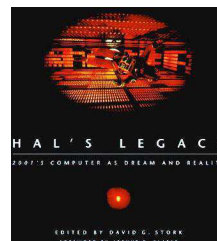
Much of what we do is automated. You don't need to think of which leg to move and what foot to put down when you walk. Reacting to danger also to a large extent is automated. It doesn't help much to start thinking about how dangerous a lion can be when it's coming after you, you'd better move fast. Our limbic system is responsible for such automated behaviour, for instance driven by emotions. The more difficult tasks and thoughts about them happen in the frontal cortex (sort of).



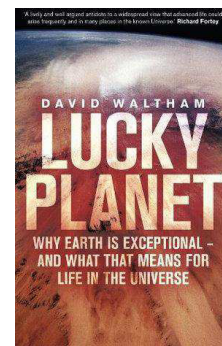
astronomy



informatics



future science



earth science

For most users \TeX is like the limbic system: there is not much thinking involved, and the easy solutions are the ones used. Just like hitting a nerve triggers a chain of reactions, hitting a key eventually produces a typeset document. Often this is best because the job needs to get done and no one really cares how it looks; just copy a preamble, key in the text and assume that it works out well (enough). It is tempting to compare \TeX 's penalties, badness and other parameters with levels of hormones and neurotransmitters. Their function depends on where they get used and the impact can be accumulated, blocked or absent. It's all magic, especially when things interact.

Existing \TeX users, developers and user groups of course prefer to think otherwise, that it is a positive choice by free will. That new users have looked around and arrived at \TeX for good reason: their frontal cortex steering a deliberate choice. Well, it might have played a role but the decision to use \TeX might in the end be due to survival skills: I want to pass this exam and therefore I will use that weird system called \TeX .

All animals, us included, have some level of intelligence but also have this hard to describe property that we think makes us what we are. Intelligence and consciousness are not the same (at least we know a bit about the first but nearly nothing about the second). We can argue about how well composed some music is but why we like it is a different matter.

We can make a well thought out choice for using \TeX for certain tasks but can we say why we started liking it (or not)? Why it gives us pleasure or maybe grief? Has it become a drug that we got addicted to? So, one can make an intelligent decision about using \TeX but getting a grip on why we like it can be hard. Do we enjoy the first time struggle? Probably not. Do we like the folks involved? Yes, Don Knuth is a special and very nice person. Can we find help and run into a friendly community? Yes, and a unique one too, annoying at times, often stimulating and on the average friendly for all the odd cases running around.

Artificial intelligence is pretty ambitious, so speaking of machine intelligence is probably better. Is \TeX an intelligent program? There is definitely some intelligence built in and the designer of that program is for sure very intelligent. The designer is also a conscious entity: he likes what he did and finds pleasure in using it. The program on the other hand is just doing its job: it doesn't care how it's done and how long it takes: a mindless entity. So here is a question: do we really want a more intelligent program doing the job for us, or do those who attend conferences like Bacho \TeX enjoy \TeX ing so much that they happily stay with what they have now? Compared to rockets tumbling down and/or exploding or Mars landers thrashing themselves due to programming errors of interactions, \TeX is surprisingly stable and bug free.

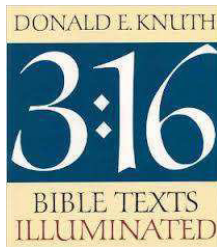
10 Individual versus group evolution

After listening for hours to Sapolsky you start getting accustomed to remarks about (unconscious) behaviour driven by genes, expression and environment, aimed at "spreading many copies of your genes". In most cases that is an individual's driving force. However, cooperation between individuals plays a role in this. A possible view is that we have now reached a state where survival is more dependent on a group than on an individual. This makes sense when we consider that developments (around us) can go way faster than regular evolution (adaptation) can handle. We take control over evolution, a mechanism that needs time to adapt and time is something we don't give it anymore.

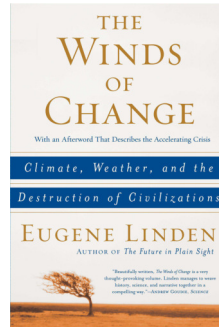
Why does \TeX stay around? It started with an individual but eventually it's the groups that keeps it going. A too-small group won't work but too-large groups won't work either. It's a known fact that one can only handle some 150 social contacts: we evolved in small bands that split when they became too large. Larger groups demanded abstract beliefs and systems to deal with the numbers: housing, food production, protection. The \TeX user groups also provide some organization: they organize meetings, somehow keep development going and provide infrastructure and distributions. They are organized around languages. According to Diamond new languages are still discovered but many go extinct too. So the potential for language related user groups is not really growing.

Some of the problems that we face in this world have become too large to be dealt with by individuals and nations. In spite of what anti-globalists want we cannot deal with our energy hunger, environmental issues, lack of natural resources, upcoming technologies without global cooperation. We currently see a regression in cooperation by nationalistic movements, protectionism and the usual going back to presumed better times, but that won't work.

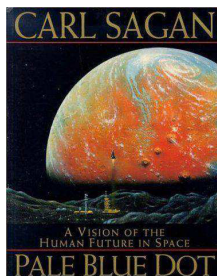
Local user groups are important but the number of members is not growing. There is some cooperation between groups but eventually we might need to combine the groups into one which might succeed unless one wants to come first. Of course we will get the same sentiments and arguments as in regular politics but on the other hand, we already have the advantage of \TeX systems being multi-lingual and users sharing interest in the diversity of usage and users. The biggest challenge is to pass on what we have achieved. We're just a momentary highlight and let's not try to embrace some " \TeX first" madness.



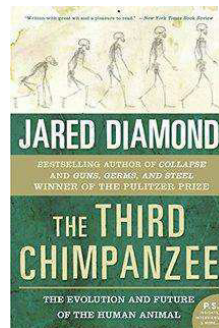
art



history



astronomy



history

11 Sexes

Most species have two sexes but it is actually a continuum controlled by hormones and genetic expression: we just have to accept it. Although the situation has improved there are plenty of places where some gender relationships are considered bad even to the extent that one's life can be in danger. Actually having strong ideas about these issues is typically human. But in the end one has to accept the continuum.

In a similar way we just have to accept that \TeX usage, application of \TeX engines, etc. is a continuum and not a batch versus WYSIWYG battle any more. It's disturbing to read strong recommendations not to use this or that. Of the many macro packages that showed up only a few were able to survive. How do users of outlines look at bitmaps, how do DVI lovers look at PDF. But, as typesetting relates to esthetics, strong opinions come with the game.

Sapolsky reports about a group of baboons where due to the fact that they get the first choice of food the alpha males of pack got poisoned, so that the remaining suppressed males who treated the females well became dominant. In fact they can then make sure that no new alpha male from outside joins the pack without behaving like they do. A sort of social selection. In a similar fashion, until now the gatherings of \TeX ies managed to keep its social properties and not been dominated by for instance commerce.

In the animal world often sexes relate to appearance. The word sexy made it to other domains as well. Is \TeX sexy? For some it is. We often don't see the real colors of birds. What looks gray to us looks vivid to a bird which sees in a different spectrum. The same is true for \TeX . Some users see a command line (shell) and think: this is great! Others just see characters and keystrokes and are more attracted to an interactive program. When I see a graphic made by MetaPost, I always note how exact it is. Others don't care if their interactive effort doesn't connect the dots well. Some people (also present here) think that we should make \TeX attractive but keep in mind that like and dislike are not fixed human properties. Some mindsets might as well be the result from our makeup, others can be driven by culture.

12 Religion

One of Sapolsky's lectures is about religion and it comes in the sequence of mental variations including depression and schizophrenia, because all these relate to mental states, emotions, thresholds and such (all things human). That makes it a tricky topic which is why it has not been taped. As I was raised in a moderate Protestant tradition I can imagine that it's an uncomfortable topic instead. But there are actually a few years older videos around and they are interesting to watch and not as threatening as some might expect. Here I just stick to some common characteristics.

If you separate the functions that religions play into for instance explanation of the yet unknown, social interactions, control of power and regulation of morals, then it's clear why at \TeX user group meetings the religious aspect of \TeX has been discussed in talks. Those who see programs as infallible and always right and don't understand the inner working can see it as an almighty entity. In the Netherlands church-going diminishes but it looks like alternative meetings are replacing it (and I'm not talking of football matches). So what are our \TeX meetings? What do we believe in? The reason that I bring up this aspect is that in the \TeX community we can find aspects of the more extremist aspects of religions: if you don't use the macro package that I use, you're wrong. If you don't use the same operating system as I do, you're evil. You will be punished if you use the wrong editor for \TeX ? Why don't you use this library (which, by the way, just replaced that other one)? We create angels and daemons. Even for quite convinced atheists (it's not hard to run into them on youtube) a religion only survives when it has benefits, something that puzzles them. So when

we're religious about \TeX and friends we have to make sure that it's at least beneficial. Also, maybe we fall in Dennett's category of "believers who want to believe": it helps us to do our job if we just believe that we have the perfect tool. Religion has inspired visual and aural art and keeps doing that. (Don Knuth's current musical composition project is a good example of this.)

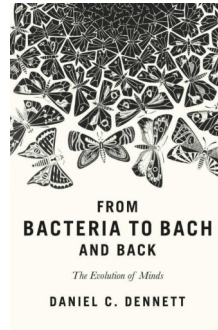
Scientists can be religious, in flexible ways too, which is demonstrated by Don Knuth. In fact, I'm pretty sure \TeX would not be in the position it is in now if it weren't for his knowledgeable, inspirational, humorous, humble, and always positive presence. And for sure he's not at all religious about the open source software that he sent viral.

I'm halfway through reading "The Good Book of Human Nature" (An Evolutionary Reading of the Bible) a book about the evolution of the bible and monotheism which is quite interesting. It discusses for instance how transitions from a hunter to a farmer society demanded a change of rules and introduced stories that made sense in that changing paradigm. Staying in one place means that possessions became more important and therefore inheritance. Often when religion is discussed by behavioral biologists, historians and anthropologists they stress this cultural narrative aspect. Also mentioned is that such societies were willing to support (in food and shelter) the ones that didn't normally fit it but added to the spiritual character of religions. The social and welcoming aspect is definitely present in for instance Bacho \TeX conferences although a bystander can wonder what these folks are doing in the middle of the night around a campfire, singing, drinking, frying sausages, spitting fire, and discussing the meaning of life.

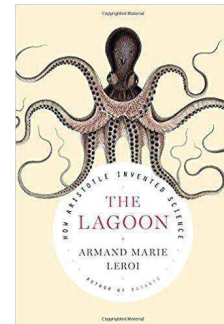
Those who wrap up the state of religious affairs, do predictions and advocate the message, are sometimes called evangelists. I remember a \TeX conference in the USA where the gospel of XML was preached (by someone from outside the \TeX community). We were all invited to believe it. I was sitting in the back of the crowded (!) room and that speaker was not at all interested in who spoke before and after. Well, I do my share of XML processing with Con \TeX t, but believe me: much of the XML that we see is not according to any gospel. It's probably blessed the same way as those state officials get blessed when they ask and pray for it in public.

It can get worse at \TeX conferences. Some present here at Bacho \TeX might remember the PDF evangelists that we had show up at \TeX conferences. You see this qualification occasionally and I have become quite allergic to qualifications like architect, innovator, visionary, inspirator and evangelist, even

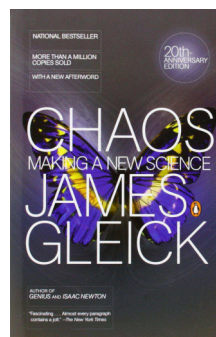
worse when they look young but qualify as senior. I have no problem with religion at all but let's stay away from becoming one. And yes, typography also falls into that trap, so we have to be doubly careful.



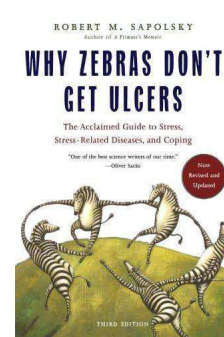
philosophy



science history



science



behavioral biology

13 Chaotic solutions

The lectures on "chaos and reductionism" and "emergence and complexity" were the highlights in Sapolsky's lectures. I'm not a good narrator so I will not summarize them but it sort of boils down to the fact that certain classes of problems cannot be split up in smaller tasks that we understand well, after which we can reassemble the solutions to deal with the complex task. Emerging systems can however cook up working solutions from random events. Examples are colonies of ants and bees.

The \TeX community is like a colony: we cook up solutions, often by trial and error. We dream of the perfect solutions but deep down know that esthetics cannot be programmed in detail. This is a good thing because it doesn't render us obsolete. At last year's Bacho \TeX , my nephew Teun and I challenged the anthill outside the canteen to typeset the \TeX logo with sticks but it didn't persist. So we don't need to worry about competition from that end. How do you program a hive mind anyway?

When chaos theory evolved in the second half of the previous century not every scientist felt happy

about it. Instead of converging to more perfect predictions and control in some fields a persistent uncertainty became reality.

After about a decade of using \TeX and writing macros to solve recurring situations I came to the conclusion that striving for a perfect \TeX (the engine) that can do everything and anything makes no sense. Don Knuth not only stopped adding code when he could do what he needed for his books, he also stuck to what to me seems reasonable endpoints. Every hard-coded solution beyond that is just that: a hard-coded solution that is not able to deal with the exceptions that make up most of the more complex documents. Of course we can theorize and discuss at length the perfect never-reachable solutions but sometimes it makes more sense to admit that an able user of a desktop publishing system can do that job in minutes, just by looking at the result and moving around an image or piece of text a bit.

There are some hard-coded solutions and presets in the programs but with \LuaTeX and \MPLib we try to open those up. And that's about it. Thinking that for instance adding features like protrusion or expansion (or whatever else) always lead to better results is just a dream. Just as a butterfly flapping its wings on one side of the world can have an effect on the other side, so can adding a single syllable to your source completely confuse an otherwise clever column or page break algorithm. So, we settle for not adding more to the engine, and provide just a flexible framework.

A curious observation is that when Edward Lorenz ran into chaotic models it was partially due to a restart of a simulation midway, using printed floating point numbers that then in the computer were represented with a different accuracy than printed. Aware of floating point numbers being represented differently across architectures, Don Knuth made sure that \TeX was insensitive to this so that its outcome was predictable, if you knew how it worked internally. Maybe \LuaTeX introduces a bit of chaos because the Lua we use has only floats. In fact, a few months ago we did uncover a bug in the backend where the same phenomena gave a chaotic crash.

In chaos theory there is the concept of an attractor. When visualized this can be the area (seemingly random) covered by a trajectory. Or it can be a single point where for instance a pendulum comes to rest. So what is our attractor? We have a few actually. First there is the engine, the stable core of primitives always present. You often see programs grow more complex every update and for sure that happened with $\epsilon\text{-TeX}$, \pdfTeX , \XeTeX and \LuaTeX . However there is always the core that is supposed

to be stable. After some time the new kid arrives at a stable state not much different from the parent. The same is true for \MetaPost . Fonts are somewhat different because the technology changes but in the end the shapes and their interactions become stable as well. Yet another example is \TeX Live : during a year it might diverge from its route but eventually it settles down and enters the area where we expect it to end up. The \TeX world is at times chaotic, but stable in the long run.

So, how about the existence, the reason for it still being around? One can speculate about its future trajectory but one thing is sure: as long as we break a text into paragraphs and pages \TeX is hard to beat. But what if we don't need that any more? What if the concept of a page is no longer relevant? What if justified texts no longer matter (often designers don't care anyway)? What if students are no longer challenged to come up with a nice looking thesis? Do these collaborative tools with remote \TeX processing really bring new long term users or is \TeX then just one of the come-and-go tools?

14 Looking ahead

In an interview (“World of ideas”) Asimov explains that science fiction evolved rapidly when people lived long enough to see that there was a future (even for their offspring) that is different from today. It is (at least for me) mind boggling to think of an evolution of hundreds of thousands of years to achieve something like language. Waiting for the physical being to arrive at a spot where you can make sounds, where the brain is suitable for linguistic patterns, etc. A few hundred years ago speed of any developments (and science) stepped up.

\TeX is getting near 40 years old. Now, for software that is old! In that period we have seen computers evolve: thousands of times faster processing, even more increase in memory and storage. If we read about spaceships that travel at a reasonable fraction of the speed of light, and think that will not happen soon, just think back to the terminals that were sitting in computer labs when \TeX was developed: 300 baud was normal. I actually spent quite some time on optimizing time-critical components of \ConTeXt but on this timescale that is really a waste of time. But even temporary bottlenecks can be annoying (and costly) enough to trigger such an effort. (Okay, I admit that it can be a challenge, a kind of game, too.)

Neil Tyson, in the video “Storytelling of science” says that when science made it possible to make photos it also made possible a transition in painting to impressionism. Other technology could make the

exact snapshot so there was new room for inner feelings and impressions. When the Internet showed up we went through a similar transition, but \TeX actually dates from before the Internet. Did we also have a shift in typesetting? To some extent yes, browsers and real time rendering is different from rendering pages on paper. In what space and time are \TeX ies rooted?

We get older than previous generations. Quoting Sapolsky “. . . we are now living well enough and long enough to slowly fall apart.” The opposite is happening with our tools, especially software: it’s useful lifetime becomes shorter and changes faster each year. Just look at the version numbers of operating systems. Don Knuth expected \TeX to last for a long time and compared to other software its core concept and implementation is doing surprisingly well. We use a tool that suits our lifespan! Let’s not stress ourselves out too much with complex themes. (It helps to read “Why zebras don’t get ulcers”.)

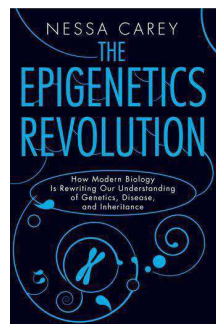
15 Memes

If you repeat a message often enough, even if it’s something not true, it can become a meme that gets itself transferred across generations. Conferences like this is where they can evolve. We tell ourselves and the audience how good \TeX is and because we spend so many hours, days, weeks, months using it, it actually must be good, or otherwise we would not come here and talk about it. We’re not so stupid as to spend time on something not good, are we? We’re always surprised when we run into a (potential) customer who seems to know \TeX . It rings a bell, and it being around must mean something. Somehow the \TeX meme has anchored itself when someone attended university. Even if experiences might have been bad or usage was minimal. The meme that \TeX is the best in math typesetting is a strong survivor.

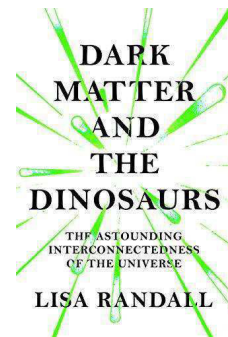
There’s a certain kind of person who tries to get away with their own deeds and decisions by pointing to “fake news” and accusations of “mainstream media” cheating on them. But to what extent are our stories true about how easy \TeX macro packages are to use and how good their result? We have to make sure we spread the right memes. And the user groups are the guardians.

Maybe macro packages are like memes too. In the beginning there was a bunch but only some survived. It’s about adaptation and evolution. Maybe competition was too fierce in the beginning. Like ecosystems, organisms and cellular processes in biology we can see the \TeX ecosystem, users and usage, as a chaotic system. Solutions pop up, succeed, survive, lead to new ones. Some look similar and slightly

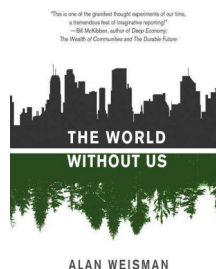
different input can give hugely different outcomes. You cannot really look too far ahead and you cannot deduce the past from the present. Whenever something kicks it off its stable course, like the arrival of color, graphics, font technologies, PDF, XML, ebooks, the \TeX ecosystem has to adapt and find its stable state again. The core technology has proven to be quite fit for the kind of adaptation needed. But still, do it wrong and you get amplified out of existence, don’t do anything and the external factors also make you extinct. There is no denial that (in the computer domain) \TeX is surprisingly stable and adaptive. It’s also hard not to see how conservatism can lead to extinction.



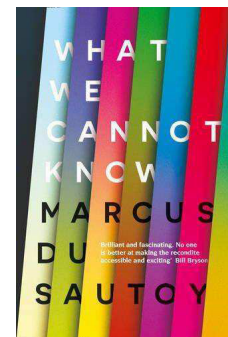
genetics



physics



history



science

16 Inspiration

I just took some ideas from different fields. I could have mentioned quantum biology, which tries to explain some unexplainable phenomena in living creatures. For instance how do birds navigate without visible and measurable clues. How do people arrive at \TeX while we don’t really advertise? Or I could mention epigenetics and explorations in junk DNA. It’s not the bit of the genome that we thought that matters, but also the expression of the genes driven by other factors. Offspring not only gets genetic material passed but it can get presets. How can the \TeX community pass on Knuth’s legacy? Do we need to hide the message in subtle ways? Or how about the quest for dark matter? Does it really exist or do

we want (need) it to exist? Does \TeX really have that many users, or do we cheat by adding the users that are enforced during college but don't like it at all? There's enough inspiration for topics at \TeX conferences, we just have to look around us.

17 Stability

I didn't go into technical aspects of \TeX yet. I must admit that after decades of writing macros I've reached a point where I can safely say that there will never be perfect automated solutions for really complex documents. When books about neural networks show up I wondered if it could be applied (but I couldn't). When I ran into genetic algorithms I tried to understand its possible impact (but I never did). So I stuck to writing solutions for problems using visualization: the trial and error way. Of course, speaking of $\text{Con}\text{\TeX}t$, I will adapt what is needed, and others can do that as well. Is there a new font technology? Fine, let's support it as it's no big deal, just a boring programming task. Does a user want a new mechanism? No problem, as solving a reduced subset of problems can be fun. But to think of \TeX in a reductionist way, i.e. solving the small puzzles, and to expect the whole to work in tandem to solve a complex task is not trivial and maybe even impossible. It's a good thing actually, as it keeps us on edge. Also, $\text{Con}\text{\TeX}t$ was designed to help you with your own solutions: be creative.

I mentioned my nephew Bram. He has seen part of this crowd a few times, just like his brother and sister do now. He's into artificial intelligence now. In a few years I'll ask him how he sees the current state of \TeX affairs. I might learn a few tricks in the process.

In "The world without us" Weisman explores how fast the world would be void of traces of humankind. A mere 10.000 years can be more than enough. Looking back, that's about the time hunters became farmers. So here's a challenge: say that we want an ant culture that evolves to the level of having archaeologists to know that we were here at $\text{Bach}\text{\TeX}$. . . what would we leave behind?

Sapolsky ends his series by stressing that we should accept and embrace individual differences. The person sitting next to you can have the same makeup but be just a bit more sensitive to depression or be the few percent with genes controlling schizophrenic behaviour. He stresses that knowing how things work or where things go wrong doesn't mean that we should fix everything. So look at this room full of \TeX ies: we don't need to be all the same, use all the same, we don't need some dominance, we just need to accept and especially we need to under-

stand that we can never fully understand (and solve) everything forever.

Predictions, one of the themes, can be hard. It's not true that science has the answer to everything. There will always be room for speculation and maybe we will always need metaphysics too. I just started to read "What we cannot know" by Sautoy. For sure those present here can not predict how \TeX will go on and/or be remembered.

18 Children of \TeX

I mentioned "Children of time". The author lets you see their spidery world through spider eyes and physiology. They have different possibilities (eyesight, smell) than we do and also different mental capabilities. They evolve rapidly and have to cope conceptually with signals from a human surveillance satellite up in the sky. Eventually they need to deal with a bunch of (of course) quarrelling humans who want their place on the planet. We humans have some pre-occupation with spiders and other creatures. In a competitive world it is sometimes better to be suspicious (and avoid and flee) that to take a risk of being eaten. A frequently used example is that a rustle in a bush can be the wind or a lion, so best is to run.

We are not that well adapted to our current environment. We evolved at a very slow pace so there was no need to look ahead more than a year. And so we still don't look too far ahead (and choose politicians accordingly). We can also not deal that well with statistics (Dawkins's "Climbing Mount Probability" is a good read) so we make false assumptions, or just forget.

Does our typeset text really look that good on the long run, or do we cheat with statistics? It's not too hard to find a bad example of something not made by \TeX and extrapolate that to the whole body of typeset documents. Just like we can take a nice example of something done by \TeX and assume that what we do ourselves is equally okay. I still remember the tests we did with $\text{pdf}\text{\TeX}$ and hz . When Hàn Th   Thành and I discussed that with Hermann Zapf he was not surprised at all that no one saw a difference between the samples and instead was focusing on aspects that \TeX ies are told to look at, like two hyphens in a row.

A tool like \TeX has a learning curve. If you don't like that just don't use it. If you think that someone doesn't like that, don't enforce this tool on that someone. And don't use (or lie with) statistics. Much better arguments are that it's a long-lived stable tool with a large user base and support. That it's not a waste of time. Watching a designer like Hermann

Zapf draw shapes is more fun than watching click and point in heavily automated tools. It's probably also less fun to watch a \TeX converge towards a solution.

Spiders are resilient. Ants maybe even more. Ants will survive a nuclear blast (mutations might even bring them benefits), they can handle the impact of a meteorite, a change in climate won't harm them much. Their biggest enemy is probably us, when we try to wipe them out with poison. But, as long as they keep a low profile they're okay. \TeX doesn't fit into the economic model as there is no turnaround involved, no paid development, it is often not seen at all, it's just a hit in a search engine and even then you might miss it (if only because no one pays for it being shown at the top).

We can learn from that. Keeping a low profile doesn't trigger the competition to wipe you out. Many (open source) software projects fade away: some big company buys out the developer and stalls the project or wraps what they bought in their own stuff, other projects go professional and enterprise and alienate the original users. Yet others abort because the authors lose interest. Just like the ideals of socialism don't automatically mean that every attempt to implement it is a success, so not all open source and free software is good (nature) by principle either. The fact that communism failed doesn't mean that capitalism is better and a long term winner. The same applies to programs, whether successful or not.

Maybe we should be like the sheep. Dennett uses these animals as a clever species. They found a way to survive by letting themselves (unconsciously) be domesticated. The shepherd guarantees food, shelter and protection. He makes sure they don't get ill. Speaking biologically: they definitely made sure that many copies of their genes survived. Cows did the same and surprisingly many of them are related due to the fact that they share the same father (something now trying to be reverted). All \TeX spin-offs relate to the same parent, and those that survived are those that were herded by user groups. We see bits and pieces of \TeX end up in other applications. Hyphenation is one of them. Maybe we should settle for that small victory in a future hall of fame.

When I sit on my balcony and look at the fruit trees in my garden, some simple math can be applied. Say that one of the apple trees has 100 apples per year and say that this tree survives for 25 years (it's one of those small manipulated trees). That makes 2.500 apples. Without human intervention only a few of these apples make it into new trees, otherwise the whole world would be dominated by apple trees.

Of course that tree now only survives because we permit it to survive, and for that it has to be humble (something that is very hard for modern Apples). Anyway, the apple tree doesn't look too unhappy.

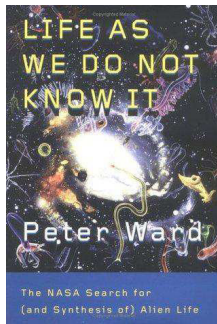
A similar calculation can be done for birds that nest in the trees and under my roof. Given that the number of birds stays the same, most of energy spent on raising offspring is wasted. Nevertheless they seem to enjoy life. Maybe we should be content if we get one enthusiastic new user when we demonstrate \TeX to thousands of potential users.

Maybe, coming back to the themes of the conference, we should not come up with these kinds of themes. We seem to be quite happy here. Talking about the things that we like, meeting people. We just have to make sure that we survive. Why not stay low under the radar? That way nothing will see us as a danger. Let's be like the ants and spiders, the invisible hive mind that carries our message, whatever that is.

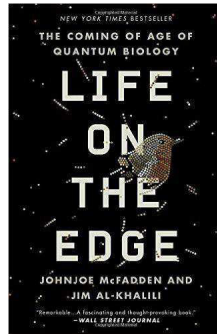
When Dennett discusses language he mentions (coined) words that survive in language. He also mentions that children pick up language no matter what. Their minds are made for it. Other animals don't do that: they listen but don't start talking back. Maybe \TeX is just made for certain minds. Some like it and pick it up, while for others it's just noise. There's nothing wrong with that. Predilection can be a user property.

19 The unexpected

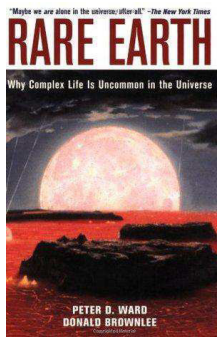
In a discussion with Dawkins the well-spoken astrophysicist Neil deGrasse Tyson brings up the following. We differ only a few percent in DNA from a chimp but quite a lot in brain power, so how would it be if an alien that differs a few percent (or more) passes by earth. Just like we don't talk to ants or chimps or whatever expecting an intelligent answer, whatever passes earth won't bother wasting time on us. Our rambling about the quality of typesetting probably sounds alien to many people who just want to read and who happily reflow a text on an ebook device, not bothered by a lack of quality.



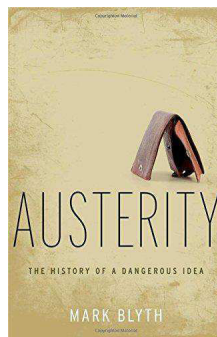
astrobiology



quantum biology



astrophysics



economics

We tend to take ourselves as reference. In “Rare Earth”, Ward and Brownlee extrapolate the possibility of life elsewhere in the universe. They are not alone in thinking that while on one hand applying statistics to these formulas of possible life on planets there might also be a chance that we’re the only intelligent species ever evolved. In a follow up, “Life as we do not know it” paleontologist and astrobiologist Ward (one of my favourite authors) discusses the possibility of life not based on carbon, which is not natural for a carbon based species. Carl Sagan once pointed out that an alien species looking down to earth can easily conclude that cars are the dominant species on earth and that the thingies crawling in and out them are some kind of parasites. So, when we look at the things that somehow end up on paper (as words, sentences, ornaments, etc.), what is dominant there? And is what we consider dominant really that dominant in the long run? You can look at a nice page as a whole and don’t see the details of the content. Maybe beauty hides nonsense.

When T_EXies look around they look to similar technologies. Commands in shells and solutions done by scripting and programming. This make sense in the perspective of survival. However, if you want to ponder alternatives, maybe not for usage but just for fun, a completely different perspective might be needed. You must be willing to accept that communicating with a user of a WYSIWYG program might be impossible. If mutual puzzlement is a fact, then

they can either be too smart and you can be too dumb or the reverse. Or both approaches can be just too alien, based on different technologies and assumptions. Just try to explain T_EX to a kid 40 years younger or to an 80 year old grandparent for that matter. Today you can be very clever in one area and very stupid in another.

In another debate, Neil deGrasse Tyson asks Dawkins the question why in science fiction movies the aliens look so human and when they don’t, why they look so strange, for instance like cumbersome sluggish snails. The response to that is one of puzzlement: the opponent has no reference of such movies. In discussions old T_EXies like to suggest that we should convert young users. They often don’t understand that kids live in a different universe.

How often does that happen to us? In a world of many billions T_EX has its place and can happily coexist with other typesetting technologies. Users of other technologies can be unaware of us and even create wrong images. In fact, this also happens in the community itself: (false) assumptions turned into conclusions. Solutions that look alien, weird and wrong to users of the same community. Maybe something that I present as hip and modern and high-T_EX and promising might be the opposite: backward, old-fashioned and of no use to others. Or maybe it is, but the audience is in a different mindset. Does it matter? Let’s just celebrate that diversity. (So maybe, instead of discussing the conference theme, I should have talked about how I abuse LuaT_EX in controlling lights in my home as part of some IoT experiments.)

20 What drives us

I’m not a fan of economics and big money talk makes me suspicious. I cannot imagine working in a large company where money is the drive. It also means that I have not much imagination in that area. We get those calls at the office from far away countries who are hired to convince us by phone of investments. Unfortunately mentioning that you’re not at all interested in investments or that multiplying money is irrelevant to you does not silence the line. You have to actively kill such calls. This is also why I probably don’t understand today’s publishing world where money also dominates. Recently I ran into talks by Mark Blyth about the crisis (what crisis?) and I wish I could argue like he does when it comes to typesetting and workflows. He discusses quite well that most politicians have no clue what the crisis is about.

I think that the same applies to the management of publishers: many have no clue what typesetting is about. So they just throw lots of money into the wrong activities, just like the central banks seem

to do. It doesn't matter if we \TeX ies demonstrate cheap and efficient solutions.

Of course there are exceptions. We're lucky to have some customers that do understand the issues at hand. Those are also the customers where authors may use the tools themselves. Educating publishers, and explaining that authors can do a lot, might be a premise, predilection and prediction in one go! Forget about those who don't get it: they will lose eventually, unfortunately not before they have reaped and wasted the landscape.

Google, Facebook, Amazon, Microsoft and others invest a lot in artificial intelligence (or, having all that virtual cash, just buy other companies that do). They already have such entities in place to analyze whatever you do. It is predicted that at some point they know more about you than you know yourself. Reading Luke Dormehl's "The Formula" is revealing. So what will that do with our so-called (disputed by some) free will? Can we choose our own tools? What if a potential user is told that all his or her friends use WhateverOffice so they'd better do that too? Will subtle pressure lead them or even us users away from \TeX ? We already see arguments among \TeX ies, like "It doesn't look updated in 3 years, is it still good?" Why update something that is still valid? Will the community be forced to update everything, sort of fake updates. Who sets out the rules? Do I really need to update (or re-run) manuals every five years?

Occasionally I visit the Festo website. This is a (family owned) company that does research at the level that used to be common in large companies decades ago. If I had to choose a job, that would be the place to go to. Just google for "festo bionic learning network" and you understand why. We lack this kind of research in the field we talk about today: research not driven by commerce, short term profit, long term control, but because it is fundamental fun.

Last year Alan Braslau and I spent some time on BIB \TeX . Apart from dealing with all the weird aspects of the APA standard, dealing with the inconsistently constructed author fields is a real pain. There have been numerous talks about that aspect here at Bacho \TeX by Jean-Michel Hufflen. We're trying to deal with a more than 30-year-old flawed architecture. Just look back over a curve that backtracks 30 years of exponential development in software and databases and you realize that it's a real waste of time and a lost battle. It's fine to have a text based database, and stable formats are great, but the lack of structure is appalling and hard to explain to young programmers. Compare that to the Festo projects and you realize that there can be more challenging projects. Of course, dealing with the old data can

be a challenge, a necessity and eventually even be fun, but don't even think that it can be presented as something hip and modern. We should be willing to admit flaws. No wonder that Jean-Michel decided to switch to talking about music instead. Way more fun.

Our brains are massively parallel bio-machinery. Groups of neurons cooperate and compete for attention. Coming up with solutions that match what comes out of our minds demands a different approach. Here we still think in traditional programming solutions. Will new ideas about presenting information, the follow up on books come from this community? Are we the innovative Festo or are we an old dinosaur that just follows the fashion?

21 User experience

Here is a nice one. Harari spends many pages explaining that research shows that when an unpleasant experience has less unpleasantness at the end of the period involved, the overall experience is valued according to the last experience. Now, this is something we can apply to working with \TeX : often, the more you reach the final state of typesetting the more it feels as all hurdles are in the beginning: initial coding, setting up a layout, figuring things out, etc.

It can only get worse if you have a few left-over typesetting disasters but there adapting the text can help out. Of course seeing it in a cheap bad print can make the whole experience bad again. It happens. There is a catch here: one can find lots of bad-looking documents typeset by \TeX . Maybe there frustration (or indifference) prevails.

I sometimes get to see what kind of documents people make with Con \TeX t and it's nice to see a good looking thesis with diverse topics: science, philosophy, music, etc. Here \TeX is just instrumental, as what it is used for is way more interesting (and often also more complex) than the tool used to get it on paper. We have conferences but they're not about rocket science or particle accelerators. Proceedings of such conferences can still scream \TeX , but it's the content that matters. Here somehow \TeX still sells itself, being silently present in rendering and presentations. It's like a rootkit: not really appreciated and hard to get rid of. Does one discuss the future of rootkits other than in the perspective of extinction? So, even as an invisible rootkit, hidden in the workings of other programs, \TeX 's future is not safe. Sometimes, when you install a Linux system, you automatically get this large \TeX installation, either because of dependencies or because it is seen as a similar toolkit as for instance Open (or is it Libre) Office. If you don't need it, that user might as well start seeing it as a (friendly) virus.

22 Conclusion

At some point those who introduced computers in typesetting had no problem throwing printing presses out of the window. So don't pity yourself if at some point in the near future you figure out that professional typesetting is no longer needed. Maybe once we let machines rule the world (even more) we will be left alone and can make beautiful documents (or whatever) just for the joy, not bothering if we use outdated tools. After all, we play modern music on old instruments (and the older rock musicians get, the more they seem to like acoustic).

There are now computer generated compositions that experienced listeners cannot distinguish from old school. We already had copies of paintings that could only be determined forgeries by looking at chemical properties. Both of these (artificial) arts can be admired and bring joy. So, the same applies to fully automated typeset novels (or runtime rendered ebooks). How bad is that really? You don't dig channels with your hand. You don't calculate logarithmic tables manually any longer.

However, one of the benefits of the Internet is watching and listening to great minds. Another is seeing musicians perform, which is way more fun than watching a computer (although googling for "animusic" brings nice visuals). Recently I ran into a wooden musical computer made by "Wintergatan" which reminded me of the "Paige Compositor" that we use in a LuaTeX cartoon. Watching something like that nicely compensates for a day of rather boring programming. Watching how the marble machine x (mmx) evolves is yet another nice distraction.

Now, the average age of the audience here is pretty high even if we consider that we get older. When I see solutions of ConTeXt users (or experts) posted by (young) users on the mailing list or stack exchange I often have to smile because my answer would have been worse. A programmable system invokes creative solutions. My criterion is always that it has to look nice in code and has some elegance. Many posted solutions fit. Do we really want more automation? It's more fun to admire the art of solutions and I'm amazed how well users use the possibilities (even ones that I already forgot).

One of my favourite artists on my weekly "check youtube" list is Jacob Collier. Right from when I ran into him I realized that a new era in music had begun. Just google for his name and "music theory interview" and you probably understand what I mean. When Dennett comments on the next generation (say up to 25) he wonders how they will evolve as they grow up in a completely different environment of connectivity.

I can see that when I watch family members. Already long ago Greg Bear wrote the novel "Darwin's Children". It sets you thinking and when looking around you even wonder if there is a truth in it.

There are folks here at BachoTeX who make music. Now imagine that this is a conference about music and that the theme includes the word "future". Then, imagine watching that video. You see some young musicians, one of them probably one of the musical masterminds of this century, others instrumental to his success, for instance by wrapping up his work. While listening you realize that this next generation knows perfectly well what previous generations did and achieved and how they influenced the current. You see the future there. Just look at how old musicians reflect on such videos. (There are lots of examples of youth evolving into prominent musicians around and I love watching them). There is no need to discuss the future, in fact, we might make a fool of ourselves doing so. Now back to this conference. Do we really want to discuss the future? What we think is the future? Our future? Why not just hope that in the flow of getting words on a medium we play our humble role and hope we're not forgotten but remembered as inspiration.

One more word about predicting the future. When Arthur Clarke's "2001: A Space Odyssey" was turned into a movie in 1968, a lot of effort went into making sure that the not so far ahead future would look right. In 1996 scientists were asked to reflect on these predictions in "Hal's Legacy". It turned out that most predictions were plain wrong. For instance computers got way smaller (and even smaller in the next 20 years) while (self-aware) artificial intelligence had not arrived either. So, let's be careful in what we predict (and wish for).

23 No more themes

We're having fun here, that's why we come to BachoTeX (predilection). That should be our focus. Making sure that TeX's future is not so much in the cutting edge but in providing fun to its users (prediction). So we just have to make sure it stays around (premise). That's how it started out. Just look at Don Knuth's 3:16 poster: via TeX and METAFONT he got in contact with designers and I wouldn't be surprised if that sub-project was among the most satisfying parts. So, maybe instead of ambitious themes the only theme that matters is: show what you did and how you did it.

◇ Hans Hagen
Pragma ADE
<http://pragma-ade.com>

MFLua 0.8 — Prologue

Luigi Scarso

Abstract

Reflections on the roles of $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{u}}\text{a}\text{T}_{\text{E}}\text{X}$, $\text{M}_{\text{E}}\text{T}_{\text{A}}\text{F}_{\text{O}}\text{N}T$ and $\text{M}_{\text{F}}\text{L}_{\text{u}}\text{a}$, in the spirit of the theme of the TUG@Bach $\text{T}_{\text{E}}\text{X}$ 2017 conference.

1 Introduction

The opening talk of the TUG@Bach $\text{T}_{\text{E}}\text{X}$ 2017 meeting, given by Hans Hagen, was explicitly focused on the theme of the conference:

Premises — the starting point, what we have, what do we use, what has been achieved,

Predilections — how do we act now, how do we want to act, what is important to us and what do we miss,

Predictions — what is the future of $\text{T}_{\text{E}}\text{X}$, what we'd like to achieve and can we influence it.

Reading the draft of the proceedings, I started to mentally note some thoughts, and later I decided to try to organize them in a consistent way. The original paper was supposed to be focused on the technical details of $\text{M}_{\text{F}}\text{L}_{\text{u}}\text{a}$ 0.8, the new version of $\text{M}_{\text{F}}\text{L}_{\text{u}}\text{a}$ shipped with $\text{T}_{\text{E}}\text{X}$ Live 2017, but I have decided to postpone that to a future paper, preferring a more narrative one here. I consider this a kind of prologue that tries to explain the motivations behind $\text{L}_{\text{u}}\text{a}\text{T}_{\text{E}}\text{X}$ and $\text{M}_{\text{F}}\text{L}_{\text{u}}\text{a}$ — of course, from my personal point of view.

2 Philosophy and history

Hans' talk touched several themes, from artificial intelligence to religion and chaos theory, viewed from the point of view (quite popular nowadays) that mixes “hard” science (as math, physics, information theory, biology) with “soft” science (psychology, sociology, anthropology). Curiously, philosophy is often left out from these considerations — the thing doesn't disturb so much the philosophers because for them “everything follows from philosophy” — but as a result the conclusions always look a bit too provisional. Sometimes the mix returns a blurred image, sometimes a defined one that is already outdated by the course of events, but seldom does a guiding principle emerge from the past to the future; more often, the becoming is sensed as movement that nullifies the past and hides the future.

We can assume as indisputable at least two “facts”: 1) the spread of global communication and 2) the mathematisation of the society. The first had a big impulse about six hundred years ago, with Gutenberg, but now the time required to exchange

a message is roughly several thousand times faster than 100 years ago, and sender and recipient can be almost anywhere — a *novel* situation for the human race. The sensation that the quality of the global communication is not high enough, the naturalness of our act of communicating and the presumption to know that the global communication is not so global after all, hide from us the re-evolutionary step forward we have made in the last 30 years: many common people in the world *can* communicate easily and cheaply. The distinction between synchronous and asynchronous communication, as also between human agent and artificial agent, is irrelevant: we exchange knowledge on a daily basis — and knowledge is a subject of philosophy.

The other fact is rooted in the past, some five hundred years ago, when Galileo asserted that Nature has its own language, and that it is a mathematical language. Being both a mathematician and a philosopher, Galileo understood that math is a common language *also* of mankind, and his observations of Jupiter showed the disruptive power of these facts: suddenly we (as mankind, not only a “select few”) can understand the universe — and perhaps we can control it. From here the process of describing Nature with the language of mathematics — the *mathematisation* of Nature — started (slowly) down its own path.

It's a successful process. Another important milestone (from our perspective) was reached with Hilbert in 1900, when his second problem began reflections on formal systems and computations. After only 100 years (i.e., four times faster than the previous step from Galileo), supported by the fundamental results by several first-class mathematicians and logicians such as Gödel, Turing and Church, this mathematisation process manifested a twofold consequence.

Firstly, the transition from the initial determinism to the probabilistic description of Nature. It could be considered as completion of Hume's reflections on causality: if at the elementary level Heisenberg's uncertainty principle rules, then the Universe cannot be completely understood (and hence controlled) — or, which is the same, the future is not completely determined from the past: we still have chances to learn from the past to make a better future. Secondly, by means of information theory and computer science, math has started to come into human society (from western society and slowly reaching the rest of the world) in a pervasive manner: not only the mechanisation and automation of the means of production (the *hard* part) but the informatisation of the services (the *soft* part).

This, again, is possible because math is a common language of mankind, and, as such, it does not prevent the course of the global communication: the two facts in some way reinforce each other. The mathematisation of Nature affirms the indeterminism of reality; the mathematisation of the society tends to the determinism of its components and relations. Both these tendencies reconcile themselves into the *ontology* — which masked itself behind the more fascinating term of *semantic web*.

3 \TeX and \LuaTeX

The other important assertion from Galileo about math and Nature is the need for an *alphabet* of the math language. It’s a recurring theme: classical Greek and Latin first and English now have been the “lingua franca” that translates our subjective, private, internal and a-symbolic language to a common and shareable one with fixed symbols. It’s interesting to observe how our languages, under the pressure of the global communication, are evolving by accepting new visual symbols (emoticons), showing how fast a language can adapt itself to new demands. The need for a unified math alphabet was always secondary to correctness of reasoning — even logic has been somewhat timid in this area — but in the second half of the 20th century the pressure of mathematisation and global communication was so high that now, in retrospect, the birth of \TeX appears as obvious and unavoidable.

\TeX is the answer to the eternal question in math: “Can we do it better?”, where the problem in this case is the exchange of math knowledge. Without formulas (i.e. algebra) and graphics (i.e. geometry) pure prose is “only” philosophy. As soon as we move to logic, the need for an alphabet emerges as *the* way to avoid the ambiguity of pure prose, and to produce compact and “portable” proofs — and from here it spreads into math.

Different alphabets play against this need, slowing down the process of mathematisation, but, on the other hand, the same process *needs* new symbols when creating new descriptions of concepts (as with category theory) or reality. It was only by means of Knuth that this problem found an optimal solution. Being (for that time) a unique combination of mathematician, computer scientist (winner of a Turing Award) and typographer, the solution was a macro programming language (\TeX) to write math as a mathematician would like to, a procedural language (METAFONT), clearly rooted in algebra and geometry, to create new fonts (hence alphabets), while a similar language (METAPOST) was developed a

bit later for graphics. Finally a “device independent” (DVI) final format of the document, easily portable to other formats. Another procedural language, WEB (Pascal-based), was used to write the tex family of programs, and this raised another problem: a program is a mathematical proof and as such it must be written. Literate programming was Knuth’s answer, of course using \TeX .

The distinctiveness of theorems is that they are forever (like diamonds and extinctions): after 2500 years, the Pythagorean theorem still doesn’t show any wear patterns. Is it the same for \TeX ? Of course the line-breaking algorithm is still valid (and that a page-breaking algorithm is still NP) and \TeX has no known bugs (only “features”) so, if the theorem is “Is this language correct?” for which the tex program is the proof, we can say again yes. But is the language still suitable for the mathematisation of the society? How does it behave with the exponential growth of communication?

Quite surprisingly, for a 40-year-old program, \TeX stands up well. A set of macros, \LaTeX , is almost a standard de facto; the concept of literate programming, not as widespread as it should be, has made it possible to extend the program, ultimately resulting, after some intermediate steps, in the \pdfTeX engine. A quick look at https://arxiv.org/help/stats/2016_by_area/index shows about one hundred thousand submissions for 2016, and the rate is increasing: most of them are in $(\text{\La})\text{\TeX}$ and arXiv, as well as \pdfLaTeX , still accepts documents that compile to DVI. Even without taking into account other sources, it’s wrong to conclude that the role of \TeX was and is marginal in this process.

On the other side, METAFONT, after an initial period, was never adopted in the mainstream as such but always subsumed by other font formats: one of the main purposes of \pdfTeX was, besides natively supporting the PDF format, using the Type 1 font format by Adobe, even though it lacks full support for the more widespread TrueType format developed by Microsoft.

The next challenge, that of the OpenType format, was taken up by \XeTeX and, later, by \LuaTeX . During the first decade of the 21st century the coexistence of three engines was easily resolved by specializing the \LaTeX format (and with \LuaTeX , the \ConTeXt format), apparently showing that the new engines, ultimately, are not as significant a change compared to the original one. The GUST team, with the active support of almost all \TeX user groups, has managed to produce Type 1, TrueType and OpenType Latin Modern versions of Computer Modern

using METAPOST and AFDKO (Adobe Font Development Kit for OpenType), and later FontForge, safeguarding the ability to create new alphabets.

As mentioned, apparently these were minor adjustments of Knuth’s solution but, under the surface, they reveal a lack of being able to keep up with the environment. Right from the start, the global communication pushed T_EX to interact with other languages, going outside the realm of math symbols and “standard” English (i.e. that used in scientific publications). The hyphenation capability of T_EX was the answer but the Unicode standard and OpenType demand greater flexibility and, outside T_EX, WEB and METAFONT, are simply ignored. Ω and $\mathcal{N}\mathcal{T}\mathcal{S}$ tried to create a new starting point, but perhaps they used an abstract-to-concrete approach which led to overwhelming complexity.

LuaT_EX was something different. The pragmatic approach of “implementation-over-specification”, the incremental update cycle (always release a runnable program, even if incomplete or with known bugs) and the “extend (not replace!) the capabilities of T_EX” paradigms were the keys to avoiding death due to complexity. First, the original source was completely rewritten in CWEB, translating the original WEB source, gaining the ability to feasibly use modern external libraries to deal with Unicode and OpenType. Second, the introduction of Lua *in addition to* T_EX for typesetting documents. Initially Lua was a glue language, a sort of “hub” to connect the T_EX core with the other libraries (the PDF backend, the OpenType loader, the Unicode module, the METAPOST library) but gradually, starting as an extension of `\scantokens`, this has changed. Lua interacts with T_EX as a companion language which lives, being procedural, in a space orthogonal to the traditional macro language.

The potential of this orthogonal pair is still unfolding: at the beginning, Lua was used as an input adapter, then its dynamic loading feature was explored, with the SWIGLIB project, through work on how to extend the features of LuaT_EX *at runtime*. Suddenly LuaT_EX can become a graphics converter program, or a tool for number theory, or a PostScript interpreter, or load at runtime a different text shaper — and the latest release pushes the dynamic loading feature forward, avoiding the compilation of a separate wrapper module. Again, this was an evolution (or “extension”) of the `\write18` macro — with the prominent difference that calling an external program is many times slower than calling a function (of course it must be compared with the time required to typeset the document). This opens a new

perspective: T_EX not only *to write about* math but *to do* math.

Lua interacts also with the T_EX internals. With the `nodelib` module the procedural nature of Lua sheds new light on some complex T_EX mechanisms and, in interacting with the PDF backend or with the font loader, opens new ways to do old things or even make new ones possible. For example, ConT_EXt MKiV was the first format to produce PDF/A-2a — thus revealing the need for a widespread and freely available validator — and exporting a faithful copy of the PDF in XML from a T_EX source is now more a CSS issue than anything else. The integration of METAPOST into LuaT_EX leads, on the path of the virtual fonts, to *artificial* fonts: the font can be created directly by T_EX injecting a METAPOST outline (as well as a bitmap, perhaps from METAFONT). For the first time T_EX gets back the control of the alphabet, even if it’s outside the mainstream of OpenType — but still inside PDF. And again, by writing a new font loader in Lua, it is possible to manage color fonts and even variable fonts, reaching, probably after many years, a new breakthrough: T_EX goes beyond the known tools, being the only one (at the present moment) able to produce a valid PDF with this new font technology — which, it must be said, looks *very* similar to METAFONT.

But there is also the other side of the coin: the reference format is now PDF, not DVI.

4 MFLua

As seen in the previous section, METAFONT was quickly considered outdated — again a consequence of the global communication: the rising rate of document exchange led to consuming more data on video screens than on printed paper, and the antialiasing technology of PDF viewers, given the already existing outline format, was not suitable for handling bitmaps. But, as described in *The METAFONTbook*, METAFONT internally uses outlines, and these are clearly written to the log file with `tracingall`.

In the light of the above, the use of Lua to manage these outlines looks like a natural step, but there are fundamental differences. It should never be forgotten that these are carefully designed math programs and being in line with the time, talking of math, is a double edged sword. The experience of METAPOST, with the translation of the original WEB source code to CWEB, shows that when the math is tightly coupled with the implementation and the semantics of the program is complex, bug-free translations come at a price — on the other side, the four different numeric modes of METAPOST (scaled,

double, decimal and binary) was another area that deserved to be explored. METAFONT doesn't need to be modernized, scaled numerics are not “showing their age”, nor is the concept of the pen outdated: in short, there is no need to translate the METAFONT WEB source code.

The role of Lua then is simply to collect enough data from the METAFONT state (in practice, outlines and bitmaps), store them into tables, and let the user manage these tables. And this can be done by merely adding a few procedures in the original source code, by means of a traditional change file, and, as with LuaTeX, the two interpreters can talk between themselves by mean of `scantokens`. In this sense, MFLua started as METAFONT plus a logging facility.

There is another key difference between LuaTeX and MFLua: in the latter, Lua is not really orthogonal to METAFONT. This was clear after the first use case, the natural one: take a METAFONT font and produce an OpenType version. ConcreteOT, an OpenType proof-of-concept font developed from Concrete Roman, shows that the design of the font must consider the outlines as output right from the beginning: Lua is not of great help to elaborate the outlines *after* the bitmap is drawn; they are too closely tailored to the image. It's only while METAFONT is doing its job — producing clean outlines — that Lua can add value: the `sourcecode-regular` presentation at the meeting shows that the natural role of Lua is the *backend*, i.e. translating the now abstract METAFONT code into a font instance.

It's now possible to have an SVG font, or ttx, and nothing prevents us from having FontForge output, or OTF directly: it's only a matter of having a clear

specification. So, suddenly MFLua puts METAFONT back in the game of font design: it took only a few days to modify the ttx backend and make from `sourcecode-regular` a proof-of-concept variable font.

5 Today's challenges

It may seem that “just adding a scripting language” is the solution, and yes *it is* a solution, or better a counter-measure, but only to the pressure of the global communication. Today challenges require fast answers, which are better managed by loading code at runtime, avoiding hardcoded solutions.

On the other side, the ongoing mathematisation of society demands a stronger and stronger grounding in math. What TeX and METAFONT show is that, in the long run, this is more important than the choice of the language of implementation, and, to a lesser extent, of the language implemented. We need to be careful talking about the future of TeX: SQL and COBOL are older than TeX and there are no signs that they are dying — and they do not occupy niche sectors either. At the latest meeting, there was a talk by the GUST team about LuaTeX as a *font editor*; we have seen, perhaps for the first time, that TeX can even go a little further in implementing solutions and that Lua can give a fresh impulse for the development of new strategies for page breaking. Let's take all these as good omens: the future is still to be written.

◇ Luigi Scarso
luigi dot scarso (at) gmail dot com

**Off topic (completely):
Many faces (and types) of beer**

Michał Gasewicz

Abstract

Welcome to the world of beer tasting! Are you ready to learn about the diversity and richness of the oldest alcoholic beverage on Earth? Stop associating beer with the common, cheap, non-absorbing drink which doesn't require a second thought. Smell, taste and have fun! This article summarizes beers included in the BachoTeX 2017 beer tasting with short descriptions.

Beer is associated with tasteless, golden beverages whose only purpose of existence is to make watching football games more pleasant. In theory this is true because 90% of consumed beer is pale lager and it is often difficult to find anything else in some regular grocery stores. Gigantic breweries, more similar to modern factories, produce enormous amounts of beer that does not need to taste good, it only needs to be cheap to produce. Beer is something more — this is one of the oldest beverages in the world and it is made in more than 100 different styles; to be honest, the number of combinations of different kinds of malt, hops from different corners of the world, yeast strains and all kinds of additions is endless.

During the tasting I tried to introduce participants to the wide world of beer. I told a bit about its history and a bit about the newest trends. I chose Polish beers that represent traditional Polish styles and some new-wave variants.

This year we tried the following. All photos are courtesy of Harald König.

**Toruńskie Piernikowe Jasne
(Gingerbread Pale Beer from Toruń)**

Poland is known for its mead and honey beer, because of the clear and intense taste of honey. A nearby town — Toruń — is famous for gingerbread with honey being one very important ingredient. In this beer we have the taste of honey beer with a lot of root spices used in gingerbread. Many visitors to Toruń cannot leave the city without trying this unique gingerbread beer.



**Piwo z Grodziska — edycja
specjalna Piwobranie 2016
(Beer from Grodzisk — Special Edition
2016)**

Grodziskie is the only style of beer coming from Poland (other styles are historical beers with unknown recipes). This is a light, low-alcohol, wheat beer, but very unusual, because it is made with the usage of wheat malt smoked with oak wood. Bottle conditioning caused inconsistent overcarbonation — that is the reason to call this beer Polish champagne. The brewery in Grodzisk was closed in the 1990s, but craft breweries started to brew this style and the original brewery was reopened a few years ago. This special edition of grodziskie beer is enhanced by the addition of Earl Grey tea.



Fruit Wheat — Grand Champion 2016

This beer was brewed in Browar Zamkowy Cieszyń as the winning recipe in the biggest homebrew competition in Poland. Judges chose this beer brewed by Piotr Machowicz as the best one from around 350 entries. Usually wheat beer is refreshing by itself and in this case huge amounts of dry frozen strawberries were added. Fruits enhanced the aroma and taste in a very pleasant way and gave a bit of acidity. It is a great thirst quencher for hot summer days.



ART 9 — Oatmeal Hoptart

Berliner Weisse is very light, low-alcohol, sour wheat beer coming from Berlin. The Stu Mostów brewery from Wrocław in collaboration with Bristol Brewing from Colorado made beer which is at the same time slightly sour and very smooth from oat flakes and has a fruity aroma from aromatic American and Australian hops varieties. This combination was greatly appreciated by beer enthusiasts, causing this brew to skyrocket in the rankings.



Vermont IPA

India Pale Ale from Vermont (or New England IPA, or Northeast IPA) is a new trend in new-wave brewing. In this style, the focus is on getting a fruity aroma from aromatic American hops and perfect drinkability. On the other hand, bitterness does not have to be as intense as in a typical IPA. The use of a special yeast strain and no filtration cause such beers to become very hazy and the colour often resembles juice.



Imperator Bałtycki

Polish breweries are specialists in brewing Baltic porters and they are very often awardees at international contests. Baltic porters are appreciated for the rich and complex chocolate and caramel sweetness developing into dark fruit sweetness. Baltic Emperor is brewed as an imperial, even stronger, version of Baltic porter which has even more richness, thickness and deliciousness. No wonder this beer is considered one of the best Polish beers.



Miss Big Foot

Brewed by the Birbant brewery for the Piwna Stopa (Beer Foot) pub with the pub owners' recipe. Smoking is a very important process in Polish cuisine. In this stronger version of stout they used traditional Polish smoked plums, muscovado sugar and vanilla. The addition of smoked plums gives a brand new kind of smokiness, different than smoked malt, and it also brings some fruit notes. Even though outside Poland smoked plums are not popular, beers with this ingredient gain top rankings worldwide.



Beers presented during this tasting are only a few representatives of all beer styles. I encourage you to taste beer on your own. You only need to find an unusual beer and think about its aroma and taste. You can take some notes or compare your feelings with some reviews. This will be a very good first step into the wide world of beer.

◇ Michał Gasewicz
Toruń
Poland
genn (at) umk dot pl



History of accidentals in music*

Jean-Michel HUFFLEN

Abstract

Signs used throughout music scores in order to change a note's pitch slightly are well-known: the sharp (\sharp) to raise it, the flat (\flat) to lower it, and the natural (\natural) to restore it to its normal pitch. First we give the etymology of these names, then we show that the conventions used in the past are very different from those used nowadays, especially if we consider *double* accidentals. In addition, accidentals present interesting typographic problems because there are several conventions with precise meanings: accidentals left to the note (with or without parentheses) or upwards.

Accidental signs used in classical or popular music are included in Unicode, as have some signs used for micro-intervals, such as quarter tones. From our point of view, the selection made by Unicode is debatable. In order to clarify the situation, we show the accidentals mainly used for micro-intervals in *musique orientale* and contemporary music. This article requires only basic knowledge in reading music scores. *Keywords* History, accidentals' origin, putting accidentals, music typography, micro-intervals, Unicode.

Streszczenie

Znaki używane w zapisie nutowym w celu niewielkiej zmiany wysokości dźwięku są powszechnie znane: krzyżyk (\sharp) do podniesienia, bemol (\flat) do obniżenia i kasownik (\natural) do anulowania innych znaków. Najpierw zostanie omówiona etymologia tych nazw, następnie zaś to, że konwencje używane w przeszłości bardzo się różniły od obecnych, zwłaszcza jeśli weźmie się pod uwagę *podwójne* znaki chromatyczne. Na dodatek znaki chromatyczne sprawiają ciekawe problemy typograficzne, gdyż istnieje kilka konwencji przypisywania im precyzyjnego znaczenia: znaki chromatyczne na lewo od nuty (w nawiasach bądź bez nich) albo podniesione.

Do unikodu włączono znaki chromatyczne używane w muzyce klasycznej i popularnej, jak też niektóre znaki używane do oznaczania mikrointerwałów, takich jak ćwiartki dźwięków. Z naszego punktu widzenia można dyskutować z tym doborem znaków. W celu wyjaśnienia sytuacji zostaną pokazane znaki chromatyczne używane do oznaczania mikrointerwałów w muzyce orientalnej i współczesnej. Do zrozumienia tej prezentacji wystarczy jedynie podstawowa znajomość zapisu nutowego.

Słowa kluczowe Historia, pochodzenie znaków chromatycznych, umieszczanie znaków chromatycznych, typografia muzyczna, mikrointerwały, Unicode.

* Polish title: *Historia znaków chromatycznych w muzyce.*

Introduction

Within music scores, an **accidental** is a sign usually put before a note figure, signalling a slight change of its pitch. Let us consider the keys of a piano, *white* keys are denoted by letters from 'A' to 'G',¹ whereas *black* keys are reached by means of accidentals. These signs are well-known: the **sharp** (\sharp) raises a note's pitch by a semitone,² the **flat** (\flat) lowers it by a semitone, and the **natural** (\natural) restores it to its normal pitch. In L^AT_EX, these signs are respectively produced by the commands `\sharp`, `\flat`, and `\natural` in math mode.

From a typographic point of view, writing accidentals in music scores obeys rules that are not always precisely known. Besides, these rules have evolved over time. In addition, there are many other signs related to accidentals, some of which are included in Unicode [32]. We personally think that the selection made by Unicode is quite debatable. So the purpose of this article is to give an overview of the conventions related to accidentals. We discuss the etymology of these signs in the first section, then the rules are given in Section 2. Section 3 is devoted to some remarks about characters, software and encodings' organisation. In Section 4, we study accidentals for *micro-intervals*—smaller than semitones—used in *musique orientale*³ and also in contemporary music. Reading this article requires only basic knowledge of music scores. Readers interested in precise definitions of music terminology can consult [16]. More information about historical points can be found in [1], and also in [22], but in French. To help readers situate musicians cited throughout the article, we give their dates, either in the text or in the bibliography.

1 Origins

In the early Middle Ages, three *modes* were known. These modes cannot be viewed as 'modern' scales

¹ In the English-speaking world. Some other countries—including France, Spain, Italy and Russia—use names coined by Guido d'Arezzo (991 or 992–after 1033) after the verses of a Latin hymn in honour of John the Baptist: *ut ré, mi, fa, sol, la*, for C, D, E, F, G, A. Later, the *si* name was added for B in the xvith century, and *ut* was renamed to *do* in the xvith century; the origins of these last two names are controversial.

² The interval between two notes played by adjacent notes on a piano, regardless of colour.

³ This French term encompasses Andalusian classical music, beginning in the Emirate of Cordoba in the ixth century, and including music of countries of North Africa, Near and Middle East. In English, the word 'oriental' is most commonly used to refer to the Far East, whereas 'orientale' in French is often applied to the Near or Middle East. That is why we use the French expression.



Figure 1: Nicolas BERNIER (1664–1734), *Diane*, excerpt [6, p. 3].

including 7 *degrees*, since they were based on *hexachords*, that is, 6-note groups. They were:

- the *natural hexachord*⁴ (*hexachordum naturale*): C, D, E, F, G, A;
- the *soft hexachord* (*hexachordum molle*): F, G, A, B \flat , C, D;
- the *hard hexachord* (*hexachordum durum*): G, A, B, C, D, E.

Thus, only the B note could be flattened.⁵ The ‘ \flat ’ sign derives from a *round b* (*b rotundum*)—originally written ‘ \flat ’—in connection with the soft hexachord; the French name *bémol* comes from medieval French *bé mol*⁶ for *soft b*. The signs ‘ \natural ’ and ‘ \sharp ’ derive from a *square b* (*b quadratum*)—originally written ‘ \natural ’—in connection with the hard hexachord; the French name *bécarre* (\natural) comes from medieval French *bé carré*.⁷ The natural and sharp signs are both derived from this ‘square b’ using two different ways to extend sides of this square. The French name *dièse* (for ‘sharp’) comes from the Latin word *diesis*: initially, this word denoted a quarter tone interval in ancient music; at the Roman Empire’s end, it was used for a semitone interval. To conclude the etymology questions, it seems that the English name ‘sharp’ (resp. ‘flat’) comes from ‘so high (resp. low) as to be out of tune’.

Let us go back to the natural and sharp signs. The difference between them was vague for a long period of time. Often, ancient scores used sharps in order to raise a note lowered previously. For such a sharp sign, as shown in Fig. 1, a natural sign would

⁴ As an example, the Hymn to John the Baptist (cf. *supra*) is written using this mode. Guido d’Arezzo was unable to use this piece to give a name to the B note, since it does not appear within this natural hexachord.

⁵ In addition, let us mention that at this time, scribes were hesitating over whether ‘B’ denotes B \natural or B \flat . This ambiguity is removed by the German notation system, still in use today: ‘B’ stands for B \flat , ‘H’ for B \natural . This system is also in use in Central and Eastern Europe, and in Scandinavia. Also, hexachords’ names have survived in the German words for the modes of ‘classical’ tonal scales: *moll* for *minor*, *Dur* for *Major*.

⁶ In modern French: *bé mou*.

⁷ In modern French: *bé carré*.

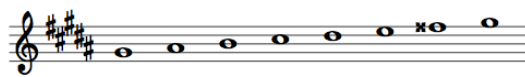


Figure 2: G \sharp minor scale.

be substituted in modern notation. In fact, these two signs were clearly separated only at the beginning of the classical era. During the pre-classical period, modern scales—C Major, F Major, G Major, etc.—were progressively emerging, from the xvth to the xviiith century. The first ‘actual’ sharp in use was F \sharp , then the second flat—E \flat —came, and so on: C \sharp , A \flat , G \sharp , D \flat , D \sharp , G \flat , A \sharp , C \flat , E \sharp , F \flat , B \sharp . The notion of *enharmonic intervals* appeared, e.g., A \flat and G \sharp are the same note, although they have different names.⁸

Double accidentals, such as *double sharp* ($\sharp\sharp$) and *double flat* ($\flat\flat$), were introduced in the xviiith century.⁹ They raise or lower a note’s pitch by *two* semitones. Initially, the goal was to express *sensible notes*¹⁰ for some minor scales, e.g., F $\sharp\sharp$ is the sensible note of the G \sharp minor scale, as shown in Fig. 2. The $\sharp\sharp$ sign was introduced before the $\flat\flat$ sign,¹¹ whereas the \flat sign was coined before the \sharp sign.

2 Rules

Most of the rules we give hereafter are well-known for musicians. We examine them from a typographical point of view.

2.1 Restoring accidentals

Often, accidentals are implicit in scores from the Middle Ages. In other words, an accidental may be omitted whenever it was obvious for the musicians of the time, who were used to restoring them.¹² When modern music gravers restore such implicit accidentals,

⁸ In fact, that is true only for instruments based on twelve-tone *equal temperament*, e.g., a piano in today’s standard tuning. Dealing with other temperaments is outside this article’s scope. In addition, let us remark that such enharmonic notes are not limited to those which are played using black keys of a piano: as a counter-example, E \sharp and F \natural are enharmonic, too.

⁹ [9, §45] gives another—old—notation for the double sharp: ‘ $\sharp\sharp$ ’, surrounded by four dots. We personally have never seen this sign in any score, even very old ones.

¹⁰ A *sensible note* is located just below a scale’s basic note and is *attracted* by it, so a sensible note must be at a distance of a semitone. Only modes—major and minor—with sensible notes are used in classical harmony.

¹¹ The $\flat\flat$ sign does not belong to any ‘classical’ scale; it has been introduced ‘symmetrically’ to the $\sharp\sharp$ sign.

¹² This may seem surprising, but analogously, many jazz scores today let some details remain implicit: for example, rhythms in jazz scores are often simplified forms in comparison to what jazz musicians actually play. Chords in jazz scores are often simplified, too.

Figure 3: P. Attaignant [2, no. 19]: *Basse dance*, start.

Figure 4: G. Gershwin [12, p. 5]: 1st movement, 7 bars after §2.

they have to put them *above* or *below* the corresponding note head — not at the left — as shown in Fig. 3. Such an accidental only applies to the corresponding note, not to the following ones.¹³

2.2 More double accidentals

As mentioned in §1, a sharp sign sometimes had a relative effect in ancient scores, since it could be used to raise a note already lowered. This view — which appears strange nowadays — has survived through the ages in the use of the double signs $\natural\flat$, $\natural\sharp$, and $\natural\natural$ [7, §82]. For example, if a double flattened note is followed by the same note with a ‘single’ flat sign, it

¹³ Thus the repetition of the \flat sign for two adjacent notes within the bass voice of Fig. 3. Let us also notice a kind of *polytonality*, usual in the music of this time. Contrary to what many people think, polytonality and polymodality were not introduced in the XXth century.

Figure 5: J.-M. Hufflen [15], bar 132.

was incorrect to insert the \flat sign. Using $\natural\flat$ means that the first \natural sign cancels one of the two semitones of $\flat\flat$, so the note can now be flattened. As shown in Fig. 4, the same sign is sometimes used after a sharpened note, before flattening it. Symmetrically, the use of $\natural\sharp$ is analogous to that of $\natural\flat$. The $\natural\natural$ sign means that a note is restored at its normal pitch after the use of \sharp or \flat . Nowadays this complicated rule — it introduces ‘compound’ signs that are actually useless — becomes more and more obsolete, and a ‘simple’ accidental sign always denotes its original effect, regardless of accidentals used before.

2.3 Accidentals and bars

In most music scores [7, §79], an accidental sign takes effect for the following note and any repetition of that note at the same octave and in the same bar, unless cancelled by another accidental. If a note is tied into the following bar, the accidental takes effect just until this tie’s end. If a system of multiple staves is used, an accidental used on one staff never affects others. This convention gradually emerged over the XVIIIth century. Before, accidentals only applied to immediately repeated notes or short groups for which it was obvious that the accidental should go on.¹⁴ In some scores from the XIXth century, accidentals apply to the same notes in the same bar, regardless of octaves. The standard rules can be observed in the example given in Fig. 5. The \natural sign used with the circled A note at the staff for the piano’s right hand avoids any ambiguity but is formally useless: first, this note is not at the same octave as the $A\flat$ at the immediate left, and second the $A\flat$ at the same octave and inside the same bar does not belong to the same staff.

Some contemporary composers use accidentals regardless of bars, that is, accidentals are not continued until a bar’s end. The last scores of Hans-Werner Henze (1926–2012) are examples where accidentals apply only to one note or immediately re-

¹⁴ Let us recall that at this time, some accidentals were implicit, supplied by interpreters.

Figure 6: W. Lutosławski [21, p. 30], before Section 72.

peated notes. As other — non-limitative — examples, accidentals apply to only one note in the last scores of Witold Lutosławski and Henryk Mikołaj Górecki (1933–2010). W. Lutosławski uses repeated notes without heads when an accidental applies to some adjacent notes — as shown in Fig. 6 — whereas H. M. Górecki explicitly repeats the same accidental before each repeated note.¹⁵

When in doubt, music composers and publishers sometimes put extra accidentals down, even if they are ‘formally’ useless. Such accidentals are called *courtesy* or *cautionary*. For example, if a note has an accidental within a bar, such a courtesy accidental within the next bar allows an interpreter not to be confused about this note’s pitch. In particular, courtesy accidentals should be used for notes with lengthy ties at a new system’s beginning. Courtesy accidentals should be surrounded by parentheses, but in practice, this convention is often not followed and courtesy accidentals are written as ‘actual’ accidentals.¹⁶

2.4 Key signatures

A *key signature* is a set of sharps or flats associated with a scale. The placement of accidentals on key signatures obeys precise rules about the succession of sharps or flats and their placement, depending on the clef used,¹⁷ but now music software programs generating scores do that correctly. So we just give some examples in Fig. 7:

- (a), (c), (f) use the well-known *treble* clef,
- (b) uses the *tenor* clef, devoted to the high range of instruments such as the bassoon, cello, or trombone,

¹⁵ Let us remark that ♯ signs become useless if such conventions — that accidentals apply to one note only — are used with empty key signature scores (cf. §2.4).

¹⁶ ... as we did for the circled note in Fig. 5. The score [15] has been typeset with MuseScore [24].

¹⁷ Different key signatures may be used in *musique orientale* and popular music, but this is outside this article’s scope.

Figure 7: Key signatures.

Figure 8: J.-M. Hufflein [15], bars 143–144.

- (d) uses the *mezzo-soprano* clef and (e) the *soprano* clef.¹⁸

In (e) and (f), we can remark that if a key signature changes throughout a piece, the accidentals absent in the new key signature should be cancelled by ♯ signs, as shown in Fig. 7-(e,f). We can observe that this rule is less used nowadays, so ♯ signs in a key signature change are only used when the new signature is empty, as shown in Fig. 8.

Let us mention that in pre-classical music, some key signatures seem to be incorrect, especially for minor scales. For example, a piece that appears to be in G minor uses the D minor key signature, as shown in Fig. 3. More generally, minor scales sometimes use a key signature with one flat fewer, or one

¹⁸ Nowadays, these two keys are no longer in frequent use. The *soprano* key is still used scholastically (in harmony exercises), while the *mezzo-soprano* key is only used for transposition purposes. To give an idea about relationships among these keys, let us mention that examples (a–e) begin with notes at the same pitch.

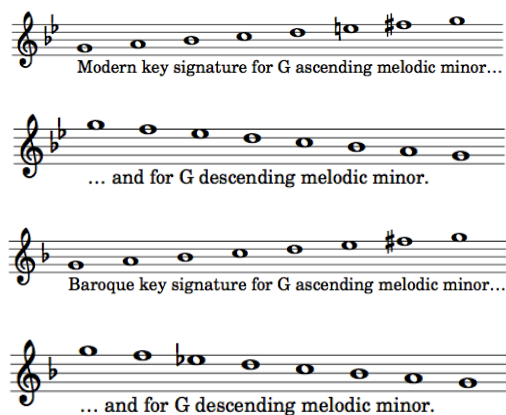


Figure 9: Modern and baroque key signatures for melodic minor scales.

more sharp, or an empty signature for D minor. More examples—including ‘incomplete’ key signatures for major scales—can be found in [8]. A complete explanation is given in [11]. We can demystify this *modus operandi* about minor scales:¹⁹ it allows *ascending melodic* minor scales²⁰ to be specified with as few accidentals as possible, with respect to the key signature’s signs, as shown in Fig. 9 for the ascending melodic scale of G minor.

3 Intermezzi

The previous sections are based on ‘purely’ musical material; now, we give some details related to computer science and music software.

3.1 ‘#’ vs ‘#’

Often the characters ‘#’ (U+266F) and ‘#’ (U+0023) are confused. They are graphically different since the latter is a combination of level horizontal strokes and right-tilting vertical ones, whereas the former is based on vertical strokes and slanted horizontal ones. The ‘#’ character is the *number sign* and is present on standard keyboards, so it often replaces ‘#’ in practice. A good example is the name of the C# programming language [23], written with a number sign but pronounced ‘C sharp’.

¹⁹ The explanation of this *modus operandi* about major scales would need the introduction of *ancient modes*, out of this article’s scope.

²⁰ In this scale, the third is minor, the sixth and seventh are major [7, §151.II]. As pictured in Fig. 9, the sixth and seventh are minor within a *descending melodic* minor scale. In classical harmony, such melodic minor scales should be used only for a *melody*, the successive chords harmonising a melody are based on the *harmonic* minor scale, as defined in Fig. 2 for the G# minor scale.

```
\version "2.18.0"
\score {
  \new Staff {
    \clef "treble" \time 3/4
    \accidentalStyle Score.default
    r8 bes'8 fes'4. ges'8 | ees'2 f'4 |
    r8 a'8[ d'8. cis'16] g'4 |
  }
  \layout {}
}
```

Figure 10: Example using LilyPond.

3.2 Accidentals in Unicode

In Unicode [32], the ‘basic’ accidentals —**b**, **♭**, **#**—are encoded in the *Miscellaneous Symbol Block* (U+266D, U+266E, U+266F). The other accidentals are encoded in the *Musical Symbol Block*, from the code-point U+1D12A to U+1D133. The first two code-points of this range are for **✖** and **♭**.

3.3 Coding more musical signs

Presently, Unicode has retained only a few musical signs from the multitude of signs used through the ages. Let us mention the SMuFL²¹ project. This is a specification providing a standard way of mapping musical symbols required by conventional music notation into the Private Use Area in Unicode’s Basic Multilingual Plane (U+E000–U+F8FF). In particular, all the symbols introduced in the following have been mapped. In addition, several music software packages—e.g., MuseScore [24]—use this encoding.

3.4 Accidentals and LilyPond

The GNU²² LilyPond music engraver [29] provides a character-based language to specify the rhythm and pitch of a note—a short introduction and example are given in [14]. When LilyPond compiles a piece’s specification into a music score, it uses *accidental styles* to decide whether or how accidentals actually appear. These accidental styles—w.r.t. LilyPond’s terminology, they may be viewed as *strategies*—include [20]:

default accidentals are inserted or are implicit, according to common practice,

modern some courtesy accidentals, without parentheses, are added to avoid ambiguity,

neo-modern accidentals are repeated if the same note appears again in the same bar, unless this note is immediately repeated,

²¹ **Standard Music Font Layout**. See <http://smufl.org> for more technical details.

²² Recursive acronym: **GNU’s Not UNIX**.

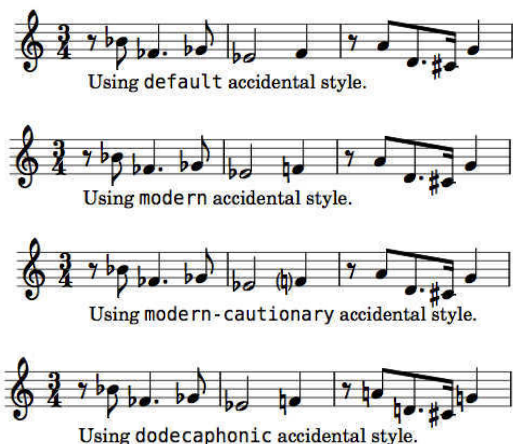


Figure 11: Examples of accidental styles for Fig. 10.

dodecapronic every note gets an accidental sign, including natural signs,

forget accidentals are not remembered at all.

The styles:

modern-cautionary, neo-modern-cautionary, teaching

are respectively similar to:

modern, neo-modern, dodecapronic,

except extra accidentals are surrounded by parentheses, as cautionary accidentals. As a very short example, Fig. 10 gives a LilyPond specification of the *Tema*'s beginning of Arnold Schoenberg's *Variations for Orchestra* [28, p. 7, bars 34–36]. For each note, its accidental is given as ‘...is’ for \sharp , ‘...es’ for \flat , and no suffix for \natural . Results according to some of LilyPond's accidental styles are given in Fig. 11. Other customisation features are available: for example, allowing the use of double accidentals such as $\flat\flat$, $\sharp\sharp$, and $\sharp\flat$; inserting or removing the accidental of a tied note at the start of a new system, and more.

In addition, let us mention that the *lilyglyphs* package²³ allows music glyphs pictured by LilyPond to be handled by means of T_EX-like commands, provided that X_LLaT_EX [18] or LuaLaT_EX [13] are used.²⁴ We have personally added some new commands, but all the glyphs for ‘basic’ accidentals used throughout this article come from this package.

4 Accidentals for micro-intervals

4.1 What are micro-intervals?

Micro-intervals are *smaller* than semitones. In the

²³ Included in the T_EX Live distribution.

²⁴ All these commands—including redefinitions of the commands `\flat`, `\natural`, and `\sharp`—are to be used in text mode.



Figure 12: A. Berg [4, p. 53]: bars 274–277.

early XXth century, composers started to use such intervals, in particular, *quarter tones*.²⁵ At this time, these intervals were specified by means of *ad hoc* notations: for example, Alban Berg in his *Chamber Concerto* (premiered in 1927, cf. Fig. 12). A later example is given by the last movement of Béla Bartók's *Sonata for Solo Violin*²⁶ [3], written in 1944. Some composers built totally new organisations of sounds and intervals. Two historical and representative examples are Alois Hába (1893–1973) and Ivan Wyschnegradsky (cf. [36]); both went in this direction after the First World War. In addition, quarter tones are used in some popular music, especially *musique orientale*, but this music is not really based on quarter tones, in the sense that classical music may be viewed as based on semitones (for example, in classical scales, the interval between B and C). In fact, *musique orientale* does not deal with quarter tones between adjacent notes: in addition to tones and semitones, it uses *great tones* (5/4 tone), and *small tones* (3/4 tone). Expressing this organisation by means of our occidental notation causes quarter tone notations to appear, but this interval does not exist in *musique orientale*.

Other divisions of tones have existed, too. For example, Maurice Ohana (1913–1992) utilised a division of tones by three,²⁷ and I. Wyschnegradsky divided a tone into 12 parts, and more (!) [34]. As mentioned in §3.2, some accidentals for micro-intervals

²⁵ [6] explains that the evolution of music, throughout the ages, has developed chords incorporating more and more sounds, according to the successive harmonic notes within harmonic series (first the octave, then the fifth, before the third, and so on). In particular, the introduction of micro-intervals at this time can be explained by this theory [6, p. 77–79].

²⁶ This sonata was composed for violinist Yehudi Menuhin (1916–1999). In a letter dated April 21, 1944, B. Bartók wrote that ‘quarter tone steps may be eliminated and replaced by alternative versions.’ He would have liked ‘to hear both played versions, and then decide if it is worth while to use these 1/4 tones.’ Unfortunately, he never heard this work before his death, and the alternatives, only retained within the Menuhin edition, are often played instead.

²⁷ According to his notation, raising a note by a third tone (resp. 2/3 tone) is signalled by ‘/’ (resp. ‘//’) to the left of the note head. Third tones were also used in the last movement of B. Bartók's *Sonata for Solo Violin* [3, bars 58–62], although an alternative version, retained by Y. Menuhin, avoids them (cf. footnote 26).

Figure 13: A. Schnittke [27, p. 2]: 1st movement, Section 3.

belong to the Unicode encoding, but they do not specify the more frequent and more precise intervals.

4.2 Exact micro-intervals

As a semitone is the *exact* division of tones by two, a quarter tone is the exact division of semitones by two, that is, this division yields something *precise*.²⁸ If quarter tones are used throughout a score, some explanations make the signs' meaning precise. Even if there is no 'official' standardisation, the more frequently used signs are ♯₄ for a *half sharp*, raising a note by a quarter tone,²⁹ and ♯₃ for a *sharp and half*, raising a note by three quarter tones. In particular, these notations are used by Iannis Xenakis [37]. Alternative notations exist: ♯_{1/2} and ♯_{3/4}.

A *half flat*, lowering a note by a quarter tone is often denoted by ♭_{1/2}, with a *flat and half*, lowering a note by three quarter tones, by ♭_{3/4}, alternative notations being ♭_{1/2} and ♭_{3/4}.

As examples, we can see the notations used by Alfred Schnittke in Figs. 13 and 14. Those used by Krzysztof Penderecki, Ivan Wyschnegradsky, and Witold Lutosławski are shown in Figs. 15–17. In these scores, the glyphs for half sharps and sharps and a flat are quite similar. Concerning half flats and flats and a half, A. Schnittke and W. Lutosławski use 'open' glyphs for half flats. K. Penderecki uses black-filled flats for half flats and ♭ for flats and a half. More details about these notations and variants can be found in [17]. Let us observe that none of them have been

²⁸ We assume an equal temperament (cf. footnote 8, on p. 148). In any case, unequal temperaments complicate the definition of quarter tones, but still lead to precise results.

²⁹ As mentioned in §3.3, all the signs introduced in the present section have been included in the mapping done as part of the SMuFL project. For example, the code point of the ♯₄ sign is U+E282.

Figure 14: A. Schnittke [27, p. 42]: 3rd movement, Section 7.

included into Unicode. The glyphs defined by Unicode at present are ♯₄ (U+1D132) and ♯₃ (U+1D133): we have *never* seen them in *any* score.

We end this short study of quarter tones with giving two examples of modes within *musique orientale* in Fig. 18: *rast* and *soznak* (cf. [5, p. 2] & [10, p. 38]). In considering the *rast* mode, we can notice a small tone between the 2nd and 3rd degrees and a great tone between the 3rd and 4th degree. If you are interested in such modes, you can find more details in [5, 10].

4.3 Approximate micro-intervals

Table 1 lists signs derived from the classical accidentals and expressing *indeterminate* pitch [33, p. 138–139]. We include the corresponding code-point for those included in Unicode, preceded by '★' if the Unicode's glyph is slightly different.³⁰ An up (resp. down) arrow means that the note is to be slightly raised (resp. lowered). For example, if an interpreter plays C♯₄ (half sharp) for C♯₃, that is correct but not required; the notation merely expresses that this note must be located between C♯_{1/2} and than C♯. In addition, it *should be* closer to C♯₃ than C♯_{1/2}. An up-down arrow means 'around' the corresponding note, e.g., C♯_{1/2} may be slightly higher or lower than C♯_{1/2}. We can imagine only with difficulty such a notation when several instrumentalists play the same part, e.g., all the violins of a symphonic orchestra, but it has been used in chamber music, an example being given in Fig. 19.

Other notations expressing the same behaviour come from the *breaks in the voice* in Byzantine chant: ♭_{1/2}, ♯_{1/2}, ♯_{3/4}, as shown in Fig. 20.

5 Conclusion

Handling accidentals in music scores is error-prone,

³⁰ The arrow of the Unicode character U+1D131 is at the bottom right corner, whereas the 'actual' sign's arrow is at the bottom left corner, as shown in Table 1.



Figure 15: K. Penderecki [26, p. 10]: Section 9.



Figure 16: I. Wyschnegradsky [35, p. 34]: Prelude X, bars 24-27.



Figure 17: W. Lutosławski [21, p. 4]: *cadenza*.

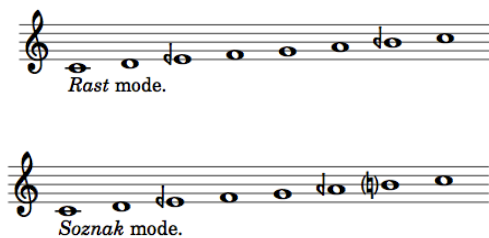


Figure 18: Two examples of modes within *musique orientale*.

↑	(U+1D12C)	↕	-
↓	(U+1D12D)	↕	(U+1D130)
↕	-	↕	* (U+1D131)
↕	(U+1D12E)	↕	-
↕	(U+1D12F)		

Table 1: Signs for approximate micro-intervals.

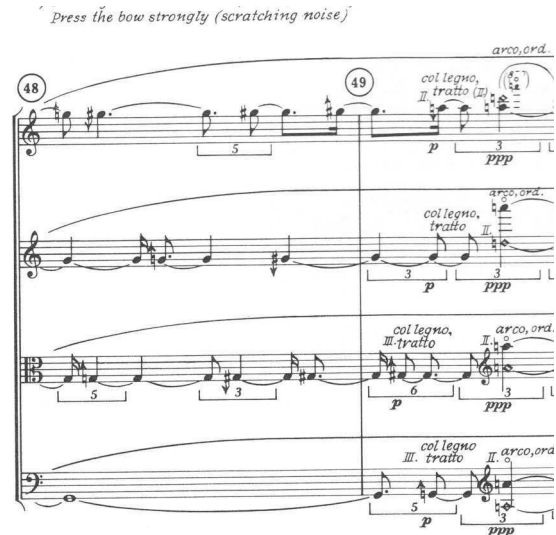


Figure 19: G. Ligeti [19, p. 16]: 2nd movement, bars 48-49.

especially for older scores. Often musicologists doubt their interpretation. However this system has been in use for several centuries, and some attempts to replace them — e.g., [25] — have failed. From a point of view of software generating music scores, LilyPond’s *modus operandi* seems to us to be good, in the sense that we can precisely customise a score’s final look. Such advanced functions do not exist in MusiX_{TEX} [30] or MuseScore [24]. With respect to Unicode, we speak in favour of adding accidental signs for exact quarter tones. Of course, Unicode does not aim to incorporate all new notations in contemporary music, but including these signs may be interesting for typesetting studies about *musique orientale*. As far as we know, most of the references about this topic use signs for exact quarter tones, not signs for approximate ones, as defined in Unicode.

6 Acknowledgements

Many thanks to the Polish translators: Ryszard Kubiak for the abstract and Jerzy B. Ludwichowski for

Figure 20: J. Tavener [31, p. 29]: Section N, 2nd movement, beginning.

the keywords. Thanks to GUTenberg, the French-speaking T_EX Users Group, that offered me a grant for participating in this TUG@BachOT_EX 2017 conference. I am also grateful to this definitive version's proofreaders: Karl Berry and Barbara Beeton. Last but not at least, Brian Bartling and Gail Berry provided valuable advice about some points related to musical terminology.

References

- [1] Denis ARNOLD, ed.: *The New Oxford Companion to Music*. Oxford University Press. 1983.
- [2] Pierre ATTAINGNANT (~1494–1551 ou 1552) : *Danceries à 4 parties (second livre, 1547)*. Heugel & C^{ie}. Édition par Raymond MEYLAN. 1993.
- [3] Béla BARTÓK (1881–1945): *Sonata for Solo Violin*. Boosey & Hawkes. Urtext edition. 1994.
- [4] Alban BERG (1885–1935): *Kammerkonzert*, Bd. 423. Philharmonia. 1925.
- [5] Elie BOHBOT : *Abrégé théorique et pratique de musique orientale traditionnelle à 1/4 de ton. Initiation des musiciens occidentaux au 1/4 de ton*. Gérard Billaudot, éditeur. 1983.
- [6] Jacques CHAILLEY (1910–1999) : *Traité historique d'analyse harmonique*. 2^e édition. Alphonse Leduc. 1977.
- [7] Jacques CHAILLEY et Henri CHALLAN (1910–1977) : *Théorie complète de la musique, 1^{er} volume*. Alphonse Leduc. 1947.
- [8] Arcangelo CORELLI (1653–1713): *Concerti grossi for 2 Violins, Violoncello, Strings and Basso continuo, op. 6/1–12*. Eulenburg, London. 1997.
- [9] Adolphe-Léopold DANHAUSER (1835–1896) : *Théorie de la musique*. Éditions Henry Lemoine, Paris. Édition revue et corrigée par Henri RABAUD. 1929.
- [10] Salah EL MAHDI : *La musique arabe*. Alphonse Leduc, Paris. Juillet 1983.
- [11] Laurent FICHET : *Le langage musical baroque : éléments et structures*. Minerve. 2014.
- [12] George GERSHWIN (1898–1937): *Concerto in F for Piano and Orchestra*, Vol. 1819. Eulenburg, London. 1987.
- [13] Hans HAGEN: “The Luaification of T_EX and ConT_EXt”. In: *Proc. BachOT_EX 2008 Conference*, pp. 114–123. April 2008.
- [14] Jean-Michel HUFFLEN: “A Comparison of MusiX_T_EX and LilyPond”. In: Tomasz PRZECHLEWSKI, Karl BERRY and Jerzy B. LUDWICHOWSKI, eds., *Twenty Years After. Proc. BachOT_EX 2012 Conference*, pp. 103–108. Bachotek, Poland. April 2012.
- [15] Jean-Michel HUFFLEN: “Dijon Concerto, for Trombone, String Orchestra, and Piano”. 2015.
- [16] Arthur JACOBS (1922–1996): *The New Penguin Dictionary of Music*. 4th edition. Penguin Books. 1988.
- [17] Franck JEDRZEJEWSKI : *Dictionnaire des musiques microtonales*. L’Harmattan. 2004.
- [18] Jonathan KEW: “X_EL_AT_EX in T_EX Live and beyond”. *TUGboat*, Vol. 29, no. 1, pp. 146–150. EuroBachOT_EX 2007 proceedings. 2007. <https://tug.org/TUGboat/tb29-1/tb91kew.pdf>.
- [19] György LIGETI (1923–2006): *String Quartet No. 2*, Vol. 6639. B. Schott’s Söhne, Mainz. 1968.
- [20] *LilyPond*. March 2014. <http://www.lilypond.org/doc/v2.18/Documentation/web/index.html>.
- [21] Witold LUTOSŁAWSKI (1913–1994): *Concerto for Cello and Orchestra*. PWM, London. 1971.
- [22] Brigitte MASSIN et Jean MASSIN, rédacteurs en chef : *Histoire de la musique occidentale*. Fayard. Octobre 1985.
- [23] MICROSOFT CORPORATION: *Microsoft C# Specifications*. Microsoft Press. 2001.
- [24] *MuseScore Handbook*. April 2017. <http://musescore.org>.

- [25] Nicolas OBOUHOW (1892–1954) : « L’harmonie totale ». *La revue musicale*, Vol. 290–291, p. 25–70. Août 1972.
- [26] Krzysztof PENDERECKI (1933–): „*Als Jakob erwachte*“, Bd. 6623. B. Schott’s Söhne, Mainz. 1975.
- [27] Alfred Garrievich SCHNITTKE (1934–1998): *Concerto Grosso for Two Violins, Harpsichord (Also Piano) and String Orchestra*, Vol. 488. Philharmonia. 1977.
- [28] Arnold SCHOENBERG (1874–1951): *Variationen für Orchester, op. 31*, Bd. 12 196. Universal Edition. 1956.
- [29] Lambert M. SURHONE, Mariam T. TENNOE and Susan F. HENSSONOW, eds.: *GNU LilyPond*. VDM Verlag Dr. Muller Aktiengesellschaft & Co. KG. September 2010.
- [30] Daniel TAUPIN, Ross MITCHELL and Andreas EGLER: *MusiX_{TEX}. Using T_EX to Write Polyphonic or Instrumental Music. Version T.104*. January 2002. <https://ctan.org/pkg/musixtex>.
- [31] John TAVENER (1944–2013): *The Protecting Veil, for cello and string orchestra*, Vol. 59030. Chester Music, London, UK. 1993.
- [32] THE UNICODE CONSORTIUM: *Unicode 9.0.0*. June 2016. <http://www.unicode.org/versions/Unicode9.0.0/>.
- [33] Hans VOGT, mit Maja BARD, Mathias BIELITZ, Hans-Peter HALLER, Hans-Peter RAISS und Angelus SEIPT: *Neue Musik seit 1945*. 3. Auflage. Philipp Reclam, Stuttgart. 1982.
- [34] Ivan WYSCHNEGRADSKY (1893–1979) : « L’ultrachromatisme et les espaces non octavians ». *La revue musicale*, Vol. 290–291, p. 71–141. Août 1972.
- [35] Ivan WYSCHNEGRADSKY: *24 Preludes in Quarter-Tone System, op. 22*, Vol. 418. Mitrofan Petrovich Belaieff, Frankfurt. 1979.
- [36] Ivan WYSCHNEGRADSKY : *Manuel d’harmonie à quarts de ton*. Éditions Max Eschig, Paris. 1980.
- [37] Iannis XENAKIS (1922–2001) : *Nuits, musique pour 12 voix mixtes*. Éditions Salabert. 1969.

◇ Jean-Michel HUFFLEN
 FEMTO-ST (UMR CNRS 6174)
 & University of Bourgogne
 Franche-Comté
 16, route de Gray
 25030 Besançon Cedex
 France
 jmhuffle (at) femto-st dot fr
<http://members.femto-st.fr/jean-michel-hufflen>

Bookbinding workshop: Making a portfolio

Willi Egger

Material

2	Board	320 × 225 mm
1	Spine cover material (book cover cloth)	350 × 50 mm
1	Corner cover material (book cover cloth)	100 × 100 mm
2	Outside cover paper	350 × 220 mm
1	Inside spine cover paper (black)	313 × 50 mm
2	Inside cover paper (black)	313 × 215 mm
1	Side flap paper (black)	300 × 125 mm
2	Top/bottom flap paper (black)	210 × 125 mm
2	Piece of twill-ribbon	200 mm

Board and paper should in any case be cut such that the grain direction is along the spine of the portfolio. You need bookbinders glue (PVA) for gluing the book cover cloth parts to the boards and for gluing the twill-ribbon in place. You need paste for gluing all paper parts to the portfolio.



Figure 1: The result of participation in the workshop by a happy attendee who wasn't as careful as she should have been to make sure all the pieces were aligned properly. —bb

Instructions

- On each board, make a recess with an incision through the board:

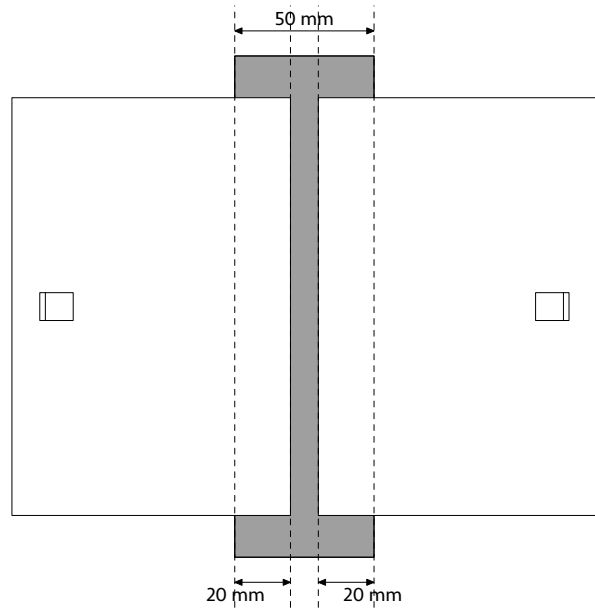


Figure 2: Layout of the recess

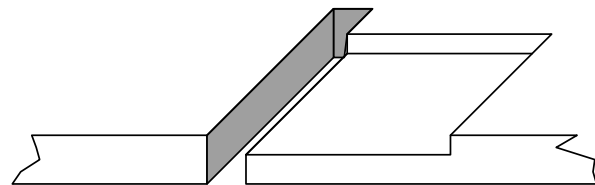


Figure 3: Recess and incision

- Glue the spine cover material and the boards together, distance 10 mm:

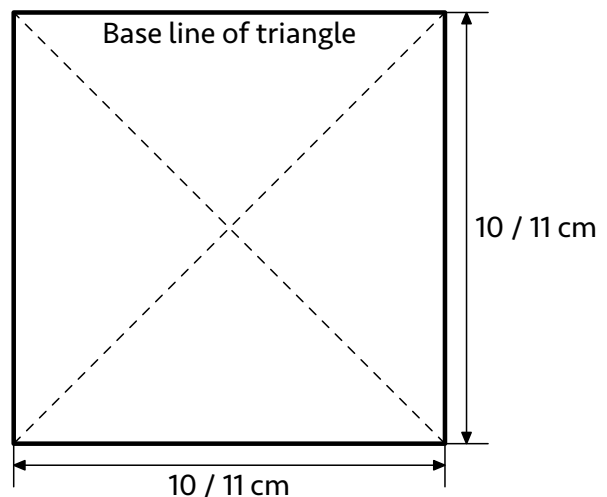


Figure 4: Corner cover material

- Cut the edge cover material diagonally in order to get 4 triangles, as shown above.
- Draw two lines along the shorter sides of the triangles, which are 15 mm from the sides.
- Glue the triangles on the outside of the boards, cut about 1 board thickness outside the corner of the board under 45 degrees. Turn in first the top turn-in, and after forming the corner-covering turn in the other turn-in.
- Prepare the outside cover paper. Use paste to glue it onto the boards. Paste down the turn ins.
- Open the previously prepared incision with a cutter. Cut the twill-band in half. Push one end of each piece through the incision from the outside. Cut the end square and glue it neatly into the recess.
- Paste the spine inside cover paper into place.
- Paste the inside cover paper on the boards.

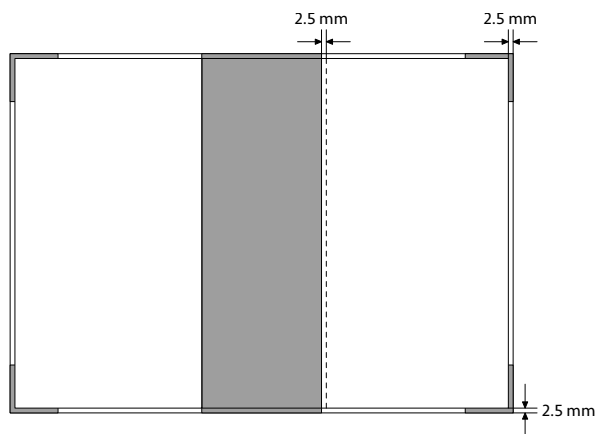


Figure 5: Inside cover material

- Cut the paper flaps to shape, as shown below, and crease 4 lines.
- Glue the three flaps onto one of the boards.

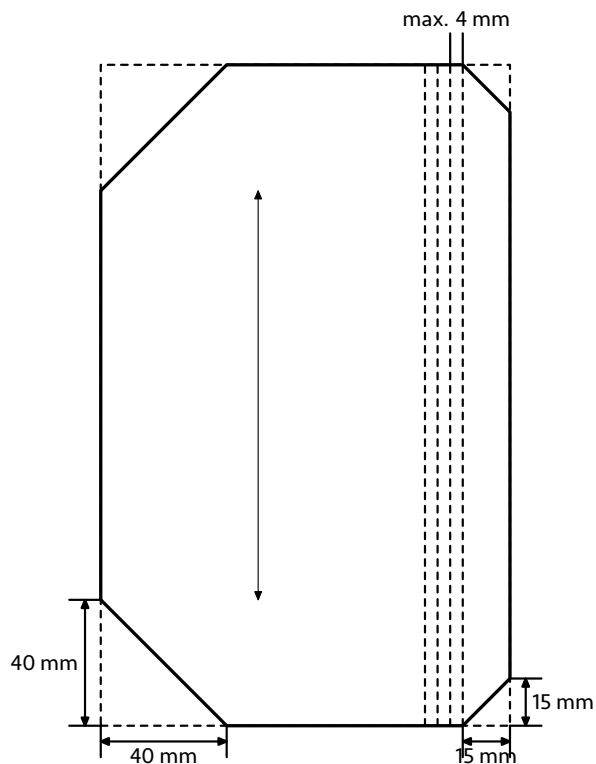


Figure 6: Flaps

Now your portfolio is finished! However, before using it, let it dry under weight in order to keep it flat!

◇ Willi Egger
w dot egger (at) boede dot nl

Debugging L^AT_EX files — Illegitimi non carborundum

Barbara Beeton

Abstract

Every L^AT_EX user has, at least once in her career, been faced with a thorny problem when compilation shuts down for some obscure reason. How to deal with simple problems is reasonably well known, but there are situations when the time-honored methods fall short.

This article will present strategies and tactics for dealing with the many types of problems that have arisen during long experience as a member of the AMS technical support staff, handling questions from authors and the editorial staff. Both common and uncommon glitches will be visited, with a bias toward avoiding problems in one’s own work — something for everyone.

1 Background

Last year, the AMS published on the order of 60,000 pages of books and journals, most of them produced from L^AT_EX files prepared and submitted by authors. The acceptance of a journal article is based on scientific merit, judged by an editorial committee and referees reviewing a paper or electronic document; it might even be handwritten. No consideration is supposed to be given to the presentation, only to the content. Books are contracted by the acquisitions staff, all of whom are professional mathematicians familiar with L^AT_EX, but by no means T_EXnically skilled. What comes in for production is what we have to deal with.

Assume that the accepted work *is* prepared in L^AT_EX (if it is not, it will be (re)keyboarded by a competent entry operator and delivered in usable condition); the quality of submissions varies greatly, providing a wealth of opportunity to test (and improve) one’s debugging skills.

Production is carried out on networked Linux systems. The available macro library is in three parts: T_EX Live, which is updated at most once a year; local versions of “public” macro files and fonts (sometimes including updated versions that will become part of next year’s T_EX Live collection); and macros, fonts and other tools that are entirely local to AMS. Everything is archived with Subversion, with archives of published books and articles extending back a couple of decades. The versions of (L^A)T_EX and all used packages are recorded within the main file for a published work using the `snapshot` package, so that if reprocessing is necessary, the original environment can be recreated. This setup provides the stability

necessary to produce a steady flow of new publications while handling reprints, revised editions, and conversion of existing publications to other formats such as ebooks.

As described so far, this workflow is effective and reliable once the files representing a manuscript are ready to be sent to the printer. But all sorts of things can go wrong before that happy moment. One guiding principle tops all others: If something goes wrong, it must be possible to recover a known, stable starting point quickly and reliably.

2 Preparation — plan ahead

There are certain conventions that, if followed diligently, can make one’s life easier in the long run. First, choose good tools and become familiar with them.

The most important tool is a competent editor or IDE. The author uses emacs, but other options are available, some for single users on one platform, some intended for cooperative authoring online, and a number of alternatives somewhere in between. A list of such tools can be found in answer to a question on the `TeX.stackexchange` site (hereafter “`tex.sx`” [4]).¹

The author also prefers to process files from the command line. This makes it possible to correct simple errors, such as misspelled control sequences, interactively, avoiding delays and the possibility of a cascade of irrelevant error messages as a consequence of a possibly trivial error. (But don’t forget to correct the file as well before the next run.)

Among the features most useful for debugging are these:

- good search facilities;
- brace and `\begin/\end` matching;
- multiple windows viewable at the same time;
- “go to” a specified line number;
- ability to match strings and to ask “how many?”

Another important consideration is how directories and files are laid out and addressed. It’s advisable to *avoid spaces in file names*; not all operating systems handle such spaces gracefully (or at all). Similarly, some operating systems are case sensitive — to avoid problems here, *use only the lowercase alphabet for file names*; digits and hyphens are also “neutral” in this regard, but (extra) periods and characters with special meanings to T_EX (e.g., the underscore) are best avoided.

Keeping files at a manageable size will pay off in the long run. For a large work like a book or dissertation, place each chapter in a separate file,

¹ LaTeX Editors/IDEs, <http://tex.stackexchange.com/q/339>

controlled by a main or “driver” file. This will permit you to work on just one chapter at a time, taking advantage of the `\includeonly` facility. If you have large tables or figures, placing each in a separate file can also be handy, as it is then possible to exclude one with a single `%` to comment it out (this also makes it easy to move it to a different place in the main file if that becomes necessary).

Finally, when preparing files, it’s usually a good idea to end files other than the main file with a separate line `\endinput`; this avoids problems from garbage that is sometimes added on, unasked, when a file is shipped from one system to another. And *never* put a line `\end{document}` in any file but the main driver file.

Another suggestion: Learn where the log file can be found, before you ever need to look at one. Some IDEs hide this from a user; if your job goes south and you cannot check what is happening by looking in the log, you are going to have a very difficult time figuring out how to make things right.

And one more:

**Don’t update your system
in the middle of an important project.**

New versions of packages can have new, incompatible features, and old packages can disappear. Of course, if your hardware decides to conk out at that point, this is not useful advice. But you *do* keep a full current backup, don’t you?

3 Isolate and insulate your testing

Use *copies* of your files to test.

If the error you’re trying to fix isn’t something like a simple typo, protect yourself against possible disasters: set up a special debugging environment. At the very least, make a backup of your files, maybe even a zip of the full working directory tree, and put it in a safe place. You know your current situation, and you want to be able to return to it safely.

**Under no circumstances make experimental
changes to your only copy of any files.**

Better yet, if you have the space, create a separate test area, identical in all important respects to the “live” work area, and do your experimenting there.

If the job consists of more than one file, start by copying *only* the driver file—the file that reads in all the others—into the test area. This will be your guinea pig.

Use a “soft link” to access other files in the job. For a Linux system, this involves issuing the command

```
ln -s <directory name>
```

and adding the name of that location to the path.

(It should be possible with a web search to find out how to do this for other systems.)

Process the job interactively. Then simple errors can be corrected at once, before they spawn meaningless and confusing error messages. (Remember to make the corrections in *both* test and real files.) And if an error is detected that can’t be corrected interactively (such as an unrecognized or unended environment), the job can be stopped at once and the problem fixed before continuing.

Processing a job in nonstop mode (the usual procedure when launching a compile from within an IDE) will, of course, list all errors in the log file (up to a maximum of 100), but a single error that is not the simple misspelling of a symbol name can cause a cascade of spurious messages that would not have been necessary unless the first error was encountered.

More about this approach below, under “Divide and conquer”.

4 Some tools for interactive diagnosis

Some diagnostic commands are available to send information to both the terminal and the log file.

- `\message{...}` writes out a message in the log and on the screen; it can be used to report when processing has reached a predetermined point. For example,

```
\message{last section, page \number\thepage^^J}
      last section, page 904
```

- `\show` reports the current meaning of a command; processing is suspended to permit additional interaction. Example:

```
\show\LaTeX
> \LaTeX=macro:
->\protect \LaTeX .
\show\protect
> \protect=\relax.
```

Following the halt, more input can be inserted by typing `i` followed by a command or text. “Enter” will restart the session.

- `\showthe` reports the *value* of a command; processing is likewise suspended.

```
\showthe\hfuzz
> 1.0pt.
```

A number of tracing commands are available to provide details of the processing flow. (Caution: tracing requests can deliver more information than you usually want, so be selective.) The result is sent only to the log unless requested otherwise. These are the tracing commands used most often by the author:

- `\tracingoutput` can be set to 1 to report, in symbolic form, the contents of all boxes that are written to the output;

- `\tracingcommands` and `\tracingmacros` give the gory details of L^AT_EX processing;
- `\errorcontextlines=200` sets the maximum number of lines associated with a single error message; the default value (5) often shows too few lines to understand the entire operation;
- `\tracingonline` directs the report of the other tracing commands to the screen as well as to the log.

Details of these commands (and many `\tracing...` relatives) can be found in *The T_EXbook* [2] or in *T_EX by Topic*² [1].

5 The log file is your friend

The (L^A)T_EX log file records every action taken—what files and fonts are read, assignment of boxes and counters, redefinition of important commands, and so on. More importantly, from a debugging point of view, errors are reported in (sometimes excruciating) detail, all identified by line number in the source file.

Always check the log file for error messages:

```
! Undefined control sequence.
1.457 \fobx
      {%
```

Warnings are noted too, but without line number:

```
LaTeX Warning: There were undefined references.
Interpreting these messages can be a challenge, but this information should direct your first line of inquiry. If the system you are using hides the log file, ask how to find it. And don't delete the log file without looking at it.
```

Not every line number reported in an error message clearly identifies the exact line where the problem is located. The scope of math content (what is between `$` signs or other math specifiers) is not permitted to include paragraph breaks, so a missing `$` may not be reported until the next paragraph break, which may be a number of lines later in the input. (This restriction is also the reason that blank lines are not permitted within multi-line math display environments.) The other error associated with math mismatches is

```
! Missing $ inserted.
```

when a closing `$` is forgotten. This too is limited to the current paragraph, and should be easy to diagnose and repair.

An error within a figure, table, or multi-line display will also usually report the line number at the end of the environment, rather than on the line where it occurs, but again, the scope is relatively limited.

Another reason for a report far away from where the error occurred is an unmatched group—an errant `{`, `\bgroup`, `\begingroup` or `\begin{env}`. In the case of a mismatched environment, this error will be reported as

```
! LaTeX Error: \begin{env1} on input line
      nnn ended by \end{env2}
```

This will be reported as soon as the (incorrect) end is encountered and the line number should be correct. A mismatched grouping element, on the other hand, will not be reported until the end of the job, and then not even as the usual warning. The report consists of several lines:

```
(\end occurred inside a group at level m)
### semi simple group (level m) entered
      at line nnn (x)
### bottom level
```

Here, *m* will identify how many of these open groups remained at the end of the job. *x* will identify the unmatched grouping element: `\begingroup`, or `{` for either `{` or `\bgroup`. Again, the line number should be correct, just not in the place where the omission occurred.

Other possible error messages are shown in the documentation of various packages. Most messages include some line number, and in general localization is reasonably good; this is often enough to locate an error so it can be corrected without having to progress to more complicated searching steps. As soon as the error is identified and the fix verified, you can correct your *real* file, continue with the main task, and forget about the copies, which have now performed their intended function.

But, you may ask, when the job consists of multiple files, how can one be sure in which file the reported line number actually exists? See the next section.

The important lesson here is this:

Don't delete the log file until after you've extracted every bit of useful information.

It has also been suggested to the author that saving a log file for even longer (under a different name) has merit, as it makes possible the comparison of two logs when there is a question about what changed between two runs.

6 E pluribus unum—but which one?

Let us assume that the error was reported in a text file, not a package.

When the log file reports a line number, the first reaction is to look in the main file. But if that file is only 95 lines long, and the reported line number is 2345, that does not compute.

² In T_EX Live; access with `texdoc texbytopic`.

Make a copy of the log file, and work backward from the relevant error message. If some pages have actually been output, the page number (shown in square brackets: [17]) can point to a chapter, which ideally should be in a file of its own. Failing that, eliminate material that is, for this purpose, useless.

Messages about overfull boxes can be ignored—delete those lines. A matching pair of parentheses will usually enclose a file name and some more material. Look for a “completed” parenthesized group, such as

```
(C:/tech-support/debug/preface.tex
Preface
[1] [2]
)
```

and delete the whole group. What will finally remain is an opening parenthesis followed by a file name—the name of the file that was open when the error was reported. The reported line number should be found there.

But what if the line number was reported only at the end of the job, a `level m` situation? Here’s where an extra `\end{document}` comes into play.

Keep working only with test files.

Don’t touch the real files until the source of the problem is identified.

Start from the end of the driver file and insert `\end{document}` between two `\include` statements. The “binary” approach is appropriate here—start in the middle. (More about this under “Divide and conquer”.) Process what’s left. If the `level m` condition is still reported, the target file is in the first half; if it’s absent, look in the last half. Comment out `\include` statements that have been absolved of blame, and move `\end{document}` around until the target file is identified. This gets more complicated, of course, if $m > 1$, but the principle is the same.

7 Housecleaning

At some point, you will find the file where you think the error should be. Maybe you have a tightly defined line number. But maybe you still have only a general idea of where to look. Since you want to process only one file, clean out the clutter so it won’t cause confusion.

Modify the driver file, adding an `\includeonly` line that specifies only the suspect file. Comment out commands that will include irrelevant pieces that aren’t launched with an `\include` command:

- unnecessary (for the test) packages;
- `\tableofcontents`;
- anything related to the bibliography;
- `\printindex`.

Clean out your suspect file too. Don’t worry about destroying the file; this is a copy, right? Here are the things that can be removed—carefully.

- lines commented with `%` at the beginning;
- lines between `\begin` and `\end{comment}`, inclusive;
- lines between `\iffalse ... \fi`, inclusive (this is equivalent to a comment).

Make sure that all groups are completely specified. This means matching all `\begin` and `\end` environments and all methods of “bracing”. Check for these elements using your editor’s “how-many” function:

- number of opening braces `{` = number of closing braces `}` (sometimes the string `% }` is added when an opening brace stands alone in the code, so be aware of this possibility);
- number of `\begin{` = number of `\end{`;
- number of `\begingroup` = number of `\endgroup`;
- number of `\bgroup` = number of `\egroup`;
- number of `\[` = number of `\]`;
- number of `$` signs is even, as is number of `$$`.

Process what’s left, and look at the log for help as you go along.

Many problems are the result of an “unmatched” condition, so you might get lucky and not have to go any further. But let’s assume it’s still unidentified.

8 Divide and conquer

What you want to do is isolate the paragraph, or the smallest portion of the file, that is triggering the error. (Work with a copy, and keep another copy, just in case.)

Find a good paragraph break halfway through the file. Insert `\endinput` preceded by a blank line. Make sure it doesn’t split up a `\begin/\end` pair or any group. Process this reduced file. If no error is reported, the problem is in the last (unprocessed) half. Remove the part that works, and keep moving `\endinput` until the source of the problem is located. If the solution is obvious, fix it and test. Apply the fix to the full *test* copy and try processing it. Once you are sure the fix is correct, insert it in the *real* version and test again.

But what if the solution isn’t obvious?

If what remains is still too large for you to identify the problem quickly—perhaps it is a long proof, with steps presented as a list—make a copy of the file under another name and keep only the test material in the “working” file. (Many times this author has modified her *copy*, which is not the one that the driver file will input. This leads to exasperation.)

Reduce the size of this file by commenting out items that look harmless. Don't delete anything yet — what you think is harmless may actually be part of the problem. Keep iterating this process until there is no way to get rid of more material without also eliminating the (not yet located) error. This is now your “minimum (non-)working example”, an “MWE”.

Examine what's left in the file, and

Pay attention to the clues in the log.

Of course, once you know what needs to be fixed and how to fix it, you can verify this by making the necessary changes in your test file and rerunning L^AT_EX to confirm. If this works, install the fix in your real file, process it, and *if you find no other problems*, you're on your way!

If you do find more problems, it's back to the start, but now you know how to proceed.

One area hasn't yet been addressed — an error reported before

```
\begin{document}
```

is found. See below.

And there are more techniques that you can apply yourself, before calling for help.

9 Sometimes, more drastic action is required

In this section, we're still discussing problems in the body of the document.

Once a problem has been reduced to an MWE, it's time to take advantage of the available diagnostic tools to obtain more information. In addition to the commands shown on page 160 in the section on interactive diagnosis, these are also useful. (More detailed information on these commands can be found in *T_EX by Topic* [1].)

- `\tracingmacros` reports the details of macro expansion, along with the values of the arguments.
- `\showboxdepth` specifies the number of box levels to display, usually set to `\maxdimen` for tracing.
- `\showboxbreadth` specifies the number of successive elements displayed on each level.

There are more, but these are generally the most useful.

If you are desperate, and a real masochist, you can specify `\tracingall`, but sorting through this information will tax both your patience and your sanity, and usually a “simpler” approach can be found. See the definition of `\tracingall` in the file `plain.tex` to see what is unleashed.

But if you have to resort to tracing, there may be an easier way.

10 In case more help is needed

Some useful resources are available online. You may not be the first to encounter a particular problem.

The archives at `tex.sx` [4] are a good place to look. If you don't find anything resembling the problem in your file, ask a new question. (You should register if you're not already a participant in the forum.) For best results, include a complete MWE; you already have one — the minimum (non-)working example that you have been struggling with. Clean out any commented material, and, if appropriate (and possible), “anonymize” it by substituting dummy text; make it as minimal as possible while still demonstrating the problem. Include relevant lines from the log of the example you're posting, and an explanation of what you've tried. The participants in the forum are knowledgeable and friendly, and they enjoy a good puzzle — but they do need enough information to be able to experiment, and providing an MWE that they can copy and paste will yield results more quickly than if guesswork is needed.

11 Errors reported before `\begin{document}`

- Make a copy of the log file, and find the open file.
- If this isn't a `\usepackage`, back up until you find one.
- Do you have experience of L^AT_EX internals?
- No. This is a good time to seek expert advice. Go to `tex.sx` [4]. If the problem hasn't been reported, post a question. Be specific, and include your preamble and log.
- Yes. Figure out what the problem is. Check reports at `tex.sx` [4]. If it hasn't been reported, notify the package author.

This ends the discussion of problems that may occur in *your* files. The next section describes an actual problem of the author that took far too long to understand, and, in the end, wasn't really a “L^AT_EX problem”, although that's where it reared its ugly head.

12 A real puzzlement

Once in a while, not even tracing can direct you to the solution of the problem.

Two facts are important:

- I live and work in the U.S. and my workstation is set up with (presumably appropriate) local defaults, i.e., ASCII.
- I compile from the command line, and don't use `\batchmode` or `\nonstopmode`.

What showed up on the screen:

```
Overfull \hbox (23.1113pt too wide) in paragraph at lines 3288--3301
\OML/cmm/m/it/10.95 A$\T1/ptm/m/n/10.95 , as in
```

The corresponding content of the log:

```
Overfull \hbox (23.1113pt too wide) in paragraph at lines 3288--3301
\OML/cmm/m/it/10.95 A$\T1/ptm/m/n/10.95 , as in Ÿ[], is a degree-
```

What was in the source file:

```
..., as in \S\ref{SS:changing}, is a degree-1 ...
```

Figure 1: A most puzzling problem

A few months ago, a file was persistently stalling before processing was finished, locking my screen. In order to regain control, it was necessary to open another session and kill the job. This allowed me to enter `ctrl-C` to the stalled session, to return to a prompt. The last thing shown on the screen was a partial report of an overfull box. Enough text was present to be able to locate the problem area in the source ... except that the source looked perfectly valid. (See figure 1.) There was, fortunately, a log produced, albeit incomplete.

After going through the steps described above, I managed to cut the file down to a single, brief, paragraph; if I removed anything more from the beginning of the paragraph, the error didn't occur. The problem appeared to be related to the overfull box. At this point, I sought help from someone with more systems knowledge than I have.

After looking closely at what was on the screen and what it corresponded to in the log, we noticed a “strange” character — Ÿ. (This is in position "78 in the `cmsy` font, and is Unicode character U+0178.) Since I'm used to working with English texts, and only infrequently deal with accents, I'm not used to seeing non-ASCII characters, and certainly not in a log from an entirely English text. What was happening was that the physical environment I've been working in is not set up to recognize UTF-8 input, and the screen was freezing as a result.

The workaround I was given was to put this line into a file named `.i18n` in my home directory:

```
LANG="en_US.utf8"
```

This doesn't solve the problem entirely — the file still freezes the screen, but the job completes, and I can issue `ctrl-C` to regain a prompt. But figuring out the problem and how to get around it were sorely trying.

Sometimes what one thinks is a \LaTeX bug isn't. Keep an open mind.

13 Oddments (post-conference additions)

There are some easily identified problems that occur frequently, but the source may not be generally known. This seems a worthwhile place to identify them.

- **The Missing character:** warning
`There is no ; in font nullfont!`
 is almost invariably the result of a syntax error — a missing semicolon — in a `tikzpicture`. Other repeating punctuation characters citing `nullfont` may also be associated with some `tikz` expression.
- Similar warnings citing other fonts need to be researched. No line number is given in the log, but the number of the last completed page will be there. Compare the input with the output to see what is missing.

14 Acknowledgments

Thanks to GUST for hosting TUG'17 at Bachotek along with their annual meeting, and thanks to the participants whose questions after my talk provided interesting and helpful new ideas.

References and resources

- [1] Victor Eijkhout, *TEX by Topic: A TEXnician's Reference*, Addison-Wesley (U.K.), 1991. eijkhout.net/texbytopic/texbytopic.html or `texdoc texbytopic`.
- [2] Donald E. Knuth, *The TEXbook*, Addison-Wesley, Reading, 1986.
- [3] Leslie Lamport, *L^AT_EX, A Document Preparation System*, 2nd edition, Addison-Wesley, Reading, 1994.
- [4] tex.stackexchange.com, a question and answer forum with extensive archives.

◇ Barbara Beeton
 American Mathematical Society
 Providence, RI, USA
[bnb \(at\) ams dot org](mailto:bnb@amsdotorg)

Revealing semantics using subtle typography and punctuation

Kumaran Sathasivam, S.K. Venkatesan and Yakov Chandy

Abstract

The semantics of a language has deeply nested structures, which is revealed by typography using a hierarchy of paragraphs, with different font sizes and styles. At the paragraph level, a paucity of typographical features forces us to use punctuation heavily to reveal semantics. Paragraphs are further broken into smaller semantic units such as sentences using end-punctuation and initial capitals. Sentences are further broken down using semi-colons, colons, commas and hyphens into even smaller chunks. Word-spaces are used to break language into the smallest atoms of semantics, namely words or phrases. In this paper, we look at newer devices, both typographic settings and punctuation elements, that can disambiguate and reveal deeply nested semantic structures.

“To reveal art and conceal the artist is art’s aim.” — Oscar Wilde

1 Introduction

The indivisible elements, the atoms as it were, of the written forms of languages like English are letters. But reading text built of these basic units alone is a difficult exercise. An elaborate set of conventions, auxiliary symbols (punctuation symbols), spacing and typography are used in publishing today as aids to reading, as removers of obstacles to understanding. The salutary effects that these jointly have will be readily appreciated by a comparison of the first two sentences of this paragraph with a version written entirely in uppercase and unencumbered by punctuation symbols or word spaces (Figure 1). Text was, in fact, written historically thus. There was no spacing, paragraphing or punctuation in manuscripts before the development of printing (Boorstin, 1983).

Uppercase characters are still used amidst lowercase characters. Uppercase letters are used to display title and headings in a prominent way, similar to how they were used to display text in ancient Roman buildings. Uppercase letters are also used as the beginning character of prominent names (proper nouns) and later evolved to shorter form initials, abbreviations and acronyms. They are also used at the beginning of a sentence as a device to prominently mark the beginning. Special uppercase characters are used as dropped capitals in some books as a decorative element at the beginning of a chapter or other part of the document.

THEINDIVISIBLEELEMENTSTHE
ATOMSASITWEREOFWRITTEN
ENGLISHARELETTERS BUTREADING
TEXTBUILTOFTHESEBASICUNITS
ALONEISADIFFICULTEXERCISE

Figure 1: Text in uppercase only, stripped of punctuation symbols and word spaces.

The lowercase letters started initially in the cursive italic form, but now they have evolved their own upright roman form. Bold and display fonts have more or less replaced the necessity of displaying title and heading in uppercase characters. Interestingly, another form known as the small-caps has evolved from capital letters with its own variety of lowercase characters, an interesting hybrid in typography.

Fonts may be broadly classified into serifs and sans-serifs. Slab serifs are a significant subgroup of serif fonts, usually with relatively thick strokes and with flat, rectangular serif shapes. Apart from these we have other categories of fonts used, such as monospaced fonts, used to display telegrams (such as in Wikileaks messages) or computer code and other types of fonts such as Gothic and script fonts to reveal specific content or context.

At the paragraph level, the document title, headings, paragraphs, footnotes are distinguished by typographic elements such as use of different font styles and font sizes. However, once we come down to the paragraph level, except for occasional embellishments with bold and italic, upper- and lowercase characters, we are left with only the punctuation marks to describe the underlying structure of the sentences. The role of punctuation transcends that of merely providing assistance for the eye; it now encompasses interpretation of text. In other words, punctuation is vital for the meaning of written material.

It is therefore not surprising that the first known use of punctuation is related to a system that was used to help the delivery of speeches from written texts. This system dates back to the fifth century BC, when the Greeks introduced vertically arranged dots in text. Subsequently, when Greek playwrights (for example, Aristophanes and Euripides) wrote drama, they used symbols to distinguish the ends of phrases so that the play’s cast knew when to pause. Even relatively recently, school children have learned to associate punctuation marks with pauses in reading. One mnemonic poem actually quantifies the duration of the pause associated with each major punctuation symbol:



Figure 2: The dramatic change in meaning that a single comma can introduce is illustrated by this example. [Source: <http://www.themodernausten.com/2012/09/04/teacher-tuesdays-9-4-12/>]

**Charles the First walked and talked half an hour after his head was cut off.
Charles the First walked and talked; half an hour after, his head was cut off.**

Figure 3: In this example, the meaning of the first sentence is intriguing though rather macabre. A semi-colon and a comma change the meaning of the same string of words entirely, to something more likely. [Source: The American Printer, 1885 edition]

The stop point out, with truth, the time of pause

A sentence doth require at ev'ry clause.

At ev'ry comma, stop while one you count;

At semicolon, two is the amount;

A colon doth require the time of three;

The period four, as learned men agree.

The semantic role of punctuation is easily highlighted using a couple of well-known facetious examples (Figure 2, Figure 3). Thus a particular string of words can have different meanings, depending on the punctuation. In fact, a particular sequence of words can even have two opposite meanings (Figure 4).

This is not paradoxical. If such texts with different meanings are delivered *orally*, they are spoken very distinctly, depending on the intended meaning. The written forms, since they are reduced to identical strings of symbols, require auxiliary support, *marking up*, in the form of punctuation to make the distinction.

2 Evolution of punctuation

Punctuation was developing rapidly at a time when large numbers of copies of the Bible were produced by copyists in Europe, in the fifth century AD. These copies were designed for reading aloud, and so a range of marks were introduced in the text. An early

**Woman, without her man, is nothing.
Woman: without her, man is nothing.**

Figure 4: The first of these sentences emphasizes the importance of men; the second, the importance of women.

version of initial capitals (the use of lowercase letters to write sentences, except for the first letter of the sentence, which is in uppercase) made its appearance at this time. In the eighth century AD, Irish scribes introduced the practice of separating words. This was a major step in semantics, as now words began to have a standalone existence, making them candidates for a study on their own, reinforcing a socially shared context, through the use of dictionaries. The English language also evolved as an isolating language, making isolation of words possible, but at the same time increasing the importance of the position of the words, creating a need for position-based syntax and grammar. Over the next several centuries, the movement was from words to phrases and several systems of punctuation appeared, some of them disappearing after a spell of popularity, others persisting unchanged or evolving with time.

The use of movable type and the rise of printing in Europe in the 15th century led to an increase in the amount of material printed and in its readership. The printing press spread to hundreds of cities in Europe within decades. It is estimated that by 1500 the printing presses of western Europe had produced 20 million volumes. The need for a standard system of punctuation was keenly felt. Two printers of Venice, both named Aldus Manutius, one the grandson of the other, are credited with the invention of such a system. To the printers Aldus Manutius are attributed the development of punctuation practices that continue to this day, such as the one of ending sentences with full stops, and the development of symbols such as the modern comma. The younger Manutius said in 1566 that the main object of punctuation was the clarification of syntax. The trend of punctuation reflecting sentence structure continued. Notable in this context is Ben Jonson's book *English Grammar*, published posthumously in 1640, which provided the foundation for the punctuation rules followed today. This is not to say that there is total agreement about the rules of punctuation — there is still some range in punctuation usage.

Printing presses spread further, and in the 16th century they produced between 150 and 200 million copies. In the 19th century, the hand-operated press was replaced with the steam-powered rotary

press, which allowed printing to be performed on an industrial scale.

3 Punctuation rules and style manuals

As early as the late 17th century, manuals (such as Moxon's *Mechanick Exercises*, 1683–1684) were being produced for the printing trade. With the passage of time, these increasingly addressed the general reader. One of the most successful printing manuals of the 19th century, *The American Printer*, was published in 18 editions between 1866 and 1893. The preface to the first edition of this manual said that ‘Authors and publishers, as well as typographical amateurs, may consult the volume with profit; and indeed, any intelligent person will find it a serviceable companion.’ *The American Printer* (15th edition, 1885) touches very lightly upon the subject of punctuation when it outlines the work of the proof-reader: ‘The compositor is bound to “follow the copy,” in word and sentiment, unless, indeed, he meets with instances of wrong punctuation or false grammar, (and such instances are not rare,) which his intelligence enables him to amend.’

Just what correct punctuation might be was being defined by academic presses around this time and in the early 20th century, by which time they were drawing up their own rules or standards for typography. Horace Hart, controller of the Oxford University Press (OUP), had worked some three decades at printing establishments, compiling best practices over this period. In 1893 these were printed as a single broadsheet page for use at the OUP. They developed over the years, and were published in 1904. Hart's *Rules* quickly became a source of authoritative instructions of not just typesetting style but also English usage, grammar and punctuation. Similarly, in the 1890s a proofreader at the University of Chicago Press had drawn up a single sheet of typographic fundamentals. In 1906, the *Chicago Manual of Style* (CMS) was published as a book. The CMS16 (2010) is now in its 16th edition, and its guidelines have been shaped by ideas from both within the press itself and outside. The print version of CMS16 has more than a thousand pages, and there are more than 2000 hyperlinked paragraphs online. The Web site of the CMS says that it ‘has become the authoritative reference work for authors, editors, proofreaders, indexers, copywriters, designers, and publishers’.

Editors and other users of style manuals tend to follow their prescriptions slavishly although the manuals themselves point out that there is nothing hard and fast about their ‘rules’:

As always, most Chicago rules are guidelines, not imperatives; where options are offered,

the first is normally our preference. Users should break or bend rules that don't fit their needs, as we often do ourselves. Some advice from the first edition (1906), quoted in the twelfth and thirteenth editions and invoked in the fourteenth, bears repeating: “Rules and regulations such as these, in the nature of the case, cannot be endowed with the fixity of rock-ribbed law. They are meant for the average case, and must be applied with a certain degree of elasticity.” (CMS15, 2003)

The desire to adhere zealously to the guidelines of style manuals possibly arises in response to the intricacy of the rules: the CMS and the New York Public Library's style manual (Sutcliffe, 1994) devote close to 50 pages each to the use of punctuation symbols alone. It is evident that the prescriptions have been drawn up with great thoroughness to deal with every conceivable situation that may arise when using punctuation symbols.

Perhaps as a caution against overenthusiastic enforcement of the recommendations, CMS15 reminds users that ‘[p]unctuation should be governed by its function, which is to promote ease of reading. Although punctuation, like word usage, allows for subjectivity, authors and editors should be aware of certain principles lest the subjective element obscure meaning. The guidelines offered in this chapter [Punctuation] draw for the most part from traditional American practice.’ In other words, the essence of punctuation is to disambiguate.

4 Semantic inadequacies in current methods of punctuation and typography

The rules of punctuation are evolving continuously. New, revised editions of style manuals are published periodically. To quote words from Wikipedia, ‘The rules of punctuation vary with language, location, register and time and are constantly evolving.’ This continuous evolution of punctuation is due partly to developments in the language and partly to the fact that the ‘rules’ laid out in style manuals are often merely descriptive. Further, some aspects of these rules are rather whimsical. As a result, there are inadequacies in the prescriptions regarding punctuation. In Box 1 we present two cases where redundancy is present in the rules of punctuation.

In fact, there are serious limits to how well even the traditional function of punctuation, namely, indicating how text is to be read, can be carried out. As the Wikipedia entry on ‘Punctuation’ puts it, ‘Even today, formal written modern English differs subtly from spoken English because not all emphasis and disambiguation is possible to convey in print, even

Box 1: Redundancy in punctuation rules.

- *Demarcation of sentences.* Sentences are clearly marked off using both end-punctuation (full stops, question marks, exclamation marks) and initial capital letters. It would be more rational for a style manual to prescribe the use of a single sentence-separation punctuation symbol.
- *‘Which’ versus ‘that’.* A distinction is made by style manuals, most American ones, between the relative pronouns ‘which’ and ‘that’. These manuals prescribe the use of ‘that’ with a restrictive purpose, to narrow a category or identify a particular term being talked about. ‘Which’, on the other hand, is recommended for nonrestrictive use, not to identify a particular item or narrow a class but to add something about an item that has already been identified. The style manuals redundantly prescribe that ‘which’, when used nonrestrictively, should always be preceded by a comma, a parenthesis or a dash (CMS15, page 230).

Person 1: (Aggressively) Where did you get that?

Person 2: (Inquisitively) What?

Figure 5: Playwrights must provide stage directions to indicate how lines are to be spoken by actors. [Source: <http://www.tes.com>]

with punctuation.’ Nowhere are these limits felt more keenly than in play scripts, where the dramatist must liberally add adverbial stage instructions (Figure 5). Furthermore, we present in Box 2 a couple of instances we found where there may be ambiguity even when the rules of punctuation are followed faithfully.

The development of punctuation is proceeding perhaps faster than ever before. New punctuation symbols have been proposed. Houston (2013) describes the deliberate creation of a symbol to convey a mixture of surprise and doubt. This symbol, known as the interrobang, enjoyed some popularity during the late 1960s and early 1970s. It was lost in the transition of the printing industry from hand-set, hot metal printing to phototypesetting. Houston (2013)

Box 2: Ambiguity in punctuation rules.

- *The comma in an unclear role.* The comma is used in a great many situations. Sometimes, it is not clear what role it is playing. For instance, according to one CMS15 rule, ‘A word, abbreviation, phrase, or clause that is in apposition to a noun is set off by commas if it is nonrestrictive—that is, omissible, containing supplementary rather than essential information.’ The first example provided for such a use of the comma is ‘The committee chair, Gloria Ruffolo, called for a resolution’. Another CMS rule describes the use of the Oxford comma: ‘Items in a series are normally separated by commas . . . When a conjunction joins the last two elements in a series, a comma—known as the serial or series comma or the Oxford comma—should appear before the conjunction. The first example provided for this rule is ‘She took a photograph of her parents, the president, and the vice president.’ If, however, the sentence ‘She took a photograph of her father, the president, and the vice president’ is encountered, how is the reader to interpret it? Did she take a photograph of three persons? Or was her father the president, so that she took a photograph of only two persons, namely her father and the vice president? This type of ambiguity is encountered when the Oxford comma is followed.
- *A relative clause ambiguity.* Sometimes it is difficult to distinguish between interrogative and nominal relative noun clauses. An example is the sentence ‘I forgot what he asked for.’ One interpretation is ‘I know what he asked for. But I forgot to bring it.’ In this case, the object of the sentence is a relative noun clause. Another interpretation is ‘I do not know any longer what it is he asked for.’ Here, the object is an interrogative noun clause.

also describes how, even more recently, a new symbol, the sarcasm mark, was proposed by a blogger who observed that written sarcasm was regularly misinterpreted as sincerity in online interactions. The need for enhanced disambiguation through new punctuation has been both necessitated and facilitated by

the development of electronic communication. Smileys, emojis, emoticons and sarcasm tagging have all gained widespread recognition and usage through text messages and emails. It is quite possible that these new punctuation symbols will be absorbed into regular print typography.

5 Extending the L^AT_EX solution for deep structures

Like XML and HTML, L^AT_EX provides a mechanism for separation between style and content. The user provides hints and hooks for the typesetting engine and the typesetting engine then provides the rendering, incorporating some hyphenation and justification of paragraphs, and paginating the document. There is now adequate computer power to take L^AT_EX typesetting to higher levels of sophistication.

We note that the rules of punctuation and typography need to be revisited. We believe that the present setting provides opportunities to advance disambiguation, for both the human reader and the machine reader, through imaginative typography and punctuation.

First we construct some simple solutions to existing disambiguation problems, before we move on to the general problem of structure and semantics.

- *The comma in a dual role.* We could disambiguate the sentence ‘She took a photograph of her father, the president, and the vice president.’ in the following ways: ‘She took a photograph of her father ,the president, and the vice president.’ Here we try to convey the information that her father is the president by putting ‘the president’ within parenthetical comma, which is comma that is shifted after the word-space to indicate the parenthetical nature of it. Likewise, when there is a list with items containing coordinating conjunctions, and there is no Oxford comma, we would compress, expand or letterspace the text automatically. The example we have is the sentence ‘We had ice cream, fish and chips and strawberries and cream at the tennis match.’ We could disambiguate this by writing it this way: ‘We had ice cream, fish · and · chips and strawberries · and · cream at the tennis match.’ We hope that the typography of L^AT_EX will be clever enough to distinguish this nesting indicated here through curly braces and provide subtle typographic effects to indicate the nested structure. When Zapf’s hz-program squeezed or stretched individual characters by a few percentage points or letterspaced text (to avoid rivers) in the mid-1990s, it was considered sacrilege by purists. Be that as it may, these features, which

Zapf referred to as ‘micro-typographic features’, have entered L^AT_EX and other typesetting programs. We believe that these features, when used appropriately, will serve the purpose of disambiguation.

- *Relative clause ambiguity.* Can we disambiguate the following sentence? ‘I forgot what he asked for.’ As mentioned above, this has two possible interpretations:

- 1 I *forgot* that he asked for something;
- 2 I cannot remember *what* it was that he asked for.

The solution:

- 1 ‘I forgot_i what he asked for.’
- 2 ‘I forgot what_i he asked for.’

We hope that the introduction of new punctuation after ‘forgot/what’ solves the problem. It may require subtle education of both the readers and authors.

- *Sentence break.* The sentence demarcation problem from Box 1. A simple solution would be to introduce a hidden (nonprinting) demarcation symbol between sentences. We could introduce a new character, say a square dot, ‘▪’ (a small version of the Halmos square box). This would provide unambiguous information to a ‘machine reader’.

At present, with all the progress in AI, especially Deep Learning, there are now quite a few natural language processing (NLP) libraries that could also be used by the L^AT_EX engine to interpret the input and produce typeset output as desired, with options to switch on/off such features by the authors of L^AT_EX. Such interactive NLP systems can also query the author when in doubt. Of course, all this becomes feasible only if we have a rich set of font families with subtle variations and a set of new glyphs for proposing new punctuation marks. A sentence tagged with parts of speech by NLP, with additional indicators introduced by L^AT_EX authors, can provide a wide scope for L^AT_EX to typeset the output with subtle typographic variations revealing the underlying structure. Noun phrases and verb phrases can be indicated by either a different font-face variant and/or replacing word-spaces within the phrases by a middle dot, ‘·’, as in the example, ‘We had ice cream, fish · and · chips and strawberries · and · cream at the tennis match.’

Zapf’s ‘micro-typographic features’ have entered L^AT_EX and other typesetting programs, and these features along with newer typefaces for representing grammatical structures and parts of speech will help us evolve language and typography to higher levels.

6 Conclusion

Evolution of style, typography and punctuation have been studied in some detail here. In this article we have also indicated the flavour of the interesting new world that lies before us, while at the same time indicating the need for a huge family of fonts in the great tradition of Knuth's computer modern font or even a myriad of fonts in the MetaPost tradition. We also need a repertoire of new glyphs for punctuation, to express ourselves fully in writing as we would indicate in speech (in tonal variations) and actions (hand, head movements) in our real life. There are quite a few glyphs in musical notation that can give us the inspiration and direction: ♪ ♫ ♬ ♭ ♯. Music is an interesting area to start searching for, as it was here in our conception of music that our baby steps in speech and language took shape; there could be something interesting that we left behind in our early creative outpourings.

References

- Boorstin, D. J. (1983). *The Discoverers*. Random House.
- CMS15 (2003). *The Chicago Manual of Style*. The University of Chicago Press, 15th edition.
- CMS16 (2010). *The Chicago Manual of Style*. The University of Chicago Press, 16th edition.
- Houston, K. (2013). *Shady Characters: The Secret Life of Punctuation, Symbols & Other Typographical Marks*. W.W. Norton & Company.
- Sutcliffe, A. J. (1994). *The New York Public Library Writer's Guide to Style and Usage*. The New York Public Library and The Stonesong Press Inc.

- ◇ Kumaran Sathasivam
Writer and consultant, Chennai, India
kumaran.sathasivam (at) gmail dot com
- ◇ S.K. Venkatesan
TNQ Books and Journals, Chennai, India
skvenkat (at) tnqsoftware dot co dot in
- ◇ Yakov Chandy
TNQ Books and Journals, Chennai, India
yakov (at) tnq dot co dot in

To justify or not to justify? Why bad typography may be harmful for your readers

Boris Veytsman and Leila Akhmadeeva

1 Introduction

One of the most well-known algorithms in $\text{T}_{\text{E}}\text{X}$ is the famous hyphenation algorithm, which implements true justification. Some other computer typesetting systems do not bother with hyphenation, just increasing spacing between the words until the lines of text have the same width. While this method is frowned upon by typesetters, it is a valid question whether it makes a measurable difference for readers. This question is especially important for readers with cognitive impairments, for example, post-stroke patients.

In one of our previous works [1] we studied the difference in reading speed and comprehension between justified hyphenated text, *and* ragged right non-hyphenated text. It showed that justified texts were read slightly faster than ragged right, but on delayed (see below) tests gave slightly worse results. However, one can argue that in [1] we measured two different factors: justification and hyphenation, and their influence was confounding. Fortunately, $\text{T}_{\text{E}}\text{X}$ allows us to separate them, and study hyphenation and justification separately.

In this work we compared the speed of reading and comprehension of two sets of unhyphenated texts: justified (“sloppily justified”, using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ terminology), and ragged right. We measured these factors for post-stroke patients.

2 Experimental methods

The experimental methods were the same as in our previous papers [1–4]. A group of $n = 20$ post-stroke patients (Ufa, Russia) was given two texts, *A* and *B*. Each text was typeset with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ using ParaType Serif fonts. Half of the participants were given text *A* justified and text *B* ragged right, while the other half had text *B* justified and text *A* ragged right. The participants were asked to read the text. After a minute they marked their current reading position. Immediately after the reading the participants were given a multiple choice test (10 questions with 4 variants of answers to choose from). To test long-term memory, we repeated the test 60 minutes later.

The Babel package and `\selectlanguage{nil}` was used to switch off hyphenation. The justified texts were typeset with the setting `\sloppy`. The ragged right texts were typeset with `\raggedright`.

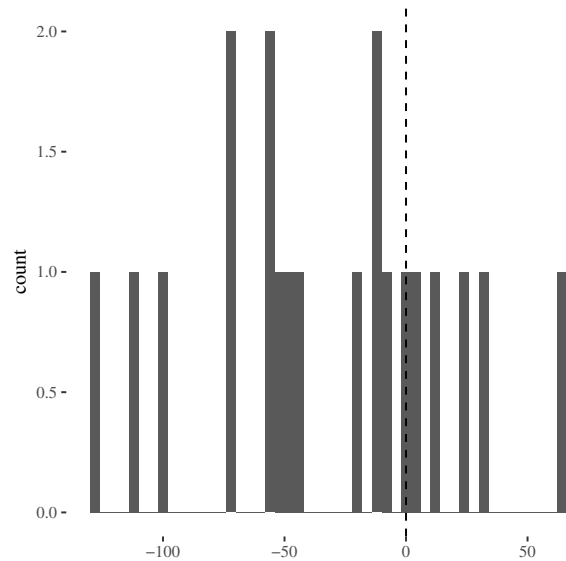


Figure 1: Histogram of difference between justified and ragged right in reading speed results (words per minute).

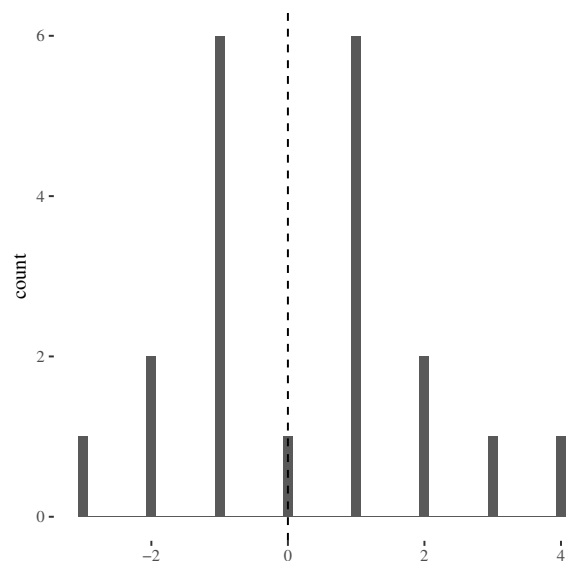


Figure 2: Histogram of difference between justified and ragged right in immediate comprehension results (correct answers).

3 Results

The results of the experiment are shown in Figures 1, 2 and 3. While there is no noticeable difference between justified and ragged right texts with respect to reading comprehension (in either immediate or delayed tests), there *is* a difference in reading speed: sloppily justified texts are being read significantly *slower*: $p = 0.01$. The average difference was -32.3 words per minute.

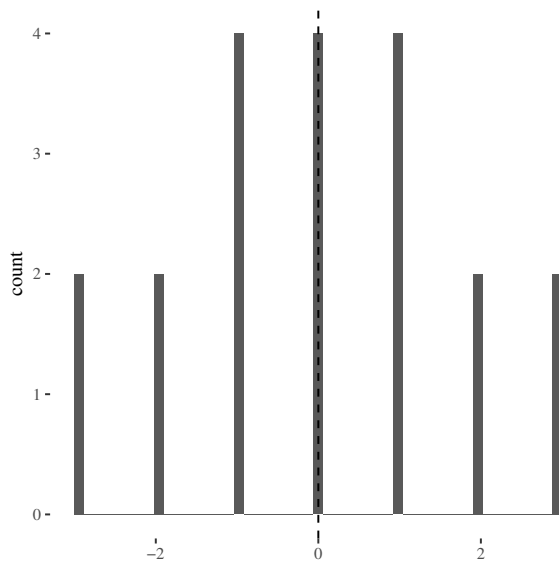


Figure 3: Histogram of difference between justified and ragged right in delayed comprehension results (correct answers).

4 Discussion

The results of this study provide an interesting complement to the conclusions of [1], where the difference in speed of reading between justified and ragged right texts was quite small — unlike our present results.

It is unknown what causes this difference between “sloppily” justified and ragged right texts. One can speculate that uneven spacing between the words produced by “sloppy” justification disturbs reading, especially for patients with cognitive challenges.

Of course our sample was quite small. However, if the results hold, they might have important practical implication. Namely, they suggest that it is better not to justify at all than to justify without hyphenation. Or, to say it succinctly, `\sloppy` is not your friend.

Acknowledgments

We gratefully acknowledge the TUG Bursary grant which made possible the participation of one of the authors in the conference. We are grateful to Darya Popenova, Irina Muhamadieva and Albina Kireeva for their help with the experiments.

References

- [1] Leila Akhmadeeva, Rinat Gizatullin, and Boris Veytsman. Are justification and hyphenation good or bad for the reader? First results. *TUGboat*, 37(2):148–151, 2016. <https://tug.org/TUGboat/tb37-2/tb116akhmadeeva.pdf>.
- [2] Leila Akhmadeeva, Ilnar Tukhvatullin, and Boris Veytsman. Do serifs help in comprehension of printed text? An experiment with Cyrillic readers. *Vision Research*, 65:21–24, 2012.
- [3] Boris Veytsman and Leila Akhmadeeva. Towards evidence-based typography: First results. *TUGboat*, 33(2):156–157, 2012. <https://tug.org/TUGboat/tb33-2/tb104veytsman-typo.pdf>.
- [4] Leila Akhmadeeva and Boris Veytsman. Typography and readability: An experiment with post-stroke patients. *TUGboat*, 35(2):195–197, 2014. <https://tug.org/TUGboat/tb35-2/tb110akhmadeeva.pdf>.


◇ Boris Veytsman
Systems Biology School &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA, 22030, USA
`borisv (at) lk (dot) net`
<http://borisv.lk.net>


◇ Leila Akhmadeeva
Bashkir State Medical University
3 Lenina Str.
Ufa, 450000, Russia
`la (at) ufaneuro (dot) org`
<http://www.ufaneuro.org>


Making ltxsparklines: The journey of a CTAN contributor into the world of CRAN

Boris Veytsman

Edward Tufte defines a sparkline as [...] *a small intense, simple, word-sized graphic with typographic resolution. Sparklines mean that graphics are no longer cartoonish special occasions with captions and boxes, but rather sparkline graphics can be everywhere a word or number can be: embedded in a sentence, table, headline, map, spreadsheet, graphic. Data graphics should have the resolution of typography* [5].

For example, to convey an idea of the evolution over time of TUG membership, we can just insert in the text a simple graph . This graph, better than many words, describes the growth of membership in the pre-Internet era, when the only way to get T_EX was to join TUG — and the relative stability for many years now.

Similarly, the famous data set about the flow of Nile at Aswan can be described by a simple word-like graph  with the gray line showing the interquartile range.

The “resolution of typography” mentioned by Tufte is the natural realm of T_EX. Naturally there is a L^AT_EX package *sparklines* [2] implementing these small graphs. In this package, the code producing a typical sparkline  looks like this:

```
\begin{sparkline}{10}
  \sparkrectangle 0.3 0.8
  \sparkdot 0.5 0.62 gray
  \sparkdot 1 0.2 black
  \spark 0 0 0.1 0.95 0.2 0.8 0.3
          0.3 0.4 0.52 0.5 0.62 0.6
          0.7 0.7 0.5 0.8 0.4 0.9
          0.25 1 0.2 /
\end{sparkline}
```

While the package is quite versatile, and can make many different kinds of sparklines, it leaves all calculations to the user. It would be much more convenient to plot data using a simple command. While it is possible to write a T_EX-only interface for this package from a data processing software also written in pure T_EX, for example, *datatool* [4], in this paper we describe a different approach based on R [3].

Many of us are quite familiar with R. Some, like me, learned to love it after the brilliant lecture by Uwe Ziegenhagen at TUG 2010 [8]. An R-using T_EXnician spends most of her time editing and compiling *.rnw* files. These files look like L^AT_EX, but contain “chunks” of R code. Processed by R, these files become *.tex* files, with R code substituted for the calculation results, figures, etc. The result is a

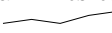


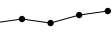
typeset document containing the full report of the research project.

Thus it was a natural decision for me to write an R interface to L^AT_EX *sparklines* package. This interface is released as the R package *ltxsparklines* [6].


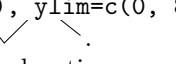
The package defines a single R command, named, naturally enough, `sparkline`. It outputs a L^AT_EX `sparkline` environment. The command has a number of possible arguments. Below these arguments are listed with the default values:

```
sparkline(x = NULL, y = NULL,
  xspikes = NULL, yspikes = NULL,
  xdots = NULL, ydots = NULL,
  dotcolor = NULL,
  width =
    getOption("ltxsparklines.width"),
  rectangle =
    c(NA, NA),
  xlim = c(NA, NA),
  ylim = c(NA, NA),
  clip =
    getOption("ltxsparklines.clip"),
  na.rm =
    getOption("ltxsparklines.na.rm"),
  bottomline =
    getOption("ltxsparklines.bottomline"),
  bottomlinelength =
    NA,
  bottomlinex =
    getOption("ltxsparklines.bottomlinex"),
  startdotcolor =
    getOption("ltxsparklines.startdotcolor"),
  enddotcolor =
    getOption("ltxsparklines.enddotcolor"),
  output =
    getOption('ltxsparklines.output'))
```

The options are fully documented in the package itself. Here we describe the general idea.

Three kinds of sparklines can be created: lines (defined by *x* and *y*) , bars (defined by *xspikes* and *yspikes*) , and dots (defined by *xdots* and *ydots*) . It is possible to combine them, for example, . One can set both *x* and *y* coordinates, or just *y* coordinates. In the latter case the sequence 1, 2, ... will be used for the missing *x* coordinates. Alternatively the command can be given a time series as an argument, and in this case the time/date values will be used for *x* coordinates, and the value of the series for *y* coordinates.

Other arguments can change the appearance of the graphics. Compare, for example, the result of the call `sparkline(c(2, 20, 1, 16, 4),`

```
ylim=c(0, 8), xlim=c(2, 5)): ,
and the similar call clipping the output:
sparkline(c(2, 20, 1, 16, 4), ylim=c(0, 8),
xlim=c(2, 5), clip=TRUE): .
```

There are several color-related options, so one can create bright and distinct sparklines.

With this package the sparklines in the beginning of this paper were typeset as simple as

```
# TUG membership
sparkline(xspikes=tug$Date,
          yspikes=tug$Members,
          ylim=c(0,NA))

# Nile flow
sparkline(Nile,
          rectangle=quantile(Nile,
                              c(0.25, 0.75)),
          enddotcolor='black',
          width=20)
```

Here `tug` is a data frame, obtained by reading the comma separated file `tug.csv` using `R` function `read.csv`, and `Nile` is the time series included in the standard `R` distribution.

There are two ways to generate `.tex` files from `R`: the more traditional `Sweave` package [1,8] and the newer and spiffier `knitr` [7]. (A review of the recent book on `knitr` was published in *TUGboat* 35:1: tug.org/books/reviews/tb109reviews-xie.html.) As might be expected, the `ltxsparklines` package works with both.

This is my first contribution to CRAN, The Comprehensive R Archive Network, and it was interesting for me to observe the differences between CRAN and CTAN. (By the way, CTAN was the first archive network of this kind, and other projects like CRAN and CPAN (for *Perl* developers) used our site as an inspiration.)

It looks as if CRAN has a more strict editorial policy than either CTAN or CPAN. Each package must follow a certain structure including detailed documentation, and contributions not obeying this are automatically rejected by the site upload scripts. After the contribution is approved by the robots, there is the next line: human maintainers. My package was accepted after several rounds of e-mail exchanges with CRAN admins, and anecdotally this is rather the rule than an exception. One of the problems was the compilation of examples. While the official policy of CRAN allows for documentation in PDF format, the admins really wanted it to be compilable on their machines from sources. This was problematic since CRAN machines had a rather old `TeX` installation, and my examples needed a fairly recent version of `LATEX sparklines` package (some of

the recent changes in the `LATEX` package were written by me to facilitate `R` interface). At the end I just included the full code in my `LATEX` source rather than call `\usepackage{sparklines}`. I would venture to say that the interaction with CTAN (or, for this matter, CPAN) admins is easier for the authors than dealing with CRAN. On the other hand, I can understand the rationale behind CRAN strictness.

In summary, it was fun for me to write my first `R` package. I hope it might be useful for fellow `TeX`ers to create dynamic reports with `R`.

References

- [1] Friedrich Leisch and R-core. *Sweave User Manual*, November 2016. <https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf>.
- [2] Andreas Loeffler and Dan Luecking. *Sparklines*, December 2016. <https://ctan.org/pkg/sparklines>.
- [3] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011.
- [4] Nicola L.C. Talbot. *User Manual for datatool bundle*, July 2016. <https://ctan.org/pkg/datatool>.
- [5] E.R. Tufte. *Beautiful Evidence*. Graphics Press, 2006.
- [6] Boris Veytsman. *Package 'ltxsparklines'*, January 2017. CRAN: <https://CRAN.R-project.org/package=ltxsparklines> Github: <https://github.com/borisveytsman/ltxsparklines>.
- [7] Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. ISBN 978-1498716963.
- [8] Uwe Ziegenhagen. Dynamic reporting with `R/Sweave` and `LATEX`. *TUGboat*, 31(2):189–192, 2010. <https://tug.org/TUGboat/tb31-2/tb98ziegenhagen.pdf>.

◇ Boris Veytsman
Systems Biology School &
Computational Materials
Science Center, MS 6A2
George Mason University
Fairfax, VA, 22030, USA
borisv (at) lk (dot) net
<http://borisv.lk.net>

T_EX in Schools? Just Say Yes: The use of T_EX at the Faculty of Informatics, Masaryk University

Petr Sojka and Vít Novotný

Abstract

Students at Masaryk University (MU) use T_EX for many purposes, such as writing theses, essays, and papers. It is also used by the staff for teaching electronic publishing and literate programming, for writing scientific papers, quizzes and teaching resources, and for generating documents and web pages from university databases by the university information system. T_EX and related technologies have been systematically supported and deployed at the Faculty of Informatics of MU (FI MU) for more than two decades. In this paper, we describe the T_EX-related support and projects that we have realized at various levels. These include the design of the Faculty's visual identity, resources for teaching electronic publishing, and for database publishing directly from the University's information system. We evaluate the outcomes, and consider some possible future deployments of T_EX-related technologies. With the data analytics of fithesis3 class support and its use at MU, we give arguments why the answer to the often-asked question in the title is in the affirmative, at least for computer science schools like ours and for authoring math publications.

Why not just hope that in the flow of getting words on a medium we play our humble role and hope we're not forgotten but remembered as inspiration. (Hans Hagen, [7, p. 32])

1 Introduction — basic premises

T_EX was born at a university, in the Stanford Computer Science Department, but primarily for one project of its author. Should it be used and taught widely in schools? Such questions have often been asked and answered [22, 4, 19]. Under which premises and for what purposes should T_EX and its friends be used in schools? The most appropriate answer is that it depends on the type of school, on the tasks, and on the end users:

- T_EX as a programming (macro) language? Probably not.
- T_EX as an example of a literate programming paradigm? Maybe.
- T_EX as a low level typesetting tool? In some cases, it depends on the type of school.
- T_EX in the L^AT_EX format as a reusable scientific authoring markup tool? Probably yes.



Figure 1: Hàn Thế Thành studied at FI MU in Brno from 1991 through 2001

- T_EX as a community building tool? No reason not to.

Working in academia for more than a quarter of a century, let us share our experience with T_EX in the context of the Institute of Computer Science and the FI MU in Brno. The rest of this paper should be understood in this light; the implications are specific to this type of school, place, time and other factors.

Historia magistra vitae (Latin proverb)

2 History of T_EX at Czech schools — just a predilection or an objective good?

Let us start with some historical remarks.

1980s T_EX found its way to Czechoslovakia at the end of the eighties, and was probably first used by the dissidents when preparing books and booklets that were forbidden to be printed officially [5]. For this reason, Czech diacritics had to be added to Computer Modern fonts [47].

1990s Within a year of the Velvet Revolution, the Czechoslovak T_EX Users Group (C_STUG) was founded. With the vast majority of the individual and institutional members of C_STUG being part of academia, high schools and universities became natural hubs of T_EX know-how.

To put this into a historical context — Hàn Thế Thành (Figure 1) came from socialist Vietnam and started to learn Czech at a Czech school and subsequently enrolled in the FI MU. The first Internet ADSL 56 kbps line from Linz in Austria was rented by the consortium of Czech universities to share. And at 290 kB, `latex.tex` was easy to both search and edit even on a PC XT with 640 kB of memory and two floppy diskettes.

As T_EX began to gain momentum, a group of enthusiasts decided to organize a T_EX conference in Prague [48]. Thus, EuroT_EX 92 was born with about 300 participants from all over the world. T_EX started to be used for book and database publishing [40].

A new Czechoslovak variant of the Computer Modern fonts (`csfonts`) was created. Math journals started switching to \TeX . *Czechoslovak Mathematical Journal*, *Applications of Mathematics*, and *Mathematica Bohemica* in Prague, *Archivum Mathematicum* in Brno, and *Mathematica Slovaca* in Bratislava all used \TeX as their primary typesetting tool.

Thus the community was already starting to grow. Groups of mathematicians started to typeset their reviews for the German *Zentralblatt Math* journal, and (\LaTeX) courses started to find their way into schools, primarily as *tools* for typesetting mathematics. One such a course was even taught at TUG 1993 in Aston, UK.

At that time, the first author was working at the Institute of Computer Science, Masaryk University, and promoted the use of \TeX there. There was a series of popular articles about \TeX published in a university bulletin *Zpravodaj MU* and in \mathcal{CS} TUG's bulletin *Zpravodaj \mathcal{CS}TUG*. MU became an institutional member of TUG. \TeX was actively supported and customized versions of \TeX supporting the Latin2 input encoding were created and compiled on shared \TeX installations within the university.

The first computer science faculty in the Czech Republic — the Faculty of Informatics, Masaryk University, Brno (FI MU) — was founded in 1994. Jiří Zlatuška, a proponent of \TeX , became its first dean. The faculty logo was designed by the first author as a *ligature* FI based on Escher's Penrose triangle, as seen in Figure 2. The motto of the logo comes from Blaise Pascal's *Pensées*: “The eternal silence of these infinite spaces terrifies me”.



Figure 2: The logo of the Faculty of Informatics: the ligature FI, as a symbol of quality typography, was implemented in METAFONT [49]. The optically scaled Computer Modern letters in the circular text were recursively joined using the ligature mechanism of METAFONT.

Rozvrh pro skupinu 1MI, 1994–95–léto

	6:50-7:35	7:40-8:25	8:30-9:15	9:20-10:05	10:15-11:00	11:05-11:50	12:00-12:45	12:50-13:35	13:45-14:30	14:35-15:20	15:30-16:15	16:20-17:05	17:10-17:55	18:00-18:45	18:50-19:35
Po	Pr. 1MI-a Cv. teorie množin Hodnotil:	Pr. 1MI-a Cv. lineární algebra Slovák Jan	1MI-a	J. 1MI Angličtina slova zařadí. Kat. jazyk			D. 1MI Sem. z funkcionálního prog. Hajn Pavel	1MI-n Sem. z funkcionálního prog. Skravada Libor			M. 1MI Lineární algebra Slovák Jan				
Út				A. Seminář z programování Seveřková Michaela	1MI-a	A. Seminář z programování Seveřková Michaela	1MI-a	1MI-a Angličtina pok. Kat. jazyk		J. 1MI-a Němčina Kat. jazyk					
St				TV Kat. TV	1MI		J. 1MI-a Angličtina pok. Kat. jazyk	1MI-n Ada 1MI-n Teorie množin Kad'ourek Jiří		D. 1MI Návrh algoritmů I Ochrana Reza					
Čt															

Vytváření zrnky a opravy hlavy na studijní oddělení Ivt Holanová

Figure 3: An example of a timetable for the 1MI study group at FI MU in 1994.

\TeX became the mainstay of everyday life at the Faculty. There was a need to typeset timetables, e.g. for lecture rooms, for individuals and for study groups. \TeX has proven itself to be an ideal tool for the job (see Figure 3). \TeX has been used for the typesetting of almost all database outputs of the Faculty administration [26], including phone directories, course catalogues — as seen in Figure 4 — and study diplomas.

A course on electronic document preparation opened in 1994. It was designed as a blend of both the theory and practice [18] of document preparation. The course teaches students about how information is transferred from the mind of an author via a markup language (\LaTeX) to the reader's mind. They are taught about the separation of form and content and about the particulars of both paper and digital output formats of PDF and (X)HTML. Since document development and program development have much in common, the students are taught to use versioning systems and automation tools such as `make`. As far as \TeX is concerned, the students learn both the practicalities, such as the typesetting of documents with an emphasis on theses, and the theory covering \TeX 's line-breaking and hyphenation algorithms.

Every effort was made to ensure the Faculty was a safe playground to experiment with \TeX toys and tools, for the benefit of all, and as part of the studies [27]. For students like Hàn Thé Thành, \TeX was the obvious choice for typesetting their essays and theses. Hàn Thé Thành picked \TeX and the recently designed PDF format as the topic of his Master's thesis. \TeX has been extensively used by the staff for their academic output and most research publications have been prepared in \TeX . The Faculty's technical report series has been designed in its own

PB029 – Elektronická příprava dokumentů

zk, 2/1, 3 kr., podzim

RNDr. Petr Sojka

–P029

Doporučení: Je vhodné mít základy algoritmizace, základní znalosti práce s počítačem v unixovém prostředí (vhodné absolvovat například předmět P004 *UNIX*) a mít ponětí o formálních jazycích.

Úvod. Vymezení předmětu. Cyklus přípravy a ladění dokumentů. Analogie s vývojem programů. ✦ **Značkování.** Logická vs. vizuální struktura dokumentu. Značkovací jazyky, SGML, XML, HTML. Gramatiky dokumentů, DTD. Validace dokumentů, NSGMLS. ✦ **Design.** Principy knižního designu. Specifika designu na WWW. ✦ **Sazba.** Základy typografie, základní typografické pojmy, míry, terminologie. ✦ Písma, typy formáty písem, způsoby reprezentace a designu písem. Rastrovací algoritmy, techniky redukce tvaru písem. ✦ Pravidla sazby. Mikrotypografie. Specifika sazby českých textů. Korektura, značky. ✦ Sázeční systémy. \TeX jako příklad dávkového sázečního systému. WYSIWYG systémy. DSSSL, XSL. ✦ \TeX . Historie. Princip makrojazyka. Algoritmy řádkového a stránkového zlomu použité v \TeX u. *hz*-systém. Algoritmus dělení slov, ✦ **Předtisková příprava.** Jazyky pro popis stránek. Postscript. Bézierovy křivky. SPDL. Direct Imaging. Archivová montáž. ✦ **Tisk a distribuce.** Výstupní zařízení. Osvět, tisk a vazba. Portable Document Format, Adobe Acrobat. $\mathbb{E}\TeX$ 2html. pdf \TeX . Publikace databází. Konverze, aktualizace a údržba dokumentů. ✦ **Závěrečné shrnutí.** Sdílení zkušeností, anketa.

Doporučená literatura:

- Knuth, Donald Ervin. *Digital typography*. Stanford : Center for the Study of Language and Information, 1999. xv, 685 s.
- Beran, Vladimír. *Typografický manuál : učebnice počítačové typografie*. 1. vyd. Náchod : MANUÁL, 1994. přeruš. st.
- Bringhurst, Robert. *The elements of typographic style*. Vancouver : Hartley & Marks, 1992. 254 s.

Figure 4: The syllabus of the Electronic publishing course typeset in Minion by `pdf \TeX` as a part of the Yellow book of courses taught at FI MU in 2004.

$\mathbb{E}\TeX$ style with Hermann Zapf's Palatino as the faculty's primary font.

To automate the typesetting of longer texts and database publishing, quality hyphenation was required. The results of the first author's research [45, 32] were reported at TUG 1995 (and elsewhere), where the first author met Donald Knuth and took the photo in Figure 5. Don was subsequently invited to Brno to receive his twentieth honorary doctorate.

When he arrived in Brno, Don saw his Computer Modern fonts on the timetables of public transport tram stops (see Figure 7). He was delighted to see the fruits of his 'labour of love' being used on the other side of the globe, both in theory and in practice. He mentioned this in his inaugural speech (Figure 6) when he became the first recipient of an honorary doctorate from FI MU.

In 1996, Hàn Thê Thành defended his masters thesis [10]; the program called `tex2pdf` [31] was presented to the \TeX community at the TUG 1996 conference in Dubna, Russia. The program caught the eye of the \TeX community and was subsequently renamed `pdf \TeX` and its manual was drafted [15].

The new toy needed users willing to test it in day-to-day \TeX authoring work. We maintained faculty-wide installations for multiple operating systems that shared the same `texmf` trees; in addition,



Figure 5: Donald Knuth's finger raised when talking to Jiří Zlatuška at the TUG 1995 conference in Florida; photo taken by Petr Sojka.

we kept historical \TeX Live installations and made them available via a module switching mechanism. Twenty years later, most \TeX Live versions of the past are still installed and ready to use; this makes it easy for authors to go back in time and retypeset decades-old material. Lowering the bar for starting with \TeX , by having the tools ready to use and a local community ready to help, made \TeX the go-to



Figure 6: Donald Knuth’s talk at the Faculty of Informatics, Masaryk University, Brno, 1996

<p>▼ Kohoutovice</p> <p>1 Pavlovská</p> <p>2 Tělichova (o)</p> <p>3 Bellova</p> <p>4 Voříškova</p> <p>5 Sramčova</p> <p>6 Glínkova</p> <p>7 Borodínova (o)</p> <p>8 Libušina třída (o)</p> <p>9 Libušino údolí (z)</p> <p>10 Antonína Procházky (z)</p> <p>12 Pískový</p> <p>13 Výstaviště (o)</p> <p>17 Mendlovo náměstí</p> <p>19 Tvrďého (o)</p> <p>21 Úvoz</p> <p>23 Komenského náměstí</p> <p>z : zastávka celodenně na znamení o : zastávka od 20 do 5 hodin na znamení</p>	<p>137</p> <p>Odjezdy ze zastávky Kohoutovice</p> <p>PRACOVNÍ DNY</p> <table border="1"> <tr><td>0</td><td>30M</td></tr> <tr><td>1</td><td>30M</td></tr> <tr><td>2</td><td>30M</td></tr> <tr><td>3</td><td>30T</td></tr> <tr><td>4</td><td>00 15 30 42 51</td></tr> <tr><td>5</td><td>05 11 16 22 27 33 38 44 49 55</td></tr> <tr><td>6</td><td>00 06 11 17 22 28 33 39 44 50 55</td></tr> <tr><td>7</td><td>01 06 12 17 23 28 33 40 46 54</td></tr> <tr><td>8</td><td>02 08 14 20 26 32 38 44 50</td></tr> <tr><td>9</td><td>02 14 26 38 50</td></tr> <tr><td>10</td><td>02 14 26 38 50</td></tr> <tr><td>11</td><td>02 14 26 38 50</td></tr> <tr><td>12</td><td>02 14 26 38 50</td></tr> <tr><td>13</td><td>02 14 26 38 50</td></tr> <tr><td>14</td><td>02 08 14 20 26 32 38 44 50 56</td></tr> <tr><td>15</td><td>02 07 13 18 24 29 35 40 46 51 57</td></tr> <tr><td>16</td><td>02 09 15 22 28 35 42 43T 49 57</td></tr> <tr><td>17</td><td>05 06T 13 21 24T 30 37 45 54</td></tr> <tr><td>18</td><td>03 12 14T 21 31 40 49</td></tr> <tr><td>19</td><td>00 12 23 33 45 51T 58</td></tr> <tr><td>20</td><td>10 13T 22 34 46 58</td></tr> <tr><td>21</td><td>10 22 34 47 57T</td></tr> <tr><td>22</td><td>02 10T 22 42</td></tr> <tr><td>23</td><td>09T 13M 30M 51T</td></tr> </table>	0	30M	1	30M	2	30M	3	30T	4	00 15 30 42 51	5	05 11 16 22 27 33 38 44 49 55	6	00 06 11 17 22 28 33 39 44 50 55	7	01 06 12 17 23 28 33 40 46 54	8	02 08 14 20 26 32 38 44 50	9	02 14 26 38 50	10	02 14 26 38 50	11	02 14 26 38 50	12	02 14 26 38 50	13	02 14 26 38 50	14	02 08 14 20 26 32 38 44 50 56	15	02 07 13 18 24 29 35 40 46 51 57	16	02 09 15 22 28 35 42 43T 49 57	17	05 06T 13 21 24T 30 37 45 54	18	03 12 14T 21 31 40 49	19	00 12 23 33 45 51T 58	20	10 13T 22 34 46 58	21	10 22 34 47 57T	22	02 10T 22 42	23	09T 13M 30M 51T	<p>linka: 137</p> <p>kód stopu: 137</p> <p>datum: 12.3.1996</p> <p>strana: 1</p>																																															
0	30M																																																																																																
1	30M																																																																																																
2	30M																																																																																																
3	30T																																																																																																
4	00 15 30 42 51																																																																																																
5	05 11 16 22 27 33 38 44 49 55																																																																																																
6	00 06 11 17 22 28 33 39 44 50 55																																																																																																
7	01 06 12 17 23 28 33 40 46 54																																																																																																
8	02 08 14 20 26 32 38 44 50																																																																																																
9	02 14 26 38 50																																																																																																
10	02 14 26 38 50																																																																																																
11	02 14 26 38 50																																																																																																
12	02 14 26 38 50																																																																																																
13	02 14 26 38 50																																																																																																
14	02 08 14 20 26 32 38 44 50 56																																																																																																
15	02 07 13 18 24 29 35 40 46 51 57																																																																																																
16	02 09 15 22 28 35 42 43T 49 57																																																																																																
17	05 06T 13 21 24T 30 37 45 54																																																																																																
18	03 12 14T 21 31 40 49																																																																																																
19	00 12 23 33 45 51T 58																																																																																																
20	10 13T 22 34 46 58																																																																																																
21	10 22 34 47 57T																																																																																																
22	02 10T 22 42																																																																																																
23	09T 13M 30M 51T																																																																																																
<p>SOBOTA</p> <table border="1"> <tr><td>0</td><td>30M</td></tr> <tr><td>1</td><td>30M</td></tr> <tr><td>2</td><td>30M</td></tr> <tr><td>3</td><td>30T</td></tr> <tr><td>4</td><td>00 15 30 45</td></tr> <tr><td>5</td><td>00 15 28 41 54</td></tr> <tr><td>6</td><td>06 18 30 42 54</td></tr> <tr><td>7</td><td>06 18 30 40 50</td></tr> <tr><td>8</td><td>00 08 16 24 32 40 48 56</td></tr> <tr><td>9</td><td>04 12 20 28 36 44 52</td></tr> <tr><td>10</td><td>00 10 20 30 40 50</td></tr> <tr><td>11</td><td>00 10 20 30 36 44 52</td></tr> <tr><td>12</td><td>00 08T 10 20 31 43 54</td></tr> <tr><td>13</td><td>06 08T 17 29 40 52</td></tr> <tr><td>14</td><td>03 15 26 38 49</td></tr> <tr><td>15</td><td>01 12 23 34 45 56</td></tr> <tr><td>16</td><td>06 16 26 36 46 56</td></tr> <tr><td>17</td><td>07 18 29 40 51</td></tr> <tr><td>18</td><td>02 13 24 35 46 58</td></tr> <tr><td>19</td><td>11 23 25T 36 50</td></tr> <tr><td>20</td><td>04 11T 17 30 43 57</td></tr> <tr><td>21</td><td>10 24 37 51</td></tr> <tr><td>22</td><td>04 14 24 43 48T</td></tr> <tr><td>23</td><td>06M 08T 30M 49T</td></tr> </table>	0	30M	1	30M	2	30M	3	30T	4	00 15 30 45	5	00 15 28 41 54	6	06 18 30 42 54	7	06 18 30 40 50	8	00 08 16 24 32 40 48 56	9	04 12 20 28 36 44 52	10	00 10 20 30 40 50	11	00 10 20 30 36 44 52	12	00 08T 10 20 31 43 54	13	06 08T 17 29 40 52	14	03 15 26 38 49	15	01 12 23 34 45 56	16	06 16 26 36 46 56	17	07 18 29 40 51	18	02 13 24 35 46 58	19	11 23 25T 36 50	20	04 11T 17 30 43 57	21	10 24 37 51	22	04 14 24 43 48T	23	06M 08T 30M 49T	<p>NEDELE</p> <table border="1"> <tr><td>0</td><td>30M</td></tr> <tr><td>1</td><td>30M</td></tr> <tr><td>2</td><td>30M</td></tr> <tr><td>3</td><td>30T</td></tr> <tr><td>4</td><td>00 15 30 50</td></tr> <tr><td>5</td><td>05 20 35 50</td></tr> <tr><td>6</td><td>05 20 35 50</td></tr> <tr><td>7</td><td>05 20 35 48</td></tr> <tr><td>8</td><td>03 15 27 39 51</td></tr> <tr><td>9</td><td>03 15 27 39 51</td></tr> <tr><td>10</td><td>03 15 27 39 51</td></tr> <tr><td>11</td><td>05 19 31 40 52</td></tr> <tr><td>12</td><td>04 16 28 40 52</td></tr> <tr><td>13</td><td>04 16 28 39 50</td></tr> <tr><td>14</td><td>01 12 23 34 45 59</td></tr> <tr><td>15</td><td>13 22 28 36 45 53</td></tr> <tr><td>16</td><td>02 11 21 30 39 48 57</td></tr> <tr><td>17</td><td>05 11T 23 31 38 45 51 57</td></tr> <tr><td>18</td><td>03 09 15 23 31 39 47 55</td></tr> <tr><td>19</td><td>03 13 23 27T 36 50 51T</td></tr> <tr><td>20</td><td>04 11T 17 30 43 57</td></tr> <tr><td>21</td><td>10 24 37 51</td></tr> <tr><td>22</td><td>04 14 24 43 48T</td></tr> <tr><td>23</td><td>06M 08T 30M 50T</td></tr> </table> <p>M : jede jen na Mendlovo náměstí T : jede do Husovic na Tomkovo náměstí</p> <p>DPmB, Hlinky 151, tel. 4217 1111 Informace o MHD, tel. 4281 0584</p> <p>Platí od 1.prosince 1995</p>	0	30M	1	30M	2	30M	3	30T	4	00 15 30 50	5	05 20 35 50	6	05 20 35 50	7	05 20 35 48	8	03 15 27 39 51	9	03 15 27 39 51	10	03 15 27 39 51	11	05 19 31 40 52	12	04 16 28 40 52	13	04 16 28 39 50	14	01 12 23 34 45 59	15	13 22 28 36 45 53	16	02 11 21 30 39 48 57	17	05 11T 23 31 38 45 51 57	18	03 09 15 23 31 39 47 55	19	03 13 23 27T 36 50 51T	20	04 11T 17 30 43 57	21	10 24 37 51	22	04 14 24 43 48T	23	06M 08T 30M 50T
0	30M																																																																																																
1	30M																																																																																																
2	30M																																																																																																
3	30T																																																																																																
4	00 15 30 45																																																																																																
5	00 15 28 41 54																																																																																																
6	06 18 30 42 54																																																																																																
7	06 18 30 40 50																																																																																																
8	00 08 16 24 32 40 48 56																																																																																																
9	04 12 20 28 36 44 52																																																																																																
10	00 10 20 30 40 50																																																																																																
11	00 10 20 30 36 44 52																																																																																																
12	00 08T 10 20 31 43 54																																																																																																
13	06 08T 17 29 40 52																																																																																																
14	03 15 26 38 49																																																																																																
15	01 12 23 34 45 56																																																																																																
16	06 16 26 36 46 56																																																																																																
17	07 18 29 40 51																																																																																																
18	02 13 24 35 46 58																																																																																																
19	11 23 25T 36 50																																																																																																
20	04 11T 17 30 43 57																																																																																																
21	10 24 37 51																																																																																																
22	04 14 24 43 48T																																																																																																
23	06M 08T 30M 49T																																																																																																
0	30M																																																																																																
1	30M																																																																																																
2	30M																																																																																																
3	30T																																																																																																
4	00 15 30 50																																																																																																
5	05 20 35 50																																																																																																
6	05 20 35 50																																																																																																
7	05 20 35 48																																																																																																
8	03 15 27 39 51																																																																																																
9	03 15 27 39 51																																																																																																
10	03 15 27 39 51																																																																																																
11	05 19 31 40 52																																																																																																
12	04 16 28 40 52																																																																																																
13	04 16 28 39 50																																																																																																
14	01 12 23 34 45 59																																																																																																
15	13 22 28 36 45 53																																																																																																
16	02 11 21 30 39 48 57																																																																																																
17	05 11T 23 31 38 45 51 57																																																																																																
18	03 09 15 23 31 39 47 55																																																																																																
19	03 13 23 27T 36 50 51T																																																																																																
20	04 11T 17 30 43 57																																																																																																
21	10 24 37 51																																																																																																
22	04 14 24 43 48T																																																																																																
23	06M 08T 30M 50T																																																																																																

Figure 7: Brno public transport timetables featuring Computer Modern fonts during Knuth’s visit to Brno in March 1996.

system for authoring long documents such as books or theses. The fithesis L^AT_EX class for typesetting

Petr Sojka and Vít Novotný

theses was designed, installed and offered to the students. They were given a small booklet “Getting started with T_EX at FI” on enrollment day at the Faculty.

There were conferences being organized by the Faculty, e.g. Gödel in 1996, and a multiconference on the Mathematical Foundations of Computer Science (MFCS) in 1998. T_EX was used for typesetting all conference materials from a single *textual* database; Figure 9 shows one example of this material. In the Seminar on Linux and T_EX (SLT) organized mainly by the students themselves, Linux and T_EX enthusiasts developed not only an interesting research program, but also the icons seen in Figures 8 and 10 drawn by Petra Rychlá.

The information system of the Faculty, also developed partly by the students [26], generated most of its output via a secure independent sandboxed T_EX installation. Data for the course catalogue were acquired from the teachers using web forms, then validated, converted to L^AT_EX, and typeset. The DTD for the validation of the submitted data enabled the use of special entities &T_EX; and &L_AT_EX; ©. Hyphenation pattern were further improved [33] to minimize errors in automated workflows. Students were motivated to actively participate in T_EX-related projects. Mirka Misáková implemented Gutenberg-like justification in METAFONT as a part of her thesis [21], Jan Pazdziora studied line and page breaking algorithms [25], and Pavel Janík studied digital font formats [16]. Most of N_TS [50] was programmed in Brno by the MU alumnus, Karel Skoupý [30].

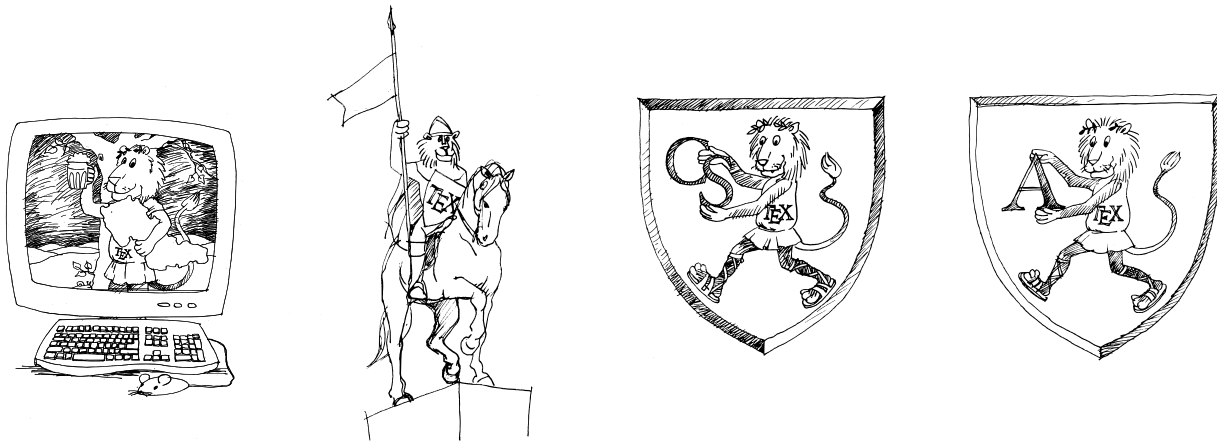


Figure 8: Icons for the Seminar on Linux and \TeX (SLT '98), drawn by Petra Rychlá.



*MFCS/CSL'98 Conference
Welcome Party Invitation
for Mr. Honza Staudek*

*You are wholeheartedly invited to the welcome party
of the MFCS/CSL'98 Conference.*

*The welcome party will take place in the open space
inside the building complex of Faculty of Informatics,
Botanická 68a, at 7:30 PM on Sunday, August 23,
1998. Starobrno beer will be served.*

Please, present this invitation card at the entrance.

Figure 9: A personalized invitation card typeset for the participants of the MFCS '98 conference held at FI MU.

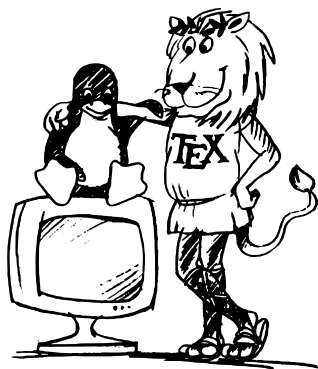


Figure 10: The logo of the Seminar on Linux and \TeX (SLT), drawn by Petra Rychlá.

2000s Hàn Thế Thành consulted on further `pdftex` improvements [11] with Herman Zapf, and conducted several microtypographic experiments together with Hans Hagen who came to give a special course on Typographic Programming in Brno. In October 2000, Hàn Thế Thành finished his dissertation [12], and

left Brno after 11 years of study. He returned to Vietnam, secured his family financially and for a short while worked in Vietnamese academia [13, 14].

As the power of electronic documents and demand for them was increasing, new coursebooks and interactive teaching materials were created [6]. There was demand for animations in PDF [34], for the automation of multiple choice testing [36], and for interactive teaching materials in PDF and JavaScript [35]. \TeX 's notation was so common for the University math teachers that they demanded an extension of the interface for creating online quizzes that would enable them to directly input \LaTeX formulae using a special `$` and `$` element. Math formulae were rendered on the fly via a pipe of \LaTeX to `dvipng`. The software for the automated scanning and evaluation of test sheets generated by \TeX [9], an extended version of `patgen` called `opatgen` that enabled the direct use of UTF-8 patterns [2, 1, 39], and the software for producing animated PDFs in `pdftex` [8] may serve as examples of other \TeX -related tools that have been designed and developed by students and staff at FI. The reuse of textbook content authored in \TeX for multiple output devices was also requested. We have been able to show that, given that form and content are separated in the markup, several different outputs can be easily generated via \TeX , namely PDFs suitable for printing, PDFs suitable for reading on a screen, HTML for web-enabled devices, and XHTML/MathML for fully standards-compliant web-enabled devices [42] without the monstrous systems of large publishers. Our \TeX -based production system is used by most of journals delivering to the DML-CZ library [29, 43].

At the time when \TeX and Knuth became widely known, many software businesses started to move to Brno, which is now known as the Silicon Valley

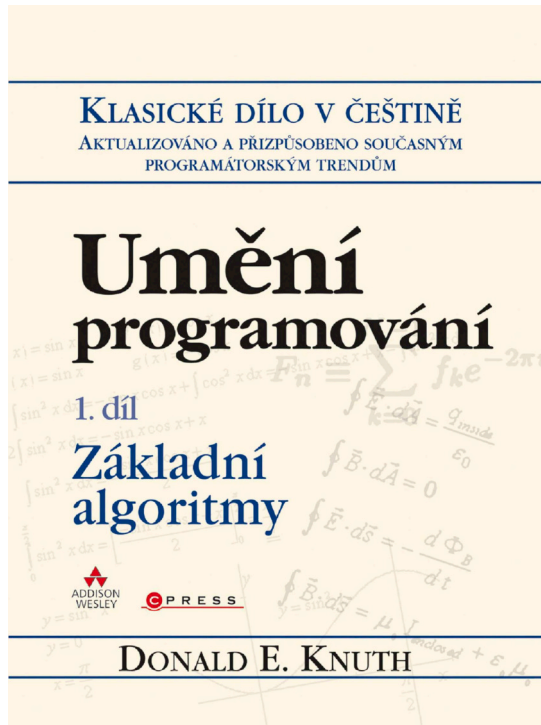


Figure 11: The Czech translation of TAOCP, Vol. 1, published by Computer Press in 2008.

of Central Europe. Consequently, a publisher based in Brno had the Art of Computer Programming (TAOCP) translated into Czech (by a FI MU alumnus) and retypeset from Knuth’s sources (Figure 11).

2010s Leveraging their \TeX typesetting know-how, the students and alumni of FI MU joined several projects related to digital mathematical libraries, namely DML-CZ and EuDML. A \TeX -based workflow for journal publishing has been developed with an automated export of an archival version that would be stored in the digital library. The *Archivum Mathematicum* journal published by MU uses the tools and the workflow developed for DML-CZ [44, 38]. Several related tools have been developed: an efficient PDF recompression technique [41] and the \TeX math indexing and searching algorithm from the MIaS project [20] deployed in EuDML [46]. As blind students needed to study math from \TeX -authored textbooks, support for Czech Braille output has been prepared as part of a master thesis [17].

The creation of \TeX -related software has been supported by the dean’s research project program and offered as a topic for theses. The second author of this paper, supervised by the first author, created a new version of the *fithesis* class [23] with fine-tuned support for all nine faculties of Masaryk University. Thousands of students across the university now au-

thor their theses in \LaTeX with the ability to discuss problems via a dedicated forum in the university information system. Students have also started to file pull requests to customize style options of other faculties, a sign of a growing faculty-local \TeX support community.

Another development was triggered by the inability of *markdown* to prevent the occurrence of Czech vowelless prepositions at the end of lines in FI MU senate minutes, which is a grave error according to Czech typography rules. The new *markdown.tex* macro package that enables the processing of *markdown* documents directly in \TeX solves this issue as a tiny side-effect [24].

The fruits of separating form and content were recently reaped when Masaryk University changed the visual style for their documents. Changes in the \TeX -based workflow were minimal and did not affect the authors much — a *muletter* style file for preparing letters, and a thesis review document template were put on the Faculty’s GitLab server shortly after the new visual style was released and smoothed the transition significantly.

The use of \TeX at MU currently celebrates a quarter century of support and development, where students and staff have contributed significantly both to the questions and solutions in the digital typography world and especially within the 40-year-old \TeX family.

So, maybe instead of ambitious themes, the only theme that matters is: show what you did and how you did it. (Hans Hagen, [7, p. 32])

3 Where we are now and what’s next — predictions

Nelson Beebe predicted the future of \TeX more than a decade ago [3]. The world we live in constantly changes, and while most predictions still hold, some have to be revisited. We have tried to evaluate the influence of the \TeX tools and predilections using statistical data about theses defended not only at FI MU, but across the entire university.

With the creation of the *fithesis3* \LaTeX class, the level of support for thesis writing entered a new era [23]. Templates in *fithesis3* were prepared for each of the nine faculties of Masaryk University. Writing a thesis is now just a few clicks away even if the author does not have a working local \TeX installation. Enchanted by the ease of the authoring process and the beauty of the resulting documents, it seems likely that many will install \TeX on their devices at some stage. Cloud \TeX environments enable much faster learning by example than before, and allow for

online consulting, commenting by supervisors, and collaborative debugging.

The portability, stability, reliability, and style uniformity enforced implicitly by visible markup, the ease of writing math, as well as the aesthetic and visual qualities of the output are the main benefits compared to WYSIWYG editor alternatives. This is attractive for students, as can be seen in Figure 12.

In parallel, a beamer theme `fibeamer` has been developed and made available in \TeX Live and cloud \TeX platforms to allow the students to prepare their presentations for thesis defense without having to bother about the visual style of their slides. This appears to be especially attractive for the students of the Faculty of Arts — see Figure 13.

Approximately 40,000 students study at Masaryk University and all theses defended are archived in the university information system. We have used heuristics to detect whether a thesis has been written in \TeX on a sample of 44,875 theses submitted at MU from 2010 through 2015. It is estimated that the number of theses written in \TeX across the entire University steadily increased from 5.67% in 2010 to 6.28% in 2014. Extrapolating this trend indicates that 100% of theses will be written in \TeX by 2783 ☺.

Theses written using \TeX had been awarded grade A statistically significantly more often and grades C and D statistically significantly less often than theses not written using \TeX [23]. The awarded grades are summarized in Table 1 and in Figure 14. There is clear evidence that theses written in \TeX received better grades than theses written using different tools. It remains to be shown that the grades students received for theses written in \TeX are consistently better than the grades the students received for their state exams — the hypothesis is that using \TeX helped the students reach grades that do not correspond to their ability to study in general.

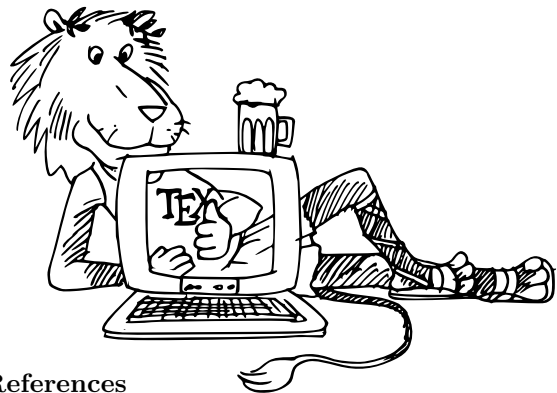
To conclude, the main lessons learned from \TeX at MU are:

- Sustainable support for [thesis] writing in \TeX and incentives for community building by university are very important. There should ideally be a playground where students and faculty members can play and experiment together, work on joint projects, and have fun.
- Using \TeX in the daily agenda of the university is motivating, and is a win-win situation for both students and faculty members — the students learn new things while the faculty administration and teaching is effective and enjoyable.
- The \TeX typesetting kernel gives visually appealing results, often superior when compared

to other alternatives, especially when math typesetting is needed, as in STEM education.

- Contrary to most WYSIWYG alternatives, the use of \TeX gives consistent results, is productive and efficient for database and automated publishing, and for long documents containing math. It is a safe choice, especially when there is official support.
- The separation of form and content and \TeX as a fixed point in document authoring is another benefit academics recognize in their ever-changing world: it allows reusing content in different portable forms and formats that appear over time.
- The usage of \TeX as a typesetting kernel in a university information system has paid off in decades of use.

Young, smart students who enjoy playing with \TeX document tools are constantly appearing, joining the community, and taking on ambitious new \TeX -related projects and challenges. This allows the retiring faculty members to take a well-earned rest.



References

- [1] David Antoš. PATLIB, Pattern Manipulation Library. Master's thesis, 2002. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/3077/fi_m/.
- [2] David Antoš and Petr Sojka. Pattern Generation Revisited. In Simon Pepping, editor, *Proceedings of the 16th European TeX Conference, Kerkrade, 2001*, pages 7–17, Kerkrade, The Netherlands, September 2001. NTG. www.ntg.nl/EuroTeX/2001/.
- [3] 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. tug.org/TUGboat/tb25-1/beebe-2003keynote.pdf.
- [4] Al Cuoco. \TeX in schools: Why not? *TUGboat*, 12(2):303–304, June 1991. tug.org/TUGboat/tb12-2/tb32letters.pdf.

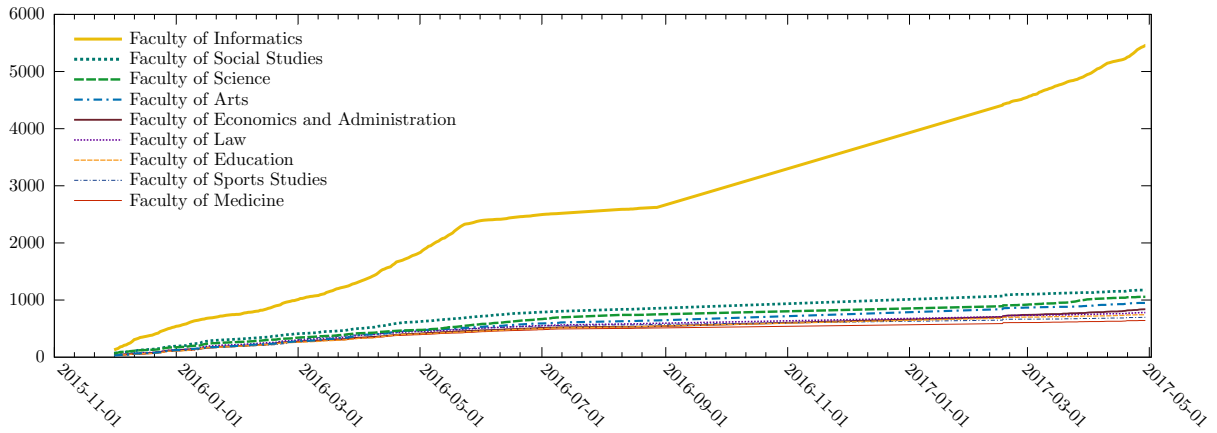


Figure 12: The cumulative number of views of the `fithesis3` document class in the online service of Overleaf.

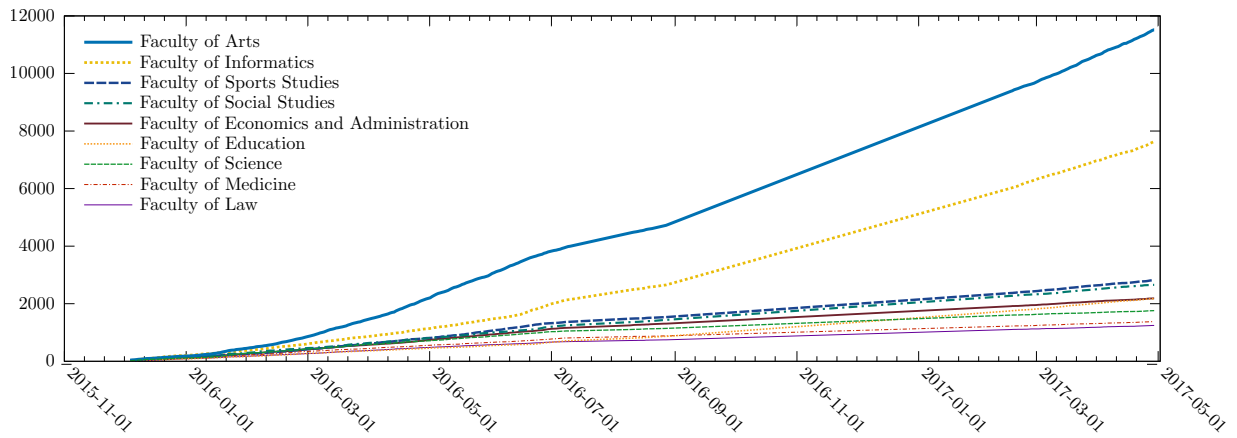


Figure 13: The cumulative number of views of the `fibeamer` beamer theme in the online service of Overleaf.

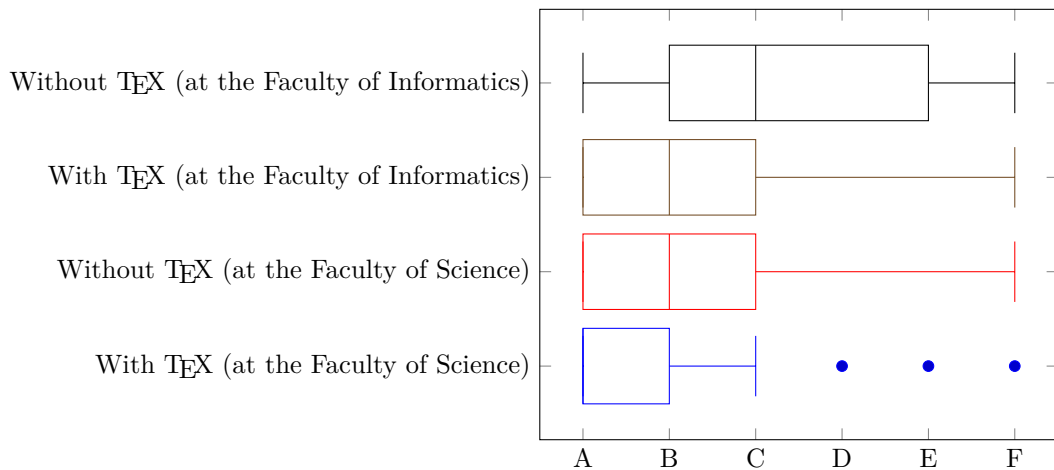


Figure 14: A box plot of the grades of theses written and defended during 2010–2015 at the Faculty of Informatics (FI MU), the Faculty of Science (Sci MU).

Table 1: The contingency table of the numbers of marks awarded to theses written and defended during 2010–2015 with Pearson’s goodness-of-fit measure $(E - O)^2/E$ between the expected (E) and the observed (O) numbers of marks awarded to theses written using \TeX .

Grade	Without \TeX	E(with \TeX)	O(with \TeX)	$(E - O)^2/E$
A	15,476	988	1,181	37.858
B	9,999	638	587	4.093
C	7,926	506	381	30.799
D	4,020	257	194	15.248
E	2,783	178	128	13.853
F	1,979	126	145	2.771
Total	42,183	2,692	2,692	104.623

- [5] Barbara Day. *The Velvet Philosophers*. A&C Black, 1999.
- [6] Zuzana Došlá, Roman Plch, and Petr Sojka. Matematická analýza s programem Maple: 2. Nekonečné řady. CD-ROM, www.math.muni.cz/~plch/nkpm/, December 2002.
- [7] Hans Hagen. Children of \TeX . In Przechlewski et al. [28], pages 18–32.
- [8] Jan Holeček and Petr Sojka. Animations in a pdf \TeX -generated PDF. *TUGboat*, 25:35–41, April 2004.
- [9] Miroslav Hrad and Petr Sojka. Automation of Typesetting and Scanning of Forms (in Czech). *Zpravodaj ČS \TeX* , 12(3–4):123–139, 2002.
- [10] Hàn Thế Thành. Portable Document Format and Typesetting System \TeX (in Czech). Master’s thesis, April 1996. Masaryk University, Brno, Faculty of Informatics (advisor: Jiří Zlatuška).
- [11] Hàn Thế Thành. Improving \TeX ’s Typeset Layout. *TUGboat*, 19(3):284–288, September 1998. tug.org/TUGboat/tb19-3/tb60than.pdf.
- [12] Hàn Thế Thành. Micro-typographic extensions to the \TeX typesetting system. *TUGboat*, 21(4):317–434, December 2000. tug.org/TUGboat/tb21-4/tb69thanh.pdf.
- [13] Hàn Thế Thành. Margin kerning and font expansion with pdf \TeX . *TUGboat*, 22(3):146–148, September 2001. tug.org/TUGboat/tb22-3/tb72thanh.pdf.
- [14] Hàn Thế Thành. Micro-typographic extensions of pdf \TeX in practice. *TUGboat*, 25(1):35–38, 2004. tug.org/TUGboat/tb25-1/thanh.pdf.
- [15] Hàn Thế Thành and Sebastian Rahtz. The pdf \TeX user manual. *TUGboat*, 18(4):249–254, December 1997. tug.org/TUGboat/tb18-4/tb57than.pdf.
- [16] Pavel Janík. Digital Font Formats in Computer Typesetting (in Czech). Master’s thesis, March 2000. Masaryk University, Brno, Faculty of Science (advisor: Petr Sojka), is.muni.cz/th/3267/fi_m/.
- [17] Martin Jarmar. Conversion of Mathematical Documents into Braille. Master’s thesis, January 2012. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/172981/fi_m/.
- [18] Donald E. Knuth. Theory and practice. Keynote address for the 11th World Computer Congress (Information Processing ’89), August 1989.
- [19] Simon Laube. \TeX in Schools: Just Say Yes! *TUGboat*, 36(3):188–189, 2015. tug.org/TUGboat/tb36-3/tb114laube.pdf.
- [20] Martin Líška. Evaluation of Mathematics Retrieval. Master’s thesis, January 2013. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/255768/fi_m/.
- [21] Miroslava Misáková. Typography of Quality in Computer Typesetting (in Czech). Master’s thesis, 1998. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/2660/fi_m/.
- [22] Konrad Neuwirth. \TeX in Schools: Just Say No. *TUGboat*, 12(1):171–174, March 1991. tug.org/TUGboat/tb12-1/tb31kneuwirth.pdf.
- [23] Vít Novotný. Form of Theses Written in \LaTeX (in Czech). Bachelor’s thesis, 2015. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/409729/fi_b/.
- [24] Vít Novotný. Using Markdown inside \TeX Documents. In Przechlewski et al. [28], pages 50–53.
- [25] Jan Pazdziora. Algorithms of Line and Page Breaking in Computer Typesetting (in Czech). Master’s thesis, 1997. Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), is.muni.cz/th/2644/fi_m/.
- [26] Jan Pazdziora and Michal Brandejs. University Information System Fully Based on WWW. In *ICEIS 2000 Proceedings*, pages 467–471. Escola Superior de Tecnologia do Instituto Politécnico de Setúbal, 2000. is.muni.cz/auth/clanky/2000_ICEIS.pl.

- [27] Zuzana Popelková. Macros for Typesetting of Timetables (in Czech). Bachelor's thesis, January 2001. Masaryk University, Brno, Faculty of Informatics (advisor: Libor Škarvada), is.muni.cz/th/3839/fi_b/.
- [28] Tomasz Przechlewski, Karl Berry, and Jerzy Ludwiczowski, editors. *XXV Międzynarodowa Konferencja Użytkowników Systemu T_EX: Materiały konferencyjne*. GUST, 2017.
- [29] Michal Růžička. Automated Processing of T_EX-typeset Articles for a Digital Library. In Sojka [37], pages 167–176. dml.cz/dmlcz/702564.
- [30] Karel Skoupý. *N_TS*: a New Typesetting System. *TUGboat*, 19(3):318–322, September 1998. tug.org/TUGboat/tb19-3/tb60nts.pdf.
- [31] Petr Sojka, Hàn Thê Thành, and Jiří Zlatuška. The Joy of T_EX2PDF — Acrobatics with an alternative to DVI format. *TUGboat*, 17(3):244–251, 1996. tug.org/TUGboat/tb17-3/tb52sojk.pdf.
- [32] Petr Sojka. Notes on compound word hyphenation in T_EX. *TUGboat*, 16(3):290–296, September 1995. tug.org/TUGboat/tb16-3/tb48soj2.pdf.
- [33] Petr Sojka. Hyphenation on Demand. *TUGboat*, 20(3):241–247, 1999. tug.org/TUGboat/tb20-3/tb64sojka.pdf.
- [34] Petr Sojka. Animations in PDF. In *Proceedings of the 8th SIGCSE Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2003*, page 263, Thessaloniki, 2003. Association for Computing Machinery.
- [35] Petr Sojka. Interactive Teaching Materials in PDF using JavaScript. In *Proceedings of the 8th SIGCSE Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2003*, page 275, Thessaloniki, 2003. Association for Computing Machinery.
- [36] Petr Sojka. Rapid Evaluation using Multiple Choice Tests and T_EX. In *Proceedings of the 8th SIGCSE Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2003*, page 265, Thessaloniki, 2003. Association for Computing Machinery.
- [37] Petr Sojka, editor. *Towards a Digital Mathematics Library*, Birmingham, UK, July 2008. Masaryk University. dml.cz/dmlcz/702564.
- [38] Petr Sojka. Digitization Workflow in the Czech Digital Mathematics Library. In Ruyong Feng, Wen-shin Lee, and Yosuke Sato, editors, *Computer Mathematics*, pages 147–156. Springer-Verlag, October 2014.
- [39] Petr Sojka and David Antoš. Context Sensitive Pattern Based Segmentation: A Thai Challenge. In Pat Hall and Durgesh D. Rao, editors, *Proceedings of EACL 2003 Workshop on Computational Linguistics for South Asian Languages — Expanding Synergies with Europe*, pages 65–72, Budapest, April 2003.
- [40] Petr Sojka, Rudolf Červenka, and Martin Svoboda. T_EX for database publishing. In Zlatuška [48], pages 53–58.
- [41] Petr Sojka and Radim Hatlapatka. Document Engineering for a Digital Library: PDF recompression using JBIG2 and other optimization of PDF documents. In *Proceedings of the ACM Conference on Document Engineering, DocEng 2010*, pages 3–12, Manchester, September 2010. Association for Computing Machinery. portal.acm.org/citation.cfm?id=1860563.
- [42] Petr Sojka and Roman Plch. Technological Challenges of Teaching Mathematics in a Blended Learning Environment. *International Journal of Continuing Engineering Education and Life-Long Learning*, 18(5-6):657–665, 2008.
- [43] Petr Sojka and Jiří Rákosník. From Pixels and Minds to the Mathematical Knowledge in a Digital Library. In Sojka [37], pages 17–27. dml.cz/dmlcz/702564.
- [44] Petr Sojka and Michal Růžička. Single-source publishing in multiple formats for different output devices. *TUGboat*, 29(1):118–124, 2008. tug.org/TUGboat/tb29-1/tb91sojka.pdf.
- [45] Petr Sojka and Pavel Ševeček. Hyphenation in T_EX — Quo Vadis? *TUGboat*, 16(3):280–289, September 1995. tug.org/TUGboat/tb16-3/tb48soj1.pdf.
- [46] Wojtek Sylwestrzak, José Borbinha, Thierry Bouche, Aleksander Nowiński, and Petr Sojka. EuDML—Towards the European Digital Mathematics Library. In Petr Sojka, editor, *Proceedings of DML 2010*, pages 11–24, Paris, France, July 2010. Masaryk University. dml.cz/dmlcz/702569.
- [47] Jiří Zlatuška. Automatic generation of virtual fonts with accented letters for T_EX. *Cahiers GUTenberg*, 10–11:57–68, September 1991.
- [48] Jiří Zlatuška, editor. *Proceedings of the 7th European T_EX Conference, Prague, 1992*. Masaryk University, Brno, September 1992.
- [49] Jiří Zlatuška. When METAFONT does it alone. *TUGboat*, 16(3):227–232, September 1995. tug.org/TUGboat/tb16-3/tb48zlat.pdf.
- [50] Jiří Zlatuška. *N_TS*: Programming Languages and Paradigms. In *EuroT_EX Proceedings*, pages 241–246, Heidelberg, 1999. DANTE.
- ◇ Petr Sojka
The Faculty of Informatics at Masaryk University
Brno, Czech Republic
[sojka \(at\) fi dot muni dot cz](mailto:sojka@fi.muni.cz)
- ◇ Vít Novotný
The Faculty of Informatics at Masaryk University
Brno, Czech Republic
[witiko \(at\) mail dot muni dot cz](mailto:witiko@mail.muni.cz)

Implementing bioinformatics algorithms in \TeX — the Gotoh package, a case study

Takuto Asakura

Abstract

\TeX is appropriate for implementing many bioinformatics algorithms because they can be programmed with short codes, calculated with a limited range of numbers, and produce visual results. As a case study, I present Gotoh, a \LaTeX package which implements the Gotoh algorithm, a popular biological sequence alignment algorithm.

1 Motivation

\TeX makes for a good programming language to implement many bioinformatics algorithms, such as those for sequence alignment. There are several reasons for this.

First, code for such algorithms tends to be brief. While it is theoretically possible to program even complex algorithms with \TeX since it is a Turing machine, it is difficult to write algorithms requiring lengthy source code. Sequence alignment algorithms can be stated with a few lines of recursions.

Secondly, the calculation processes use only a limited range of numbers (usually integers), making it possible to easily store them in \TeX 's registers. Though the exact limit of what \TeX can handle depends on the computing environment, they are usually within the range of what is required by bioinformatics algorithms.

Thirdly, bioinformatics algorithms often build visual output such as charts and strings. These results can be easily incorporated into documents produced by \TeX . They can also be utilized as \LaTeX packages. As \LaTeX is one of the most widely used front-end systems for typesetting academic papers, it is convenient for researchers if the algorithms that generate contents that go directly into the papers are available as \LaTeX packages. Users of the packages do not need to execute any commands other than `latex`, and they are freed from the hassles of installing and understanding dedicated tools. It is also possible to link seamlessly with a number of other \LaTeX packages.

Finally, the stability of the \TeX macro language provides for a long-lasting code repository for bioinformatics algorithms. Since the primitives designed by Knuth are extremely stable (Knuth, 1990), an implementation that uses these primitives will continue to function for a long time. However, this might not be necessarily true for implementations using primitives which are available in other engines.

A

1	2	3	4	5
G	A	C	T	A
G	A	.	G	A

B

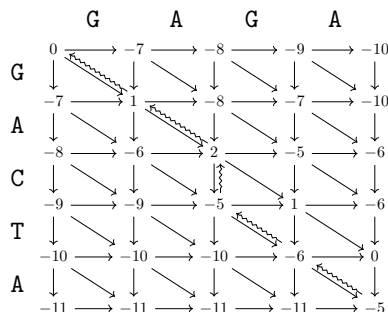


Figure 1: A. An example of pairwise DNA sequence alignment. Here, the third column is a *gap*, the fourth column is a *mismatch* and the others are *matches*. B. The edit graph corresponding to the matrix H . The squiggly arrows (\rightsquigarrow) show the result of trace back.

Here, I consider the Gotoh algorithm, a popular sequence alignment algorithm and implement it in \TeX . I also show how to produce publication-ready output by combining my package with other \LaTeX packages.

2 Sequence alignment

Sequence alignment is often used in bioinformatics to compare the similarity of biological sequences such as DNA, RNA, and amino acid sequences. In the pairwise sequence alignment problem, we are given a pair of sequences

$$A \equiv a_1 a_2 a_3 \dots a_m, \quad B \equiv b_1 b_2 b_3 \dots b_n$$

where a_i and b_j are chosen from a finite alphabet, e.g. $\{A, T, G, C\}$, and the output is a sequence alignment (Figure 1A).

The Longest Common Subsequence (LCS) problem, which is strongly related to the `diff` utility, can be considered as a simple form of sequence alignment in which we score 1 for a *match* and 0 for a *gap*. The optimal score $s_{m,n}$ can be evaluated with the following dynamic programming recursion:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1. \end{cases}$$

Sequence alignment can be solved by a similar approach, though scoring schemes can be slightly more complex, e.g.

$$\begin{aligned} \text{match} &= c_+, \quad \text{mismatch} = c_-, \\ g(l) &= -d - (l-1)e, \end{aligned}$$

where c_+, c_-, d, e are fixed integers, and $g(l)$ is a penalty for an l -length *gap*. One of the most well-known solutions for the problem is the Needleman–Wunsch algorithm (Needleman and Wunsch, 1970; Waterman, Smith, and Beyer, 1976), which calculates the optimal score using the recursion:

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + c_{ij} \\ H_{i-k,j} + g(k) \\ H_{i,j-k} + g(k) \end{cases} \quad (1)$$

where

$$c_{ij} = \begin{cases} c_+ & \text{if } a_i = b_j \quad (\text{match}) \\ c_- & \text{otherwise} \quad (\text{mismatch}). \end{cases}$$

After calculating the entries of the dynamic programming matrix H , an optimal alignment can be obtained by trace back of the edit graph (Figure 1B).

3 The Gotoh package

Whereas the Needleman–Wunsch algorithm requires $O(m^2n)$ time, the Gotoh algorithm (Gotoh, 1982) solves the same problem in $O(mn)$ time. The Gotoh package is an implementation of this algorithm. It is available from CTAN.

3.1 Algorithm

The Gotoh algorithm uses the following formulae transformed from Equation (1):

$$M_{i+1,j+1} = \max \{M_{ij}, I_{ij}^x, I_{ij}^y\} + c_{ij}$$

where

$$I_{i+1,j}^x = \max \{M_{ij} - d, I_{ij}^x - e, I_{ij}^y - d\}$$

and

$$I_{i,j+1}^y = \max \{M_{ij} - d, I_{ij}^y - e\}.$$

An optimal alignment can be obtained by trace back of the three edit graphs corresponding to the matrices M, I^x, I^y . I omit the details.

3.2 Usage and features

The Gotoh package provides two commands: `\Gotoh` for executing the algorithm and `\GotohConfig` for setting parameters with a key–value interface. The package is implemented with only primitives specified by Knuth and some L^AT_EX macros; it also requires the `xkeyval` package (Adriaens, 2014).

The usage of `\Gotoh` is simple (Figure 2). This command takes two sequences, assigns the optimal score to `\GotohScore`, and returns the alignment to `\GotohResultA` and `\GotohResultB`. Macros to store the score and results can be changed with the `\GotohConfig` command as follows.

```
\GotohConfig{
  score = \GotohScore,
  result A = \GotohResultA,
  result B = \GotohResultB}
```

```
A \Gotoh{<sequence A>}{<sequence B>}
B \Gotoh{ATCGGCGCACGGGGGA}{TTCCGCCAC}
  \texttt{\GotohResultA} \ \
  \texttt{\GotohResultB}
C ATCGGCGCACGGGGGA
  TTCCGCCAC.....A
```

Figure 2: Usage of `\Gotoh`. A. Command syntax. B and C. Simple example input and its output. This alignment was calculated with the default parameters of the Gotoh package, which are shown in Equation (2), and the optimal score is -6 .

The Gotoh package by default uses the scoring parameters:

$$c_+ = 1, c_- = -1, d = 7, e = 1. \quad (2)$$

They also can be set with `\GotohConfig` as follows.

```
\GotohConfig{
  match = 1, mismatch = -1, d = 7, e = 1}
```

3.3 Collaborating with T_EXshade

T_EXshade is a L^AT_EX package designed for typesetting, shading, and labeling preprocessed sequence alignments (Beitz, 2000). This package is also available from CTAN. The Gotoh package can be easily combined with this package.

For example, suppose you define the following macros in the preamble of a L^AT_EX document.

```
% output file
\newwrite\FASTAfile
\def\writeFASTA#1{%
  \immediate\write\FASTAfile{#1}}

% print alignment
\newcommand{\PrintAlignment}[3][\relax]{%
  \Gotoh{#2}{#3}%
  \immediate\openout\FASTAfile=\jobname.fasta
  \writeFASTA{> Seq 1^J\GotohResultA}%
  \writeFASTA{> Seq 2^J\GotohResultB}%
  \immediate\closeout\FASTAfile
  \texshade{\jobname.fasta}#1\endtexshade}
```

At this point, by simply including `\PrintAlignment` in the document, the Gotoh algorithm is executed, the result is formatted by T_EXshade, and is output directly in your article (Figure 3).

Note that `\PrintAlignment` communicates between the two packages via a FASTA file, a simple and standard bioinformatics format for recording sequences. This is because T_EXshade does not have any user interface to read sequences directly from L^AT_EX sources (Beitz, 2011). Even so, this macro requires only one `latex` execution.

A `\PrintAlignment[⟨TeXshade commands⟩]{⟨sequence A⟩}{⟨sequence B⟩}`

B

```

seq1      .....GGAGTGAGGGGAGCAGTTGGGC TGAAGATGGTCAA CGCCGAGGGAACG 48
seq2      CGCATGCGGAGTGAGGGGAGCAGTTGGG. AACAGATGGTC. CGCCGAGGGAACG 53
consensus *****!!!!!!!!!!!!!!!!!!!!!!!!!!!!* !!!!!!!!!!!* !!!!!!!!!!!!!!! !!

seq1      GTAAAGGCGACGG...AGCTGTGGCAGACCTGGCTTCCTAACCACGTCCCGTGT 99
seq2      GT. GGGCAGACGGGGCCAGCTGTGGCAGACACTGGCTTCCTAACCACGAACGT. T 106
consensus !!* ! !!!!!*****!!!!!!!!!!!!!!!!!!!!!! ! !!!!!!!!!!!!!!! !!*!

seq1      TTTGCGGCTCCGCGAGGACTG 120
seq2      CTTTCCGCTCCG.....GG 120
consensus !! ! !!!!!***** !
    
```

Figure 3: Usage of the macro `\PrintAlignment`. A. Command syntax. The first argument `⟨TeXshade commands⟩` is optional. B. A sample output.

4 Future directions

It would be conceivable to add to Gotoh a few user interfaces for easier cooperation with other packages that deal with biological sequences. Functional extensions to display more detailed information, such as edit graphs, may also be beneficial. It will be interesting to develop related packages, for instance, one which provides the functionality of multiple-sequence alignments.

Furthermore, it is also interesting to write \TeX implementations of algorithms producing visual results to be incorporated into documents. For example, it would be useful if \LaTeX packages for printing source code such as listings have a `diff` function.

5 Acknowledgements

I would like to thank Shun Sakuraba for his engaging lecture on the Gotoh algorithm which inspired me to develop this package. I am grateful to Anish M. S. Shrestha for helping with the manuscript.

References

Adriaens, Hendri. “The xkeyval package (v2.7a)”. <https://ctan.org/pkg/xkeyval>, 2014.

Beitz, Eric. “ \TeXshade : shading and labeling of multiple sequence alignments using $\text{\LaTeX} 2_{\epsilon}$ ”. *Bioinformatics* **16**(2), 135–139, 2000.

Beitz, Eric. “The \TeXshade package (v1.24)”. <https://ctan.org/pkg/texshade>, 2011.

Gotoh, Osamu. “An improved algorithm for matching biological sequences”. *Journal of Molecular Biology* **162**(3), 705–708, 1982.

Knuth, Donald E. “The future of \TeX and METAFONT”. *TUGboat* **11**(4), 1990. <https://tug.org/TUGboat/tb11-4/tb30knut.pdf>.

Needleman, Saul B., and C. D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. *Journal of Molecular Biology* **48**(3), 443–453, 1970.

Waterman, Michael S, T. F. Smith, and W. A. Beyer. “Some biological sequence metrics”. *Advances in Mathematics* **20**(3), 367–387, 1976.

◇ Takuto Asakura
 The University of Tokyo
 Department of Bioinformatics and
 Systems Biology
 2-11-16 Yayoi
 Bunkyo, Tokyo, 113-0032
 Japan
 tkt.asakura (at) gmail dot com

updmap and fmtutil — past and future changes (or: cleaning up the mess)

Norbert Preining

Abstract

This article serves first as an introduction to two of the central utility programs in any T_EX Live installation, `updmap` and `fmtutil`, describing the general functionality as well as the syntax of the configuration files. In addition, we report on changes that we have carried out over the last few years relating to the operation mode. These changes include switching to multiple configuration files, and the user-mode versus system-mode changes to be introduced in T_EX Live 2017. Last but not least, we close with a list of best practices to help guide users.

If you only want to know how best to install fonts (or formats) and are not particularly interested in the details, jump to Section 5.

1 Introduction

Two central utility programs in any T_EX installation are `updmap`, responsible for creating font maps for various programs, and `fmtutil`, responsible for (re)creating format dumps.

For many years the venerable shell scripts by Thomas Esser were used on Unix-like systems with only minimal changes. For Windows, T_EX Live used binary programs developed independently. Having two independent implementations hindered development of new features. Thus, some years ago we started rewriting them in Perl: first `updmap` (T_EX Live 2012), and later `fmtutil` (2015).

With the rewrites in place, the first new feature added was already a considerable change in internal behavior: While the original shell scripts used a single configuration file, the new versions read configuration files on a per-tree basis. This helped users preserve their configuration across TL upgrades, and gave OS distributions better ways of integration into their respective packaging infrastructure.

With T_EX Live 2017, we will go further and eliminate the biggest source of confusion: Users invoking the scripts in the so-called *user mode* (in contrast to *system mode*), thus generating local configuration files shadowing the global ones. The origin of this confusion is the widespread misinformation to call simply `updmap` (`fmtutil`) when the available fonts change.

T_EX Live 2017 and later disable calls to `updmap` and `fmtutil` without an explicit mode request. This means that users who unknowingly call them will get a warning message — and hopefully afterwards will use the right mode.

1.1 Layout of the article

Section 2 will start with an explanation of the functionality of the scripts and how they fit into a T_EX (Live) installation. While the general functionality of these scripts will be similar in other T_EX distributions, some options described here are probably not available in other installations. In this section we also introduce the original system and user modes.

Section 3 describes the changes introduced with multiple configuration files, and explains how this can be used in single and multi-user environments.

Section 4 introduces the changed operational mode introduced in T_EX Live 2017.

Section 5 has recommendations and best practices for dealing with local fonts and formats.

A running example for the installation of the MathProII fonts will exhibit the usage changes.

2 Functionality of updmap and fmtutil

Although `updmap` and `fmtutil` are central to T_EX operations and are automatically executed on many occasions, both scripts have remained relatively mysterious and are often misused.

2.1 updmap

Many of the fonts shipped in a T_EX system are PostScript Type 1 fonts. The original T_EX does not know anything about this (or any glyph) font format; it only uses the metrics from TFM files. The output drivers on the other hand need to know how TFM names map to glyphs. Typical output drivers are

pdf(la)tex the T_EX engine extended with direct PDF output. Since producing PDF needs the actual fonts, **pdftex** is also an output driver.

dvips the classical output driver. T_EX engines can produce DVI (DeVice Independent) files, which can be translated to PostScript (or other) formats. To do this, the fonts have to be embedded.

(x)dvipdf(m(x)) the family of DVI-to-PDF converters. Instead of going to PostScript first, these programs support direct translation of DVI into PDF. X_YT_EX uses one of these in the background. Japanese users often use **dvipdfmx**, since it has good support for Japanese fonts.

xdvi online X11 display program, which of course needs access to the fonts to render the glyphs.

These output drivers have supported font mapping in slightly different ways, changing over the years, and here is where `updmap` comes into the game: It reads a list of specifications, and creates configuration files in the needed formats.

2.1.1 What does updmap do?

Font definitions are necessarily a complicated beast in the \TeX world; many components have to play well together for the final document to contain the correct fonts. Here is an overview of the main items necessary to understand updmap:

font definition maps a TFM file name to an external font (font name and file name), with optional additional transformations. A simple example:

```
eufm10 EUFM10 <eufm10.pfb
```

which says that the TFM name `eufm10` should be resolved by a font internally named `EUFM10`, which is defined in the file `eufm10.pfb`. Far more complex font definitions are possible, catering to different encodings and more, but the basic purpose of mapping a TFM to an external font always remains.

font map file is a file of font map definitions, normally collecting together related fonts from a package. The above definition for `eufm10` is contained in `euler.map`, which contains all the Euler-related font definitions.

updmap config file lists the font map files, with additional specifications concerning bitmap vs. outline fonts, as well as a few settings for updmap itself (details in the next section). Continuing our example, in a normal \TeX Live installation the font map file `euler.map` is listed in `texmf-dist/web2c/updmap.cfg`:

```
Map euler.map
```

generated files Finally, updmap generates configuration files in various formats (see above).

Output drivers don't have (or need) the slightest idea that updmap and the related intermediate files even exist; they only read the ultimately-generated configuration file to determine which fonts are available. This means that if, somewhere in the middle, one of the steps fails or is incorrect, the output will probably not have the right fonts.

2.1.2 Configuration of fonts in updmap.cfg

The central configuration file for updmap is (always) named `updmap.cfg`. In former times, only the first one found by the Kpathsea library was used, but now all `updmap.cfg` files are read (see below). Each `updmap.cfg` can contain the following items:

1. Empty lines, comments beginning with '#'; these are ignored.

2. Map directives, in one of the forms:

```
Map foo.map
MixedMap bar.map
KanjiMap baz.map
```

`Map` is used for fonts that are available only in PostScript Type 1 format; `MixedMap` is for fonts where both Metafont and PostScript variants are present; and `KanjiMap` is for creating the special Kanji map file.

3. updmap configuration lines, of the form

```
<settingName> <value>
```

with the following setting names and values (* indicates the default):

```
dvipsPreferOutline values *true, false
```

Whether `dvips` prefers bitmaps or outlines, when both are available.

```
dvipsDownloadBase35 values *true, false
```

Whether `dvips` includes the 35 standard PostScript fonts in its output.

```
pdftexDownloadBase14 values *true, false
```

Whether `pdftex` includes the 14 standard PDF fonts in its output.

```
pxdviUse values true, *false
```

Whether maps for `pxdvi` (Japanese-patched `xdvi`) are under updmap's control.

```
(ja|sc|tc|ko)Embed, jaVariant values strings
```

Controls kanji font embedding for Japanese (`ja`), Simplified Chinese (`sc`), Traditional Chinese (`tc`), and Korean (`ko`).

```
LW35 values *URWkb, URW, ADOBEkb, ADOBE
```

Controls which fonts are used for the 35 standard PostScript fonts.

The `..Embed` and the `jaVariant` settings were added to the \TeX Live implementation recently, and might not be supported in other \TeX distributions.

2.2 fmtutil

In the years long ago, when memory was scarce, computers slow, and Knuth went forth to create the most advanced typesetting system, he devised a way to speed things up and at the same time conserve space: format dumps. This is not the place for details but in short, you can think of them as dumps of the state of the program (\TeX , Metafont, ...) after a (slow, painful) initialization, which can be easily and quickly loaded and used as a starting point for actual typesetting and font design work.

When there was only one \TeX program and one Metafont program, managing these dumps was a simple task, but over time the situation grew more complex: more programs, more formats, various additions for internationalization. Nowadays, we're at a point that people often do not know what is going on when a *formats are rebuilding* message appears.

2.2.1 What does fmtutil do?

Written long ago by Thomas Esser for his `teTeX`, `fmtutil` supports specifying the available format in

a line-based configuration file, and for rebuilding them in various ways. The script has served the $\text{T}_{\text{E}}\text{X}$ community for many years. The shell script mentions a first change in 2001, but the script is much older than that (considerably predating $\text{T}_{\text{E}}\text{X}$ Live).

2.3 Configuration of fonts in `fmtutil.cnf`

`fmtutil` is a rather friendlier colleague than `updmap`, with no need for all the complicated layers of definitions. The configuration files for `fmtutil`, named `fmtutil.cnf`, define the formats which can be made. The most commonly used format is $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, but there are many more, some of which are quite esoteric (e.g., `utf8mex`).

Each format definition is on exactly one line, and consists of four parts:

`<fmtname> <engine> <hyphenfile> <options>`

Let us look at two examples from $\text{T}_{\text{E}}\text{X}$ Live:

```
aleph aleph - *aleph.ini
latex pdftex language.dat
      -translate-file=cp227.tcx *latex.ini
```

The first one defines the format `aleph`, the second one the format `latex`. (The second is broken across lines only for *TUGboat*; in the actual source file, it's all on one line.)

name `aleph`, `latex`—the first item in a format definition is the format name, which (usually) coincides with the program name.

engine `aleph`, `pdftex`—the second item defines the base engine, the program that is run to load the definitions and dump the image. As shown, sometimes the format and the engine have the same name. For the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ format, $\text{T}_{\text{E}}\text{X}$ Live has used the `pdf $\text{T}_{\text{E}}\text{X}$` engine for many years.

hyphenfile `-`, `language.dat`—the third item specifies a file name for hyphenation pattern definitions, or a literal `-` to indicate that no patterns are used.

options —the rest of the line comprises command line arguments passed to the engine. In the `aleph` line we see that only one file is passed to the engine, while in the `latex` case we also pass an additional option.

As specified on its own command line, `fmtutil` reads `fmtutil.cnf`, invokes some or all of the engines with the respective options in turn, and puts the resulting dump files in the right place so that the engine can load the dump.

2.4 Previous behavior and system mode vs. user mode

The original shell scripts read only one configuration file, found by searching with `Kpathsea`. This is

the very same method $\text{T}_{\text{E}}\text{X}$ uses to find files when they are read (e.g., via `\include`) To cater for user-supplied font maps, the original `updmap` program allowed for enabling and disabling, adding and removing individual entries from the configuration file.

While this approach works nicely in a single user installation where the user has complete control over all files, in a multi-user setting it would be chaos if users changed a system-wide configuration file, adding their private fonts. Thus, soon after their inception, Thomas Esser added an additional *system mode* to these scripts, distinguished from the normal invocation style in *user mode*. The only difference between user mode and system mode is *where* generated files are saved: In user mode this was the directory defined by the `Kpathsea` variable `TEXMFVAR`, while in system mode it was `TEXMFSYSVAR`.

System mode was specified by invoking the program under the name `updmap-sys` (`fmtutil-sys`), while user mode was the default.

This was the state of affairs for more than a decade. The advantages of this system were that all configurations were contained in a single file, and the operation mode was easy (easier?) to understand.

In my case, as I had purchased the MathProII fonts, every year and on every computer I used I had to manually disable the open-source clone enabled by default in `belleek.map`, add the necessary map file for the MathProII fonts, and run `updmap`. While this is not much to do, it is easy to forget and error-prone.

3 Per tree configuration

With the Perl reimplementations of the scripts we have also switched to a different way of handling configuration files: the two programs now read not just a single configuration file, but *all* configuration files found, in a stacked manner, meaning that files read later can override parameters from those read earlier. *Override* here means the following: disabling a map that is enabled in a lower level configuration file, and changing settings from a value set in a lower level configuration file.

To see which configuration files will be used, these two commands will output the list of all configuration files used by the two programs:

```
kpsewhich -all updmap.cfg
kpsewhich -all fmtutil.cnf
```

This new method allows configuration of available fonts and formats to be put in the same tree where the respective fonts or formats are installed. Formerly, activation of a map file or format would not survive (re)installing a release of $\text{T}_{\text{E}}\text{X}$ Live. Now, local fonts can be installed under `TEXMFLOCAL`, and listed in `TEXMFLOCAL/web2c/updmap.cfg`, and they

will automatically be picked up across updates.

Similarly, users can have personal fonts or formats without needing to maintain a copy of the system's `updmap.cfg` or `fntutil.cnf`.

3.1 Default locations searched

By default, `updmap` and `fntutil` check the following directories for `updmap.cfg` and `fntutil.cnf`, in the order given.

User mode only:	<code>TEXMFCONFIG/web2c</code>
	<code>TEXMFVAR/web2c</code>
	<code>TEXMFHOME/web2c</code>
Both user and	<code>TEXMFLOCAL/web2c</code>
system modes:	<code>TEXMFSYSCONFIG/web2c</code>
	<code>TEXMFSYSVAR/web2c</code>
	<code>TEXMFDIST/web2c</code>

with these default values for those variables:

<code>TEXMFSYSCONFIG</code>	<code>TL/YYYY/texmf-config</code>
<code>TEXMFSYSVAR</code>	<code>TL/YYYY/texmf-var</code>
<code>TEXMFDIST</code>	<code>TL/YYYY/texmf-dist</code>
<code>TEXMFLOCAL</code>	<code>TL/texmf-local</code>
<code>TEXMFHOME</code>	<code>~/texmf</code>
<code>TEXMFCONFIG</code>	<code>~/.texliveYYYY/texmf-config</code>
<code>TEXMFVAR</code>	<code>~/.texliveYYYY/texmf-var</code>

Making use of this information, let's continue the previous example of the MathProII fonts. As mentioned above, \TeX Live ships the free Belleek fonts which use the same TFM names; thus, we have to disable `belleek.map` and add `mtpro2.map`:

- Put the MathProII files, including `mtpro2.map`, in `TEXMFLOCAL`.
- Edit `TEXMFLOCAL/texmf/web2c/updmap.cfg`:
 - disable Belleek by adding


```
#! Map belleek.map
```
 - enable MathProII by adding


```
Map mtpro2.map
```
- Run `updmap-sys`.

Now, when I update my \TeX Live installation from one year to the next *no* additional work is needed: `updmap` find the local configuration file, duly disabling the one map and activating the other.

Similarly, these per-tree configuration files have brought considerable simplification for distributors like Debian (indeed, this was the original reason why I implemented this feature).

4 Explicit user mode in TL 2017

4.1 What was the problem?

Let's suppose a user wants to add a private font to the \TeX setup (as I had to do during my studies, when I purchased the Lucida fonts for writing my thesis). The steps were these:

- Copy `updmap.cfg` into `TEXMFHOME`;

- add the additional map entries to it;
- run `updmap`.

In itself this was not a problem. The problem comes when the fonts on the system side change (because of an update or addition of new font packages): The user had to re-execute these steps, every time. Not doing so would leave the user with outdated information; in the worst case (but unfortunately a very common case!), some font definitions would no longer be correct, and thus output files would be broken.

The reason was mentioned above: The configuration files for the output drivers generated by `updmap` in the user's home directory override the ones in the system directory.

We might hope for users to know about this problem, but unfortunately the Internet is full of instructions on how to install fonts for \LaTeX , and the typical recommendation is to call `updmap`, and not `updmap-sys`. From my experience as the maintainer of the \TeX Live packages in Debian, as well as from the \TeX Live mailing lists, I can report that this is the single most common point of failure.

That is, most users were simply unaware that calling `updmap` (as is, thus in user mode) creates copies of configuration files which will *never be updated* unless the user calls `updmap` again; system changes in the meantime are immaterial.

For `fntutil` the problem is the same: Format dumps would remain in the user's home directory and never be updated. As a glaring example, I recall a Debian bug report where a user had called `fntutil` once, and *years* later some \LaTeX packages stopped working, because he still used the format dump from years ago, all unknowing.

4.2 New operation mode

For TL17, we (that is Karl Berry and I) decided to try to get out of this interminable chaos once and for all. Thus, from now on *user mode* cannot be invoked by calling `updmap` or `fntutil` as is; to activate user mode, it's now required to give the option `-user`, or call the separate scripts `updmap-user` or `fntutil-user`. To summarize:

System mode is invoked by using `updmap-sys` or `fntutil-sys`, or by giving the `-sys` option.

User mode is invoked by using `updmap-user` or `fntutil-user`, or by giving the `-user` option.

Calling `updmap` or `fntutil` without `-sys` or `-user` now results in a fatal error, with a link to an explanatory web page.

Our hope is that this will prevent some (perhaps many) users from hurting themselves by unintentional switching to user mode. Furthermore, by introducing this new behavior we are explicitly in-

validating plenty of documentation on the web that we know to be wrong, and force users to make a conscious decision. We will see next year how it has worked out!

5 Best practice and use cases

There is probably only one thing we should write here, and if you take one thing from this article, it should be this one:

Use system mode.

Anything else will very likely cause trouble. One might ask, so why didn't we abolish user mode completely? Indeed, we pondered this, but firstly, it would be a radical step after so many years, and secondly, there remain rare cases where user mode is needed; see the following use cases.

5.1 Use cases

The following use cases are also listed on a TUG page (tug.org/texlive/scripts-sys-user.html); the scripts refer to this same page in case of missing mode specifications.

5.1.1 Single user computer — add fonts

One of the most common cases: One user, one computer, \TeX Live is installed system-wide, and fonts should be available to all (1) users of the machine:

- put the fonts into `TEXMFLOCAL` according to the TDS (tug.org/tds);
- enable the font map(s) in the file `TEXMFLOCAL/web2c/updmap.cfg`;
- run (once) `updmap-sys` (no options needed).

Future (re)installations of \TeX Live will pick up these local fonts automatically.

5.1.2 Multi-user computer — add system-wide fonts

A common need in a department or company with organization-specific fonts, which all users should have access to: This case is handled exactly like the previous case, without any changes.

5.1.3 Multi-user computer — private user fonts

This is the only case where user mode is required: A computer with multiple users, but some fonts are private to specific users. Here we cannot install the fonts system-wide, as other users would gain access to them. Thus `TEXMFHOME` is used instead of `TEXMFLOCAL`, and `updmap-user` is run:

- Put fonts into `TEXMFHOME`, following the TDS;
- enable the font map(s) in `TEXMFHOME/web2c/updmap.cfg`;
- run (once) `updmap-user`.

A repeated warning is necessary here, because this is the prime case of misbehavior we have seen: After doing this, changes in the font setup of the system are *invisible* until `updmap-user` is rerun. Thus, we recommend running it regularly, e.g., from Unix `cron`, to make sure no discrepancy creeps in between the fonts as actually installed and those registered in the per-user `updmap.cfg`.

5.1.4 Single user computer — additional formats

While it is uncommon for users create their own formats, in principle the procedure is the same as with `updmap`. In most cases, the additional formats need not be private, so following the first use case above is suggested:

- adjust `TEXMFLOCAL/web2c/fmtutil.cnf`
- run (once) `fmtutil-sys` (no options needed).

5.2 Switching back to system mode

Last but not least, here is how to switch back to system mode if by chance one has called `updmap` or `fmtutil` in user mode. This is never done automatically, and (at least for now) there is no interface to the two programs to allow easily switching.

To switch back to system mode, what has to be done is to remove the following directory trees (after backing them up, of course):

- for `updmap`: `TEXMFVAR/fonts/map`
- for `fmtutil`: `TEXMFVAR/web2c`

where under normal circumstances, `TEXMFVAR` is `~/.texliveYYYY/texmf-var`.

6 Conclusion

We hope that the changes made over the last years have made these programs easier to use, and a bit more protective for the casual user. But one should not forget that they are central configuration programs for \TeX , so messing around with them always bears some risk.

Final exhortation: USE SYSTEM MODE!

◇ Norbert Preining
Accelia Inc., Tokyo, Japan
`norbert (at) preining dot info`
<http://tug.org/texlive/scripts-sys-user.html>

TLaunch, the T_EX Live Launcher for Windows

Siep Kroonenberg

Abstract

The T_EX Live Launcher offers Windows users of a network T_EX Live installation similar conveniences as a locally-installed T_EX Live. It is easy to integrate additional T_EX-related software.

This paper describes the launcher and its configuration. As an example, it shows how it is used at the Rijksuniversiteit Groningen.

1 Overview

The T_EX Live launcher gives users on Windows workstations easy access to a T_EX Live installation already present on the network.

The launcher interface contains menus and buttons to invoke programs, and to access related local and online resources (see figure 1).

It also takes care of the usual Windows-specific configuration: at first run, T_EX Live is added to the search path and relevant filetype associations are set up.

Because of prior experience with users running the initializer or installer when they really want to run the already initialized or installed T_EX Live, I opted for a launcher that configures itself automatically, without requiring a separate initialization step.

Users can replace the default T_EX editor from within the launcher interface, either with an editor defined in an ini file or with a third-party editor present on the filesystem (see figure 2).

For the sake of full access to the Windows API, I wrote the launcher in C. It has no dependencies whatsoever, aside from a T_EX Live installation and Windows version 7 or later.

The launcher makes it easy for the T_EX Live installation maintainer to add menu- or button controls and filetype associations for additional T_EX-related software.

Filetypes, menus and buttons are defined in a Windows ini file. If necessary, pre- and post configuration script files can be configured as well.

The ini file included in T_EX Live provides functionality more or less equivalent to the classic Windows T_EX Live installation.

In the following sections, we have a more detailed look at the launcher and its configuration. The package documentation contains the full details.

Section 6 looks at the T_EX Live installation at the Rijksuniversiteit Groningen, for which the launcher was created.

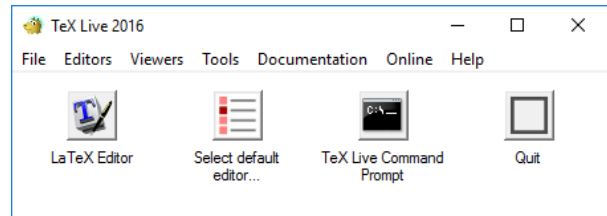


Figure 1: The default T_EX Live Launcher

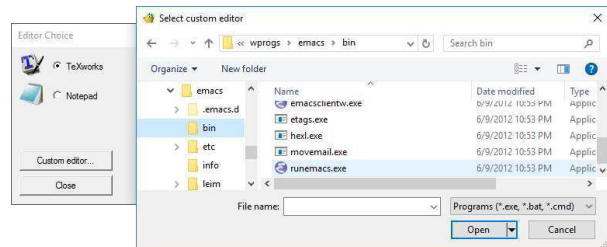


Figure 2: Selecting a custom editor

2 The ini file

The launcher reads its configuration from a conventional Windows ini file with sections, definitions and comment lines. If the T_EX Live installation contains Windows binaries, then `tlaunch.exe` will be in the `bin/win32` directory, and `tlaunch.ini` in `texmf-dist/web2c`.¹ A custom ini file, supporting different software or with localized strings, can be placed in a higher-priority tree.

It is also possible to place `tlaunch.exe` and `tlaunch.ini` together in the root of the installation.

In general, entries which refer to non-existent items are silently ignored.

2.1 Strings

There is a Strings section for string variables. String variable names are case-insensitive. Some string variables, such as `%TLCONFIG%`, are required. This variable indicates the directory where the “forgetter” (see section 3) will be placed.

The optional variables `%PRE_CONFIG%`, `%POST_CONFIG%` and `%PRE_FORGET%` are the names of scripts to be run before and after configuration, and before forgetting respectively. The default values of these variables are empty strings.

Some variables are just conveniences to simplify subsequent definitions.

Some string variables, such as `%troot%` and `%version%`, can be used outright because they are already set when the launcher starts parsing the ini file. Environment variables, *e.g.* `%appdata%` or

¹ If during installation non-default options are selected for file associations or path adjustment, then a second, modified copy will be written to `texmf-var/web2c`.

`%UserProfile%` can also be used outright. See the package documentation for the full list.

A few example string definitions:

```
[Strings]
TLNAME=TeX Live %VERSION%
; tlaunch configuration directory
TLCONFIG=%userprofile%\texlive%VERSION%\tlaunch
TLSCRIPTS=%tlroot%\scripts
POST_CONFIG=%TLSCRIPTS%\post_config.cmd
; optional announcement text
ANNOUNCE=TeX Live Launcher with extras
```

2.2 Filetype associations

In Windows, the association of a filename extension with a program is indirect: an extension is associated with a filetype and a filetype is associated with a command. An example of a filetype definition in the ini file:

```
[FT:TL.TeXworks.edit.%VERSION%]
COMMAND="%tlroot%\bin\win32\TeXworks.exe"
EXTENSIONS=.tex .cls .sty
```

The name of the ini file section consists of the filetype name with an ‘FT:’ prefix. When a file with a listed extension is double-clicked in a file manager, Windows will run `COMMAND` with the (quoted) filename appended. If a more complex command is required, *e.g.* with parameters coming after the filename, the section can define a more complex command-line with a `SHELL_CMD` entry.

2.3 Menus and buttons

The ini file can contain a buttons section and sections for menus, with the latter indicated by an `MN:` prefix to the section name. Within the section entries, the key is the string to be displayed and the value is the action to be taken.

In the case of a button, the display string is put underneath the button (see figure 1). The launcher tries to find a suitable icon to place on the button itself, but has a fallback icon if it cannot find anything. This fallback icon is used for the Quit button in figure 1.

A few examples:

```
[MN:File]
Browse installation=explorer.exe "%tlroot%\.."
Quit=FU:quit
```

```
[MN:Viewers]
PostScript Viewer=FT:TL.PSView.view.%VERSION%
DVI Viewer=FT:TL.DVIOUT.view.%VERSION%
```

```
[MN:Documentation]
LaTeX Introduction=SO:%tlroot%\..\lshort.pdf
FAQ=SO:%tlroot%\..\newfaq.pdf
```

[Buttons]

```
LaTeX Editor=FU:default_editor
Select default editor...=FU:editor_select
Quit=FU:quit
```

The value, which is the associated action, can take several forms:

- No prefix: a command to be executed.
- With a prefix `FT:`, the associated action is the `COMMAND` of the indicated filetype, which should be defined earlier in the file.
- Prefix `SO:` (shell object) meaning in this case a file or url that Windows should know how to open.
- Prefix `SC:` indicates a script object defined earlier in the ini file; see the package documentation.
- Prefix `FU:` indicates a predefined function; see the package documentation.

2.4 The General section

The most important options in this section replicate options from the TeX Live installer:

`Filetypes` Allowed values are `none`, `new` (default) and `overwrite`

`searchpath` Allowed values are 0 and 1 (default)

Both entries and the section itself are optional. For example:

```
[General]
FILETYPES=new
SEARCHPATH=1
```

3 Forgetting

The launcher has functions to undo and redo configuration, which can be assigned to menu items. However, the installation may not be under the user’s control and may no longer be around when the configuration is to be cleared out.

Therefore, the launcher creates a so-called forgetter as part of its first-time initialization. This forgetter consists of a copy of the launcher and a modified copy of the configuration file, both placed under the user’s profile. This copy knows from its location that it is intended to run as forgetter and not as launcher.

4 Scripts

The launcher can run scripts and command-line utilities, and display their output in a window. The ini file can specify scripts for *e.g.* supplemental initialization and cleanup (see section 2.1). Section 6.1 shows some examples. It is also possible to assign scripts to menu entries and to buttons. More about scripts is in the package documentation.

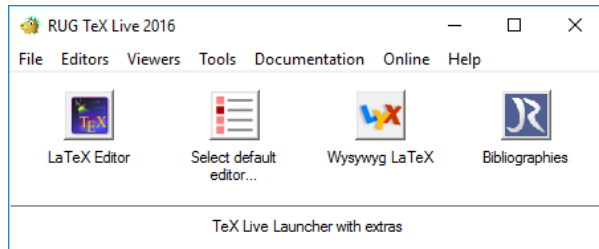


Figure 3: The TeX Live Launcher at the Rijksuniversiteit Groningen

5 Launcher-based installations

The 2017 TeX Live installer offers the option of creating a launcher-based installation, as an alternative to creating menu shortcuts. If this option is selected, then no path adjustment is done and no filetype associations are created by the installer itself. The installer invokes the launcher with a special option to ‘install’ itself, *i.e.* to create a start menu shortcut and an uninstaller registry entry for itself. In case of a single-user install, it also performs a first-time initialization.

With such an installation, the TeX Live installer no longer has direct dealings with the Windows API, or with the Perl modules providing API access.

For purposes of trying out the launcher, TeX Live includes a script `tlaunchmode` which can switch the installation between classic and launcher mode without reinstalling TeX Live.

6 The launcher at the Rijksuniversiteit Groningen

Workstations at our university are mostly centrally managed. Typically, users have a centrally managed Start menu on their Windows workstation. The IT people put the TeX Live Launcher in this menu, so users are just one click away from starting to use TeX Live.

Settings are centrally backed-up on logout and restored on login. So a user’s desktop looks very similar, whatever physical workstation [s]he works on. This same desktop is also available remotely. In addition, users have a network share for storing their own files. This share is also available from any workstation on which they log in.

6.1 Additional software

The additional programs at our university include:

- More editors: TeXnicCenter and TeXstudio. Both offer extensive assistance in editing math.
- The PDF viewer SumatraPDF. This viewer provides source–PDF synchronization for TeXnicCenter, which has no built-in PDF viewer.

- The Java-based bibliography manager JabRef.
- The `epspdf` GUI with bundled single-file Tcl/Tk runtime (<https://ctan.org/pkg/epspdf>).
- The pseudo-WYSIWYG L^AT_EX L^AT_EX editor.

There are menu items for additional documentation, such as the L^AT_EX classes for the university house style. Controls for the TeX Live Manager and for uninstalling TeX Live itself are omitted, since those tasks are reserved for the maintainer of the installation.

All these programs were installed on a scratch system and from there copied into the TeX Live installation tree. ‘Installed’ this way, most of them run more or less ok from their new location.

However, some fixes were desirable and were implemented via a postconfig script (see section 2.1):

TeXnicCenter While TeXnicCenter can autoconfigure itself nicely for MiKTeX, it asks TeX Live users a series of questions about what is where. To spare users those questions, I wrote a vbscript which emulates the MiKTeX autoconfiguration for TeX Live, and which is invoked by the postconfig script.

TeXstudio This editor by default checks at startup whether there is a new version. The postconfig script turns this option off in an existing or newly-created TeXstudio configuration file.

TeXworks borrows some dictionaries from TeXstudio.

SumatraPDF This PDF viewer also tests for updates, which are dealt with in the same way as for TeXstudio. It also requires a registry setting to specify that this is *not* a portable installation, and it should store its settings under the user’s profile.

L^AX First-time initialization can take a very long time. Therefore, a L^AX user configuration directory has been prepared in advance. The postconfig script copies it to the user’s profile.²

The launcher documentation contains a file `rug.zip` with slightly sanitized versions of the scripts and configuration files actually used at our university installation.

7 Problems

7.1 Non-roaming filetype associations

In a standard Roaming Profiles setup, filetype associations do not roam. I plan to add an option to the launcher to restore missing filetype associations

² There is also a shared L^AX configuration file which had to be patched, but this is not a task for a per-user postconfig script.

on login. This is not a problem with the centrally-managed desktops at our university. On the other hand, on those centrally-managed desktops some filetype associations are pre-empted and cannot be permanently changed. This includes PDF files.

7.2 Search path

Another problem associated with our desktop management software is that programs ignore the user search path.

This is not a problem for software started from the launcher.

Some programs do not absolutely need \TeX Live on the search path. Others, such as \TeX works and the DVI- and PostScript viewers included in \TeX Live, are invoked via a wrapper which takes care of the search path. But for \TeX studio I had to provide a wrapper myself to take care of the search path.

And then the university offers software such as R and WinEdt which also need \LaTeX on the search path, but which are not under my control; the IT department has to handle these.

7.3 Uninstalling

The third problem I want to mention is uninstalling under Windows 10. This is not specific to the launcher.

There are two ways to give a user access to an uninstaller:

- Via a Start menu item. However, Windows 10 may somehow decide not to display such an item.
- Via an uninstaller registry key. This way, it will show up in Settings / Apps / Apps & features. However, Windows may decide to pop up a User Account Control (UAC) prompt even if it is a user install. Still, a user-installed launcher *can* be uninstalled via right-clicking its icon under the Start menu, or from within the launcher itself.

8 Finding out more

Earlier, I mentioned the `tlaunch` manual. If you have a fully updated 2016 or a later installation of \TeX Live with Windows platform support, then you should have the `tlaunch` binary and documentation on your system. But you can also visit its CTAN directory at <https://ctan.org/pkg/tlaunch>.

For experimentation, you can run the script `tlaunchmode` mentioned at the end of section 5. With this script you can switch an existing \TeX Live installation to launcher mode and back.

◇ Siep Kroonenberg
Groningen
The Netherlands
`siepo (at) cybercomm dot nl`

Xdvi_{ps}k: Dvips ready for OpenType fonts and more image formats

Sigitas Tolušis, Arūnas Povilaitis and
Valentinas Kriaučiukas

Abstract

We present two extensions to `dvips`. One allows flexible inclusion of bitmap images and was implemented on top of the FreeImage library. The second extension solves quite a long-standing task: adding OpenType font support to `dvips`. Our extended `dvips`, `xdvipsk`, goes the “Lua_{TeX} way” in OpenType font management: it works on DVI files compiled by Lua_{TeX} and expects to find the necessary Unicode map files, obtained as by-products of the compilation. The providing of these map files is ensured by a special `LaTeX` package.

1 Motivation and history

The Dvips(k) page [5] says that “it would be great to add OpenType and perhaps TrueType support to `dvips`”.

We had our own motivation too. We saw that complete elimination of the PostScript stage from our publishing workflow `LaTeX`→PDF would be either very costly or almost impossible, for different reasons. Among them are requirements to produce web-optimized PDF, use of Adobe’s Acrobat Distiller and of other local utilities built into the workflow going through PostScript.

Some `dvips` shortcomings, such as restricted support of graphic formats, can be quite easily compensated for by graphics preprocessing tools. An attempt to use OpenType fonts meets bigger problems. It is possible to transform one OT font to many new Type1 fonts, but then one needs to introduce the new fonts in `TeX` styles and ensure their correct use in `LaTeX` texts. As a side effect of this, all advantages of OpenType fonts are lost. Plus, these steps are painful, so the wish to avoid them by extending `dvips` is natural.

The locally used versions of `dvips` were modified long ago (before 2000), but behavior modifications were not deep, like ignoring unknown and recognizing private specials, and writing a log file. As the `dvips` program was not actively developed at that time, the local patching to new versions of `dvips` was an easy task.

About five years ago, more advanced handling of graphic files was implemented, based on the FreeImage [6] library. Standard `dvips` mainly works with EPS files only, so all non-EPS graphics had to be converted into EPS format. It allows restricted

use of bitmap images (BMP, PCX, PICT formats, no scaling or rotation), but this is not exposed in the main documentation [8]. The extended `dvips` now accepts BMP, PCX, TIFF, JPEG, PNG formats and performs the same actions as with EPS: scaling, rotating, trim, viewport (but the `graphics` package does not yet implement the operations of clipping, trimming and viewport).

The work on providing OpenType font support started about a year ago, when possible components were tested; later they were connected into a working chain. The current stage of `xdvipsk` development can probably be characterized as beta.

The program name `xdvipsk` starts with ‘x’ denoting the Unicode (OT fonts) extension and ends with ‘k’ denoting use of the Kpathsea library (as with `dvipsk`). The standard Kpathsea library (from `TeX Live`) does not work with our main development environment on MS Windows (Visual Studio), so it was separately compiled for `xdvipsk`.

2 New options

New features of the extended `dvips` can be switched on or off using new command-line options. The summary of the options, presented in Fig. 1, is output when `xdvipsk` is called with no arguments or with the standard `--help` option. For more convenient review, in this presentation the new options are printed in frame-boxes. A more detailed description of the new options, as included in the documentation (again differing only in formatting details) is as follows:

- g Mode of logging into the file named
`<dvi file name>.xdvips.log`;
 default on. For a successful run, the log file contains only the message `!!!Success!!!`.
- H 32-bit turbo mode for inclusion of PostScript graphics (writes EPS files directly to PS file) using 10 MB dynamic buffer; default off.
- I<*pixel-form filters*> Resizing mode for bitmap images included with `em: graph` specials; default off. <*pixel-form filters*> is a comma-separated tuple of up to four pairs <*pixel-form*>:<*filter*>, where <*pixel-form*> can be one of
 BW: black/white 1-bit pixels,
 GR: gray 8-bit pixels,
 RGB: colored 24-bit pixels,
 CMYK: colored 32-bit pixels,
 and <*filter*> can be one of the following:
 b: box filter,
 t: bilinear filter,
 B: B-spline filter,
 m: Mitchell–Netravali bicubic filter,

```

Usage: dvips [OPTION]... FILENAME[.dvi]
Convert DVI input files to PostScript.
Options:
-a* Conserve memory, not time      -A Print only odd (TeX) pages
-b # Page copies, for posters e.g. -B Print only even (TeX) pages
-c # Uncollated copies            -C # Collated copies
-d # Debugging                    -D # Resolution
-e # Maxdrift value               -E* Try to create EPSF
-f* Run as filter                 -F* Send control-D at end
-g* write log file                -G* Shift low chars to higher pos.
-h f Add header file             -H* Turbo mode for PS graphics
-i* Separate file per section     -I* Resize mode for emTeX graphics
-j* Download [T1] fonts partially -J* Download OpenType fonts partially
-k* Print crop marks             -K* Pull comments from inclusions
-l # Last page                   -L* Last special papersize wins
-m* Manual feed                  -M* Don't make fonts
-mode s Metafont device name     -N* No structured comments
-n # Maximum number of pages
-noomega Disable Omega extensions
-noptex Disable pTeX extensions
-no luatex Disable LuaTeX extensions
-noToUnicode Disable ToUnicode CMap file generation for OpenType fonts
-o f Output file                  -O c Set/change paper offset
-p # First page                   -P s Load config.$s
-pp l Print only pages listed
-q* Run quietly                   -Q* Skip VTeX private specials
-r* Reverse order of pages        -R* Run securely
-s* Enclose output in save/restore -S # Max section size in pages
-t s Paper format                 -T c Specify desired page size
-u s PS mapfile                   -U* Disable string param trick
-v Print version number and quit  -V* Send downloadable PS fonts as PK
                                  -W* Extended search for emTeX graphics
-x # Override dvi magnification  -X # Horizontal resolution
-y # Multiply by dvi magnification -Y # Vertical resolution
-z* Hyper PS                      -Z* Compress bitmap fonts
# = number f = file s = string * = suffix '0' to turn off
c = comma-separated dimension pair (e.g., 3.2in,-32.1cm)
l = comma-separated list of page ranges (e.g., 1-4,7-9)

```

Figure 1: Xdvipsk option summary with new options indicated.

l: Lanczos-windowed sinc filter,
c: Catmull–Rom and Overhauser splines,
r: resample image (remove rows and columns
in the bitmap),
w_i: MS Windows GDI filter, where $i = 1, 2, 3, 4$
means modes BLACKONWHITE, WHITEON-
BLACK, COLORONCOLOR and HALFTONE,
respectively.

Not all $\langle pixel\ form \rangle : \langle filter \rangle$ combinations are possible:

- filters w_i can be used on MS Windows systems only and just for BW, GR, and RGB pixel forms; for CMYK, any w_i filter is replaced by the **r** filter;
- on Linux and other systems, filters w_i are also changed to **r** filter;
- for monochrome graphics, only filters **r** and w_i are applicable.

-I (without filters) Resizing mode with the following filter tuples:

BW: **w1**, **GR**: **w3**, **RGB**: **w3**, **CMYK**: **r** on Windows;
BW: **r**, **GR**: **r**, **RGB**: **r**, **CMYK**: **r** on other systems.

- j Type 1 fonts partial download; default off (contrary to dvips).
- J Download only needed characters from OT fonts; default on.
- no luatex Disable LuaTeX extensions and support of OpenType fonts.
- noToUnicode Omit generation of map (to Unicode) files for OT fonts, which can be used by Acrobat Distiller to enable text search; default on.
- Q Mode of skipping VTeX specials: any content of `\special` commands prefixed with `mt:`, `vtex:`, `MC:`, `BMC:` or `EMC:` is silently ignored; default off.
- W Extended search mode for image files indicated by **em**: **graph** specials: when no file with the specified name is found, the file names with other extensions (`.pcx`, `.bmp`, `.tif`, `.jpg`, `.png`) are tried; default off.

3 Extension for graphics

The extension for bitmap images does not require changes to the user-level syntax; the \LaTeX command `\includegraphics` should work as described in the documentation of `graphics` and `graphicx` [2]; that is, after inclusion in the preamble of either

```
\usepackage[<driver>]{graphics}
```

or

```
\usepackage[<driver>]{graphicx}
```

where the file `<driver>.def` contains all the necessary declarations and is registered in `graphics.sty` (examples can be found in the presentation [10]). As `xdvi` accepts images in formats BMP, JPEG, PCX, PNG, and TIFF, they should all be declared in the form of graphic inclusion rules in the driver file, most likely `dvips.def`:

```
\@namedef{Gin@rule@.tif}#1{{bmp}{.tif.bb}{#1}}
\@namedef{Gin@rule@.tiff}#1{{bmp}{.tiff.bb}{#1}}
\@namedef{Gin@rule@.jpeg}#1{{bmp}{.jpeg.bb}{#1}}
\@namedef{Gin@rule@.jpg}#1{{bmp}{.jpg.bb}{#1}}
\@namedef{Gin@rule@.png}#1{{bmp}{.png.bb}{#1}}
```

Several things for authors of \TeX packages and papers to know:

- Bitmap image file names are included in DVI files inside arguments of `\special` commands with prefix `em:graph` (the name has roots in the time of the $\text{Em}\TeX$ distribution).
- Bitmaps can be of different color models: BW, gray, RGB, CMYK, indexed RGB.
- The program ignores the content of `\special` commands with unknown prefixes.
- For more precise image positioning, `Xdvi` inserts the PostScript `HiResBoundingBox`.

4 How Xdvi works with OpenType fonts

Our solution comes from the decision to use OpenType font information directly, as with $\text{Lua}\TeX$ and the `luaotfload` package [9]. The current version of `luaotfload` operates with only one writable cache, which incorporates file paths specific to an OS, which for us is inconvenient. Our production environment contains different operating systems: Linux servers, Linux and Windows workstations, and a shared \TeX -tree resource with multiple \TeX Live versions on a Linux server, accessible by Windows clients through the local network. We wanted to have things as flexible as possible in presence of different OSes.

Some additional tools are necessary for `xdvi` to work:

1. PostScript header file `texcid.pro` is used for inclusion of OpenType fonts in PostScript files.

It is an analogue of `texps.pro` that is used in case of Type1 fonts.

2. A \LaTeX package `luafonts`, which is just an interface to Lua code generating two additional maps. It is loaded like any other \LaTeX package:

```
\usepackage{luafonts}
```

One map generated by the package at compilation time is analogous to `psfonts.map` and contains information about OpenType fonts used in a particular article. The map format is as follows:

```
<tfm name>_{<ps name>}_{<tefont name>}_{<file name>}
```

where `<tfm name>` is the same as what is written in the DVI file by $\text{Lua}\TeX$, and `<ps name>` and `<file name>` come from `luaotfload` Lua tables. This `<ps name>` is a PostScript font name and `<tefont name>` is an internal font name seen by `luaotfload` as `fullname`. `<file name>` is modified so that the directory prefix, corresponding to the actual \TeX tree used, is replaced by variable `$$SELFAUTOPARENT`. Examples of `<tfm name>`, `<file name>` and `<ps name>` are given, respectively, in Figs. 2, 3 and 4.

Another map generated by `luafonts` stores information about characters. It consists of triples `<internal tex character code>`, `<opentype font glyph index>`, `<unicode equivalent>`; examples are in Fig. 5.

These maps are used by `Xdvi` (a) to find CIDs (character identifiers [1]) for insertion in PS files and (b) to prepare `TOUNICODE` cmaps [3], like the one shown in Fig. 6. They are needed for searching in PDF files. A utility `make2unc` was created to incorporate `TOUNICODE` cmaps for PDF.

4.1 Process in steps

Step 1. Run `dvilualatex <article>.tex`

where file `<article>.tex` uses `luafonts`:

```
Input:  <article>.tex
        tex/luatex/luafonts/luafonts.sty
        tex/luatex/luafonts/luafonts.lua
        ...
```

```
Output: <article>.dvi
        .xdvi/ps/<ps name>.encodings.map
        ...
        .xdvi/<article>.opentype.map
```

Step 2. Run `xdvi <article>.dvi`:

```
Input:  <article>.dvi
        .xdvi/ps/<ps name>.encodings.map
        ...
        .xdvi/<article>.opentype.map
        texmf-dist/dvips/base/texcid.pro
        ...
```

```

FandolFang-Regular
FandolFang-Regular:mode=node;script=latn;language=DFLT;+tlig;
TeXGyreAdventor
TeXGyreAdventor/B
TeXGyreAdventor/BI
TeXGyreAdventor/I
TeXGyreAdventor:mode=node;script=latn;language=DFLT;+pnum;+onum;
[lmroman10-bold]:+tlig;
[lmroman10-italic]:+tlig;
[lmroman10-regular]:+tlig;

```

Figure 2: Examples of $\langle tfm name \rangle$.

```

>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/fandol/FandolFang-Regular.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/tex-gyre/tegyreadventor-regular.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/tex-gyre/tegyreadventor-bold.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/tex-gyre/tegyreadventor-bolditalic.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/tex-gyre/tegyreadventor-italic.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/tex-gyre/tegyreadventor-regular.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/lm/lmroman10-bold.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/lm/lmroman10-italic.otf
>$SELFAUTOPARENT/texmf-dist/fonts/opentype/public/lm/lmroman10-regular.otf

```

Figure 3: Examples of $\langle file name \rangle$.

FandolFang-Regular	59964,707,00AF
TeXGyreAdventor-Regular	59965,708,00AF
TeXGyreAdventor-Bold	59966,709,00200331
TeXGyreAdventor-BoldItalic	59967,710,0304
TeXGyreAdventor-Italic	59968,711,02DA
TeXGyreAdventor-Regular	59969,712,0020030A0301
LMRoman10-Bold	59970,713,0020030A0301
LMRoman10-Italic	59971,714,030A
LMRoman10-Regular	59972,715,02DC

Figure 4: Examples of $\langle ps name \rangle$.

Figure 5: An excerpt from a map, specifying T_EX characters' OpenType glyph indices and Unicode equivalence codes.

Output: $\langle article \rangle$.ps

```

...
.xdvipsk/ $\langle article \rangle$ -cid $\langle num \rangle$ .tounicode
...

```

where $\langle num \rangle$ is a font index in the DVI file and is used here to have distinct file names.

Step 3. Convert the PostScript file to PDF (using Ghostscript, Acrobat or any other tool):

Input: $\langle article \rangle$.ps
Output: $\langle article \rangle$.pdf

Step 4. Add TOUNICODE cmaps to the PDF file using `make2unc` utility:

```

Input:  $\langle article \rangle$ .pdf
...
.xdvipsk/ $\langle article \rangle$ -cid $\langle num \rangle$ .tounicode
...

```

Output: $\langle article \rangle$.pdf (searchable)

5 Development environment

At present, we use a rather split and mixed environment, compared with the T_EX Live build ecosystem. As mentioned above, the main development and building of executables is done on a MS Windows workstation using the Visual Studio 2013 IDE.

In parallel, we build the code on two more architectures: Linux and Mac OS X. For these, we are quite close to the T_EX Live build environment except for prebuilt architecture-dependent versions of `tiff`, `lzma` and `jbig` libraries for `xdvipsk` and `MuPDF` [7] library for `make2unc`. There is no doubt that this is easier than incorporating the mentioned libraries into the T_EX Live build ecosystem in the proper way. It allowed us to provide, with minimum effort, our time-limited solution for incorporating OpenType fonts into a Dvips-based workflow.

```

/CIDInit /ProcSet findresource begin
12 dict begin
begincmap
/CMAPName /t6-cid002 def
/CMAPType 2 def
/CIDSystemInfo <<
  /Registry (TeX)
  /Ordering (BHCDARZO+002)
  /Supplement 0
>> def
1 begincodespacerange
<0000> <FFFF>
endcodespacerange
24 beginbfchar
<001D> <0061>
<0024> <0062>
<002E> <002C>
<0030> <0064>
<0033> <0065>
<0042> <0049>
<0043> <0069>
<0049> <006C>
<004C> <006D>
<004E> <006E>
...
<040B> <0037>
<040E> <0036>
<0415> <0033>
<0419> <0032>
endbfchar
endcmap
CMAPName currentdict /CMAP defineresource pop
end
end

```

Figure 6: An example of a TOUNICODE map.

Other needed libraries are taken from T_EX Live distributions. The current `xdvipsk` version is based on `dvips` 5.996, `web2c+kpathsea` 6.22, T_EX Live 2016.06.07, `jpeglib` 9b, `libpng` 1.6.2, `libtiff` 4.06, `zlib` 1.2.8.

Comparing with the `dvips` source, the changes in the code structure are the following:

- New modules:

```

charcode.c    emspecial.c
luamap.c     sfntload.c
utarray.h    uthash.h
writecid.c

```

and `texcid.lpro`—a PostScript procset with comments.

- Removed modules:

```

emspecial.c

```

- Changes made in 22 modules. All changes are tagged with markers:

```

//AP--begin
//AP--end

```

- New directories:
 - `graflib`: simplified and adapted code from the FreeImage [6] library;
 - `otflib`: adapted code from `dvipdfmx` [4].

6 Availability

The source code is available from <https://github.com/vtex-soft/texlive.xdvipsk>.

References

- [1] Adobe Systems Inc. *Adobe CMap and CIDFont Files Specification*, June 1993. Version 1.0. https://www.adobe.com/content/dam/Adobe/en/devnet/font/pdfs/5014.CIDFont_Spec.pdf.
- [2] David P. Carlisle. *Packages in the ‘graphics’ bundle*. The L^AT_EX3 Project, December 2016. <http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf>.
- [3] Cmap—character to glyph index mapping table. <https://www.microsoft.com/typography/otspec/cmap.htm>. Visited 2017-04-10.
- [4] Dvipdfmx—an extended version of `dvipdfm`. <https://ctan.org/pkg/dvipdfmx>, March 2010.
- [5] Dvips(k). <https://tug.org/dvips/>, April 2017.
- [6] The FreeImage project. <http://freeimage.sourceforge.net/>, April 2017.
- [7] MuPDF. <http://mupdf.com/>.
- [8] Tomas Rokicki. *Dvips: A DVI-to-PostScript Translator*, April 2016. <https://tug.org/texlive/Contents/live/texmf-dist/doc/dvips/dvips.pdf>.
- [9] Elie Roux, Khaled Hosny, and Philipp Gesang. *The luaotfload package*, January 2017. <https://ctan.org/pkg/luaotfload>.
- [10] Sigitas Tolušis, Arūnas Povilaitis, and Valentinas Kriaučiukas. *Xdvipsk: dvips ready for opentype fonts and more image formats*. <http://www.gust.org.pl/bachotex/2017-pl/presentations/stolusis-et-al-1-2017.pdf>, May 2017. Slides of presentation at TUG@BachoT_EX 2017.

- ◇ Sigitas Tolušis
VTeX, Mokslininkų 2a, Vilnius
LT-08412, Lithuania
`sigitas (at) vtex dot lt`
- ◇ Arūnas Povilaitis
VTeX, Mokslininkų 2a, Vilnius
LT-08412, Lithuania
`arunas (at) vtex dot lt`
- ◇ Valentinas Kriaučiukas
VTeX, Mokslininkų 2a, Vilnius
LT-08412, Lithuania
`valius (at) vtex dot lt`

GUST e-foundry current font projects

Jerzy B. Ludwichowski

Abstract

This is a short description of GUST's e-foundry plans for the more or less immediate future. Until now they have not been presented widely, but only in a different form to some LUG boards.

Introduction

For readers not familiar with the GUST e-foundry (<http://www.gust.org.pl/projects/e-foundry>), a list of its achievements follows:

- text fonts: Latin Modern, T_EX Gyre fonts (seven families), Antykwa Półtawskiego, Antykwa Torunska, Kurier, Iwona, Cyklop.
- OTF math fonts (6 of the total 10 free and 3 commercial available): Latin Modern Math, four T_EX Gyre Math fonts (Bonum, Pagella, Schola, Termes), T_EX Gyre DejaVu Math.

Various proposals have been made to the team to do more work based on its members' OTF math fonts expertise. The resulting projects are briefly outlined here.

Math symbols subsets

Define subsets of math symbols for several uses:

- a sans-serif font (with the MATH table and a limited repertoire of glyphs); to be used in headings and slides;
- a heavy font (with the MATH table and a limited repertoire of glyphs); again, to be used in headings and for slides;
- a monospaced font (without the MATH table), to be used with text editors
- a text font for technical texts (in-line references to symbols and quoting of simple formulas without deploying the math fonts machinery);

This is a study project, with no direct deliverables, except for the selection of glyphs. It is, however, prerequisite for most of the following projects.

A sans-serif math OpenType font

Make a sans-serif OpenType math font, based on DejaVu, with an eye on doing the same for other sans-serif fonts; for use in headings.

A heavy math OpenType font

Starting from the heavy version of one of the T_EX Gyre OTF math fonts, possibly T_EX Gyre Termes, with an eye on doing the same for other serif fonts. Also for use in headings.

A monospace font with math symbols

A monospace (text) font enhanced with math symbols without extensibles (a proper subset of math symbols required), most probably DejaVu based; for use in editing and source code. The main difficulties:

- “squeezing” of wide math symbols into the monospaced dimensions;
- the incompleteness of the Unicode standard (e.g., the incomplete set of superscript glyphs) may turn out to be troublesome.

Enhancing the T_EX Gyre text fonts

The T_EX Gyre fonts will certainly benefit from enhancement with a subset of math symbols. Possible (open) problems:

- might require a revision of glyph selection, sans-serif OTF math and heavy OTF math.
- should sans-serif fonts also be enhanced?
- and should they share the same repertoire of extra glyphs?

It makes little sense to enhance the T_EX Gyre Chorus (the Zapf Chancery replacement) font with math oriented glyphs. In addition:

- The fonts do require maintenance;
- until now done only when requests or bug reports were received.

To keep uniformity and spare users unpleasant surprises this must involve all GUST fonts, even when no changes/modifications ensue. This should be done carefully, on a planned schedule; the team proposes regular yearly (calendar) revisions.

Enhancements to existing fonts

The GUST e-foundry's math fonts will profit from being enhanced with math kerns and math oriented features like variant extra alphabets, e.g., double-struck or calligraphic, implemented using the “stylistic set” features, ss01–ss20.

Summary

The priorities will certainly influence the order in which the projects will be tackled, but the glyph selection is the prerequisite.

As there is a considerable amount of work involved in all of these projects, we requested funding from some T_EX user groups. Support has been promised from: NTG, C_STUG, CG (Context Group), DANTE e.V., TUG, GUST (non-material). Given time, the team will work on the projects, even without funds.

◇ Jerzy B. Ludwichowski
GUST, Poland
jerzy.ludwichowski (at) gmail dot com

Variable fonts

Hans Hagen

1 Introduction

History shows the tendency to recycle ideas. Often quite some effort is made by historians to figure out what really happened, not just long ago, when nothing was written down and we have to do with stories or pictures at most, but also in recent times. Descriptions can be conflicting, puzzling, incomplete, partially lost, biased, . . .

Just as language was invented (or evolved) several times, so were scripts. The same might be true for rendering scripts on a medium. Semaphores came and went within decades and how many people know now that they existed and that encryption was involved? Are the old printing presses truly the old ones, or are older examples simply gone? One of the nice aspects of the internet is that one can now more easily discover similar solutions for the same problem, but with a different (and independent) origin.

So, how about this “new big thing” in font technology: variable fonts. In this case, history shows that it’s not that new. For most \TeX users the names `METAFONT` and `MetaPost` will ring bells. They have a very well documented history so there is not much left to speculation. There are articles, books, pictures, examples, sources, and more around for decades. So, the ability to change the appearance of a glyph in a font depending on some parameters is not new. What probably *is* new is that creating variable fonts is done in the natural environment where fonts are designed: an interactive program. The `METAFONT` toolkit demands quite some insight in programming shapes in such a way that one can change look and feel depending on parameters. There are not that many meta fonts made and one reason is that making them requires a certain mind- and skill set. On the other hand, faster computers, interactive programs, evolving web technologies, where real-time rendering and therefore more or less real-time tweaking of fonts is a realistic option, all play a role in acceptance.

But do interactive font design programs make this easier? You still need to be able to translate ideas into usable beautiful fonts. Taking the common shapes of glyphs, defining extremes and letting a program calculate some interpolations will not always bring good results. It’s like morphing a picture of your baby’s face into yours of old age (or that of your grandparent): not all intermediate results will look great. It’s good to notice that variable fonts are a revival of existing techniques and ideas used in, for

instance, multiple master fonts. The details might matter even more as they can now be exaggerated when some transformation is applied.

There is currently (March 2017) not much information about these fonts so what I say next may be partially wrong or at least different from what is intended. The perspective will be one from a \TeX user and coder. Whatever you think of them, these fonts will be out there and for sure there will be nice examples circulating soon. And so, when I ran into a few experimental fonts, with PostScript and TrueType outlines, I decided to have a look at what is inside. After all, because it’s visual, it’s also fun to play with. Let’s stress that at the moment of this writing I only have a few simple fonts available, fonts that are designed for testing and not usage. Some recommended tables were missing and no complex OpenType features are used in these fonts.

2 The specification

I’m not that good at reading specifications, first of all because I quickly fall asleep with such documents, but most of all because I prefer reading other stuff (I do have lots of books waiting to be read). I’m also someone who has to play with something in order to understand it: trial and error is my *modus operandi*. Eventually it’s my intended usage that drives the interface and that is when everything comes together.

Exploring this technology comes down to: locate a font, get the OpenType 1.8 specification from the Microsoft website, and try to figure out what is in the font. When I had a rough idea the next step was to get to the shapes and see if I could manipulate them. Of course it helped that in `ConTeXt` we already can load fonts and play with shapes (using `MetaPost`). I didn’t have to install and learn other programs. Once I could render them, in this case by creating a virtual font with inline PDF literals, a next step was to apply variation. Then came the first experiments with a possible user interface. Seeing more variation then drove the exploration of additional properties needed for typesetting, like features.

The main extension to the data packaged in a font file concerns the (to be discussed) axis along which variable fonts operate and deltas to be applied to coordinates. The `gdef` table has been extended and contains information that is used in `gpos` features. There are new `hvar`, `vvar` and `mvar` tables that influence the horizontal, vertical and general font dimensions. The `gvar` table is used for TrueType variants, while the `cff2` table replaces the `cff` table for OpenType PostScript outlines. The `avar` and `stat` tables contain some meta-information about the axes of variations.

It must be said that because this is new technology the information in the standard is not always easy to understand. The fact that we have two rendering techniques, PostScript `cff` and TrueType `ttf`, also means that we have different information and perspectives. But this situation is not much different from OpenType standards a few years ago: it takes time but in the end I will get there. And, after all, users also complain about the lack of documentation for ConTeXt, so who am I to complain? In fact, it will be those ConTeXt users who will provide feedback and make the implementation better in the end.

3 Loading

Before we discuss some details, it will be useful to summarize what the font loader does when a user requests a font at a certain size and with specific features enabled. When a font is used the first time, its binary format is converted into a form that makes it suitable for use within ConTeXt and therefore LuaTeX. This conversion involves collecting properties of the font as a whole (official names, general dimensions like x-height and em-width, etc.), of glyphs (dimensions, Unicode properties, optional math properties), and all kinds of information that relates to (contextual) replacements of glyphs (small caps, old-style, scripts like Arabic) and positioning (kerning, anchoring marks, etc.). In the ConTeXt font loader this conversion is done in Lua.

The result is stored in a condensed format in a cache and the next time the font is needed it loads in an instant. In the cached version the dimensions are untouched, so a font at different sizes has just one copy in the cache. Often a font is needed at several sizes and for each size we create a copy with scaled glyph dimensions. The feature-related dimensions (kerning, anchoring, etc.) are shared and scaled when needed. This happens when sequences of characters in the node list get converted into sequences of glyphs. We could do the same with glyph dimensions but one reason for having a scaled copy is that this copy can also contain virtual glyphs and these have to be scaled beforehand. In practice there are several layers of caching in order to keep the memory footprint within reasonable bounds.¹

When the font is actually used, interaction between characters is resolved using the feature-related information. When for instance two characters need to be kerned, a lookup results in the injection of a

¹ In retrospect one can wonder if that makes sense; just look at how much memory a browser uses when it has been open for some time. In the beginning of LuaTeX users wondered about caching fonts, but again, just look at how much browsers cache.

kern, scaled from general dimensions to the current size of the font.

When the outlines of glyphs are needed in Metafun the font is also converted from its binary form to something in Lua, but this time we filter the shapes. For a `cff` this comes down to interpreting the `charstrings` and reducing the complexity to `moveto`, `lineto` and `curveto` operators. In the process subroutines are inlined. The result is something that MetaPost is happy with but that also can be turned into a piece of a PDF.

We now come to what a variable font actually is: a basic design which is transformed along one or more axes. A simple example is wider shapes:



We can also go taller and retain the width:



Here we have a linear scaling but glyphs are not normally done that way. There are font collections out there with lots of intermediate variants (say from light to heavy) and it's more profitable to sell each variant independently. However, there is often some logic behind it, probably supported by programs that designers use, so why not build that logic into the font and have one file that represents many intermediate forms. In fact, once we have multiple axes, even when the designer has clear ideas of the intended usage, nothing will prevent users from tinkering with the axis properties in ways that will fulfil their demands but hurt the designers' eyes. We will not discuss that dilemma here.

When a variable font follows the route described above, we face a problem. When you load a TrueType font it will just work. The glyphs are packaged in the same format as static fonts. However, a variable font has axes and on each axis a value can be set. Each axis has a minimum, maximum and default. It can be that the default instance also assumes some transformations are applied. The standard recommends adding tables to describe these things but the fonts that I played with each lacked such tables. So that leaves some guesswork. But still, just loading a TrueType font gives some sort of outcome, although the dimensions (widths) might be weird due to lack of a (default) axis being applied.

An OpenType font with PostScript outlines is different: the internal `cff` format has been upgraded to `cff2` which on the one hand is less complicated but on the other hand has a few new operators—

which results in programs that have not been adapted complaining or simply quitting on them.

One could argue that a font is just a resource and that one only has to pass it along but that's not what works well in practice. Take LuaTeX. We can of course load the font and apply axis values so that we can process the document as we normally do. But at some point we have to create a PDF. We can simply embed the TrueType files but no axis values are applied. This is because, even if we add the relevant information, there is no way in current PDF formats to deal with it. For that, we should be able to pass all relevant axis-related information as well as specify what values to use along these axes. And for TrueType fonts this information is not part of the shape description so then we in fact need to filter and pass more. An OpenType PostScript font is much cleaner because there we have the information needed to transform the shape mostly in the glyph description. There we only need to carry some extra information on how to apply these so-called blend values. The region/axis model used there only demands passing a relatively simple table (stripped down to what we need). But, as said above, `cff2` is not backward-compatible so a viewer will (currently) simply not show anything.

Recalling how we load fonts, how does that translate with variable changes? If we have two characters with glyphs that get transformed and that have a kern between them, the kern may or may not transform. So, when we choose values on an axis, then not only glyph properties change but also relations. We no longer can share positional information and scale afterwards because each instance can have different values to start with. We could carry all that information around and apply it at runtime but because we're typesetting documents with a static design it's more convenient to just apply it once and create an instance. We can use the same caching as mentioned before but each chosen instance (provided by the font or made up by user specifications) is kept in the cache. As a consequence, using a variable font has no overhead, apart from initial caching.

So, having dealt with that, how do we proceed? Processing a font is not different from what we already had. However, I would not be surprised if users are not always satisfied with, for instance, kerning, because in such fonts a lot of care has to be given to this by the designer. Of course I can imagine that programs used to create fonts deal with this, but even then, there is a visual aspect to it too. The good news is that in ConTeXt we can manipulate features so in theory one can create a so-called font goodie file for a specific instance.

4 Shapes

For OpenType PostScript shapes we always have to do a dummy rendering in order to get the right bounding box information. For TrueType this information is already present but not when we use a variable instance, so I had to do a bit of coding for that. Here we face a problem. For TeX we need the width, height and depth of a glyph. Consider the following case:



The shape has a bounding box that fits the shape. However, its left corner is not at the origin. So, when we calculate a tight bounding box, we cannot use it for actually positioning the glyph. We do use it (for horizontal scripts) to get the height and depth but for the width we depend on an explicit value. In OpenType PostScript we have the width available and how the shape is positioned relative to the origin doesn't much matter. In a TrueType shape a bounding box is part of the specification, as is the width, but for a variable font one has to use so-called phantom points to recalculate the width and the test fonts I had were not suitable for investigating this.

At any rate, once I could generate documents with typeset text using variable fonts it became time to start thinking about a user interface. A variable font can have predefined instances but of course a user also wants to mess with axis values. Take one of the test fonts: Adobe Variable Font Prototype. It has several instances:

extralight	It looks like this!	weight=0 contrast=0
light	It looks like this!	weight=150 contrast=0
regular	It looks like this!	weight=394 contrast=0
semibold	It looks like this!	weight=600 contrast=0
bold	It looks like this!	weight=824 contrast=0
black high contrast	It looks like this!	weight=1000 contrast=100
black medium contrast	It looks like this!	weight=1000 contrast=50
black	It looks like this!	weight=1000 contrast=0

Such an instance is accessed with:

```
\definefont [MyLightFont]
  [name:adobevariablefontprototypelight*default]
```

The Avenir Next variable demo font (currently) provides:

regular	It looks like this!	weight=400 width=100
medium	It looks like this!	weight=500 width=100
bold	It looks like this!	weight=700 width=100
heavy	It looks like this!	weight=900 width=100
condensed	It looks like this!	weight=400 width=75
medium condensed	It looks like this!	weight=500 width=75
bold condensed	It looks like this!	weight=700 width=75
heavy condensed	It looks like this!	weight=900 width=75

Before we continue I will show a few examples of variable shapes. Here we use some Metafun magic. Just take these definitions for granted.

```
\startMPcode
draw outlinetext.b ("\definedfont
  [name:adobevariablefontprototypeextralight]%
  foo@bar")
(withcolor "gray")
(withcolor red withpen pencircle scaled 1/10)
xsize .45TextWidth ;
\stopMPcode
\startMPcode
draw outlinetext.b ("\definedfont
  [name:adobevariablefontprototypelight]%
  foo@bar")
(withcolor "gray")
(withcolor red withpen pencircle scaled 1/10)
xsize .45TextWidth ;
\stopMPcode
\startMPcode
draw outlinetext.b ("\definedfont
  [name:adobevariablefontprototypebold]%
  foo@bar")
(withcolor "gray")
(withcolor red withpen pencircle scaled 1/10)
xsize .45TextWidth ;
\stopMPcode
\startMPcode
draw outlinetext.b
("\definefontfeature[whatever]%
  [axis={weight:350}]%
  \definedfont
  [name:adobevariablefontprototype*whatever]%
  foo@bar")
(withcolor "gray")
(withcolor red withpen pencircle scaled 1/10)
xsize .45TextWidth ;
\stopMPcode
```

The results are shown in figure 1. What we see here is that as long as we fill the shape everything will look as expected but using an outline only won't. The crucial (control) points are moved to different locations and as a result they can end up inside the shape. Giving up outlines is the price we evidently need to pay. Of course this is not unique for variable fonts although in practice static fonts behave better. To some extent we're back to where we were with METAFONT and (for instance) Computer Modern: because these originate in bitmaps (and probably use similar design logic) we also can have overlap and bits and pieces pasted together and no one will notice that. The first outline variants of Computer Modern also had such artifacts while in the static Latin Modern successors, outlines were cleaned up.

The fact that we need to preprocess an instance but only know how to do that when we have got-



Figure 1: Four variants

ten the information about axis values from the font means that the font handler has to be adapted to keep caching correct. Another definition is:

```
\definefontfeature[lightdefault]
  [default]
  [axis={weight:230,contrast:50}]
\definefont[MyLightFont]
  [name:adobevariablefontprototype*lightdefault]
```

Here the complication is that where normally features are dealt with after loading, the axis feature is part of the preparation (and caching). If you want the virtual font solution you can do this:

```
\definefontfeature[inlinelightdefault]
  [default]
  [axis={weight:230,contrast:50},
  variablesshapes=yes]
\definefont[MyLightFont]
  [name:adobevariablefontprototype
  *inlinelightdefault]
```

When playing with these fonts it was hard to see if loading was done right. For instance not all values make sense. It is beyond the scope of this article, but axes like weight, width, contrast and italic values get applied differently to so-called regions (subspaces). So say that we have an x coordinate with value 50. This value can be adapted in, for instance, four subspaces (regions), so we actually get:

$$x' = x + s_1 \times x_1 + s_2 \times x_2 + s_3 \times x_3 + s_4 \times x_4$$

The (here) four scale factors s_n are determined by the axis value. Each axis has some rules about how to map the values 230 for weight and 50 for contrast to such a factor. And each region has its own translation from axis values to these factors. The deltas x_1, \dots, x_4 are provided by the font. For a PostScript-based font we find sequences like:

```
1 ⟨setvstore⟩
120 [10 -30 40 -60] 1 ⟨blend⟩ ... ⟨operator⟩
100 120 [10 -30 40 -60] [30 -10 -30 20]
  2 ⟨blend⟩ ... ⟨operator⟩
```

A store refers to a region specification. From there the factors are calculated using the chosen values on the axis. The deltas are part of the glyph specification. Officially there can be multiple region specifications, but how likely it is that they will be used in real fonts is an open question.

For TrueType fonts the deltas are not in the glyph specification but in a dedicated `gvar` table.

```
apply x deltas [10 -30 40 -60] to x 120
apply y deltas [30 -10 -30 20] to y 100
```

Here the deltas come from tables outside the glyph specification and their application is triggered by a combination of axis values and regions.

The following two examples use Avenir Next Variable and demonstrate that kerning is adapted to the variant.

```
\definefontfeature[default:shaped][default]
  [axis={width:10}]
\definefont[SomeFont]
  [file:avenirnextvariable*default:shaped]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on. Hermann Zapf

```
\definefontfeature[default:shaped][default]
  [axis={width:100}]
\definefont[SomeFont]
  [file:avenirnextvariable*default:shaped]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on. Hermann Zapf

5 Embedding

Once we're done typesetting and a PDF file has to be created there are three possible routes:

- We can embed the shapes as PDF images (inline literal) using virtual font technology. We cannot use so-called xforms here because we want to support color selectively in text.
- We can wait till the PDF format supports such fonts, which might happen but even then we might be stuck for years with viewers getting there. Also documents need to get printed, and when printer support might arrive is another unknown.
- We can embed a regular font with shapes that match the chosen values on the axis. This solution is way more efficient than the first.

Once I could interpret the right information in the font, the first route was the way to go. A side

effect of having a converter for both outline types meant that it was trivial to create a virtual font at runtime. This option will stay in ConTeXt as pseudo-feature `variablesapes`.

When trying to support variable fonts I tried to limit the impact on the backend code. Also, processing features and such was not touched. The inclusion of the right shapes is done via a callback that requests the blob to be injected in the `cff` or `glyf` table. When implementing this I actually found out that the LuaTeX backend also does some juggling of charstrings, to serve the purpose of inlining subroutines. In retrospect I could have learned a few tricks faster by looking at that code but I never realized that it was there. Looking at the code again, it strikes me that the whole inclusion could be done with Lua code and some day I will give that a try.

6 Conclusion

When I first heard about variable fonts I was confident that when they showed up they could be supported. Of course a specimen was needed to prove this. A first implementation demonstrates that indeed it's no big deal to let ConTeXt with LuaTeX handle such fonts. At the conference Adam Twardoch demonstrated the website `axis-praxis.org`, and we currently can support most of the fonts there quite well.

Of course we need to fill in some gaps which can be done once we have complete fonts. And then of course users will demand more control. In the meantime the helper script that deals with identifying fonts by name has been extended and the relevant code has been added to the distribution. At some point the ConTeXt Garden will provide the LuaTeX binary that has the callback.

I end with a warning. On the one hand this technology looks promising but on the other hand one can easily get lost. Probably most such fonts operate over a well-defined domain of values but even then one should be aware of complex interactions with features like positioning or replacements. Not all combinations can be tested. It's probably best to stick to fonts that have all the relevant tables and don't depend on properties of a specific rendering technology.

Although support is now present in the core of ConTeXt the official release will happen at the ConTeXt meeting in 2017. By then I hope to have tested more fonts. Maybe the interface has also been extended by then because after all, TeX is about control.

◇ Hans Hagen
 Pragma ADE
<http://pragma-ade.com>

Parametric math symbol fonts

Bogusław Jackowski, Piotr Strzelczyk and
Piotr Pianowski

1 Introduction

In 2007, Microsoft released their math-equipped MS Office along with the math OpenType (OTF) font Cambria. In the past 10 years, a dozen more OTF math fonts have been released — half of which were developed by the GUST e-foundry [4, p. 908].

Given the huge number of font vendors (see, e.g., [2]) and the correspondingly huge number of offered fonts, the nearly negligible number of math OTF fonts is somewhat puzzling. Leaving aside the reasons for such a state of the art, one conclusion seems obvious: math OTF fonts, despite having a well-defined standard which is undoubtedly an important advantage, are not particularly popular.

Thus, the question arises: is concentrating efforts on generating more math fonts reasonable? As far as the \TeX society is considered, the answer is equivocal: yes and no. Certainly, \TeX ies are interested in typesetting math texts, as \TeX is still the best tool for this purpose, therefore they would gladly use a broad variety of math fonts. However, \TeX ies do not actually need complete OTF math fonts. Thanks to new \TeX engines, notably Lua \TeX , math fonts can be assembled out of already existing text fonts and a “math trunk” — a set of math symbols from another font.

Below we present the idea of assembling math fonts on the fly using the Lua \TeX engine. We will try to justify that this approach is less laborious than the making of a regular math font, yet general enough for \TeX users.

2 What is a math font?

The contents of an OTF (also called Unicode) math font is specified by Microsoft documentation [9], and the Unicode Consortium report on Unicode support for mathematics [12]. The former specifies a special MATH table, a pivotal table for math OTF fonts. It contains information about glyph chains, stretchable glyphs, positioning of subscripts and superscripts, fractions, etc. The latter defines component alphabet sets (scripts) that are expected to be present in a math OTF font. The required components of a typical math OTF font are schematically shown in Figure 1.

As one can see, a math OTF font is, in fact, a collection of various fonts assembled into one entity. One of the reasons, the most important in our opinion, behind this arrangement is that nowadays

Composites (subfonts) of a math OTF font:



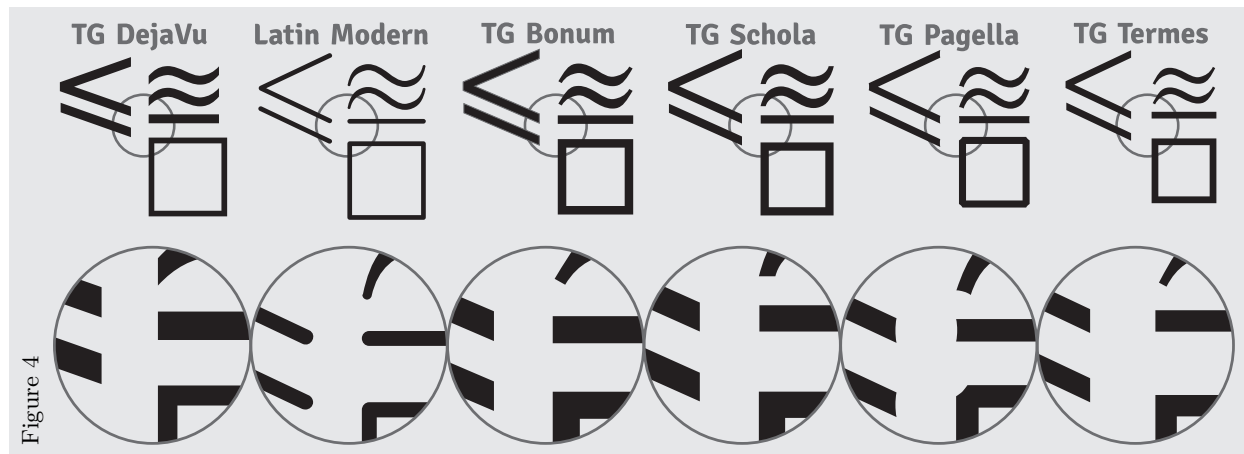
Figure 1

operating systems do not enable flexible handling of user-defined families (collections) of fonts — formatting editors usually handle 4-member families comprising regular, regular italic, bold and bold italic variants. \TeX users, however, are not bound to follow that restriction. The solution proposed in this paper follows from this observation.

3 Subscripts and superscripts

Subscripts and superscripts (by tradition, of the 1st and 2nd order) are obligatory for typesetting math; therefore, math fonts are expected to contain special glyphs which can be used for this purpose, also used in fractions and as root degree in radicals; for the sake of brevity, we’ll call these glyphs *pars pro toto* subscripts. They are accessed by the OTF feature mechanism, more precisely by the math extension feature `ssty` [10, 11].

Neither the Microsoft documentation nor the Unicode Consortium report ([9] and [12]) mentioned above specify which glyphs should be accompanied by subscripts; in the GUST e-foundry fonts, we have tried to limit their number, nevertheless, they make up about 30 percent of all glyphs.



mixing the \TeX Gyre DejaVu OTF math font with the DejaVu sans-serif variant.

Such a relatively simple header allows even inexperienced users to easily type math formulas with a chosen main font (in general, an arbitrary text font, DejaVu sans-serif in this case) along with the math symbols, i.e., braces, radicals, etc., taken from a chosen math font (in general, a math OTF font, TG DejaVu Math in this case) — see Figures 3a–3c.

Not only subscript sizes and proportions can be defined on the fly; also sidebearings can be controlled by appropriate font family definitions using the $\text{Lua}\TeX$ font loading option `extend` and the $\text{Lua}\TeX$ (originally from $\text{pdf}\TeX$) primitive command `\letterspacefont`, respectively.

5 What else do we need?

In the previous section we substantiated the statement that $\text{Lua}\TeX$ can be used, in a sense, as a “poor man’s font editor”. What cannot be easily handled from within $\text{Lua}\TeX$? The answer is: subtle details should be taken into account, provided that one cares — we do.

As we emphasized in our paper on the GUST e-foundry font projects [4, p. 326], an important aspect of a math font is the visual harmonizing of the alphanumeric glyphs and the symbols. Seemingly trivial glyphs, such as operator and relational symbols, may serve as a convenient example: they have slightly different shapes in each of our math fonts — see Figure 4 above. Another example is the optical similarity between the shape of integrals and the letter ‘long s’, which in turn is similar to the letter ‘f’ [4, p. 326].

Such details, in principle, could be controlled from within $\text{Lua}\TeX$; however, we would consider this to be overloading the functionality of $\text{Lua}\TeX$. Furthermore, we prefer to fiddle around with glyph

shapes using MetaType 1 [3], our favorite MetaPost-based tool.

6 How to tackle the problem?

We can pinpoint the problem to solve as follows: *given (say, by a customer) a font, add an adequate, i.e., optically consistent, math companion to be used in $\text{Lua}\TeX$ with the given font.* The solution consists of a few more or less obvious steps:

- ◇ prepare a generic set of $\text{Lua}\TeX$ macros;
- ◇ prepare a generic set of MetaPost/MetaType 1 macros for generating the basic set of math symbol glyphs;
- ◇ for this set of MetaPost/MetaType 1 macros, prepare a set of relevant parameters for a given font controlling ovalness, incisions, thickness of stems, x-height, etc.

The good news is that all the steps listed above are to a great extent accomplished or at least commenced:

- ◇ we use $\text{Lua}\TeX$ with the `unicode-math` package [5, 8], in our office (heavily exploiting Hans Hagen’s font handling macros — thanks!);
- ◇ a lion’s share of MetaType 1 macros which we use for generating GUST e-foundry fonts can also be used for this purpose;
- ◇ moreover, the MetaType 1 macros are, of course, parameterized — this is why we were able to release a new math OTF font once a year on average.

Our experience is thus optimistic, although it does not mean that nothing remains to be done. On the contrary. Putting it figuratively: it takes a few minutes to saw a plank, burnishing it takes a few hours. So far, we “have sawn the plank”.

7 Conclusions

A canonical math OTF font has many advantages, such as, e.g., universality — it can be used with various programs and various operating systems. At the same time, it is a “frozen” (unmodifiable) object — it is impossible to modify it without employing a font editor; e.g., none of the subfonts can be replaced with a user-chosen variant.

The method described in this paper is, on one hand, certainly less universal as it is restricted to the \TeX environment, but, on the other hand, provides a flexible tool that may prove useful (we hope) in practical applications.

Our thinking about implementing such an approach was triggered by customers’ demands, who (rarely, but still) wanted to have math formulas typeset with their “flagship” font; unflinchingly, it was none of the dozen math fonts mentioned in Section 1. Needless to say, the making of a respective complete math OTF font was not feasible.

Thus, we have a natural motivation to continue the work on this subject. We believe that before long we will be able to notify the \TeX community about some results.

8 Acknowledgements

Permanent and hearty thanks go to Hans Hagen for providing us with a marvelous pastime.

All trademarks belong to their respective owners and have been used here for informational purposes only.

References

Presentations, publications and packages:

- [1] Donald E. Knuth, *Computer Modern Typefaces*, Computers & Typesetting, vol. E, Addison-Wesley, Reading, Massachusetts, 1986
 - [2] Luc Devroye, *Font vendors*, <http://luc.devroye.org/vendors.html>
 - [3] Bogusław Jackowski, Janusz M. Nowacki, Piotr Strzelczyk, *MetaType 1: A MetaPost-based engine for generating Type 1 fonts*, 2001
Article: <http://www.ntg.nl/maps/26/15.pdf>
Presentation: <http://www.gust.org.pl/bachotex/2015/presentations/2015BJackowski.pdf>
Download: <https://ctan.org/pkg/metatype1>
 - [4] Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski, *GUST e-foundry font projects*, TUGboat, Vol. 37 (2016), No. 3, pp. 317–336, <https://tug.org/TUGboat/tb37-3/tb117jackowski.pdf>
 - [5] Will Robertson, Philipp Stephani, Khaled Hosny, *Experimental Unicode mathematical typesetting: The unicode-math package*, ver. 0.8d, 2017
<https://ctan.org/pkg/unicode-math>
- General purpose documentation:
- [6] *AMS Euler typeface*, https://en.wikipedia.org/wiki/AMS_Euler
 - [7] *FontForge* — George Williams and the FontForge project contributors
<http://fontforge.github.io/en-US/>
 - [8] *Lua \TeX Reference Manual*, ver. 1.0.4, 2017
<http://www.luatex.org/svn/trunk/manual/luatex.pdf>
 - [9] *MATH — The mathematical typesetting table*, updated 2017
<https://www.microsoft.com/typography/OTSPEC/math.htm>
 - [10] *OpenType specification*, ver. 1.8.1, updated 2017
<https://www.microsoft.com/en-us/Typography/OpenTypeSpecification.aspx>
 - [11] *Registered features — definitions and implementations (p-t)*, updated 2017
https://www.microsoft.com/typography/otspec/features_pt.htm
 - [12] *Unicode Technical Report #25. Unicode Support for Mathematics*, revision 15, 2017 — Barbara Beeton, Asmus Freytag, Murray Sargent III
<http://unicode.org/reports/tr25/>
- All links were tentatively accessed 2017-06-09.
- ◊ Bogusław Jackowski
Gdańsk, Poland
b_jackowski (at) gust dot org dot pl
 - ◊ Piotr Strzelczyk
Gdynia, Poland
p.strzelczyk (at) gust dot org dot pl
 - ◊ Piotr Pianowski
Trąbki Wielkie, Poland
p.pianowski (at) bop dot com dot pl

L^AT_EX News

Issue 27, April 2017

Contents

ISO 8601 date format	1
Further TU encoding improvements	1
Disabling hyphenation	1
Discretionary hyphenation	1
Default document language	1
Line spacing in parboxes	1

ISO 8601 date format

Since before the first releases of L^AT_EX 2_ε, L^AT_EX has used a date format in the form YYYY/MM/DD. This has many advantages over more conventional formats, as it is easy to sort and avoids the unfortunate ambiguity between different communities as to whether 01/02/2017 is the 1st of February or 2nd of January.

However there is another date format, formalised by the International Standard ISO 8601. The basic format defined by this standard is functionally equivalent to the L^AT_EX format, but using - rather than /. This date format is now supported in many Operating Systems and applications (for example the `date --iso-8601` command in Linux and similar systems).

From this release, L^AT_EX will accept ISO format date strings in the date argument of `\ProvidesPackage`, `\usepackage`, etc. Currently we recommend that you do not use this format in any packages that need to work with older L^AT_EX releases; the `latexrelease` package may be used with older releases to add this functionality. This change is handled in a special way by `latexrelease`: The package always adds support for ISO dates whatever format date is requested; this is required so that the necessary date comparisons may be made.

The new functionality can be seen in the startup banner which advertises `LaTeX2e <2017-04-15>`.

Further TU encoding improvements

The 2017/01/01 release saw the introduction of the new TU encoding for specifying Unicode fonts with LuaT_EX and X_ƎT_EX. There were a number of small corrections and additions in the patch releases updating 2017/01/01, and a further addition in this release, notably extended support for the dot-under accent, `\d`.

Disabling hyphenation

The existing L^AT_EX code for `\verb` and `verbatim` had some issues when used with fonts that were not loaded with hyphenation disabled via setting `\hyphenchar` to -1. In this release these `verbatim` environments use a `\language` setting, `\l@nohyphenation`, that has no hyphenation patterns associated.

The format ensures that a language has been allocated with this name. For most users this will in fact be no change as the standard `babel` language has for a long time allocated a language with this name.

In order that page breaks in `verbatim` do not influence the language used in the page head and foot, the format now normalises the language used in the output routine to a default language as described below.

Discretionary hyphenation

The L^AT_EX definition of `\-` has been adjusted so that it will insert the current font's `\hyphenchar`, as would the T_EX primitive. A comment in `source2e` has given this new definition since the first releases of L^AT_EX 2_ε, and in this release we finally acted upon this comment. Previously `\-` always inserted a - at a break point even if a different character would be used for automatic hyphenation with the current font.

Default document language

A new integer parameter `\document@default@language` is introduced; this is initialised to -1 but is set at `\begin{document}` to the language in force at that time if it has not been set by preamble code. This is very similar to the handling of the default color, and is used in a similar way to normalise the settings for page head and foot as described above. Users should not normally need to set this explicitly but it is expected that language packages such as `babel` may set this if the default behaviour is not suitable.

Line spacing in parboxes

Inside a `\parbox` L^AT_EX normalises the baseline spacing. However it has not previously reset `\lineskiplimit`. This meant that lines of a paragraph that have ascenders or descenders could be set with *closer* line spacing than lines without. This can easily happen if you use a `\parbox` in an AMS alignment, as they use a relatively large value of `\lineskiplimit`. As usual, the `latexrelease` package may be used to force the older behavior.

L^AT_EX News, and the L^AT_EX software, are brought to you by the L^AT_EX3 Project Team; Copyright 2017, all rights reserved.

L^AT_EX table columns with fixed widths

Frank Mittelbach

While attending a workshop on ConT_EXt at the TUG 2017 conference in Bachotek, I was introduced to some of the table generating macros of ConT_EXt.

The functionality they offer¹ is not very different from what is offered through L^AT_EX environments, sometimes with similar, sometimes with somewhat different syntax.

But there was one noticeable exception: In L^AT_EX, simple columns that are either centered or aligned left or right always automatically calculate their width based on the content of the cells. If a fixed column width is needed, then one needs a rather complicated set of constructions (using `p` columns and setting up the alignment manually) resulting in rather unreadable source documents. In contrast, ConT_EXt offers a simple preamble specification, such as `lw{3cm}` for something like “make this column 3cm with the material left aligned”.

This seems like a natural task, so I was a little surprised that I have never seen L^AT_EX users complain about the missing functionality or more precisely about the roundabout way that is necessary to achieve this² in a L^AT_EX tabular environment, e.g., `>\raggedright\arraybackslashp{3cm}`.

On the other hand, given `\newcolumnntype` provided by the `array` package, it should be fairly easy to provide a reasonable interface for such task. So my thought was to provide a column type `w` that takes two arguments: the alignment (such as `c`, `l` or `r`) and a dimension specifying the columns width. With that we can then specify a left-aligned column of 3 centimeters as `w{l}{3cm}` or even shorter as `wl{3cm}`. And since new column types can be defined from existing ones, we could shorten this further through a declaration such as

```
\newcolumnntype{C}[1]{wc{#1}}
\newcolumnntype{L}[1]{wl{#1}}
\newcolumnntype{R}[1]{wr{#1}}
```

so that the necessary preamble specification then simply becomes `L{3cm}`.

Simple L^AT_EX code

So how can we generate columns with a fixed width? A simple approach is to put each cell into a `\makebox` as this command allows making boxes of a specific

¹ Perhaps more accurately the functionality presented during the workshop — this was most certainly only a fraction of that is possible.

² And this is not exactly equivalent either, if the material overflows the available space.

width (first optional argument) and allows specifying the alignment within the box (second optional argument).

The `array` package supports the preamble specifiers `>{...}` and `<{...}` through which we can put material before and after the cell content. However, they do not allow us to use the cell content as an argument to a command we place into `>{...}`, so instead we need to use a slightly more elaborate way using the `lrbox` environment.

So first we load the `array` package (if that hasn't happened already) and then we allocate a box register to temporarily hold the cell content.

```
\usepackage{array} \newsavebox\cellbox
```

Then we define the `w` column type as follows:

```
\newcolumnntype{w}[2]{%
```

Before the cell content we start an `lrbox` environment to collect the cell material into the previously allocated box `\cellbox`.

```
>\begin{lrbox}\cellbox}%
```

Then comes a specifier for the cell content. We use `l`, but that doesn't matter as in the end we will always put a box of a specific width (`#2`) into the cells of that column, so `c` or `r` would give the same result.

```
l%
```

At the end of the cell we end the `lrbox` environment so that all of the cell content is now in box `\cellbox`. As a final step we put that box into a `\makebox` using the optional arguments of that command to achieve the correct width and the desired alignment within that width.

```
<\end{lrbox}%
\makebox[#2][#1]{\usebox\cellbox}}
```

The code above only uses high-level L^AT_EX constructs. It could be made slightly more efficient by using internal functions.

Going more low-level

One of the issues with the code from the previous section is that it will always set its text at natural width even when that overflows the available space (as the box register is copied). Furthermore, using `\makebox` with optional arguments means that an overflow will be silently accepted, as that command uses `\hss` internally for alignment. Thus an overprint needs to be detected visually.

Both can be appropriate depending on the circumstances, but often enough some squeezing and a warning if something overflows may be the more appropriate action. Therefore a second version (tabular column type `W`) to address these limitations can be helpful as well.

A possible implementation for this could be:

```
\newcolumnntype{W}[2]
  >{\begin{lrbox}\cellbox}%
  1%
  <{\end{lrbox}%
    \let\hss\hfil
    \makebox[#2][#1]{\unhbox\cellbox}}}
```

This is a bit sneaky, as it temporarily disables `\hss`, but given that the cell content is supposed to be fairly plain LR material this should be sufficient in essentially all cases.

Example usage

After providing this declaration we can now easily code tables with all or some column widths fixed, e.g.,

```
\begin{tabular}{|l|wr{12mm}|Wr{12mm}|r|}
  flexible & fixed (w) & fixed (W) & flexible \\
  123 & & & \\
  123456789 & 123456789 & 123456789 & 123456789 \\
  a b c d e & a b c d e & a b c d e & a b c d e \\
\end{tabular}
```

This gives the following result:

flexible	fixed (w)	fixed (W)	flexible
123	123	123	123
123456789	123456789	123456789	123456789
a b c d e	a b c d e	a b c d e	a b c d e

Observe the overflow marks in the third row as the material is wider than 12mm) while in the second column it overflows silently in rows 1, 3 and 4 (the latter being squeezed enough to fit in column 3). Also notice that `W` always overflows to the right while `w` overflows away from the alignment (i.e., to the left if the alignment is `r`).

Outlook

In my opinion it makes sense to predefine the `w` and `W` column types in the `array` package, as the short specifications such as `wc{1cm}` should be very useful to many people.

On the other hand `C`, `L` or `R` are likely to be used already for other purposes (e.g., indicating math columns), so it seems better to only mention those as one way to make use of the `w` or `W` column type if people desire to do so.

- ◊ Frank Mittelbach
Mainz, Germany
frank.mittelbach@at
[latex-project.org](http://www.latex-project.org)
<https://www.latex-project.org>

Using Markdown inside \TeX documents

Vít Novotný

Abstract

Markdown is a lightweight markup language that makes it easy to write structurally simple documents using a clean and straightforward syntax. Although various tools for rendering Markdown documents via \TeX exist, they tend to be built on top of \TeX rather than in \TeX .

This paper briefly presents existing tools and introduces a macro package for plain-based \TeX formats that takes a different approach. By making it possible to put snippets of Markdown-formatted text into arbitrary \TeX documents and exposing \TeX macros that control the rendering of Markdown elements, the package provides a convenient way of introducing Markdown into existing \TeX workflows.

1 Introduction

\TeX is a fine tool for typesetting many kinds of documents. It may, however, not always be the best language for writing them. Markup languages based on SGML and XML make it possible for an author to focus on the content of their documents without having to worry about the error messages produced by commands and unforeseen macro package interactions. The resulting documents can then be transformed not only to various \TeX formats, but also to other output formats that bear no relation to the \TeX world.

When preparing structurally simple documents, however, SGML and XML with their bulky syntax may feel too heavy-handed. For these kinds of documents, lightweight markup languages that exchange raw expressive power for clean and simple syntax are often the best choice. In this paper, I will focus on the lightweight markup language of Markdown [2].

Although the language of Markdown was originally envisioned as an HTML preprocessor, its syntax is agnostic to the output format, which makes Markdown useful as a general document markup language. Tools that provide conversion from Markdown to various \TeX formats are therefore readily available. One of the better-known free open-source programs that enable conversion from Markdown to \TeX is Pandoc [4]. Dubbed a Swiss Army Knife by its author, Pandoc enables the conversion between an impressive number of markup languages (e.g. \LaTeX , Con \TeX t, HTML, XML Docbook) and output formats (e.g. ODF, OOXML, or PDF). The tool has already been reviewed in *TUGboat* by Massimiliano Dominici [1].

Pandoc is a powerful multi-target publishing software and its ability to perform lossy conversions (such as from L^AT_EX to HTML) makes it extremely useful for document manipulation in general. However, if our sole goal is to use Markdown markup inside T_EX documents, Pandoc displays several weaknesses.

The ability to redefine the correspondence between Markdown elements in the input and T_EX macros in the output is limited. Processing the following Markdown document:

```
- Single underlines/asterisks denote emphasis.
- Double them for strong emphasis.
- The two may be freely mixed.
```

in Pandoc v1.17.2 using the command `pandoc -f markdown -t latex` produces the following L^AT_EX document:

```
\begin{itemize}\tightlist
\item Single underlines/asterisks denote
      \emph{emphasis}.
\item Double them for \textbf{strong emphasis}.
\item The \emph{two} \textbf{may be}
      \emph{freely} \emph{mixed}}.
\end{itemize}
```

While it is possible to redefine the produced L^AT_EX macros in theory, altering base macros such as `\item` or `\textbf` may break the document in subtle ways. The output is also not fixed and may vary between different versions of Pandoc.

Another desirable feature is sandboxing. Markdown is a static markup language without programming capabilities and may be used by ordinary users without much training. If Markdown documents are submitted to a system and then automatically typeset using T_EX, then these documents should certainly not be able to crash or halt the compilation, or to execute external programs using the `\write18` command and similar mechanisms. Pandoc does not offer much in these regards, since it permits T_EX macros in the input Markdown documents. There exist complex rules for deciding whether or not an occurrence of a T_EX special character should be kept or removed; the following document:

```
This {will} 2~n \begin{get} r-moved and \this
{won't} \begin{equation}2~n\end{equation}$2~n$.
```

when converted with Pandoc becomes:

```
This \{will\} 2\~{n} \textbackslash{}begin\{%
get\}r\textasciitilde{}moved and \this{won't}
\begin{equation}2~n\end{equation} \{(2~n)\}.
```

Apparently, the aim is to enable the use of mathematics and simple T_EX macros while retaining baseline compatibility with standard Markdown documents that may contain portions resembling plain T_EX.

As a result, users are limited in their ability to use T_EX inside their Markdown documents, but there is still plenty of rope left for halts, crashes, and external command execution.

Another inconvenience of Pandoc is its lack of integration with the T_EX distributions. T_EX documents without external dependencies and written in stable formats such as plain T_EX require virtually no maintenance. The use of external assets and actively developed formats such as ConT_EXt will require some attention each time there is a major T_EX distribution release. Software outside T_EX distributions such as Pandoc throws more variables into the mix, since different versions may produce different output even if the T_EX distribution stays the same. Besides the trickier maintenance, Pandoc's absence from T_EX distributions also means that it is unavailable in major T_EX services such as the collaborative text editors at <http://www.sharelatex.com/> and <http://www.overleaf.com/>.

Other major tools for rendering Markdown, such as MultiMarkdown, were reviewed and found to be plagued by similar design choices. With this knowledge, I decided to prepare a macro package for rendering Markdown inside T_EX that would take a different approach. The goals were:

- to make it easy to specify how individual Markdown elements should be rendered,
- to provide sandboxing capabilities, and
- to make sure that the package required nothing more than what was present in standard T_EX distributions.

2 The `markdown.tex` package

2.1 Architectural overview

The block diagram in figure 1 summarizes the high-level structure of the package. Working from the bottom of the diagram upwards, I will now describe the individual layers.

The translation from Markdown to T_EX is done by the Lunamark Lua library [3]. The library was modified, so that it would not depend on external libraries and so that it would produce an intermediate plain T_EX representation of the input Markdown document. This means that instead of Pandoc's T_EX representation (repeated here from the previous page for ease of reference),

```
\begin{itemize}\tightlist
\item Single underlines/asterisks denote
      \emph{emphasis}.
\item Double them for \textbf{strong emphasis}.
\item The \emph{two} \textbf{may be}
      \emph{freely} \emph{mixed}}.
\end{itemize}
```

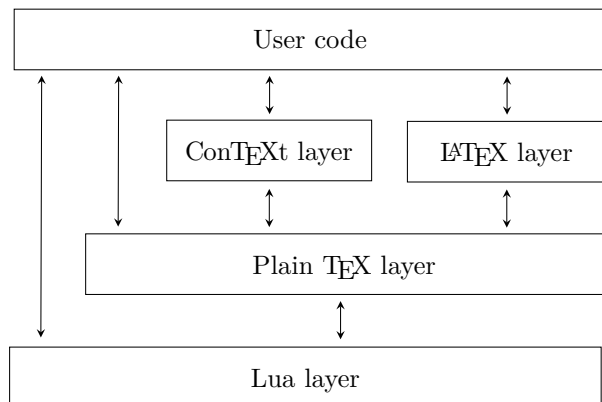


Figure 1: A block diagram of the package

the library will produce the following representation:

```
\markdownRendererUlBeginTight
  \markdownRendererUlItem
    Single underlines/asterisks denote
    \markdownRendererEmphasis{emphasis}.
  \markdownRendererUlItemEnd
  \markdownRendererUlItem
    Double them for
    \markdownRendererStrongEmphasis{strong
    emphasis}.
  \markdownRendererUlItemEnd
  \markdownRendererUlItem
    The \markdownRendererEmphasis{two}
    \markdownRendererStrongEmphasis{may be}
    \markdownRendererEmphasis{freely
    \markdownRendererEmphasis{mixed}}.
  \markdownRendererUlItemEnd
\markdownRendererUlEndTight
```

This representation has two useful properties: it works with any T_EX format that uses the same specials as plain T_EX does and it represents the logical structure of the input Markdown document in terms of macros that can be freely redefined by the user.

The plain T_EX layer exposes the capabilities of the Lua library as T_EX macros and provides default definitions for the `\markdownRenderer⟨Name⟩` macros from the intermediary representation. Macro package developers are encouraged to redefine the `\markdownRenderer⟨Name⟩Prototype` macros that correspond to the default definitions. When the LuaT_EX engine is used, the Lua library is accessed directly. Otherwise, the shell escape (`\write18`) mechanism is used (see [5, sec. 3.2.5] for details).

The L^AT_EX and ConT_EXt layers correct for some idiosyncrasies of the respective T_EX formats and provide user-friendly variants of several macros from the plain T_EX layer and sane default definitions for the `\markdownRenderer⟨Name⟩` macros. Developers are encouraged to contribute layers for other formats.

2.2 Usage examples

This section contains examples for `markdown.tex` (version 2.5.3). These should give an idea of the capabilities of the package. The examples are in L^AT_EX for ease of exposition. As noted in the previous section, the L^AT_EX layer of the package is reasonably thin and the examples can therefore be easily adapted for the plain T_EX and ConT_EXt layers. For brevity, the examples will contain only the body of a L^AT_EX document assuming the following preamble:

```
\documentclass{article}
\usepackage{markdown,graphicx}
```

To typeset the examples, you can use the `lualatex` or `pdflatex -shell-escape` commands. For further information, see the package documentation [5].

Sandboxing disables hybrid markup and is enabled by default. As a result, the following example:

```
\begin{markdown}
$\sqrt{x^2 + y^2}$
\end{markdown}
```

will produce $\sqrt{x^2 + y^2}$. To enable the use of hybrid markup, the `hybrid` option needs to be specified. The following example:

```
\begin{markdown*}{hybrid}
$\sqrt{x^2 + y^2}$
\end{markdown*}
```

will produce $\sqrt{x^2 + y^2}$ as expected. You may also create a partial sandbox; the following example enables the use of non-breaking spaces:

```
\begin{markdown*}{renderers={tilde=~}}
Bartel~Leendert~van~der~Waerden
\end{markdown*}
```

With hybrid markup, using underscores and backticks may produce unexpected results. That is because Markdown uses underscores for emphasis and backticks for inline verbatim text, whereas L^AT_EX uses underscores for math subscripts and backticks for opening quotation marks. Preceding characters with a backslash disables their special meaning in Markdown, as the following example shows:

```
\begin{markdown*}{hybrid}
\‘‘This is a quote with $a_{\text{subscript}}$.’’
\end{markdown*}
```

This, however, makes the text difficult to both read and write. Alternatively, you can disable backticks and underscores in Markdown, as the following example shows:

```
\begin{markdown*}{hybrid, codeSpans=false,
underscores=false}
‘‘This is a quote with $a_{\text{subscript}}$.’’
\end{markdown*}
```

Emphasis can still be denoted via asterisks.

Since the standard Markdown syntax covers only the essentials, the package supports a number of syntax extensions that allow you to mark up moderately complex content without hybrid markup. The following example gives a taste of what is available:

```
\begin{markdown*}{citations, contentBlocks,
                 footnotes, inlineFootnotes}
[~1] and [inline footnotes] are highly useful,
as shown in the table below.
```

```
/table.csv
```

The table was borrowed from @doe12 [p. 34].

```
[~1]: Footnotes
\end{markdown*}
```

See [5, sec. 2.1.2] for the full list of syntax extensions.

The Markdown syntax permits online images, but the package currently does not handle these in any special way. Therefore, if you would like to download and typeset online images, you will need to provide your own implementation. One such implementation is shown in the following example:

```
\begingroup
\catcode'\@=11
\catcode'\%=12
\catcode'\^^A=14
\global\def\markdownRendererImage#1#2#3#4{^^A
\immediate\write18{^^A
  if printf '%s' "#3" | grep -q ^http; then
    OUTPUT="$(printf '%s' "#3" | md5sum |
              cut -d' ' -f1).^^A
              $(printf '%s' "#3" |
                sed 's/.*[.]//')";
    if ! [ -e "$OUTPUT" ]; then
      wget -O "$OUTPUT" '#3' || rm "$OUTPUT";
      convert "$OUTPUT" png:"$OUTPUT";
    fi;
    printf '%s%' "$OUTPUT" > \jobname.fetched;
  else
    printf '%s%' "#3" > \jobname.fetched;
  fi}^^A
{\everyeof={\noexpand}^^A
\edef\filename{@@input"\jobname.fetched" }^^A
\includegraphics[width=\textwidth]{\filename}}
\endgroup
```

```
\begin{markdown}
![TUGboat](https://tug.org/tugboat/noword.jpg)
The Communications of the TeX Users Group
\end{markdown}
```

Note that this implementation expects a Unix-like operating system with a Bourne-compatible shell. It also assumes that the `md5sum`, `wget`, and `convert` binaries are installed and that the \TeX engine has shell access, among other things.

3 Conclusions

With the new `markdown.tex` package, it is now possible to typeset Markdown documents in \TeX without the need for external tools. This notably enables the use of Markdown in collaborative text editors such as <http://www.sharelatex.com/> and <http://www.overleaf.com/>, and in other services where tools from outside \TeX distributions are unavailable.

The package also gives the authors full control over how individual Markdown elements are rendered and how much access to \TeX markup the Markdown documents have. The former encourages creative domain-specific use of the Markdown syntax and the latter enables the use of \TeX for the unsupervised typesetting of user-submitted Markdown documents.

Acknowledgements

I gratefully acknowledge the funding received from the Faculty of Informatics at the Masaryk University in Brno for the development of the package.

I would also like to thank John MacFarlane, the creator of the Lunamark Lua library, for having released Lunamark under a permissive license that enabled its inclusion into the package.

References

- [1] Massimiliano Dominici. An overview of Pandoc. *TUGboat*, 35(1):44–50, 2014. <http://tug.org/TUGboat/tb35-1/tb109dominici.pdf> (visited on 2016-08-15).
- [2] John Gruber. Daring Fireball: Markdown, 2013. <http://daringfireball.net/projects/markdown/> (visited on 2016-08-15).
- [3] John MacFarlane. Lunamark, 2012. <http://jgm.github.io/lunamark/doc/> (visited on 2016-08-17).
- [4] John MacFarlane. Pandoc: A universal document converter, 2016. <http://pandoc.org/> (visited on 2016-08-15).
- [5] Vít Novotný. *A Markdown Interpreter for \TeX* , 2017. <https://ctan.org/pkg/markdown> (visited on 2017-04-10).

◇ Vít Novotný
Nad Cihelnou 602
Velešín, 382 32
Czech Republic
witiko (at) mail dot muni dot cz
<https://github.com/witiko>

**GMS, the ‘General Meta-Scenarios’:
A proper extension to the `l3expan` package
of the `expl3` bundle and language, two years
later**

Grzegorz Murzynowski

Abstract

This paper presents the current state of the GM-Scenarios, an esoteric mini-language of one-char instructions covering and extending the functionalities of the `l3expan` package of the `expl3/LATEX3` bundle.

In an automata approach, the GMSs are described as a DPDA, deterministic pushdown automaton and a respective context-free language, and the arguments (not quite formal) are given to back this point of view.

A diagram of what I believe to be the automaton actually implemented, and a formal grammar that I believe to be a grammar of the GM-Scenarios language, are included.

In the final remarks, I accept the friendly critiques received about the GMSs at TUG@BachTeX 2017, and reply in an “I’m fixing that” manner.

Contents

1	Why again?	219
1.1	The name	219
2	A brief history of logistic growth of resources or: What do we take for granted	219
3	The <i>inspiratio</i> : <code>l3expan</code>	222
3.1	The Pandora’s box of new letters . .	222
3.2	“Let’s make it shorter and don’t repeat...”, or: how the GMSs began . .	222
3.3	GMS as a nano-Copernican revolution (against <code>l3expan</code> (?))	224
4	GMS: the automaton	224
4.1	The automaton: diagram	225
5	GMS: the formal language, and program	226
5.1	The <code>\::</code> macro) and <code>\specification</code>	226
5.2	The destination, <code>\tau</code>	229
5.3	The pre-ps. and pickers, <code>\langle(\pi*\sigma^*)^*\rangle</code> .	229
5.4	The meta-operators, <code>\langle\lambda\rangle</code>	231
5.5	The general permutations, or the <code>\FSM</code> without grouping	231
5.6	Parsing the braces, or: <code>\BDSM</code> . . .	232
5.7	The <code>\subs’n’refs</code>	233
5.7.1	The replacements, ‘=:’	233
5.7.2	“The arguments from beyond”, <code>\lambda</code>	233
5.7.3	Snapshots and references, ‘*’	234
6	Rough budgeting, a.k.a. cost estimation	235

Grzegorz Murzynowski

7	Friendly critiques at TUG@BachTeX 2017	236
8	Final remarks	237
8.1	“Thank Heavens, it’s not the Premium Class”	237
8.2	The end, or <code>\epsilon\chi\alpha\tau\omicron\nu</code>	237

Motto:

Pani domu zaś, wydawszy przedtem dokładne wskazówki, sama powinna siedzieć przy stole wesoła i uśmiechnięta, i co najwyżej dawać służbie oczami znaki, gdy tego zajdzie potrzeba.

Concerning the lady of the house, she should sit at the table cheerfully and smiling and, having given the service exact instructions *before*, now give them signs with her eyes only if it is necessary.

Marja Ochorowicz-Monatowa,
“Uniwersalna książka kucharska”, 1910

Disclaimer 1. It’s not Computer Science.

I mean, I hope it to be so metaphorically, i.e., I hope the matter discussed in this paper is “crazy”. But it should be mentioned, and at the very beginning, that this paper is not a scientific article, and not on automata theory in particular. It’s just a presentation of a certain `TeX` program or set of macros, and the Reader should not be misled by the computer-scientific terms used.

The automaton and formal language presented in this paper might be at the very best considered an example or *exercise*, and the statements, especially those concerning the automaton’s and language’s classes, and computational complexity of the algorithm/program, considered hypotheses to be proven or disproved, or adjusted in their assumptions.

I’m not a computer scientist, i.e., I’m not educated in the theories of Computer Science, I’m just an “aspiring `TeX`nician” who hasn’t even read the entire *TeXbook* (you don’t use any quotation marks around the name of a Sacred Book, do you?), and just practices `TeX` in as Epicurean way as he [I] can. The only argument that might accrue to my benefit is that writing programs in `TeX` gets me my daily bread, and I’m still alive, and, moreover not sued for industrial sabotage or such.

Disclaimer 2. About (Non-)ASCII chars and the tailored font Ubu Stereo.

The `expl3` language is kept strictly ASCII. Any characters outside of ASCII that occur in this paper, especially those from the “distant far-aways” of the Unicode, or even from the Private Use Area (**PUA** henceforth), are a sin of “mine, and mine only”.

Their rôle, and the rôle of `expl3` in making me use them, is discussed in section 3.

All of them, as coming from different scripts, including the Chinese Traditional ‘記’ U+8A18 ‘write down, record, remember’, and ‘用’ U+7528 ‘use, apply, make use of’, and Math Fette Fraktur, and some even FontForge’d by myself, occur all together just in one (and only one) font in the world, named Ubu Stereo, based on Ubuntu Mono Regular, and enriched with glyphs only from fonts licensed freely.

1 Why again?

This paper follows my presentation at TUG@BachotEX 2017 and summarizes the present state of the GMS mechanism that I conceived in the beginning of the year 2015 as a “just a bit more friendly interface to the l3expan macros”. While the general concept of the machine (automaton) remains the same, and so do most of its operators and constructs, there are new concepts I have added since, and, which might be more important, “now first it seems my thought begins to span it.”¹

Which in turn gives the hope that I’ll present the subject in a more understandable manner this time.

Compared to the *TUGboat* 36:3 (2015) paper, the new things are:

- the name of the machinery, GM-Scenarios instead of GMOA;
- a meta-reflection discussed in sec. 3.3;
- shift in understanding from “DFA with Mysterious Something” to DPDA, Deterministic Push-Down Automaton;
- deprecating the pre-processors and pickers “homonymic” with the l3expan counterparts;
- a handful of new pre-ps. and pickers, ‘**ᄀ**’, ‘**ᄁ**’, ‘**ᄂ**’;
- new aliases for those now deprecated: ‘**Aa**’, ‘**Ää**’, ‘**Ee**’, ‘**Ēē**’, ‘**Vv**’, ‘**Ṽṽ**’;
- the “subs’n’refs” mechanism in FSM s;
- the “arguments from beyond” in FSM s;
- pure-ASCII and HTML-like alternate forms of operators.

What has to be mentioned, or: confessed, is that the GMS machinery is still in a state of development, and at the time of submitting this paper, some of the new things are not quite operational yet.

Much of this intense development comes from the fact I use the GMSs intensely, anywhere I can use my own T_EX packages, [the source] of this paper being no exception.

¹ Walt Whitman / Ralph Vaughan Williams, “A Sea Symphony”: “O vast Rondure”.

1.1 The name

Why did I transition from GMOA, “General Manipulation Of Arguments”, to GMS, “General Meta-Scenarios”? The obvious part is, why the two initial (nomen-omen) letters remain the same.

False humility makes me say I should put my name on my work so that Humankind knows who to blame. But that aspect should not be overestimated, Gustav Mahler has the same initials, and were this paper and its dereference of no other use, may the mention of him and his cosmic *The Eighth* advocate me *in die illa tremenda* ☺.

But, concerning the “G”, the mechanism is in fact quite general. My everyday work is programming in T_EX, X_YL^AT_EX to be precise, and I believe the fact I’m still employed by the same Company, and paid, is quite an argument for its usability and usefulness, at least in my hands. Quite general, dare I say, because the first thing it does is cover the functionality of l3expan.

The “GMOA” name focused attention on the ability of the machine to pre-process the arguments for a single expl3-function (usually, a macro). But the GMSs do more than that: they set and rearrange parts of the code before it’s actually run.

Further, because the mechanism operates on the “future” program code, it truly can be called “meta-”. And, because it tells how that “future” code should be executed, truly can it be called “scenario”. And, a “scenario” rather than “didascalica” or “markup”, as it is separated from the code it operates on.

Now, with just a tiny little modal collapse, i.e., the reasoning step “If sth. might be, then let it be”, we get—let the mechanism be called “Meta-Scenarios”, *quod erat demonstrandum*.

2 A brief history of logistic growth of resources

or: What do we take for granted

Have you ever read the L^AT_EX 2_ε sources? Thank Heavens, it’s richly commented, and the sub-structures, which in a more usual language would be called “subroutines” and “functions”, or “classes” and “methods”, are presented in pseudocode before they’re actually expressed in T_EX.

An excerpt of it is presented in Fig. 1.

Why is it so obscure, why do even the primitives not bear shorter names, not for saving memory (negligible), but for the sake of readability?

L^AT_EX 2_ε has been written in the times when memory was so precious and scarce, that William Henry Gates III, even though he didn’t utter exactly those famous words, actually was thinking, as “all”

```

\def\declare@robustcommand#1{%
  \ifx#1\@undefined\else\ifx#1\relax\else
    \@latex@info{Redefining \string#1}%
  \fi\fi
  \edef\reserved@a{\string#1}%
  \def\reserved@b{#1}%
  \edef\reserved@b{\expandafter\strip@prefix%
    \meaning\reserved@b}%
%<autoload> \aut@global
  \edef#1{%
    \ifx\reserved@a\reserved@b
      \noexpand\x@protect
      \noexpand#1%
    \fi
    \noexpand\protect
    \expandafter\noexpand\csmname
    \expandafter\@gobble\string#1 \endcsmname
  }%
  ...
}

```

Figure 1: A fragment of the L^AT_EX 2_ε source, File d: ltdefs.dtx, 2004/09/18 v1.3g

of his contemporaries, that 640 kB of RAM would be enough for “anything”, and “at least for 10 years”.

In those times “this new T_EX format, L^AT_EX”, had to do some serious “garbage collection” in order to run at all and finish the job.

That’s where `\onlypreamble` comes from, and that’s why not only was the code written with as few new macros as possible, but also with reusing names, those reuses sometimes irrelevant to their contents and goal.

That’s where DocStrip comes from, whose primary task was to Strip the comments (Documentation) from the files, so as not to slow down their *reading in*.

Also, it was pure T_EX not ε-T_EX, whose expandable primitive `\stricmp` is neatly wrapped in `expl3`’s `\str_if_eq[_x]:nn[TF]`, and `\str_case:nn[TF]`, and also with no `\numexpr` or `\dimexpr` that allow expandable integer or dimen computations.

Let us think a moment, if we could write a T_EX program or document that (with all the fonts, libraries, macro packages &c.!) works in no more than 500 kB of RAM. Yes, 500 *kilo* bytes, not megabytes.

So, in that time, and in those extreme conditions, that was the “optimum and beyond”. Let’s have this in mind and see what we take for granted in these days’ plenty, and the programmatic indulgence it’s causing.

Now, consider the following fragment of File r: ltfsdcl.dtx, dated: 2005/09/27, v3.0k, giving the definition of `\DeclareMathSymbol`:

```

\edef\reserved@a{\noexpand\in@{%
  \string\mathchar}{\meaning#1}}%
\reserved@a
\ifin@
...

```

It’s the shortest example I’ve found so far of what could be named “repetitive programmatic constructs”.

A macro is `\edefed` just to put it in the input stream immediately after it’s defined. As can easily be guessed, it’s done to give well-prepared arguments to the macro `\in@`, that checks whether its `#1` is a sub-string, or rather, a sub-tokenlist of `#2`, and sets the Boolean switch `\ifin@` accordingly.

What is at hand at the point we wish to use `\in@` needs to be expanded in a certain way first.

Those “certain ways” of expanding first, and, in my version, not only expanding, we call **pre-processing** henceforth.

This particular schema above, with the macro `\in@` “frozen” with `\noexpand`, first argument consisting of `\string` and a c.s.(1), and the second of `\meaning` and a (supposed) c.s.(2), repeats on previous and subsequent pages quite a few times, just with different control sequences (1) and (2).

And that is just one schema, and the simplest/shortest, of many found just from the beginning till File r.

Then, maybe following The Sources’ example, most (L^AT_EX 2_ε-style) macro packages and document classes repeat the same manner of repeating those “repetitive programmatic patterns”. Not even with short aliases for `\expandafter` and `\noexpand` (cf. remark 3 on p. 236.)

“Repetitive programmatic patterns”. Do you see the irony? Isn’t the very essence of computer programming to make the machine do the repetitions, if possible?

Let’s repeat: this “if possible” is the answer to the question of why the L^AT_EX 2_ε Authors repeat so many things: in those times, not-repeating them was *not* possible. But nowadays, it is. Let’s see what has l3expan got to offer.²

For the sake of disambiguation, let’s assume that ‘`#1`’ of the above code is the c.s. token ‘`\life`’. And don’t forget to set the catcode of ‘`@`’ to 11 ‘letter’, because with `expl3`, ‘`@`’ is ‘other’ by default.

² As a topic not fully relevant to the GMS, we skip the discussion on naming ‘`\in@`’ “the `expl3` way”, with ‘`:nn`’ signature, and generating its ‘`:oo`’ variant.

Let us recall that the `expl3` catcode regime is that ASCII underscore ‘`_`’ and colon ‘`:`’ are made letters, blanks are ignored, including blank lines, and tilde ‘`~`’ is made `space10` instead.³

In addition, there is Hungarian notation in the “function” names, i.e., the “function”’s signature added in its name after the colon, and other smart naming conventions.

All this results in the first impression, at least mine, “what the [...] is this??”, but then, when you get used to it, in growing and growing appreciation of the readability and “spaciousness” of the code. Also, you may see that errors are less likely to occur, and when they do, they are easier to backtrace.

Now, back to the example.

```
[\char_set_catcode_letter:N \@ ]
\::o \::o \:::
\in@ {\string\mathchar}{\meaning\life}
```

The `expl3`-function `\::o` in the first step gets the respective ⟨balanced text⟩, i.e., the argument, next to “itself”, i.e., to the 2nd-step macro. That next macro hits the ⟨text⟩ with `\expandafter` over the opening brace, and then the 3rd macro, that had been put next to `\expandafter`, puts the just-hit argument in the “storage of processed arguments”, and passes control further.

As the `\:::` macros are essential for understanding further in this paper, let me explain them in more detail. The definitions are translated here to “Traditional `TeX`”. In real `l3expan` they all use the `expl3` aliases and/or wrappers for the primitives.

```
\long\def \::o #1 \::: #2#3 {
  \expandafter \__exp_arg_next:nnn
  \expandafter {#3} {#1} {#2} }
\long\def \__exp_arg_next:nnn #1#2#3 {
  #2 \::: { #3 {#1} } }
```

So, the one-level expansion of the first `\::o` in the example above, results in:

```
#1 ->\::o % “the tail of pre-ps. sequence” in general.
#2 ->\@in
#3 ->\string\mathchar
\expandafter \__exp_arg_next:nnn
\expandafter {\string\mathchar }
{\::o } {\@in }
```

and after the `\expandafter`’s fire, we get

```
(\string)
\__exp_arg_next:nnn {\mathchar} {\::o } {\@in }
```

³ Not “ASCII tilde of catcode 10 ‘space’”, because it’s not made “funny space” of the character code `0x7e` via the `\lcode` trick, just honestly via `\catcode=`, ifx you know the difference.

[`\` to illustrate the fact that the backslash, and thus the whole control sequence, is “dead”.]

Then `__exp_arg_next:nnn` restores the order. Let’s see that in slow motion.

```
\__exp_arg_next:nnn #1#2#3 -> {#2\::: {#3{#1}}}
#1 ->\mathchar, % no space after the former c.s. is
      another sign it’s “dead”
#2 ->\::o ,
#3 ->\@in
```

and thus we get:

```
\::o \::: {\@in {\mathchar}}
```

plus what was already there,

```
{\meaning \life }
```

After performing two-level expansion of “this other `\::o`”, i.e., replacing it with its definition, and firing the `\expandafter`’s, we get

```
(\meaning )
\__exp_arg_next:nnn {> \mathchar"2A.} % \life is
      usually ‘> undefined’, but once you give it some
      Deep Thought... ☺
      {} {\@in {\mathchar}}
```

and after expansion of `__exp_arg_next:nnn`,

```
\::: {\@in {\mathchar}> \mathchar"2A.}
```

Now, the mysterious Triple Colon⁴ Macro `\:::` that served as the delimiter of the tail of the pre-processors in/for the `\:::`’s (“Double-Colon-with-a-Letter” Macros). After all the `\:::`’s have expanded, it comes out as the “identity” macro, `\@firstofone` from `LATEX 2ε`:

```
> \:::=\long macro:
#1->#1.
```

and the braces covering the main macro and all the pre-processed arguments disappear, as if the End of Time came “and all things previously hidden are now revealed”:

```
\@in {\mathchar}> \mathchar"2A.}
```

This way, we saw “in slow motion”, how the “repetitive programmatic patterns” found in the `LATEX 2ε` sources and `LATEX 2ε`-style macro packages and document classes, might be vastly simplified and made more readable using the macros defined in the `l3expan` package.

And “here comes Mommie!”⁵ — here come I, and say: look at those repeating backslashes and colons. What if we delegate repeating them to the machine, and we ourselves type just what’s essential, i.e., the final letters?

⁴ A Polish cartoon series “Kapitan Bomba” gives the term “triple colon” quite *another* meaning while describing the “Kurvinox” alien species’ anatomy.

⁵ Patrick Swayze in [spoiler alert] the opening scene of “To Wong Foo, thanks for everything, Julie Newmar”.

3 The *inspiratio*: l3expan

3.1 The Pandora’s box of new letters

Let me make another apparent digression, which is in fact an important explanation I owe the Readers and, maybe even more, the L^AT_EX3 Team.

Having the guts and nerve to abandon the Plain and L^AT_EX 2_ε convention of making ‘@’ catcode 11 ‘letter’, and to make letters of ‘_’ and ‘:’ instead, is a huge inspiration and broadens my mind horizons, comparable with some kind of spiritual enlightenment, or with Dostoyevskian “*Но если Бога нет, тогда всё довольно...*” [‘But, if there’s no God, then anything is allowed...’].

I perceive this bold move as the main inspiration for my own attempts to “think out of the box”. In my implementation it comes out more like Pandora’s box, as has been kindly and amiably pointed out to me at this BachoT_EX, as I’ll discuss later.

In order to observe the naming conventions of expl3, especially the division of a c.s. into “scope” and module parts, and seeing the need for a more structured module part, I chose the letter ‘:’ (Modifier Letter Vertical Line, U+02C8), with catcode 11 ‘letter’ in X_ƒT_EX (as it is in Unicode), as the “unofficial” word separator.

```
\__gme'int_...:...
```

denotes a module-local L^AT_EX3-function of the module ‘gme’, and its submodule ‘int’.

I use yet another letter, ‘∞’, U+1525 Canadian Syllabics SH, as the character separating the final part of a c.s., intended to describe its particular role in a multi-step construct, or in a family of macros, e.g.:

```
\∞_make'gay:nnn
\∞_make'gay"the'Yuletide:nn
\∞_make'gay"Gigi"today:nn
```

On the other hand, choosing a character rare enough so it could serve as an ideogram, like ‘☾’ U+26B8 Black Moon Lilith, or the one discussed next, let us structure the module part of names just with it and abbreviated description, like

```
\☾int_... \☾tl_... \☾dim_...
```

which I use to name my additions and “completions”⁶ to the expl3 respective modules.

As you see, it becomes “...тогда всё довольно” (‘...then anything is allowed’), indeed. But — for the

⁶ The fact that some of my views, such as expecting an Esperanto-like symmetry from “the new L^AT_EX programmers’ language”, e.g., the Boolean constants, i.e., ‘\bool_const:Nn’, just like there’s any other ‘\<type>_const:Nn[...]’, diverge a bit from what’s actually there in expl3, neither does make it “incomplete”, nor diminishes the utter respect and gratitude I have for its Authors. Hence the quotation marks.

good cause of brevity, as brevity means better readability.

3.2 “Let’s make it shorter and don’t repeat...”, or: how the GMSs began

Going still further in this direction, I make a letter also of ‘:’, U+22EE, Vertical Ellipsis (there’s also the Triple Colon character, U+2AF6, maybe it would be preferable), to get a symbol similar and referring to the Double and Triple Colons of l3expan, yet at the same time saying “I differ from them, be careful, I might be orthogonal!”

As already mentioned, one of my goals is to make code short and as free from repetitions as possible. This attitude resembles that of Webern towards music, toutes proportions gardées. And so with the T_EX code making use of the GMS.⁷

Returning to the main narrative, let’s replace Double-Colon-with-a-Letters with Just-a-Letters, i.e., let’s trim leading ‘\:::’, and precede the whole thing with a two-(newly-declared)-letter word ‘\:::’. Since I’m “thinking in type not in sound”, I’ve no idea how to pronounce this ideogram. What first, or rather *who* comes to my mind, is Aja.

```
\:::o \:::o \:::
\@in {\string\mathchar }{\meaning\life }
\::: I o o : ...
```

And that’s what the core and chronologically earliest part of the GMS does. Translates a sequence of letters into the sequence of l3expan ‘\:::’s.

One thing that needs explaining is the letter ‘I’, which doesn’t correspond to any of the l3expan “Double-Colon-with-a-Letter”s of the line above.

I found it a bit confusing in the l3expan convention that the first token, i.e., the assumed “function” for which the assumed arguments are pre-processed, is not reflected in the DCwLs, or rather is, but at the very end, in the Triple Colon. While thinking of what’s going on here not as preparing arguments for a “function”, but as a sequence of operators applied to a sequence of <text>s (cf. sec. 3.3, p. 224), it becomes clear that the Identity operator is missing in the l3expan notation, and that’s the ‘I’ in mine.

Then the desire for Symmetry wakes up and joins in, another monster conceived of L^AT_EX3 *inspiratio*,⁸ namely, from looking at the “data types”

⁷ Typical reaction of a person listening to “normal” [Western] music at first hearing of a piece by Webern is: “What?? It’s not music, it’s some separate and random sounds!” The analogy with GMS holds.

⁸ The Latin verb “inspiro” means ‘to breathe into [something/someone]’, and is used in the Vulgate and hence in the Christian narrative to describe G*d giving life to the first human after making his body out of earthly dust or clay,

and their declarators, setters, and naming conventions — a desire to make the pre-processing mechanism fully symmetric with respect to braced- and unbracedness of the results, as the `N-` and `n-` type arguments and `l3expan` pre-processors already do. Apparently. And, not quite, as Frank Mittelbach explained himself at BachoTeX 2015, correcting my (mis)understanding and thus unmasking the idea of “braced/unbraced” *result* to be “mine and mine only”.⁹

While the original `expl3`’s idea of `N-` and `n-` type arguments refers to un- and bracedness of the *arguments*, i.e., the *input*, my (mis)understanding, and the idea stemming from it in GMS, refers to un- and bracedness of the *results*, i.e., to whether the pre-processed ⟨balanced text⟩ should be “returned” in braces: when the letter is lowercase, or without them: when the letter is uppercase.

Therefore, “I’ve made my decision”¹⁰ to deprecate the `l3expan` operators (letters), and maybe turn off handling of them in the future.

So, the convention of GMS is that the lowercase letters correspond to “returning” the pre-processed ⟨text⟩ in braces, while the uppercase “return” the ⟨text⟩ unbraced, which might seem strange at first glance, e.g., when the pre-processor hits a multi-letter control sequence with `\string`, but is useful, e.g., in defining a macro with parameters delimited with detokenized (‘other’-ised) characters, e.g., `␣` or `␣` `macro` `->` .

The idea of a set of operations closed in some aspects is usually a good idea, unless we are not scarce of memory or time, and these days we are not, oh, no, we aren’t.

Also, while opposing or at best “orthogonal” to the concepts and conventions of `expl3` at first glance, this ⟨lower⟩/⟨upper⟩ convention fits with the more general paradigm of making the language fully regular, much as Esperanto is.

For those two reasons, i.e., not to mislead from the original `expl3` concepts, and to observe the ⟨lower⟩/⟨upper⟩ *result* convention, I replaced the ASCII lowercase ‘o’ with the Latin lowercase a, ‘a’, and thus the example use of `\::o` should be rewritten to:

```
\::o \::o \:::
  \@in {\string\mathchar}{\meaning\life}
\:: I a a : ...
```

But “that’s [not] all, folks”, since the goal is to make the code short. And `\string` is used so often that it’s worth its own pre-processor. And here it is:

and also, conceiving the Child by the Most Venerable Virgin “from G*d’s Spirit”.

⁹ cf. “Evita”, “Eva’s Final Broadcast”.

¹⁰ cf. “RuPaul Drag Race”.

‘s’/‘S’. Also, `\meaning` might be very useful in some applications, and although I personally haven’t yet had such a need, introducing a new pre-processor for it is a matter of five minutes of adding the respective macros, plus [indefinite time] to choose the letter/symbol.

I chose ‘`␣`’ and ‘`␣`’, having in mind that ‘`m`’ stands in the ancient `xparse`, and also in `gmcommand`, for ‘mandatory argument’, and perhaps the most famous question about meaning is Freia’s “Was deutet die Name?” in the finale of “Das Rheingold”. Then, `math Fraktur` because ASCII ‘`D`’ is already taken by the `expl3` “Don’t” pseudo-signature.

With these “particulations”, the example turns into a five-token GM-Scenario with one mandatory separator char between ‘`\:::`’ and the pre-processors (remember it’s not a `space10` in the `expl3` or `gme3u8` catcode regimes, it’s an ignored char [`space`]₉, so it’s not even officially read, yet it establishes the limit of a multi-letter control sequence, without which it would be ‘`\:::Is`’ (a bit like, say, alcoholic addiction of a parent, which is not talked about, yet keeps the children from inviting friends home), and just three tokens of things processed. Let’s put it all together:

```
\edef\reserved@a{%
  \noexpand\in@{\string\mathchar}{%
    \meaning#1}%
\reserved@a           % The LATEX 2ε sources
\::o \::o \:::
\@in {\string\mathchar}{\meaning\life} % expl3
\:: Is␣ : \@in \mathchar \life      % GMS
```

Note that the 2nd and 3rd ⟨text⟩s are written without braces, since they are not necessary either for syntax correctness or for clarity, the latter provided by simplicity of this particular GMS.

By the way, all component macros and primitives of `l3expan`’s “function” `\::o`, and thus also of GMS’s `\::: ...a... :`, are expandable.

This is the case with all the `l3expan` and GMS pre-processors, if only their very nature allows it, i.e., if the pre-processing does not involve an assignment (other than this one and only assignment of `\relax` to a c.s. raised by ‘`\K... \L`’).

As mentioned earlier, `l3expan` provides various types of arguments’ pre-expansion.

The `\::f` preprocessor applies `\romannumeral-`0`, which expands argument tokens until the first unexpandable token is seen. Because `-`0` is a complete ⟨number⟩ in the sense of *The TeXbook*, even if the argument expands to digit(s), `\romannumeral` is satisfied with the `-`0` and, as this number is negative (minus the character code of character ‘`0`’, namely `-48`),

expands to ε (empty sequence of tokens). Thus we get an “AFAP” (‘As Far As Possible’) expansion. Not many things move me as deeply as this trick.

Some of the pre-processors rendering the value of a \LaTeX variable also use `\romannumeral-`0`, depending on the \LaTeX variable’s type. The current implementation of the `_tl` type, for instance, as parameterless macros and not, e.g., as `\toks` registers, allows for rendering their values with just an ‘ $\%$ ’ (`\expandafter`), or even using that \LaTeX variable “as is”.

So far, the things described might be considered just another user interface for `l3expan`, maybe more user-friendly, if anything. Also, the operators that `l3expan` “lacks” are just superpositions of one that already exists, most often the `[o]/a`, with some of \TeX ’s expandable primitives, like `\:o\the`, `\:o\the`, `\:o\the`.

So, what are the *real* enhancements I’ve made?

3.3 GMS as a nano-Copernican revolution (against `l3expan` (?))

Besides “stripping off one backslash and two colons” shown in the previous section, probably the most important enhancement made by me (if it may at all be attributed to me with such a strong inspiration¹¹), and fundamental for any further development, is a change of the point of view, quite Copernican in this nano-scale:

Thinking of (`l3expan` and) the GM-Scenarios not as pre-processors of (individual) arguments for an `expl3` function or macro, but —

as a sequence of operations applied, not necessarily 1:1, to a sequence of \langle text \rangle s, where \langle text \rangle is an undelimited or delimited argument of a resp. macro.

“(?)” in the section’s title, because it’s not very likely that `expl3`’s Authors do not realize the power of `l3expan`, rather they deliberately abstain from using it in its full glory (*if* they do in fact), and the “revolution” declared above is more of a shift of my own *understanding* of `l3expan`. Like Dr. Pierre Abelard says in the preface to his famous “Sic et non”: “It’s rather us lacking G*d’s grace [ability] to read [and understand] than them [the Authors] lacking G*d’s grace [ability] to write.”

Let’s go back to the excerpt from the \LaTeX 2_{ε} sources and think of it this way: in the end, we wish to give \TeX a two-parameter (undelimited) macro

¹¹ It’s infinitely easier to expand/develop something than to invent it in the first place. `l3expan` does things I’ve never thought of in the 11 years of my \TeX nician’s life. Or, if I ever did, it was: “Nah ... it’s impossible; you just can’t hit the 2nd undelimited argument with `\expandafter` since you don’t know how many tokens there are in the first one”.

`\@in` with both of the arguments hit with some expandable operators.

What would be (conceptually) done, is:

1. hit the 1st argument with `\string`,
2. hit the 2nd argument with `\meaning`,
3. prepare `\@in` to go first,
4. put the tokens resulting from 1 next to `\@in` to go 2nd,
5. put the tokens resulting from 2 next to those of 4.

What if we wish to prepare the arguments as described above, but instead of knowing the “function” they are for already, give a placeholder for it and be able to pass any relevant macro “later”, i.e., as an argument?

Or, if we wish to pass those arguments twice, for two different “functions”? For instance, having a c.s.(1) and an \langle integer expression \rangle (2), first declare (1) as an ‘`_int`’ variable, and then initialize it with (2)?

One of the “basic needs” `l3expan` does not satisfy, is — changing the arguments’ order. Another is replicating them, as \LaTeX 2_{ε} ’s `\@dblarg` does, for instance. And yet another, grouping \langle text \rangle s together in one common pair of braces. The latter two actions are the reason why I mentioned it’s not always a 1:1 correspondence.

Consider the following GMS:

```
\def \what's'the'Question #1 {
  \: :  1 2 3 4 5 6 7 8 9 : \mathchar \life #1 }
```

Knowing that the Musical Natural [Pitch] Sign ‘ $\%$ ’ declares a “natural permutation”, I hope it’s clear, what those “underlined digits” mean. How many of them might there be, i.e., how many \langle text \rangle s can an `\: :` handle at a time? Currently, up to 25, referring the first 9 with `1...9`, and then with `A...P`, for the “uppercase” pre-processing, or `Q...Z`, `q...z` for “lowercase”.

But anyway, the last operator counts for the un- or bracedness, so even though the ‘`3`’, ‘`1`’, and ‘`2`’ are all “uppercase”, only the \langle text \rangle referred to as ‘`3`’ is rendered without braces, while the other with, because of the lowercase ‘`s`’ and ‘`b`’.

And this, and other features, are worth a section of their own, so —

4 GMS: the automaton

The above example of changing the order of arguments is rather simple. And, seeing what is referred to as ‘`1`’, ‘`2`’, and ‘`3`’, poses no problem.

But when there are more \langle text \rangle s to make a permutation of, it seems more reasonable to label them with some numbers. The following example uses

the characters ‘Opening Lenticular Bracket Ordinal Number Omega’, ‘ ω ’, put at PUA+E9EA, and ‘Closing Lenticular Bracket Ordinal Number Omega’, ‘ ω ’, at PUA+E9EB. (Both of them designed by me, together with the plain ‘Ordinal Number Omega’, ω , PUA+E9E9).¹²

‘ ω ’ starts a part of a GMS being a permutation of $\langle \text{text} \rangle$ s with possible repetitions and grouping, and additional pre-processing of them. Let’s call it: **Finite Sequence Manipulation**, henceforth **FSM**.¹³

‘ ω ’ delimits the (explicitly labelled) sequence of $\langle \text{text} \rangle$ that’ll become the **elements** of this FSM, so that all may be absorbed by an internal macro delimited with this char.

(By the way, those two chars are declared in my Emacs as matching parentheses, so they are highlighted properly.)

```
[\gme''on] % set the gme3u8/exp3 catcodes
\pdef % \protected\def
\make'exhyphen #1 {
  \:: I  $\omega$  1 [[ 2 $\ddot{A}$ =2 ]216 3{42 $\boxminus$ } : % a \GMS
  \lccode
  1 `
  2 \c_catcode_active_tl
  3 \lowercase
  4 {\protected\def }
  5 {\penalty\exhyphenpenalty \hskip%
    \c_zero_skip }
  6 #1 % two tokens in definition, but replaced with a
    single char in runtime.
   $\omega$ 
  \f_catcode:: `#1\{ _active::
}
[\gme''off]
```

What we consider the automaton in this paper, is a set of macros that pass control to each other subsequently, just like states and transitions of a deterministic push-down automaton, and initialized by the macro `\::` (or another of the family, as will be discussed later), and finished by a colon.

In the example above, the GM-Scenario is the code commented as such, starting with ‘`\::`’ and fin-

¹² The Ordinal Number ω not the Cardinal \aleph_0 because in the context of permutations, the ordering is fundamental, and the proper name for the Cardinal \aleph_0 in its Ordinal aspect is ω .

Then, ω not some arbitrary symbol, because “Sky is the limit”, theoretically number of $\langle \text{text} \rangle$ s handled in an FSM is arbitrary (finite), and limited only by the capacities of the hardware and Time of our Universe, as the computing complexity of this part of GMS’s seems to be at least $O(n^2)$ Time, and at least $O(n)$ Space.

¹³ The religious allusion of this acronym is deliberate, may He be always *al dente*.

ishing with ‘:’, if we take the syntactic approach, or all the code starting with `\::`, and finishing with ‘ ω ’, if we look semantically. (It is not always clear where a GMS in the latter sense ends, since its instructions might be determined dynamically in the “runtime”, thanks to the “inter^Ruptions”.)

The letter ‘I’ refers to the first (balanced text) following the colon, i.e., `\lccode`, and says ‘just take it and return as is, but without braces’. Then, the ‘ ω ’ sign declares a **labelled FSM**, which means, that the automaton should now expect a permutation, possibly with repetitions, additional pre-processing, and grouping, of a certain number of $\langle \text{balanced texts} \rangle$, and that those $\langle \text{texts} \rangle$ have been already labelled, i.e., each of them preceded by proper alpha-digit (not necessarily starting from 1 and keeping the “increment by 1” rule), and that after all those texts there’s the delimiter ‘ ω ’, so that absorbing all the permutation elements is performed in a one-level expansion of a macro with a ω -delimited parameter.

Then, the fragments ‘1’, ‘216’, and ‘3’ are translated into macros “get the element with the label (\cdot)”, and put those elements in given order. Only, before the element with the label ‘2’ is taken, the translation of the fragment ‘[[2 \ddot{A} =2]’ hits that element with the “double `\expandafter`”, and replaces the original with the result of that 2-level expansion. We’ll discuss this in more detail in section 5.7.

Then, the part ‘42 \boxminus ’ translates into grouping the elements #4, #2, and #5 in one pair of braces together, and the element #5 within braces by itself. We discuss the grouping (bracing) mechanism, and its consequences in terms of the hierarchy class of the automaton, in section 5.6.

4.1 The automaton: diagram

Presenting the GM-Scenarios automaton, we use the following conventions:

- the symbol “ $_ \bullet$ ” denotes ‘an arbitrary representative of the class \bullet ’, where “ \bullet ” is the (usually one-character) name of the class, which may be homonymic with one of its members, or even the only one. One can think of it like an abbreviated Ruby Manual convention of typing an instance of a class, say, ‘Array’ as ‘an_Array’, just with the preposition stripped off; or, as sort-of conforming to the `expl3` convention of indicating the type of a variable (a data carrier) with underscore and type at the end of its name.
- “ $\downarrow \cdot$ ” means ‘push (\cdot) down the stack’, where (\cdot) is 1 for an opening brace, or i for an opening ‘[[’ bracket, which is used to mark the start of $\langle \text{subs'n'refs} \rangle$, and matching with ‘]]’.

- “↑: 1|0|i” ‘pop from the stack, and then there’s 1|0|i resp. on top of the stack’. Note that the symbol *i* might be considered the initial symbol for the ⟨subs’n’refs⟩ sub-stack.
- “[{action}]” ‘an action, usually assignment, performed “sideways” with a default value, in case of a transition that skips some intermediate state(s).
- “_•” ‘I’m not a usual label, in fact, I’m a sub-automaton’. Both of those sub-automata are depicted in the same figure.

5 GMS: the formal language, and program

There’s a theorem about a correspondence between finite automata and formal languages, namely, coupling a formal language with an automaton that recognises it.

And since a formal language might be described with a formal grammar, there’s also a natural correspondence between automata and formal grammars, namely, that $A \sim G$ iff $\exists L$ such that L is the language recognised by automaton A , and at the same time is defined by formal grammar G .

In this sense, the automaton depicted in fig. 2, and the grammar described in fig. 3, do correspond with one another.

Therefore, henceforth, I’ll be describing the language, freely switching between its formal grammar, and the automaton recognising it.

And, since this is not Computer Science, just a presentation of a program, I’ll be also explaining how the program works, which from the point of view of Automata Theory is all “side effects” at best.

5.1 The $\langle \backslash:: \rangle$ macro and $\langle \text{specification} \rangle$

The tokens of a $\langle \text{specification} \rangle$ are hit with $\backslash\text{string}$ one-by-one so they get catcode 12 “other”, except those expanded within an “inter^ruption”.

The $\{_1$ and $\}_2$ tokens may be used as they are, and if the $\langle \backslash:: \rangle$ -macro works straightforwardly (only $\backslash::$ does, as for now), they don’t even have to be balanced.¹⁴ Also, their “rôle” might be played by $\{$ and $\}$, as they’re recatcoded to letters in the gme3u8 catcode regime, and might be translated in macros that work faster than the main GMS machinery.

Since the main iterating macro has one unde-limited parameter, even in the usual catcode regime the blank chars are skipped and may serve just to

¹⁴ To be precise, each token of the GMS-charclass ‘{’ has to be balanced with a token of GMS-charclass ‘}’, but the cat-codes are irrelevant now as all the tokens are hit with $\backslash\text{string}$ one-by-one. Also, it’s possible, e.g., to generate missing }’s in an “inter^ruption” using $\backslash\text{ucharcat}$.

improve readability. Of course, in the expl3 regime, they are already ignored on the TEX reading level.

$\backslash::$ is a parameterless macro of the initial state, called KN, ‘[I] Know Nothing’. It first $\backslash\text{expandafters}$ $\backslash\text{string}$ and then launches the macro. ‘ fK ’ is a one-letter c.s. equivalent to $\backslash\text{cname}$, Tironian Et U+E970 ‘ $\text{}$ ’ another escape char, and ‘ f ’ is $\backslash\text{expandafter}$:

```
 $\backslash:: -> \text{fK} \text{f} \backslash::\text{!}_s\text{KN:N} \backslash\text{string}$ 
```

Other $\langle \backslash:: \rangle$ -macros do the same at some point, only first they absorb an entire $\langle \text{specification} \rangle$ as an argument delimited with ‘:’₁₁, and either check if such a specification has already been parsed and recorded (predefined), and use the pre-defined if so, or perform the predefinition instead of reorganising subsequent $\langle \text{text} \rangle$ s, or, $\backslash::$ 用記‘...’, are control sequences which the $\langle \text{specification} \rangle$ is part of.

We’ll discuss the basic version, ‘ $\backslash::$ ’.

The subsequent characters of $\langle \text{specification} \rangle$ are hit with $\backslash\text{string}$ and picked one-by-one, their “char-class” is determined, and proper transitions are performed accordingly, which TEX nically amounts to inserting further and further “telescopic” $\backslash\text{cname}$ ’s, i.e., the sequences of tokens that could and should be transformed into a control sequence at the very moment the matching $\backslash\text{endcname}$ is seen. Except, the immediate predecessor of such an $\backslash\text{endcname}$ is $\backslash\text{expandafter}$, and the token next to $\backslash\text{endcname}$ is another $\backslash\text{cname}$:

```
 $\backslash\text{cname name-1} \backslash\text{expandafter} \backslash\text{endcname}$   
 $\backslash\text{cname name-2} \dots$   
 $\dots \backslash\text{endcname}$ 
```

I think of this trick as an (architecture) arc or a bridge; and I think of $\backslash\text{cname} \dots \backslash\text{endcname}$ as the stator(s) of an electric motor, which make(s) the stuff between them spin. Hence the Ubu Stereo/PUA signs based on Japanese quotation marks:

```
 $\text{fK name-1} \text{fK name-2} \text{fK} \dots \text{fK}$ 
```

So, as a “side-effect” of parsing of the GMS of section 4, a translation is made:

```
 $\text{f}_\text{::}::\text{!}_\text{prepare}'\tau\{\zeta\}:w \text{f}_\text{::}::\text{I}$   
 $\text{f}_\text{::}::\text{!}_\text{prepare}'\text{FSM}'\text{w} \text{f}_\text{::}::\text{f}\#1 \text{f}_\text{::}::\text{I}$   
 $\text{f}_\text{::}::\text{!}_\text{subs}^{\text{r}}\text{refs}^{\text{r}}\text{start} \text{f}_\text{::}::\text{f}\{0\}$   
 $\text{f}_\text{::}::\text{f}\#2 \text{f}_\text{::}::\text{in}'\text{f}: \text{f}_\text{::}::\text{B}\text{A} \text{f}_\text{::}::\text{I} \text{f}_\text{::}::\text{subs}^{\text{r}}\text{refs}^{\text{r}}\text{!}\text{!} \text{f}_\text{::}::\text{f}\{0\}$   
 $\text{f}_\text{::}::\text{f}\#2 := \text{f}_\text{::}::\text{subs}^{\text{r}}\text{refs}^{\text{r}}\text{!}\text{!}$   
 $\text{f}_\text{::}::\text{!}_\text{resume}'\text{FSM} \text{f}_\text{::}::\text{f}\{0\}$   
 $\text{f}_\text{::}::\text{f}\#2 \text{f}_\text{::}::\text{I} \text{f}_\text{::}::\text{f}\#1 \text{f}_\text{::}::\text{I} \text{f}_\text{::}::\text{f}\#6 \text{f}_\text{::}::\text{I} \text{f}_\text{::}::\text{f}\#3 \text{f}_\text{::}::\text{I}$   
 $\text{f}_\text{::}::\text{B}\text{E} \text{f}_\text{::}::\text{B}\#4 \text{f}_\text{::}::\text{B}\text{B} \text{f}_\text{::}::\text{B}\text{B} \text{f}_\text{::}::\text{B}\#2 \text{f}_\text{::}::\text{B}\text{B} \text{f}_\text{::}::\text{B}\text{B}$   
 $\text{f}_\text{::}::\text{B}\#5 \text{f}_\text{::}::\text{B}\text{B} \text{f}_\text{::}::\text{B}\text{B} \text{f}_\text{::}::\text{q}\text{i} \text{f}_\text{::}::\text{q}\text{q}::\text{!}_\text{FSM}'\text{craw}^{\text{r}}\text{start} \text{f}_\text{::}::\text{f}\}$   
 $\text{f}_\text{::}::\text{!}_\text{yield}:w \text{f}_\text{::}::\text{f}\}$ 
```

where ‘ f ’ symbolizes the ‘ fK ’ arch in the “:!-run” (parsing-translation), and then, after turning them

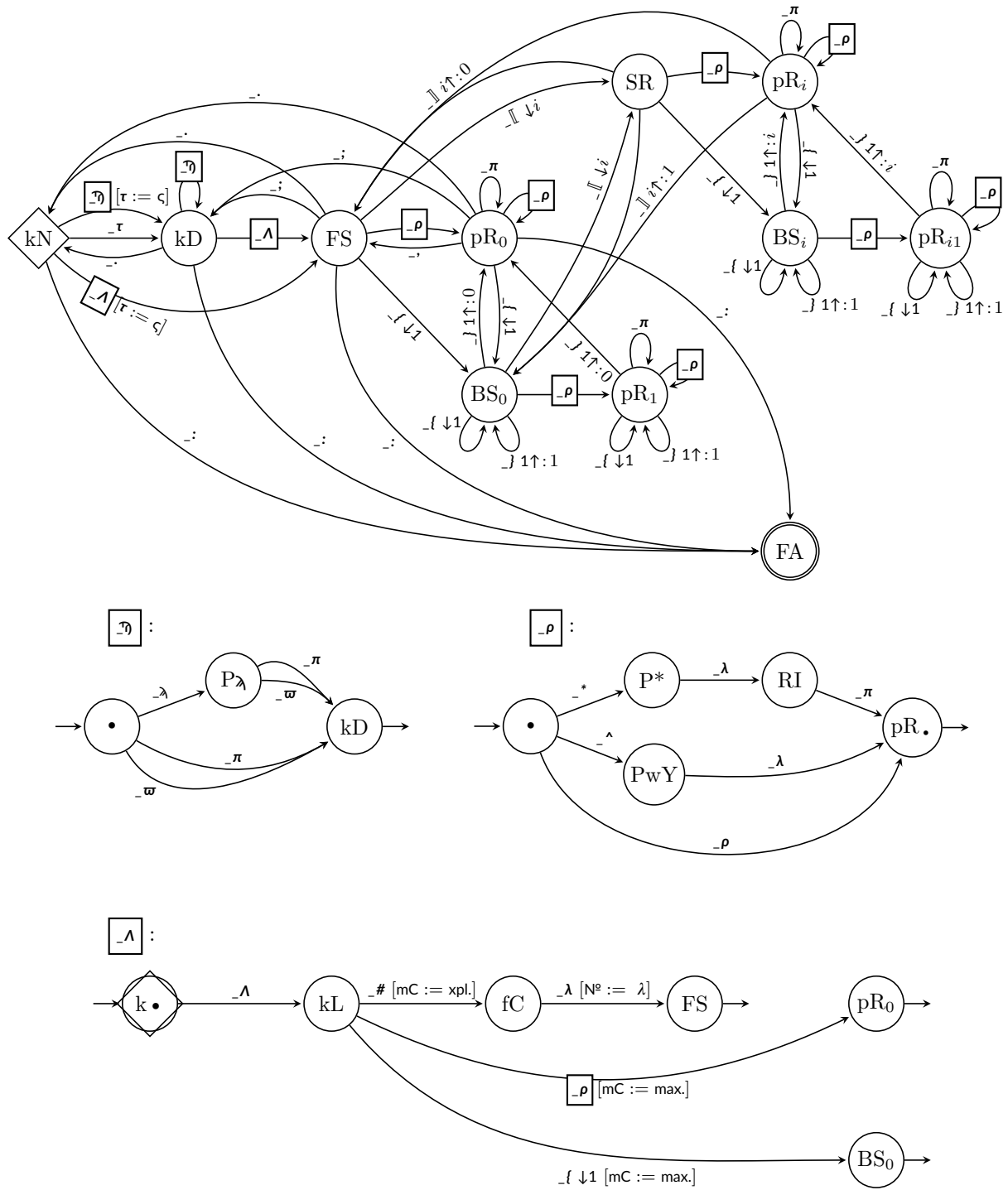


Figure 2: The GM-Scenarios deterministic push-down automaton.

GMS two years later. A complete madness. But — Turing-complete or not?

all in control sequences “backwards” from right to left, the escape chars of the resulting macros.

Not wishing to repeat everything that has already been said in the *TUGboat* 36:3 (2015) paper, let’s just restate that:

- ‘\`F#2’ is a macro of the ⟨FSM⟩ part, performing “Get the element #2 from the “shelf” and put it on the “slab”.
- ‘\`I’ is “Put the result of previous pre-processing to the FSM’s result storage, without the outer braces.
- ‘\`Bε’ begins the ⟨BDSM⟩ part, and is an “empty ⟨text⟩” argument for further binary operation of reverse concatenation, denoted with —
- ‘\`Bσ’ that takes the two most recently processed ⟨partial result⟩s, and (conceptually) glues them into one, in reverse order.
- ‘\`B⊕’ is a BDSM unary of “No wrap”/“Strip off the braces” — “Pass it further in an open envelope”, while
- ‘\`B⊗’ is (also unary) “Pass it further in a sealed envelope”, i.e., “Wrap it in braces”.
- ‘\`qι’ finishes the ⟨BDSM⟩ part and puts its result into the enclosing ⟨FSM⟩’s result container. The Epigraphic Reversed P, U+A7FC, ‘q’, indicates “Reverse Polish Notation”, as this is what’s going on in the ⟨BDSM⟩’s.

And the result of running this, is:

```
\f_lccode:: `* % a \noexpand-ed active char acquired
    by two-level expansion of \c_active_tl.
    `#1
\f_lowercase:: {\f_pdef:: *}{
  \f_penalty:: \f_exhyphenpenalty::
  \f_hskip:: \c_zero_skip }
```

The transitions are labeled not with particular characters but with equivalence classes of: ⟨ π ⟩s, ⟨ τ ⟩ (destination tokens) &c.

It’s probably not a significant savings of memory or other costs of computation, but a great simplifying of the code. And making it more change- and development-robust as e.g., adding a new argument type, which is denoted with a char of equivalence class ⟨ π ⟩, does not require any changes in the automaton.

5.2 The destination, ⟨ τ ⟩

Parsing of a ⟨specification⟩ starts with determination of the “destination”, i.e., the way the result of the next ⟨FSO⟩ is yielded:

- ε If no explicit destination token is given,¹⁵ the usual “just once” is assumed, as l3expan’s \: :ε/

¹⁵ I.e., the first char met is none of ς σ ξ .

_exp_arg_next:nnn do. This is equivalent to the use of ς .

- ς Greek letter small sigma final form, the open variant, for “συναγωγή πολύ” /synagoge poly/, ‘gather (as) many’ (with intended associations with the correlation between social diversity and open-mindedness of people). Therefore let us call this “just once and multi”.
- σ stands for “συναγωγή μόνο” /synagoge mono/, ‘gather [as] one’, Greek letter small sigma middle form, the closed variant, to be associated with enclosing of all the picked and pre-processed arguments in one common resulting pair of braces. (“Just once and as one”.) Useful if a GMS (is expandable and) is to prepare a single argument or {⟨balanced text⟩}.
- ξ for Greek “ξανα”, ‘[use] again’: the result is put back as input for further parts of the specification, quite like the ruminants do.

5.3 The pre-ps. and pickers, ⟨(π*σ*)*⟩

Most of these letters directly correspond to an expl3 “argument type” and the respective \: :ε macro, or extrapolate their ideas, even maybe towards a kind of a completeness or full(er) symmetry.

With respect to their actions, they could be divided in four groups:

- $\mathbb{1}$ the identity operators, $\mathbb{1}\iota$ (TeXnically, they *pick* an undelimited argument, so they can be also described as “pickers”);
- \mathfrak{N} the pre-processors (TeXnically also pickers, as above);
- \checkmark the special pickers, picking a delimited argument, but not applying anything to it;
- \times the discarders (or destroyers), picking an un- or delimited argument and discarding it.

We have

$$\begin{aligned}\langle\pi\rangle &= \mathbb{1} \cup \mathfrak{N}, \\ \langle\sigma\rangle &= \checkmark \cup \times,\end{aligned}$$

and the main reason for distinguishing between (the $\mathbb{1}$ s and) the \mathfrak{N} s (the ⟨ π ⟩s), and the \checkmark s and \times s (the ⟨ σ ⟩s) is that the latter are not allowed in ⟨FSM⟩s, and in fact don’t make much sense there.

So, let us see what they do. The ones homonymic with l3expan “argument types” are enclosed in [square brackets]. The group assignment of the above four is indicated, and the symbols ‘ \mathfrak{C} ’ and ‘ \mathfrak{G} ’ denote non-expandability and uncertain expandability, resp. The symbol ‘ \mathfrak{X} ’ means \times group

assignment uncertain. No symbol concerning expandability at an operator means it's expandable.¹⁶

[o] Aa \mathfrak{A} One-level expansion with `\expandafter`. (As currently implemented in `l3expan`.)

Ää \mathfrak{A} Two-level expansion with `\expandafter`. Used by me for pre-processing of macros that should be expanded to their content and that content hit once more, e.g.:

```
\def\number_of_page:{\the\c@page}
```

[c] C c \mathfrak{A} The uppercase is just an alias for `c`, i.e., applying $\mathfrak{K} \cdot \mathfrak{H}$ before passing the argument on *without* braces. The lowercase `c` does the same only passes the result on in braces. What could be it useful for? First, for all the \TeX primitives that require a `{\balanced text}`. Then, for the constructs like

```
name 1  $\mathfrak{K}$  name 2 ...
```

Đ đ \mathfrak{A} (Latin letter Eth/eth) Hits the argument with `\the`. In the current implementation of `expl3`, it's almost equivalent to `v` for some `expl3` data types, namely: `_int`, `_dim` and `_skip`.

đ is equivalent to `v` and `Đ` to `VI`. `v` is equivalent `*cđ` in stand-alone contexts or `cđ` as $\langle \pi s \rangle$ of an $\langle \text{FSM} \rangle$ or $\langle \text{BDSM} \rangle$.

However, due to the “Don't rely on implementation” rule, one should *always* use the `v` or `V` specifier to render the value of an `expl3` data carrier.

Đ đ \mathfrak{A} Hits the argument with `\meaning`. (“Was deutet die Name?”)

[x] E e \mathfrak{A} Submit the argument to `\edef`. The lowercase `e` translates to the `\:x` macro of `l3expan/expl3` which in its current implementation “returns” the “result” in braces. The uppercase variant “returns” the “result” without braces and is not present in `expl3`.

Ē ē \mathfrak{A} Submit the argument to, approximately, `\protected@edef` in the sense of $\text{\LaTeX} 2_{\epsilon}$. (We can think of the horizontal bar over the `\edef` “e”'s as a protective shield.)

[p] H h \mathfrak{A} Pick a `#{-delimited` argument and return it without braces (`H`, `p`) or wrapped in braces (`h`).

H h \mathfrak{A} Pick a `#{-delimited` argument and discard it.

H \mathfrak{A} Pick everything until the digit ($\langle \text{FSM} \rangle$ -label $\langle \lambda \rangle$) 1 and “return” without braces, leaving the $\langle \lambda \rangle$ 1 at input. (Used to jump right to a labelled FSM.) (Latin Capital Letter HV, shape modified in my Ubu Stereo font.)

I i $\mathbb{1}$ Identity operation, braced or unbraced with respect to the lettercase.

I i \mathfrak{A} Pick and discard an undelimited argument.

K k \mathfrak{A} `\detokenize` the argument.

Ł ł \mathfrak{A} Pick an argument delimited with the delimiter $\langle \phi \rangle$; if $\langle \phi \rangle$ is not yet declared, i.e., there's no internal macro with a parameter delimited with it to do the job, define it dynamically (and not expandably, in this case).

Ł ł \mathfrak{A} Pick and discard an argument delimited with $\langle \phi \rangle$, possibly declaring $\langle \phi \rangle$ dynamically, as with ‘L’ and ‘l’ above.

[N] [n] $\mathbb{1}$ No pre-processing. Equivalent to `i/I` in the current implementation of `l3expan`; listed separately here in observance of the admonition in “The $\text{\LaTeX} 3$ Interfaces” by The $\text{\LaTeX} 3$ Project [Team?]¹⁷ (henceforth “`L3Interfaces`”): “the implementation should not be relied upon”.

Ń ń \mathfrak{A} Hit the argument, which should be a single character token, with (expandable) in- or decrement by 1 respectively (expandable).

Q q \mathfrak{A} Pick an argument delimited with `\q_stop`.

Q q \mathfrak{A} (Latin Capital/Small Letter Q with Stroke through Descender, `U+A756/U+A757`) Pick and discard an argument delimited with `\q_stop`.

[f] R r \mathfrak{A} Apply `\romannumeral -`0` to the argument. This fully expands the leading token(s) of the argument until an unexpandable token is seen. So, it's called `f` for “full” and *not* called so by myself for “until first unexpandable”. I chose the letters ‘R’/‘r’ to refer clearly to the primitive `\romannumeral`, as this expansion is *not* “full” in principle; and if might be described without it, that would be “`\expandafter quantum satis`”.

S s \mathfrak{A} Hit the argument with `\string`. It's worth underlining that the uppercase version “returns” the result without braces, which for a control sequence means at least two “bare” tokens.

[v] V v \mathfrak{A} (`expl3`: Latin Capital Letter V; me: Cyrillic letter Izhitsa, `U+0474/U+0475`.) Render the value of a data carrier (an `expl3` “variable” or “constant”), given as a control sequence. Related to `đ` and `Đ`, see above.

[v] V v \mathfrak{A} (`expl3`: Latin letter lowercase v; me: Cyrillic letter Izhitsa with Kendima, `U+0476/U+0477`.) Render the value of a data carrier given as a name (first submit the argument to $\mathfrak{K} \cdot \mathfrak{H}$).

Z z \mathfrak{A} Insert the resp. $\langle \text{text} \rangle$ in the stream of this GMS translation (`:::run` macros).

¹⁶ Like, toutes proportions gardées, in Orthodox Jewish districts or state(s), “If something isn't *terefah* by its very nature, in which case there's a warning, then it's *kashrut*.”

¹⁷ as of May 18, 2016.

Since ‘Z’ is considered a symbol of things last or ultimate, and with this operator/s you can do literally anything, “from ‘A’ to ‘Z’”.¹⁸

Dangerous, experimental, liability excluded to the maximum extent permitted by Law.

Don’t use it unless you read and understood its current implementation.

ñ ñ **⌘** Submit the argument to `\the\numexpr`, `\the\dimexpr`, or `\the\glueexpr` respectively, i.e., in **Ĝ ğ** one-level expansion evaluate the argument as `\int_eval:n`, `\dim_eval:n`, and `\skip_eval:n` do in the everyday `expl3`. (Added in July, 2017.)

ř ř **⌘** Applies `\fp_eval:n` to the argument, that is, “floating point” evaluation in the sense of `expl3`. (Added in August, 2017.)

When used as $\langle \text{SALos} \rangle$, the $\langle (\pi^* \varpi^*)^* \rangle$ (pre-ps. and pickers) refer to / are applied to subsequent arguments from the input.

When following a $\langle \varrho \rangle$, the $\langle \pi \rangle$ s refer to / are applied to the resp. $\langle \lambda \rangle$ th argument from the input, counting as explained later.

When following a close brace in a $\langle \text{BDSM} \rangle$, including the outermost, the $\langle \pi \rangle$ s refer to that $\langle \text{BDSM} \rangle$ as if it were a single argument taken from the input, and a $\langle \varpi \rangle$ raises an error.

5.4 The meta-operators, $\langle \lambda \rangle$

The $*$ and $\#$ meta-operators are allowed only in the $\langle \text{SALos} \rangle$, and modify actions of $\langle (\pi^* \varpi^*)^* \rangle$. By returning a pre-processed $\langle \text{text} \rangle$ to input, they allow multiple operations on the same $\langle \text{text} \rangle$ without launching the (more expensive) $\langle \text{FSM} \rangle$ / $\langle \text{gBDSM} \rangle$ machine. They don’t increase the expressive power of the language, as they might be expressed as follows:

- $\langle * \langle \pi \rangle \rangle \equiv \langle \xi \langle \pi \rangle \rangle$.
- $\langle \# \langle \pi \rangle \rangle \equiv \langle \xi \text{ 111} \langle \pi \rangle \rangle$.

The R meta-operator (or rather: interruptor) suspends parsing of $\langle \text{specification} \rangle$, hits whatever is next with `\romannumeral -'0`, i.e., “the f-type expansion” in the `L3Interfaces`, and then hopefully resumes parsing. That allows you to branch the very specification of a given GMS, not only its arguments. Including nesting of GMS’s. Does it increase expressive power? Yes. It brings virtually the whole “mouth” of $\text{T}_{\text{E}}\text{X}$ into the GM-Scenarios, and that’s Turing-complete (cf. an expandable implementation of lambda calculus at ctan.org/pkg/lambda-lists and a more general discussion at tex.stackexchange.com/questions/35039/why-isnt-everything-expandable.)

¹⁸ Also, Zelenka’s *Missæ ultimæ* might be recalled as a mmemo.

And that (the “inter^Ruptions”) let you write code in a more “meta” way. And more obscure, yet shorter.

\times is a shorthand for `\prg_replicate:nn`, which means it requires two pairs of braces to come next, the first containing a $\langle \text{number specification} \rangle$ and the second the things you wish to replicate. This way, instead of

```
\::: ... ↓↓↓↓↓↓↓↓ ...:
```

you can type

```
\::: ... ×8↓ ... :
```

(As you may have noticed, at this point I do rely on the current implementation of `\prg_replicate:nn`, namely, on its expandability.¹⁹)

Let’s now deal with the “render-pointers” $\langle \varrho \rangle$ s, that is, the general permutations.

5.5 The general permutations, or the $\langle \text{FSM} \rangle$ without grouping

The processing of a “general permutation” can be described as two stages: (Stage One) preparation of the “**shelf**” or “substrates’ storage”, or “**craw**”, and then (Stage Two) picking labelled **elements** from the “shelf” and putting on the “**slab**”²⁰ (a permutation consists of or is applied to elements (of some set), not arguments, isn’t it?).

As labels, the “bare” digits and Latin capital letters are used: ‘1’:‘9’, ‘A’:‘P’. This is safe since the $\langle \text{GMS} \rangle$ ’s arguments’ contents are “invisible” to $\text{T}_{\text{E}}\text{X}$ ’s macro argument scanner, thanks to the braces. Two important arguments for this choice are:

- these chars are easy to type in (for the “explicit labels” version), at least with Western input devices;
- no one changes their catcodes (not even me).

The “shelf” is functionally a one-dimensional array (a vector). For each label $\langle \lambda \rangle$, an accessor or “Fetch!” macro “`\F#` $\langle \lambda \rangle$ ” exists that resp. render-pointer translates to. It “gets a copy” of the $\langle \text{text} \rangle$ put next to $\langle \lambda \rangle$ on the “shelf” onto the “slab”, i.e., absorbs that $\langle \text{text} \rangle$, and puts one copy of it on the “slab”, and another copy back on the “shelf”.

Then the $\langle \pi \rangle$ s are applied (if any), and the result is appended to what’s already in the “result container”.

It seems expensive, $O(l_s^2)$, l_s being the length of $\langle \text{specification} \rangle$, and it would probably be more effective to define index-named macros whose contents would be the $\langle \text{FSM} \rangle$ ’s elements. Then access

¹⁹ But why does the `L3Interfaces` indicate expandability of its “functions” if one should not rely on implementation? (That makes me feel confused ☹.)

²⁰ “Let’s go to the lab ’n’see what’s on the slab”, “The Rocky Horror Picture Show”.

to any of them would cost just one `\f ... \r` plus one one-level expansion of it. But practically, for up to 25 (text)s for an (FSM), it works just fine.

But when implemented the way it is now, it stays expandable. Why is that so important? I'm not quite sure. First of all, it's more fun. But also, many \TeX and $\varepsilon\text{-}\TeX$ primitives expand macros in search of \langle (balanced text) \rangle to absorb. Then, if a GMS is expandable, it is possible to write, e.g.,

```
\toks\<number>= \:: \{ { ... } : 1 ... \}
```

and get the scenario to return the \langle (text) \rangle for the `\toks` assignment.

In addition, the prefixes `\global`, `\outer`, and `\protected` expand expandable tokens, so that's possible to define, say, `\f_def:Nn` that computes and sets the proper parameters string out of its #1's signature, and at the same time accepts prefixing with `\global` or `\protected`, and acts accordingly.²¹

So, it seems we are handling a dynamic-length data structure within purely expandable sub- \TeX . Are we really? Not quite. It is dynamic in length *and* expandable only up to the largest number for which “shelf”-preparing and -referring macros were previously defined. For now, this number is 25, as I haven't needed more so far, especially since more than 10 already makes a GMS a tool of “Security by Obscurity” rather than of shortening the code or making it more bug-robust.

5.6 Parsing the braces, or: \langle BDSM \rangle

In the first go, as presented in the previous paper, I perceived the GMS as a DFA (Deterministic Finite Automaton) “with Mysterious Something” that allowed it to handle the braces properly.

That “Mysterious Something” had been implemented as the native \TeX 's argument scanner with additional tricks to roll back the effect of `\string{`, in a sense, and instead put a special token ‘q’ next to the outermost closing brace, translated to the `\“q` macro mentioned above.

Currently my understanding is that those tricks are not necessary, and we can process the \langle specification \rangle char-by-char until a colon ‘:’, which allows to use not only the braces $\{$ ₁ and $\}$ ₂ as the group opening and group closing symbols, but also other characters I'd give this charclass(es), e.g., ‘[’ and ‘]’.

The only thing we need to do is add another argument to the relevant states and transitions, one which bears the nesting level.

²¹ `\f_def:Nn` is actually defined this way in `gme3u8`, while `\expan's \cs_new:Nn` contains a preliminary/auxiliary `\def` any prefix is “earthed” at, or even raises an error.

And what is such an argument, i.e., an integer (Natural) number, that's initially 0, and is increased by 1 with each opening brace, and decreased by 1 with each closing? It's nothing (in a sense) other than a pushdown stack (i.e., a stack with top-only access), with the initial symbol 0 and just one symbol pushed down or popped, 1. Then, we can think of the number representing current nesting depth of braces as the stack storing this many 1's.

In this sense, the GMS automaton is a DPDA, Deterministic Pushdown Automaton.

The (informal) argument that it's a proper DPDA (not a Turing Machine), is that it does only what fig. 2 depicts. In particular, it rejects (raises an error at) the \langle BDSM \rangle braces and \langle subs'n'refs \rangle double-stroke brackets interlacing

```
\:: ... \{ \{ \} \} ; ... :
```

while all three ‘ $\{ \} \{ \} ;$ ’, ‘ $\{ \{ \} \} ;$ ’, and ‘ $\{ \{ \} \} ;$ ’ are accepted.

Then, the depth of the BDSM braces is limited only by the capacity of \TeX (and at the $\text{\!}^{\text{\!}}$ -run it's about memory not the maximum group level, since there are no “groups” (that belong to the semantics), just syntax, i.e., the symbols of the GMS language's alphabet). This means it's stronger than a Deterministic Finite Automaton, since it's capable of recognizing the Dyck language.

The “stack integer” is implemented as a single character interpreted as a number via ‘ $\text{\!}^{\text{\!}}$ ’ (\TeX backquote), and expandably in- or decremented thanks to `\numexpr` and `\Ucharcat`.

I consider it a data type and call it ‘N’, hence the ‘ $\text{\!}^{\text{\!}}$ ’ and ‘ $\text{\!}^{\text{\!}}$ ’ pre-processors using the expandable in- and decrement mentioned above.

With setting the number 0 to be represented by the character ‘0’, the maximum character number available in $\text{\!}^{\text{\!}}\TeX$ is 1114063. So, far far more than the maximum grouping level handled by \TeX , so it can parse (accept and translate into the rearrangement macros) far deeper nesting than \TeX can execute.

If we meet an opening brace when 0 is at the top of the stack, we move to the BDSM sub-automaton, and do basically the same things as we do in the FSM states, but pushing or popping 1's down or from the stack.

Once 0 is seen again, i.e., once all the 1's are popped from the stack, we know that the outermost opening brace (of this BDSM) has been matched, so we possibly apply some \langle π \rangle s to it, and then yield, and move back to the state of general FSM.

The case of “subs’n’refs” is conceptually analogous, although the current implementation is more tricky than honestly “automaton-ic”.

5.7 The $\langle \text{subs}'\text{n}'\text{refs} \rangle$

Whenever the automaton, being already in some of the $\langle \text{FSM} \rangle$ states, meets an opening ‘ \llbracket ’ bracket, it conceptually pushes yet another symbol down the stack, and moves us into the “subs’n’refs” sub-automaton(s).

This new symbol may be considered the initial symbol of the $\langle \text{subs}'\text{n}'\text{refs} \rangle$ sub-stack. Then, again, whenever an opening brace is met, 1 is pushed down, and popped at meeting a closing brace. Seeing that “yet another symbol” on top of the stack after a closing brace was popped, says that that brace is outermost with respect to this $\langle \text{subs}'\text{n}'\text{refs} \rangle$.

As you see, the matter discussed gets a bit complex, and hence I chose i as that “yet another symbol”. ☺

The $\langle \text{subs}'\text{n}'\text{refs} \rangle$ cannot be nested, and that’s why only one i is allowed on the stack, and why there’s no ‘ \llbracket ’- or ‘ \rrbracket ’-labelled edge between two states of the $\langle \text{subs}'\text{n}'\text{refs} \rangle$ sub-automaton.

The $\langle \text{subs}'\text{n}'\text{refs} \rangle$ sub-automaton might also be considered two distinct sub-automata, one reached from the general $\langle \text{FSM} \rangle$ states and returning to them, and the other from the $\langle \text{BDSM} \rangle$, and going back to that $\langle \text{BDSM} \rangle$.

The difference lies on the top of the stack (literally) before pushing down the symbol i , and after popping it: if the $\langle \text{subs}'\text{n}'\text{refs} \rangle$ sub-automaton is reached from a general $\langle \text{FSM} \rangle$, 0 is on top of the stack, and if the $\langle \text{subs}'\text{n}'\text{refs} \rangle$ has been reached from within a $\langle \text{BDSM} \rangle$ state, then 1 is on top of the stack.

The diagram is complex already; so as not to make it messy, those two sub-automata have been merged, in a sense. Instead of drawing the two independently, only the “foreign” edges are drawn as distinct, while the states and the “domestic” edges are present once, and the two sub-automata are topologically homeomorphic.²²

5.7.1 The replacements, ‘=:’

Consider the part ‘ $\llbracket \text{Z}\ddot{A}\text{=2 } \rrbracket$ ’ of the example in section 4. As has already been mentioned, this part of $\langle \text{specification} \rangle$, and more precisely, of its $\langle \text{FSM} \rangle$, translates into macros that replace the original element #2 with its “double $\mathcal{F}\mathcal{S}$ ” expansion, prepared to be returned without braces.

²² The macros for the two are separate, though, because within $\langle \text{BDSM} \rangle$ the local “shelf” and “slab” are prepared slightly differently, and so the parameter delimiters differ.

So far, there is no “symbol i ” used in the implementation. The transition to the $\langle \text{subs}'\text{n}'\text{refs} \rangle$ part of the automaton is implemented with “memorizing” current nesting level, which would be the current length of the stack in the terms of this paper, as a(nother) numchar, and confronting it with the $\{\}$ -nesting level at \llbracket , i.e., at the end of $\langle \text{subs}'\text{n}'\text{refs} \rangle$.

This is done in a way similar to the “usual” processing of the $\langle \text{FSM} \rangle$ elements, i.e., by taking a copy of the element into the “operation table” or “slab”, applying the $\langle \pi \rangle$ s, and, here comes the difference, putting the result not in the “result container”, but back in the “craw” or “shelf”, effectively replacing it at the given label.²³

The idea of this sub-automaton stemmed exactly from the craving for “as few repetitions as possible”, namely, in situations where I’d apply the same sequence of pre-processors $\langle \pi^* \rangle$ to the same element of an $\langle \text{FSM} \rangle$ more than once.

In the given example, it’s a bit of an overkill, as the element #2 is used only in two copies. But, even twice might be too many, if we think of the points to remember to change something, say, a single ‘ $\mathcal{F}\mathcal{S}$ ’ to “double- $\mathcal{F}\mathcal{S}$ ”.

In this example, the render-pointer $\langle \varrho \rangle$ at the left side of ‘=:’ corresponds with the label $\langle \lambda \rangle$.

But it’s not a *sine qua non*. The mechanism is general enough (at no additional cost) to process any correct $\langle \text{FSM} \rangle$ put on the left (including $\langle \text{BDSM} \rangle$), and make the replacement of it at the label typed on the right side of the ‘=:’ “assignment” symbol.

And, I used quotation marks for the word “assignment”, because the replacement operator ‘=:’ (binary infix) is expandable.

5.7.2 “The arguments from beyond”, ‘ λ ’

Also expandable is the “ambiguary” prefix operation $\lambda \langle \varpi? \rangle \langle \lambda \rangle$, that gets the next argument from the “input”, i.e., from beyond all the “slabs”, “shells”, “fridges” and “containers” of the $\langle \text{FSM} \rangle$ and of the entire $\langle \text{specification} \rangle$, and puts it as the element # $\langle \lambda \rangle$, thus replacing whatever was there before.

It also works if there was nothing at that label earlier, i.e., if $\langle \lambda \rangle$ was until now immediately followed by another label, or by a delimiter of the “shelf”. (Garbage warning as above.)

Again, it’s done with macros with a parameter delimited by the respective label(s).

²³ With the assumption that such replacements are “not too many”, and in order to allow “empty labels”, the old version of an element remains “at the back”, and is discarded only at the very end. (Garbage warning.)

Since this feature was implemented only last week, only an undelimited argument picker is handled at the moment. But other pickers are *in peccatoris*, as described in the Grammar, fig. 3.

Also, in this (early) version of this functionality, presence of the label $\langle\lambda\rangle$ already in the slab is assumed and required.

Again, this feature emerged from a need (or will) to be able to write a GMS consisting mostly of an FSM, and make an anonymous function of it,²⁴ and yet declare the FSM with labels, as it's more readable this way.

The example which follows is based on a quasi-iterator used in some *really real* T_EX program. First of all (conceptually), there is a list of control sequences that should be at some point defined (or not, that's why it's not done on the “ground level” of code). Then, if they are defined (and only if), they should be initialized, as they're defined as variables (of various types). Then, if they joined in the action, it has to be known how to set them (s), and also, how to reset (rs). That makes a 4-tuple of things for each of those control sequences, with the (s) function used also in (rs), only with the special value (rsv), and that (rsv) is specified as the 4th element of each tuple.

Each of those control sequences requires specific “methods” of its own, and initialization is performed once if at all, so instead of defining macros, I used a GMS to allow the contents of the braces (*), i.e., that anonymous 1-argument function, to be put by the loop, and given control sequence (*b) as the argument.

```
...
{\initializations of:} \g__\aux'1_str
...
{ % (*)
  \:: \u [_{5}] 15 253: } % declare & init. box to
    empty \hbox
{ \:: \u [_{5}] 254: } % reset the box to void
{ % (**
  1 \box_new:N
  2 {\f_global::\f_setbox::}
  3 {=\hbox{}}
  4 \c_f_void_box
  5
  \u
}
\c__\aux'1_box % (*b)
...
```

²⁴ Although at the time of writing this feature I was not aware it was to be an anonymous function. Not only am I not a computer scientist, but also not a graduate of a formal course of computer programming ☹.

Thus, when the code (*) and (**) are put (without braces), and followed by the c.s. (*b), the first thing done, written down as ‘[_{5}]’, is absorbing (*b) to the #5.

And then the initialization is performed, i.e., the result of the above GMS is:

```
\box_new:N \c__\aux'1_box
\f_global:: \f_setbox:: \c__\aux'1_box =
\hbox {}
```

The optional picker $\langle\varpi\rangle$, if present, makes the machine pick not the next $\langle\text{text}\rangle$ undelimited, but delimited as specified with the $\langle\varpi\rangle$ (for symmetry, specifying ‘i’ or ‘I’ is also allowed).

Note, by the way, that any GMS might be considered an anonymous function (unless a “predef” of it is made), and also an explicit sequence of l3expan ‘\::\u’'s, but not the ‘inline’ (1st) arguments of the expl3 $\langle\text{type}\rangle_map_inline:n[n|N]$ iterators, as the latter are internally assigned a (one-parameter) macro in the usual way, only hidden.

5.7.3 Snapshots and references, ‘*’

Described last, as unexpandable by their nature, are the $\ast\langle\lambda\rangle$ operations, ‘snapshot the element $\#(\lambda)$ and make a reference to it’.

The idea is very simple: allow referencing the permutation elements within the reorganized code, so as not to be forced to divide everything into the “before the element/argument part” and “after ~ part”.

So, putting ‘[\>... *7 >...]’ within an $\langle\text{FSM}\rangle$, makes a “snapshot” of the element #7 as it is at the point of ::-run of the (translation of) this operator, available as the contents of an expandable macro, or, speaking in expl3, a $_tl$ variable, that may be rendered via ‘7*7’ to get that contents wrapped in \backslashunexpanded , or via ‘7#7’, for not protected.

Nesting one GMS within another is allowed, and to avoid messing up the snapshots and references in such a case, a record of its level (depth) is kept, and updated expandably as long as purely expandable $\langle\text{subs'n'refs}\rangle$ are used, which is checked by the automaton in the ::-run.

For now, only ‘*’ and ‘#’ $\langle\text{subs'n'refs}\rangle$ operators “destroy expandability”, the latter being a superposition of expandable ‘=’ and ‘*’, as follows.

The binary infix operator ‘#’, used as

```
‘[ \>... \langle\text{FSM}\rangle =\# \langle\lambda\rangle >... ]’,
```

first replaces the element $\langle\lambda\rangle$ with the result of the $\langle\text{FSM}\rangle$ from the left side, and then also makes a snapshot of it, referable as described above, via $7\ast\langle\lambda\rangle$ for “ \backslashunexpanded ’ed”, or $7\#\langle\lambda\rangle$, for “bare”.

To be honest, I’ve used this mechanism only a few times so far, as it denies expandability by its very nature. Each of those few uses is large, of more than 10 elements, therefore I’ll show just fragments of the simplest one.

```
\:: \omega \[3=#9 % (curr. contents of) #3 is put on #9,
      and made *9]
\c{=7]
\z{#} % the c.s. built above is now defined
... :
...
2 { c_\#1 transition'from\#2 }via\#9
   \result_clist }
3 { #3 }
9
...
\omega
```

Note how the “snap’n’ref” is used: the primary goal of this feature is to allow putting placeholders in the subject code, and have them replaced with the respective element. Somewhere in the middle of the text, and possibly, also nested.

So, there are the placeholders in the middle of $\langle \text{text} \rangle$ of an element. And, they are valid only within their respective $\langle \text{FSM} \rangle$, thanks to the record of GMS nesting mentioned above, and checking it.

And outside their own FSM, those placeholders/references issue an error. So, to make any use of them, one has to apply some kind of full expansion to the elements that contain them.

And here it is: the element $\#2$ is a long cfname built with $\#9$, and the ‘ \c ’ operator, i.e., $\c\{ \cdot \}$, performs such expansion. The resulting c.s. is put instead of the original $\{ \langle \text{text} \rangle \}$.

Note BTW, that the c.s. raised from $\#2$ is not put instead of it(self), but at $\#7$. That’s because ‘ $\#3$ ’ being the contents of the dereference $\#9$, is “alive”, and “then” expands to something other than “now”.

6 Rough budgeting, a.k.a. cost estimation

If we take the “interuptions” into account, the estimation is simple: anything is possible, including arbitrary elongation of the resulting “interuption”-less $\langle \text{specification} \rangle$. That elongation might come from, e.g., $\langle N \rangle_{\{i\}}$, where N is a decimal or hex. representation of a positive integer.

Then the resulting length of $\langle \text{specification} \rangle$ becomes $O(B^{l_N})$, where B is the base of the representation of N used, and l_N its length in this representation.

So, in the full-featured version, potentially “exponentially explosive”, but no more so than any loop accepting numerical limits in power-position notation.

What about GM-Scenarios with “interuptions” put aside, i.e., the proper DPDA of it?

What basic operations should we consider here? If we think of each operator as a (constant) sequence of macros, as the definition of the operators does not change in the runtime, and getting the next $\langle \text{text} \rangle$ from input as just one step (unit cost), then the time cost of a GMS that doesn’t involve $\langle \text{FSM} \rangle$ is, putting l_s as the length of $\langle \text{specification} \rangle$, $O(l_s)$.

Then, including $\langle \text{FSM} \rangle$ in our consideration, we see that one character (plus a constant number of its “context”), may result in apparently arbitrarily large numbers of $\langle \text{text} \rangle$ s to take from input.

```
\:: ... #|7|; ... :
```

But it cannot happen, as the alphabet (not Unicode, not the charset handled by the \TeX engine used, but the theoretical alphabet of the language considered) is finite,²⁵ so there is an upper bound for the numbers expressible with the $\langle \lambda \rangle$ ’s, so, as we’re in the realm of Naturals, there exists the maximum of those numbers. Let’s denote it by M . Then, including $\langle \text{FSM} \rangle$ ’s in this “budgeting”, we get an estimation

$$O(l_s) + O(M \cdot l_s) = O(l_s),$$

still within linear time.

But, is the assumption of the unit cost of getting new $\langle \text{text} \rangle$ reasonable, no matter how far we have to jump over the tail of $\langle \text{specification} \rangle$, and over the partial result, i.e., the storage of the $\langle \text{text} \rangle$ s already processed?

It seems not. As the $\langle \text{specification} \rangle$ is executed, the partial result “pessimistically” grows at the same rate as the $\langle \text{specification} \rangle$ shortens, and we have to jump over both of them in order to get the next $\langle \text{text} \rangle$ for pre-processing. So, if we consider “jump over one $\langle \text{text} \rangle$ ” a unit cost, the estimation becomes $O(l_s)^2$, it seems. Still, polynomial time, not so bad (it seems).

The space cost appears even nicer, as no $\langle \text{text} \rangle$ at the input can be copied more than l_s times, and there can be one $\langle \text{FSM} \rangle$ “shelf” at a time, so no more than M additional $\langle \text{text} \rangle$ s at a time. That allows the following estimation of the space cost SC :

$$SC \leq 2l_s + M,$$

with given alphabet and fixed set of $\langle \rho \rangle$ ’s and $\langle \lambda \rangle$ ’s, M is constant, and so we get $O(2l_s + M)$, and that’s just $O(l_s)$. Just great, it seems.

²⁵ ...and the language is too weak to express a description like “The largest number expressible with less than 70 characters”!

However, consider

```
\:: ξ{111}. : a % (s1)
\:: ξ{111}. ξ{111}. : a % (s2)
\:: ξ{111}. ξ{111}. ξ{111}. : a % (s3)
...
```

It looks we've found an "exponential explosion", as each next ' $\xi\{111}\cdot$ ' replicates the result of the previous one three times, and n times in general, n being the number of ' $\underline{1}$'s within braces.

Even though the length of \langle specification \rangle grows by $n+K$, $K=4$ being fixed, so it's not exactly $3^{l_s/n}$ but more like $3^{l_s/(n+K)}$ — still exponential. Both in time and space, as it's the partial result that grows so fast.

But this is a weird and theoretical example.²⁶ In practice, let me repeat, it works just fine, as users intuitively avoid the ' ξ ' "destination", and don't do such silly things as mere replication of one \langle text \rangle ; the previous estimation, of $O(l_s^2)$ time and $O(l_s)$ space, seems to hold in all reasonable cases.

7 Friendly critiques at TUG@BachoTeX 2017

The respective section in the GMOA paper was called "Real-life uses of GMOA". I'm not sure whether such a machinery, which with its full features is rather an esoteric language than a friendly tool for reasonable users, might be much used in "real life". I'm afraid that, by mere using of it, the respective part of "life" would be made "un-real". At least, in the sense of total obscurity for anyone else but me.

That's the most important thing my colleagues, or better say: friends, pointed out after the presentation of GMS at TUG@BachoTeX 2017.

In more detail, it's because:

1. using "distant" and PUA Unicodes does not help at all, since most users are still in pure ASCII, at least concerning the control layer, like control sequences and special characters;
2. it's too complex and obscure, and for most people it's simply easier and clearer to write the same code twice, or more times, than to try to decipher from the one-character instructions how the pieces should be repeated, and how modified;
3. " \backslash expandafter does strange and complex things, therefore it should have a long and strange name, and not single-character!", and similar argument about other primitives;
4. it doesn't seem useful, "... and I understood, it doesn't have to be: because you don't develop

it to be useful, you develop it as your artistic expression."

@ rem. 1, I totally agree. Also, not even yet shown, non-English control sequences, such as the eschatological-appearing ' $\backslash_::_ \text{apokatastathēi}$ ' FSM' ('apokatastathēi FSM', reminding one of the Neo-Platonic or Gnostic visions of Apokatastasis at the End of Time), changed to ' $\backslash_::_ \text{S}^{\text{R}}\text{resume}$ ' FSM' herein, along with all other Greek or Latin ones.

And, to ease typing of \langle specification \rangle s to those few who may not be completely familiar with things like Opening Lenticular Bracket Ordinal Number Omega, PUA+E9EA ' Ω ', I provide a pure ASCII and HTML-like "input method": in the most general version, one may type ' $\&U+\langle$ hex \rangle ', and then \backslash Ucharcat · Ω 12 Ω will be applied to \langle hex \rangle , i.e., the resp. char12 rendered, as if it were there in the first place.

Then, there are some "ASCII approximations" of the symbols, like ' $\&w[;]$ ' and ' $\&w];$ ' for ' Ω ' and ' ω ', or ' $\&vY;$ ' for the Capital Izhitsa with Kendima, ' Ÿ ', or ' $\&\{the\};$ ' for ' δ '.

The ' $\&...;$ ' "entities" are (more or less) "inter^R-uptions", and can be used anywhere. They are translated internally to the respective original symbols, so using a native Unicode engine remains obligatory.

For the pointer-renderers of \langle FSM \rangle elements, i.e., the \langle q \rangle symbols of the formal grammar, the HTML-like forms: ' $\&_1;$ '... ' $\&_9;$ ', ' $\&_A;$ '... ' $\&_P;$ ' for the "lowercase" ' q '... ' p ', and ' $\&^1;$ '... ' $\&^9;$ ', ' $\&^A;$ '... ' $\&^P;$ ' for "uppercase" ' Q '... ' P '.

To avoid possible confusion of ' $\&_1;$ ' and ' q ', think of ASCII Underscore as the sign of "generic sub-ness", as in standard TeX for subscripts, so may it be also for lowercase, and of the graphical element ' $\underline{\underline{\quad}}$ ' "double underline" as the proofreading sign "make this uppercase".

But in case of the \langle q \rangle "render-pointers", the HTML-like notation is not necessary, i.e., the ' $\&$ ' and ' $;$ ' might be omitted, and ' q ' is also fine, as shown both in the automaton graph in fig. 2, and in the formal grammar in fig. 3.

@ rem. 2, I admit: yes, GMS are complex, and maybe even mad, and might easily become obscure. But, and this is one of its goals, they allow reducing the number of repetitions of at least some parts of code to 1, and that in turn makes things fixable at just one point. For instance, having defined two macros that differ only with the printed text, and put the same skips before it, if I wish later to change the amounts, then, having used $\backslash::$ properly, I change "both" of them only once, namely, at the label ' A ':

²⁶ Remember Murphy's Law?

```

\GMS &w[; ^1^2 ^7 {^A^3^9}
      ^1^4 ^7^8 {^A^8 _5 ^9} :
1 \def
2 \macro1 3 {indigo}
4 \macro2 5 {indigenous}
7 {#1#2#3} 8 {#4} 9 {#2}
A {\hskip 17pt\relax }% later changeable
&w];

```

As with probably all things in this world, it’s a matter of balance. Here, between only the primitives at one extreme, and just one active character expanding to the entire document, at the other. Oh, not even one: an empty file, that expands to the entire document thanks to `\everyeof`.

For me, that balance seems to be in the `l3expan` iterators and in not too long, but on the other hand, nontrivial, GM-Scenarios.

@ rem. 4, I also admit that the main reason I’m doing this is fun, or, to put it in a less hedonistic way, intense intellectual satisfaction.

But, again, that doesn’t exclude usefulness *per se*, and striving to make GMS meet not only my requirements, but also those of other people, might be as much fun, and as much art.

@ rem. 3, let me just say:

as in: $(f(x) \cdot g(x))' = f'(x)g(x) + f(x)g'(x)$. ☺

8 Final remarks

8.1 “Thank Heavens, it’s not the Premium Class”

At the end, let’s recap the question posed in the title. We already know GMS’s are a complete madness. But — are they Turing-complete?

The answer has already been given, and this answer is: No.

Ignoring the “interRuptions”, the²⁷ automaton is deterministic pushdown, and the GMS language appears to be context-free.

So, it’s not a Premium Class machine, i.e., Turing, and that’s a relief in a sense, as it shows I did not “rewrite `TEX` in `TEX`” [yet].

On the other hand, the GM-Scenarios allow for making parts of code noticeably shorter, clearer, and less repetitive, and this way more readable and bug-robust. Provided that they (GMSs) are kept at bay on their own, i.e., not too long, and not too complex.

²⁷ Actually, *an* automaton, since there exist many automata equivalent to the one just presented, in the sense of recognizing exactly the same language.

8.2 The end, or ἔσχατον

When I think of all those symbols, the automaton, its states and transitions, adding “the arguments from beyond”, the correspondence between it and the formal language of GMS, the most “finale-al” finale I know of, the eschatological and apokatasthetic “Chorus Mysticus” in the cosmic Mahler *Eighth Symphony* comes in handy. ☺

Alles Vergängliche	All things under Transition
Ist nur ein Gleichnis;	are just a Symbol;
Das Unzulängliche,	What <code>l3expan</code> couldn’t express,
Hier wird’s Ereignis;	here is performed;
Das Unbeschreibliche,	What could not be described,
Hier ist es getan;	here is just done;
Das Ewigweibliche	les Femmes Puissantes,
Zieht uns hinan.	protect and bring us beyond.
	— Goethe, “ <i>Faustus</i> ” ²⁸ & Mahler, the <i>Eighth Symphony</i>

◇ Grzegorz Murzynowski
 PARCAT.eu
 g.murzynowski (at) parcat dot eu
 natror.croolik.sryc (at) gmail dot com

²⁸ English translation mine, adapted and adjusted for the needs of this paper.

Typesetting bibliographies compliant with the ISO 690 standard in L^AT_EX

Dávid Lupták

Abstract

The preparation of bibliographic references and citations compliant with the international standard ISO 690 is required by many institutes worldwide. However, the typesetting of bibliographies conforming to the respective standard is not yet supported in the L^AT_EX document preparation system. The `blatex-iso690` package has been revised and improved to fully meet the requirements of the international standard and thus greatly simplifies the typesetting of bibliographies for all kinds of information resources.

1 Introduction

Writing an article, paper or any other kind of work requires incorporating other resources which need to be referenced and cited properly. The preparation of bibliographic references and citations is mainly required to comply with the international standard ISO 690 in Czech academia (Kratochvíl et al., 2011). This article briefly introduces the standard ISO 690 and then describes various existing software implementations that incorporate the standard. It then details the typesetting of bibliographies in the L^AT_EX preparation system and finally describes the package `blatex-iso690`, the first complete implementation in L^AT_EX that is compliant with the latest version of the standard ISO 690.

2 International standard ISO 690

The preparation of bibliographic references and citations was done in accordance with the international standard ISO 690:1987 (*ISO 690*, 1987) for printed resources and ISO 690-2:1997 (*ISO 690-2*, 1997) for electronic information resources. These two versions of the standard were unified and replaced by a new version ISO 690:2010 (*ISO 690*, 2010) in 2010. On national levels, translations of such standards are provided by offices for standards (ISO members) (International Organization for Standardization, 2015), with status at least equal to that of the original standard. Examples of such translations are Czech ČSN ISO 690:2011 (*ČSN ISO 690*, 2011), Slovak STN ISO 690:2012 (*STN ISO 690*, 2012) and German DIN ISO 690:2013 (*DIN ISO 690*, 2013).

2.1 Terminology

There are two key terms regarding the standard (*ISO 690*, 2010) which need to be explicitly defined for clear understanding of this paper. They are:

citation an indication within the text or other form of content of a relevant reference;

reference data describing a resource or part thereof, sufficiently precise and detailed to identify it and to enable it to be located.

2.2 Consistency principle

The international standard ISO 690 does not prescribe a particular style of reference or citation. The examples used in the standard are not prescriptive as to style and punctuation. These facts embrace two findings:

1. the separation of form and content is preserved,
2. the standard cannot be considered as a citation style (Hála, 2013).

At the same time, it is recommended that a uniform style, format and punctuation scheme be used for all references in a document, regardless of the particular style being used. It is up to the creator of the references to meet this requirement, drawing on examples shown in the standard itself, in various national interpretations, or in typical typesetting of bibliographies.

3 Typesetting of bibliographies in L^AT_EX

The L^AT_EX document preparation system provides three possibilities for typesetting bibliographies (Talbot, 2013). The first approach is to use L^AT_EX itself to generate the bibliographies, while the other two adhere to the principle of separation of form and content and benefit from creating an external database of bibliographic data and using an application to generate the output.

3.1 Standard L^AT_EX

The `thebibliography` environment for references and the `\cite` command family for citations are available in L^AT_EX. Each single reference is then mentioned as `\bibitem` with its unique identifier in the `thebibliography` environment.

```
\documentclass{...}
\begin{document}
... \cite{label01} ...
\begin{thebibliography}{\widest label}
\bibitem{label01}
  Author. \emph{Title: subtitle}. ...
...
\end{thebibliography}
\end{document}
```

This snippet of code shows the basic syntax of this approach and reveals how impractical it is for a large number of citations (Talbot, 2012). The main drawbacks are:

1. all entries listed in the `thebibliography` environment are typeset, regardless of whether they are cited,
2. every bibliography entry has to be entered and formatted manually for every desired bibliography style,
3. bibliography references are not sorted, but output in the order in which they are listed in the `thebibliography` environment.

ISO 690 does not prescribe any guidelines for dealing with the first disadvantage, but such results do not follow the general recommendations for bibliographies (Talbot, 2013). Regarding the second limitation, it is very difficult to ensure the consistency of references; regarding the third drawback, it is impossible to output references in the correct order for any method of citation introduced in the standard.

What is missing from this approach is reusability of the bibliographic entries and scalability of the list. On the other hand, one of its great strengths is relatively fast document compilation, as it only needs to be compiled twice using the \TeX engine.

3.2 \BIB\TeX

The preferred method of generating a bibliography is to create an external bibliography database (see section 3.4) and use an application to generate the output (Talbot, 2013). Such applications can deal with typesetting references in the correct order, solving the third issue. Also, based on the selected bibliography style, solving the second issue, they generate `thebibliography` environment which can then be input into the document. One typical representative of this method is \BIB\TeX , which adheres to the principle of separating the form and content.

The `\bibliographystyle` command is used to define the desired bibliography formatting style; the `\bibliography` command specifies an external bibliography database to use and also the location where the list of references is to be printed. The `\cite` family commands are used to create citations within the document text pointing to the desired references. It is also possible to use the `\nocite` command to add the bibliography entry to the list of references without printing a citation within the text, addressing the first problem.

```
\documentclass{...}
\bibliographystyle{(formatting style)}
\begin{document}
... \cite[(additional info)]{list of labels} ...
\bibliography{(list of database files)}
\end{document}
```

While this brief introduction to \BIB\TeX seems promising, it conceals a raft of problems, not least of which

is that development of the \BIB\TeX program is stagnant (Patashnik, 1994; Patashnik, 2003). The main disadvantages and limitations are as follows:

1. input encoding problems (“ \BIB\TeX ”, 2010) (although an alternative solution is available),¹
2. designing your own \BIB\TeX styles is rather difficult (Patashnik, 1988) (although an alternative solution for making \BIB\TeX styles is available),²
3. a shortage of citation customizations (Shell et al., 2007) (although more flexible solutions are available),³
4. absence of contemporary fields widely used nowadays, e.g. the `url` field (although an alternative solution is available),⁴
5. lack of translations and multilingual bibliographies (Harders, 2002) (although an alternative solution is available).⁵

To typeset your document properly, it is necessary to compile your document at least three times using the \TeX engine and at least once more with the \BIB\TeX program. The overall procedure to be applied (Markey, 2009) is as follows:

\LaTeX (\BIB\TeX \LaTeX)⁺ \LaTeX

Generating a bibliography using \BIB\TeX in comparison with the plain \LaTeX introduces more complexity, but it does successfully mitigate most of the aforementioned limitations.

3.3 \BIB\LaTeX

Another option for generating a bibliography via an external database and an application for compiling it is the \BIB\LaTeX package of \LaTeX . This package is a complete reimplement of the bibliographic facilities provided by \LaTeX , usually referred to as a successor of an ancient \BIB\TeX package (“ \BIB\TeX ”, 2010; Hufflen, 2011). Formatting the bibliography is entirely controlled by \TeX macros, while processing a bibliography database file (see also section 3.4) can use the new `biber` backend program (Lehman et al., 2016).

The usage of \BIB\LaTeX differs slightly from traditional \BIB\TeX since it provides more advanced bibliographic facilities for use with \LaTeX . From the user’s perspective, a different syntax is noticeable. Formatting styles are specified as a load-time package option in the optional argument to `\usepackage`.

¹ <https://www.ctan.org/pkg/bibtex8bit>

² <https://www.ctan.org/pkg/custom-bib>

³ <https://www.ctan.org/pkg/natbib>, <https://www.ctan.org/pkg/cite>

⁴ <https://www.ctan.org/pkg/natbib>, <https://www.ctan.org/pkg/babelbib>

⁵ <https://www.ctan.org/pkg/babelbib>

Bibliography database files are specified in the document preamble with the `\addbibresource` command using the full name of the file (including `.bib` extension). The list of references is generated with the `\printbibliography` command; it is output at the position of this command in the document. To create citations within a text of a document, the `\cite` command and its variants are used. The basic structure is as follows:

```
\documentclass{...}
\usepackage[...]{biblatex}
\addbibresource{database01.bib}
\addbibresource{database02.bib}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
```

BIB_ATEX successfully overcomes many of the limitations found in BIB_TEX, the most important of which are (“BIB_ATEX”, 2016):

1. full Unicode support,
2. highly customizable sorting and bibliography labels,
3. `polyglossia` and `babel` support for automatic language switching for bibliographic entries and citations,
4. more entry types and fields,
5. ease of designing new bibliography and citation styles.

This list could be extended to cover more of the rich functionality provided by the BIB_ATEX package (Lehman et al., 2016). There are very few drawbacks to this package, a notable exception being the incompatibility of BIB_ATEX auxiliary files when submitting to a journal (“Biblatex: submitting to a journal”, 2011).

Document compiling is analogous to the BIB_TEX approach. First, a document is compiled by `TEX` engine, followed by running `biber` on a generated auxiliary `.bcf` file, and then compiled by the `TEX` engine once again. Thus, the BIB_ATEX schema for compiling a document is as follows:

```
latex document[.tex]
biber document[.bcf]
latex document[.tex]
```

The file extensions are optional.

3.4 Bibliography database (.bib file)

For the sake of completeness, it is necessary to introduce the bibliography database `.bib` file as well. This file contains bibliography entries: each entry has a specific type (the word after `@`), a unique label and

a number of tags (key–value pairs) defining resource data. The general schema of an entry looks like the following (“BibTeX Format Description”, 2006):

```
@<entry type>{<label>,
  <field> = {<value>},
  ...
  <field> = {<value>},
}
```

All of the entry types supported by BIB_TEX can be used directly, or via an alias also supplied with the BIB_ATEX package. BIB_ATEX introduces more types in addition to the traditional ones, with the possibility of defining completely new ones.

The same situation applies to entry fields. The BIB_ATEX package provides backward compatibility with all of the BIB_TEX fields and adds extra ones. In addition to regular fields, there are so-called special fields which can contain additional settings related to an entry, e.g. to specify the language on a per-entry basis for multilingual bibliographies.

3.5 Summary

The basic functionality of L_ATEX for generating a bibliography can be appropriately used for a small number of citations in a document. However, in the case of a large number of citations, it is best to use an external bibliographic application. This approach adheres to the principle of separating the form and content, which results in high scalability and reusability of bibliography entries and makes working with references more flexible and efficient.

Besides BIB_TEX—the traditional representative of this method—many other applications based on it are in existence. However, all of them inherit the limitations of BIB_TEX. This is mainly the case with the formatting styles used, although some applications work towards replacing the BST (BIB_TEX STyle) language with more modern languages—mostly XML (Hufflen, 2011; Hufflen, 2008).

It emerges from the large variety of options for typesetting a bibliography in L_ATEX (Talbot, 2013; Mittelbach et al., 2004) that the best choice nowadays is the BIB_ATEX package with its backend application `biber` (Hufflen, 2011; Kime et al., 2016).

4 ISO 690 implementations

This section introduces existing software products, tools and services which incorporate the ISO 690 standard. The first two mentioned here are designed to be used with the L_ATEX document preparation system; the CSL language is covered thanks to its newfound popularity and the `OPmac-bib` package because it is a rare example among all available packages which delivers full support for this particular standard. A

more comprehensive overview of the available solutions can be found in the author's bachelor's thesis—written in Slovak (Lupták, 2016).

4.1 czechiso

For the Czech versions of the standard—ČSN ISO 690:1996 (*ČSN ISO 690*, 1996) and ČSN ISO 690-2:2000 (*ČSN ISO 690-2*, 2000)—there is an unofficial formatting style, `czechiso`, created by David Martinek in 2006 (Martinek, 2006). This implementation does not meet the requirements of the standard precisely, as it lacks some of the required fields for bibliographic entries. Many of the functions responsible for printing out a reference correctly are in need of rewriting to fully conform to the standard.

4.2 biblatex-iso690

The first version of the bibliography and citation style for `BIBLaTeX` conforming to the standard ISO 690 dates back to 2011. This implementation was based on the previous versions of the standards (*ČSN ISO 690*, 1996; *ČSN ISO 690-2*, 2000) and on its Czech interpretation (Bratková, 2008). The package was created by Michal Hoftich (Hoftich, 2011). As with `czechiso`, this solution did not precisely adhere to the standards. Many issues related to the functionality of the package as well as the usage of this style were reported on the homepage of the project. Thanks to completely revamping the style in 2016, the package fully meets the requirements of the standard at its current stage of development (see also subsection 5).

4.3 The CSL language

The Citation Style Language (CSL) is an open XML based language for working with bibliographies. It became popular with the release of the Zotero reference manager in 2006 (Fenner, 2010).

The main advantage of this language is its XML syntax, closely followed by its popularity, open source initiative and the versatility of CSL (Ansorge et al., 2013). Another undisputed benefit is its almost universal application, as testified to by the extensive list of products using CSL styles that appears on the official web page of the CSL project (Zelle, 2016a). The best known are Zotero, Papers and Mendeley.

The CSL style repository has over 8 000 styles, including 15 styles for the ISO 690 standard. These styles differ in their localization and methods of citation, hence the vast number of styles for just one standard. All of them contain minor deviations from the standard ISO 690. CSL is not, however, without its limitations (Zelle, 2016b):

- limited support for customizing the label format,

- limited support for legal styles (Multilingual Zotero can be used as an alternative),
- limited support for citing items in multiple languages within a single document (Multilingual Zotero can be used as an alternative),
- limited support for entering date ranges into the date field (no entry is generated).

It should be added that `BIBLaTeX` does not suffer from these limitations (Lehman et al., 2016).

4.4 OPmac-bib

The `OPmac` package defines additional macros on top of plain `TeX`, providing functionality similar to core `LaTeX`. The additional package, `OPmac-bib`, comes with it and is available for bibliography functionality. No external program for generating a bibliography is needed, as everything is handled by `TeX` macros and the `librarian.tex` package created by Paul Isambert. The `OPmac` package was created by Petr Olšák and has been shipping with the `csplain` package since 2015. More details about the `OPmac` package can be found in another article (Olšák, 2016).

`OPmac-bib` can process all of the traditional types and fields of `BIBTeX` and furthermore, it introduces new fields which are commonly needed when working with bibliographies nowadays. These fields are, for example, `url`, `doi` or `lang`, which eliminate the need of using a `note` field for providing such data. Hence it is possible to output this data in the correct order in accordance with the standard.

While `BIBTeX` lacks many needed types and fields, `OPmac-bib` has improved the situation considerably. But still, the standard is so complex that even `OPmac-bib` does not handle all of the requirements that the standard introduces. `OPmac-bib` can, however, deal with this very reasonably. The package provides some versatile fields which can be used for entering bibliographic data along with the formatting macros to customize the field. Hence one can achieve the desired output: `option` and `ednote` are examples of such fields.

The `option` field can be used for entering other titles, translations of titles, etc. This field allows the correct output conforming to the latest version of the standard to be achieved.

The `ednote` field can be used for entering secondary authors or other additional information. The formatting of this field is not further processed, so the entered value is output as is. Hence one has to be careful to conform to the standard when entering the data. Typical examples of such data are translators or originators of multiple editions.

The availability of these additional fields and full customization allows for generating a bibliography that conforms to the standard ISO 690.

5 The biblatex-iso690 package

Of all the implementations incorporating ISO 690 mentioned in the previous section, only one is relevant to typesetting a bibliography in L^AT_EX, namely the `biblatex-iso690` package. The original implementation deviated from the standard, but since its review, the `biblatex-iso690` package is fully compliant with the latest version of the international standard ISO 690.

The original `biblatex-iso690` package contained the following defects and drawbacks:

- followed outdated editions of the standard,
- incorrect order of elements,
- redundant or missing punctuation,
- missing some types of resources,
- missing some required elements,
- missing creator secondary responsibility,
- obsolete and deprecated code.

An analysis of the original state of `biblatex-iso690` resulted in its complete reimplementaion. Printing the bibliography elements in the correct order in a reference was crucial, but not the only change. Almost all macros, commands and definitions for parsing fields from the `.bib` database file were refactored. Many requirements of the standard could be met simply by using the author interface of the BIBL^AT_EX package. For other requirements, it was necessary to refine some of the low-level macros, and still others were left to the programmer of the bibliography database as they could not be solved algorithmically. The known limitations are:

- lack of support for the running notes citation method,
- url addresses wrapping,
- algorithmic solution for (not) printing a first edition of a resource,
- algorithmic solution for (not) printing only the first publisher,
- algorithmic solution for (not) printing only the first location (e.g., of publication),
- the term *Anon* for anonymous works,
- localization string *nodate* for no date.

5.1 Methods of citation

The ISO 690 standard prescribes three citation methods of information resources. The first is the aforementioned running notes method, then there is the so-called Harvard system (also known as author-date), and lastly the numeric system. In the `biblatex-iso690`

package they are available as `iso-authoryear` and `iso-numeric`. The formatting style is specified as a package option when loading BIBL^AT_EX, e.g.

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

5.2 Package options — customization

ISO 690 does not prescribe any particular style, format or punctuation scheme for the references to be used. Frequently requested customizations (to style, format or punctuation scheme) are available as `biblatex-iso690` package options. These are:

- `spacecolon=[true|false]` changes the printing of colons in subtitles and publication information:
 - Place : Publisher
 - Place: Publisher
- `pagetotal=[true|false]` prints a total number of pages of a resource as optional information in square brackets:
 - Place : Publisher, 2008 [60 p.]
 - Place : Publisher, 2008
- `shortnumeration=[true|false]` distinguishes volumes and pagination typographically:
 - ... 2011, **32**(3), 289–301
 - ... 2011, vol. 32, no. 3, pp. 289–301
- `thesisinfoinnotes=[true|false]` to specify the position of thesis information:
 - ... Available from: *<url>*. BT. MU, FI, Brno. Supervisor Petr SOJKA
 - ... BT. MU, FI, Brno. Supervisor Petr SOJKA. Available from: *<url>*

5.3 Integration into the fithesis3 class

`fithesis3` is the official document class for the typesetting of theses at Masaryk University (Brno, Czech Republic) in L^AT_EX (Novotný et al., 2015). This class has been designed for easy style extensibility and for local files from other academic institutions. It was also an obvious request to integrate the `biblatex-iso690` package into the `fithesis3` template. This integration has been done in cooperation with the maintainer of the `fithesis3` package — Vít Novotný — and consists of the following steps:

- the `bib` key added to the package metadata section, which can be used to specify a list of `.bib` database files,
- the citation method is loaded automatically based on the selected faculty,
- the list of references is printed automatically at the end of a document,

- all bibliography management can also be set up manually (see section 3.3).

```
\documentclass{fithesis3}
\thesissetup{
...
  bib = {database.bib}
...
}
\begin{document}
... \cite{...} ...
\end{document}
```

5.4 Availability

As already mentioned, until now there was no official support for the ISO 690 standard in L^AT_EX. However, biblatex-iso690 package has acquired official status after the revision and is now available from CTAN as the package biblatex-iso690. Under the same name it is also available in the T_EX Live distribution since T_EX Live 2016.

6 Summary

This paper describes typesetting a bibliography in L^AT_EX, compliant with the international standard ISO 690. The standard was introduced at the beginning, followed by considerations of three methods of typesetting a bibliography in L^AT_EX. There are many implementations incorporating the standard ISO 690 but the biblatex-iso690 package holds the most interest: after its initial implementation in 2011, it was revised in 2016 to fully meet the requirements of the most recent version of the standard. References in this article are generated using the reimplemented package biblatex-iso690, to serve as a demonstration.

Acknowledgements

I gratefully acknowledge the funding received from the Faculty of Informatics at the Masaryk University in Brno for the development of the package.

I would also like to acknowledge Michal Hoftich, Vít Novotný and Moritz Wemheuer for their continuous support while developing the package and Petr Sojka for supervising the whole project. And special thanks to James Thomas, Barbara Beeton and Karl Berry for their careful proofreading of this paper.

References

- BIB_TE_X*: *Process bibliographies for L^AT_EX, etc.* 2010. CTAN: *The Comprehensive T_EX Archive Network* [online] [visited on 2016-05-14]. Available from: <https://www.ctan.org/pkg/bibtex>.
- BIB_LA_TE_X*: *Sophisticated Bibliographies in L^AT_EX*, 2016. CTAN: *The Comprehensive T_EX Archive Network* [online] [visited on 2016-05-14]. Available from: <https://www.ctan.org/pkg/biblatex>.
- ANSORGE, Libor; KRATOCHVÍL, Jiří, 2013. Popis šablony ČSN ISO 690:2011 v jazyce CSL pro citační manažer Zotero. *ProInflow*. Vol. 5, no. 2. ISSN 1804-2406. Available also from: http://pro.inflow.cz/sites/default/files/pdfclanky/Kratochvil_Ansorge_Sablona_0.pdf.
- Bib_lat_ex: submitting to a journal*, 2011. T_EX – L^AT_EX Stack Exchange [online] [visited on 2016-05-14]. Available from: <http://tex.stackexchange.com/questions/12175/biblatex-submitting-to-a-journal>.
- Bib_TE_X Format Description*, 2006. *Bib_TE_X.org* [online] [visited on 2017-05-08]. Available from: <http://www.bibtex.org/Format>.
- BRATKOVÁ, Eva (comp.), 2008. *Metody citování literatury a strukturování bibliografických záznamů podle mezinárodních norem ISO 690 a ISO 690-2: metodický materiál pro autory vysokoškolských kvalifikačních prací* [online]. Verze 2.0, aktualiz. a rozšíř. Praha: Odborná komise pro otázky elektronického zpřístupňování vysokoškolských kvalifikačních prací, Asociace knihoven vysokých škol ČR [visited on 2016-02-02]. Available from: <http://www.evskp.cz/SD/4c.pdf>.
- ČSN ISO 690: Dokumentace – Bibliografické citace – Obsah, forma a struktura*, 1996. Praha: Český normalizační institut. Třídící znak 01 0197.
- ČSN ISO 690: Informace a dokumentace – Pravidla pro bibliografické odkazy a citace informačních zdrojů*, 2011. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. Třídící znak 01 0197.
- ČSN ISO 690-2: Informace a dokumentace – Bibliografické citace – Část 2: Elektronické dokumenty nebo jejich části*, 2000. Praha: Český normalizační institut. Třídící znak 01 0197.
- DIN ISO 690: Information und Dokumentation – Richtlinien für Titelangaben und Zitierung von Informationsressourcen*, 2013. Berlin: DIN Deutsches Institut für Normung e. V.
- FENNER, Martin, 2010. Citation Style Language: An Interview with Rintze Zelle and Ian Mulvany. *Gobbledygook* [online] [visited on 2016-04-16]. Available from: <http://blogs.plos.org/mfenner/2010/09/24/citation-style-language-an-interview-with-rintze-zelle-and-ian-mulvany/>.
- HÁLA, Tomáš, 2013. Komentář k nové revizi normy ČSN ISO 690 – Pravidla pro bibliografické odkazy a citace informačních zdrojů. *Zpravodaj Československého sdružení uživatelů T_EXu*. Vol. 23, no. 2, pp. 107–112. ISSN 1211-6661. Available from DOI: 10.5300/2013-2/107.
- HARDERS, Harald, 2002. Multilingual bibliographies: Using and extending the babelbib package. *TUGboat*. Vol. 23, no. 3/4, pp. 344–353. Available also from: <https://www.tug.org/TUGboat/tb23-3-4/tb75harders.pdf>.
- HOFTICH, Michal, 2011. *The biblatex-iso690 package: ISO 690 style for BIB_LA_TE_X* [GIT] [visited on 2016-12-29]. Available from: <https://github.com/michal-h21/biblatex-iso690>.
- HUFFLEN, Jean-Michel, 2008. Languages for bibliography styles. *TUGboat: Proceedings of the 2008 Annual Meeting*. Vol. 29, no. 3, pp. 401–412. Available also from: <https://www.tug.org/TUGboat/tb29-3/tb93hufflen.pdf>.

- HUFFLEN, Jean-Michel, 2011. A comparative study of methods for bibliographies. *TUGboat: TUG 2011 Proceedings*. Vol. 32, no. 3, pp. 289–301. Available also from: <https://www.tug.org/TUGboat/tb32-3/tb102hufflen.pdf>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015. *ISO Membership Manual* [online]. Geneva [visited on 2016-05-20]. ISBN 978-92-67-10611-3. Available from: http://www.iso.org/iso/iso_membership_manual.pdf.
- ISO 690: Information and documentation – Bibliographic references – Content, form and structure*, 1987. Second edition. Geneva: The International Organization for Standardization.
- ISO 690: Information and documentation – Guidelines for bibliographic references and citations to information resources*, 2010. Third edition. Geneva: The International Organization for Standardization.
- ISO 690-2: Information and documentation – Bibliographic references – Part 2: Electronic documents or parts thereof*, 1997. First edition. Geneva: The International Organization for Standardization.
- KIME, Philip; CHARETT, François, 2016. *Biber: A backend bibliography processor for biblatex* [online]. Version 2.7 [visited on 2016-12-31]. Available from: <https://www.ctan.org/pkg/biber>.
- KRATOCHVÍL, Jiří; SEJK, Petr; ELIÁŠOVÁ, Věra; STEHLÍK, Marek, 2011. *Metodika tvorby bibliografických citací* [online]. 2. revidované vydání. Návrh obálky MAZOCH, Břetislav. Brno: Masarykova univerzita [visited on 2016-03-16]. ISSN 1802-128X. Available from: http://is.muni.cz/do/rect/el/estud/prif/ps11/metodika/web/ebook_citace_2011.html.
- LEHMAN, Philipp; WRIGHT, Joseph; BORUVKA, Audrey; KIME, Philip, 2016. *The biblatex package: Programmable Bibliographies and Citations* [online]. Version 3.7 [visited on 2016-12-31]. Available from: <https://www.ctan.org/pkg/biblatex>.
- LUPTÁK, Dávid. *Typesetting of Bibliography According to ISO 690 Norm* [online] [visited on 2016-06-14]. Available from: <https://is.muni.cz/th/422640/?lang=en>. BT. Masaryk University, Faculty of Informatics, Brno, Czech Republic. Supervised by Petr SOJKA.
- MARKEY, Nicolas, 2009. *Tame the BeaST: The B to X of BIBTEX* [online]. Version 1.4 [visited on 2016-05-14]. Available from: <https://www.ctan.org/pkg/tamethebeast>.
- MARTINEK, David, 2006. *The czechiso package: Czech style for BIBTEX* [online] [visited on 2016-12-29]. Available from: <http://www.fit.vutbr.cz/~martinek/latex/czechiso.html>.
- MITTELBAACH, Frank; GOOSSENS, Michel; BRAAMS, Johannes; CARLISLE, David; ROWLEY, Chris, 2004. *The LATEX Companion*. Second Edition. Boston: Addison-Wesley. Tools and Techniques for Computer Typesetting. ISBN 0-201-36299-6. Fourth printing (with corrections), Sept. 2005.
- MORI, Lapo F., 2009. Managing bibliographies with LATEX. *TUGboat*. Vol. 30, no. 1, pp. 36–48. Available also from: <https://www.tug.org/TUGboat/tb30-1/tb94mori.pdf>.
- NOVOTNÝ, Vít; MAREK, Daniel; PAVLOVIČ, Jan; SOJKA, Petr, 2015. *The fithesis3 class for the typesetting of theses written at the Masaryk University in Brno* [GIT] [visited on 2016-05-16]. Available from: <http://github.com/witiko/fithesis3.git>.
- OLŠÁK, Petr, 2016. OPmac-bib: Citations using *.bib files with no external program. *TUGboat*. Vol. 37, no. 1, pp. 71–78. Available also from: <https://www.tug.org/TUGboat/tb37-1/tb115olsak-bib.pdf>.
- PATASHNIK, Oren, 1988. *Designing BIBTEX Styles* [online] [visited on 2016-05-14]. Available from: <http://mirrors.ctan.org/biblio/bibtex/base/btxhak.pdf>.
- PATASHNIK, Oren, 1994. BIBTEX 1.0. *TUGboat: Proceedings of the 1994 Annual Meeting*. Vol. 15, no. 3, pp. 269–273. Available also from: <https://www.tug.org/TUGboat/tb15-3/tb44patashnik.pdf>.
- PATASHNIK, Oren, 2003. BIBTEX Yesterday, Today, and Tomorrow. *TUGboat: Proceedings of the 2003 Annual Meeting*. Vol. 24, no. 1, pp. 25–30. Available also from: <https://www.tug.org/TUGboat/tb24-1/patashnik.pdf>.
- SHELL, Michael; HOADLEY, David, 2007. *BIBTEX Tips and FAQ* [online]. Version 1.1 [visited on 2016-05-14]. Available from: <http://mirrors.ctan.org/biblio/bibtex/contrib/doc/btxFAQ.pdf>.
- STN ISO 690: Informácie a dokumentácia. Návod na tvorbu bibliografických odkazov na informačné pramene a ich citovanie*, 2012. Bratislava: Úrad pre normalizáciu, metrológiu a skúšobníctvo Slovenskej republiky. Triediaci znak 01 0197.
- TALBOT, Nicola L. C., 2012. *LATEX for Complete Novices*. Norfolk, UK: Dickimaw Books. Dickimaw LATEX Series. ISBN 978-1-909440-00-5. Available also from: <http://www.dickimaw-books.com/latex/novices>.
- TALBOT, Nicola L. C., 2013. *Using LATEX to Write a PhD Thesis*. Norfolk, UK: Dickimaw Books. Dickimaw LATEX Series. ISBN 978-1-909440-02-9. Available also from: <http://www.dickimaw-books.com/latex/thesis>.
- ZELLE, Rintze, 2016a. *CitationStyles.org | The Citation Style Language: open and free citation styles* [online] [visited on 2016-05-17]. Available from: <http://citationstyles.org/>.
- ZELLE, Rintze, 2016b. *Styles. CitationStyles.org* [online] [visited on 2016-05-17]. Available from: <http://citationstyles.org/styles/>.

◇ Dávid Lupták
 Faculty of Informatics
 Masaryk University
 Brno, Czech Republic
 422640 (at) mail dot muni dot cz
<https://github.com/DavidLuptak>

MIBIB_{TEX} now handles Unicode*

Jean-Michel HUFFLEN

Abstract

A new version of MIBIB_{TEX} can deal with the full range of Unicode and can process .bib files written using most byte-based encodings. We describe the new organisation of this version and show how to use the executable files built by the installation procedure. We also summarize the syntactic extensions implemented within .bib files, some originating from new fields introduced by the biblatex package.

Keywords MIBIB_{TEX}, kernel and derived programs, interface with Scheme, recognised formats and encodings, output routines, biblatex package, Con_{TEX}t.

Streszczenie

Nowa wersja MIBIB_{TEX}-a radzi już sobie z unikiem w pełnym zakresie i potrafi przetwarzać pliki .bib zapisane z użyciem większości kodowań jednobajtowych. Zostanie opisana nowa organizacja tej wersji oraz sposób używania plików wykonywalnych, jakie buduje procedura instalacyjna. Zostaną zwięźle omówione rozszerzenia syntaktyczne zaimplementowane w plikach .bib, z których niektóre mają źródło w nowych polach pakietu biblatex.

Słowa kluczowe MIBIB_{TEX}, jądro i programy pochodne, interfejs do Scheme, rozpoznawane formaty i kodowania, procedury wyjściowe, biblatex pakiet, Con_{TEX}t.

Introduction

Let us recall that the MIBIB_{TEX}¹ program aims to be a ‘better’ BIB_{TEX}, that is, a ‘better’ bibliography processor for documents written using L_AT_EX.

Since its beginning, this project has particularly focused on multilingual features. Then it has also provided better functions from a point of view related to programming. For example, the sort function used within BIB_{TEX}’s bibliography styles [13] can only be customised by redefining *one* sort key, built by concatenating strings.² On the contrary, sort functions handled by MIBIB_{TEX} can be more easily adapted or redefined. Although MIBIB_{TEX} includes a rich collection of ‘predefined’ order relations, such a *modus operandi* means that users interested in *ad hoc* sort procedures are able to write functions in Scheme [14],

* Previously entitled: *MIBIB_{TEX} Now Deals with Unicode*. Polish title: *MIBIB_{TEX} od teraz rozumie Unicode*.

¹ MultiLingual BIB_{TEX}.

² BIB_{TEX} can only perform *lexicographic sorts*; its sort procedure cannot deal with numbers.

the implementation language of MIBIB_{TEX}. That may be viewed as restrictive, but much synergy exists among L_AT_EX users, so we think that the advantages of this approach outweigh the drawbacks: programmers can help non-programmers. On another point, MIBIB_{TEX} went beyond exclusively generating L_AT_EX ‘References’ sections: it can also generate bibliographies according to other output formats, some examples being Con_{TEX}t [1], XML³-like formats, or simple texts.

In [7], we recalled the successive steps of the development of MIBIB_{TEX} and announced a new version (1.4), more new features being described in [8]. This new version’s main point is the ability to deal with the full range of the Unicode encoding and character standard [15]. So MIBIB_{TEX} is now able to process bibliography database (.bib) files encoded with conventions other than ASCII⁴ and Latin 1, an extension suitable for western European languages. This new version will be publicly available in Summer 2017. Hereafter, after a short review of MIBIB_{TEX}’s organisation (§1), we progressively describe this new version’s state about the formats recognised (§2), the bibliography styles which may be used (§3), and the output routines for each output format (§4).

1 MIBIB_{TEX}’s organisation

We detailed MIBIB_{TEX}’s organisation in [9, Fig. 5]. Let us recall that this program gets information from an .aux file about *citation keys* and .bib files, and also looks into the preamble of a .tex master file for the languages used throughout a L_AT_EX document if the babel package is loaded. Parsing .bib files results in an (S)XML⁵ tree. A *bibliography style* is applied to this tree, and *output routines* allow the result of such a style to conform to an output format’s needs. For example, different output routines are called in order to build bibliographies for documents using L_AT_EX and Con_{TEX}t, as explained in [9].

In [4] we explained that MIBIB_{TEX} is composed of a *kernel*, upon which *executable programs* are built.⁶ The programs listed here have been updated:

`mlbibtex` aims to replace BIB_{TEX};

`mlbiblatex` builds bibliographies (.bbl source files) suitable for the biblatex package [12]; it can be an

³ eXtensible Markup Language.

⁴ American Standard Character Information Interchange.

⁵ Scheme implementation of XML [11].

⁶ We can statically determine all the modules composing such an executable program. Besides, each program has its own arguments, some being irrelevant for other programs. That is why we think that building separate programs is better. But if end-users prefer to have only one program with more options, we can do that with a wrapper program written using a script language.

```
%encoding = latin1
@BOOK{henze1973,
  AUTHOR = {first => Hans Werner,
            last => Henze},
  TITLE = {Zweites Violinkonzert für
            Sologeiger, Tonband,
            Baß-bariton und 33
            Instrumentalisten},
  PUBLISHER = {B. Scott Söhne},
  ADDRESS = {Mainz},
  YEAR = 1973,
  LANGUAGE = german}
```

Figure 1: Example using the Latin 1 encoding.

alternative to the Biber bibliography processor [10];

`mlbibcontext` generates bibliographies suitable for `ConTEXt`;

`mlbib2xml` converts `.bib` files according to the XML format internally used by `MIBIBTEX`.

The `hal` program, used to populate the `HAL`⁷ open-archive site [3] has not yet been updated.⁸

2 Formats recognised

The new `%encoding` directive at the beginning of a `.bib` file, allows the encoding of the file to be specified. Some extensions of ASCII — e.g., Latin 1, Latin 2 — are now recognised. More precisely, most *byte-based* encodings are handled, in particular UTF⁹-8. The UTF-16 encoding, based on 16-bit units, will be added to the allowed encodings later. We recommend end-users specify information about encoding explicitly, even though `MIBIBTEX` tries to guess a `.bib` file’s encoding, because it may be difficult to guess correctly. Let us consider the `file` command, generally used to determine such encodings on operating systems such as Linux and Mac OS X. Applying this command to the files of Figs. 1 and 2 reports that the used encodings belong to ISO-8859, a series of 8-bit character encodings — including Latin 1 (ISO-8859-1) for western European languages and Latin 2 (ISO-8859-2) for eastern European Latin-alphabet languages — but gives no more precise information.¹⁰

Let us be clear that a text may use citation keys belonging to *several* `.bib` files with different encodings,

⁷ *Hyper-Article en Ligne*, that is, ‘hyper-article on-line’.

⁸ Since the format used for metadata by this site has changed, a new version of this program requires major rewriting; this will be done for a future release.

⁹ Unicode Transformation Format.

¹⁰ It is unlikely that one end-user uses `.bib` files with these two encodings, so changing the default input encoding — as shown below — may fix this problem. But relying on this technique is error-prone.

```
%encoding = latin2
@BOOK{morys-twarowski2016,
  AUTHOR = {first => Michael,
            last => Morys-Twarowski},
  TITLE = {Polskie Imperium. {Wszystkie
            kraje podbite przez
            Rzeczpospolitą}},
  PUBLISHER = {Ciekawostki Historyczne.pl},
  ADDRESS = {Kraków},
  DATE = {2016-02-17},
  LANGUAGE = polish}
```

Figure 2: Example using the Latin 2 encoding.

```
%encoding = utf8
@BOOK{lem1964,
  AUTHOR = {Stanisław Lem},
  TITLE = {Bajki robotów},
  PUBLISHER = {Wydawnictwa Literackiego},
  YEAR = 1964,
  LANGUAGE = polish}
```

Figure 3: Example using the UTF-8 encoding.

for example, the three files given in Figs. 1–3 (notice the German letter ‘ß’ directly included in Fig. 1 and the Polish diacritical signs in Figs. 2 and 3). All the syntactic extensions for `.bib` files are still usable, including the new syntax for people’s names by means of *keywords* (cf. Figs. 1 and 2). Most of the fields added by the `biblatex` package are recognised,¹¹ too; an example is the `DATE` field, used within Fig. 2 instead of the fields `YEAR`, `MONTH` and `DAY`.¹²

By default, `MIBIBTEX` looks for `.bib` files for bibliographical entries, the default encoding of such files being Latin 1. It can also parse XML files for bibliographical entries, according to the `mlbiblio` format used by `MIBIBTEX`.¹³ The bibliographical entries cited throughout a document can be saved as an XML file, too. Hereafter we give two simple examples of using the interface with Scheme. It consists of Scheme definitions put in *initialisation files* located in your home directory. On Unix-based systems, the executable programs derived from `MIBIBTEX`’s kernel look for the following initialisation files:

```
mlbibtex <== ~/mlbibtex
mlbiblatex <== ~/mlbiblatex
mlbibcontext <== ~/mlbibcontext
```

¹¹ By ‘recognised’, we mean that a *type* is associated with such a field, and type-checking is performed as soon as corresponding values are parsed.

¹² This last field is recognised by `MIBIBTEX`, but is not used by ‘old’ `BIBTEX`’s standard bibliography styles.

¹³ Conventions given in [2] by means of a DTD (`Document Type Definition`) are now refined using XML Schema [17].

```

\documentclass{article}

\usepackage[T1]{fontenc}
%% \usepackage[utf8]{inputenc}

\begin{document}

Did you hear \cite{henze1973}? And did you read
\cite{lem1964,morys-twarowski2016}?

\bibliography{figure-1,figure-2,figure-3}
\bibliographystyle{plain}

\end{document}

```

Figure 4: L^AT_EX document using Figs. 1–3’s entries.

In particular, you can:

- allow MIBIB_TE_X to look for an $\langle f \rangle$ -mlbiblio.xml file when an $\langle f \rangle$.bib file is not found:

```

((bib-files-functions-pv 'set)
 (list s-parse-bib-file
       sxmlh-parse-mlbiblio-xml-file))

```
- change the default encoding of .bib files:

```

((encodings-pv
 'set-default-4-bib-files)
 'utf8)

```

You can use *prefixes* for different namespaces as described in [5], and put *inexact* information according to [6]’s syntax, but only with the two programs mlbibtex and mlbibtex2xml. The programs mlbiblatex and mlbibcontext have not incorporated these features yet.

3 Bibliography styles

BIB_TE_X’s standard bibliography styles written using [13]’s language can be used by the executable program mlbibtex, even if some fields introduced by the biblatex package are used instead of standard fields — e.g., the DATE field, instead of the standard fields YEAR and MONTH. Styles written using the nbst¹⁴ language can be used, too. The two executable programs mlbiblatex and mlbibcontext use *direct styles* — using MIBIB_TE_X’s terminology, such styles are wholly written in Scheme [4]; these styles have been updated.

4 Output routines

The encoding of an output file generated by our programs is:

¹⁴ New Bibliography STyles. Let us recall that this language is close to the first version of XSLT (eXtensible Stylesheet Language Transformations) [16].

ASCII for a file suitable for L^AT_EX, unless another encoding is given within the master file’s preamble by means of the inputenc or as an option of the mlbiblatex program;

UTF-8 for a file suitable for ConT_EXt (the option allowing the choice of an encoding has been removed) or an XML file built by the mlbib2xml program, unless another encoding is given as an option.

Now we give a simple example by considering the L^AT_EX document given in Fig. 4. Let us recall that ‘old’ BIB_TE_X operates on .aux files and never reads .tex files. On the contrary, MIBIB_TE_X reads both an .aux file and the preamble of the corresponding .tex file. If Fig. 4 is processed *as it is*, the first reference built by the executable program mlbibtex looks like:

```

\bibitem{henze1973}
Hans Werner Henze.
\newblock {\em Zweites Violinkonzert
f"\{u}r Solologeiger, Tonband,
Ba{\ss}-bariton... } ...

```

that is, all the accented letters are replaced by the T_EX commands used to produce them, since the encoding is supposed to be ASCII. If the line concerning the inputenc package in Fig. 4 is uncommented, this first reference becomes:

```

\bibitem{henze1973}
Hans Werner Henze.
\newblock {\em Zweites Violinkonzert für
Solologeiger, Tonband, Baß-bariton... } ...

```

that is, the .bbl file built by MIBIB_TE_X is encoded using UTF-8.

5 Conclusion

We need to revise the installation procedure, some points now being unsatisfactory. The complete documentation also needs to be updated. But now MIBIB_TE_X is ready to deal with Unicode.

6 Acknowledgements

Many thanks to the Polish translators: Ryszard Kubiak for the abstract and Jerzy B. Ludwichowski for the keywords. Thanks to GUTenberg, the French-speaking T_EX users group, which offered me a grant for participating in this TUG@BachoT_EX 2017 conference. I am also grateful to this definitive version’s proofreaders: Karl Berry and Barbara Beeton.

References

- [1] ConT_EXt Garden: *Bibliographies in MkII*. April 2012. <http://wiki.contextgarden.net/Bibliography>.

- [2] Jean-Michel HUFFLEN: “Multilingual Features for Bibliography Programs: From XML to MIBIB \TeX ”. In: *Euro \TeX 2002*, pp. 46–59. Bachotek, Poland. April 2002.
- [3] Jean-Michel HUFFLEN: “From Bibliography Files to Open Archives: The Sequel”. In: Karl BERRY, Jerzy B. LUDWICHOWSKI and Tomasz PRZECHLEWSKI, eds., *Proc. EuroBach \TeX 2011 Conference*, pp. 61–66. Bachotek, Poland. April 2011.
- [4] Jean-Michel HUFFLEN: “MIBIB \TeX and Its New Extensions”. In: *Proc. 6th Con \TeX t Meeting & Euro \TeX 2012*, pp. 82–91. Breskens, The Netherlands. October 2012.
- [5] Jean-Michel HUFFLEN: “Managing Name Conflicts and Aliasing with MIBIB \TeX ”. In: Tomasz PRZECHLEWSKI, Karl BERRY, Bogusław JACKOWSKI and Jerzy B. LUDWICHOWSKI, eds., *What Can Typography Gain from Electronic Media? Proc. Bach \TeX 2014 conference*, pp. 13–16. Bachotek, Poland. April 2014.
- [6] Jean-Michel HUFFLEN: “Dealing with Ancient Works in Bibliographies”. *Ars \TeX nica*, Vol. 18, pp. 81–86. In Proc. GUIT meeting 2014. October 2014. <http://www.guitex.org/home/images/ArsTeXnica/AT018/hufflen-verona.pdf>.
- [7] Jean-Michel HUFFLEN: “From MIBIB \TeX 1.3 to 1.4”. In: Tomasz PRZECHLEWSKI, Karl BERRY, Bogusław JACKOWSKI and Jerzy B. LUDWICHOWSKI, eds., *Various Faces of Typography. Proc. Bach \TeX 2015 conference*, pp. 13–17. Bachotek, Poland. April 2015.
- [8] Jean-Michel HUFFLEN: “MIBIB \TeX 1.4: The New Version”. *Ars \TeX nica*, Vol. 20, pp. 35–39. In Proc. GUIT meeting 2015. October 2015.
- [9] Jean-Michel HUFFLEN: “MIBIB \TeX & Con \TeX t: Face-to-Face”. In: *Proc. 9th Con \TeX t Meeting*, pp. 27–48. Nasbinals, France. Abridged version. September 2016.
- [10] Philip KIME and François CHARETTE: *biber. A Backend Bibliography Processor for biblatex. Version biber 2.7 (biblatex 3.7)*. 5 December 2016. <https://ctan.org/pkg/biber>.
- [11] Oleg E. KISELYOV: *XML and Scheme*. September 2005. <http://okmij.org/ftp/Scheme/xml.html>.
- [12] Philipp LEHMAN, with Philip KIME, Audrey BORUVKA and Joseph WRIGHT: *The biblatex Package. Programmable Bibliographies and Citations. Version 3.7*. 16 November 2016. <https://ctan.org/pkg/biblatex>.
- [13] Oren PATASHNIK: *Designing BIB \TeX Styles*. February 1988. Part of the BIB \TeX distribution.
- [14] Alex SHINN, John COWAN, and Arthur A. GLECKLER, with Steven GANZ, Aaron W. HSU, Bradley LUCIER, Emmanuel MEDERNACH, Alexey RADUL, Jeffrey T. READ, David RUSH, Benjamin L. RUSSEL, Olin SHIVERS, Alaric SNELL-PYM and Gerald Jay SUSSMAN: *Revised⁷ Report on the Algorithmic Language Scheme*. 6 July 2013. <http://trac.sacrideo.us/wg/raw-attachment/wiki/WikiStart/r7rs.pdf>.
- [15] THE UNICODE CONSORTIUM: *Unicode 9.0.0*. June 2016. <http://www.unicode.org/versions/Unicode9.0.0/>.
- [16] W3C: *XSL Transformations (XSLT). Version 1.0*. W3C Recommendation. Edited by James Clark. November 1999. <http://www.w3.org/TR/1999/REC-xslt-19991116>.
- [17] W3C: *XML Schema*. December 2008. <http://www.w3.org/XML/Schema>.

◇ Jean-Michel HUFFLEN
 FEMTO-ST (UMR CNRS 6174)
 & University of Bourgogne
 Franche-Comté
 16, route de Gray
 25030 Besançon Cedex
 France
 jmhuffle (at) femto-st dot fr
<http://members.femto-st.fr/jean-michel-hufflen>

TeX user habits versus publisher requirements

Lolita Tolén

Abstract

Typesetters always balance on the thin line between unlimited author creativity and strict publisher requirements to produce full-text XML. In this paper we present both sides.

TeX is designed in a way that offers wide capabilities to achieve the desired goal in many different ways. Therefore a huge collection of TeX packages has been created over the years. Even more local macros are used every day. We present which TeX packages are commonly used for the scientific content and what proportion of them comes from the standard sources, such as CTAN and TeX Live. We give some insights into authors' habits using TeX for writing scientific content. Also keeping XML in mind, we discuss how and why these habits are important for typesetters while preparing papers for publishing.

1 Introduction

Scientific content preparation for publication is a substantive task, where a typesetter must balance the researcher's vision of how the content is best presented for the scientific community with the publisher's requirements for the journal style and XML¹ content. The XML format has become a standard in storage and making the electronic documents available. Therefore almost all publishing houses we have encountered provide a DTD² for XML production, which defines structure directed not only to appearance, but very often to the meaning of the content.

In our workflow, PDF and XML are produced from L^AT_EX documents. TeX, the formatting engine of L^AT_EX 2_ε, is highly portable and free. Therefore the system runs on almost any hardware platform available [2]. So TeX has become the standard text processing system in many academic high-level scientific and research institutions.

In processing the incoming L^AT_EX³ manuscripts, typesetters strongly depend on the stability of TeX distributions and source file contents. TeX is designed in a way that offers wide capabilities to achieve the desired goal in many different ways. Therefore a huge collection of TeX packages has been created over the years. Even more local macros are used every day. This is all very attractive for the authors,

but with the XML format in mind, it often becomes a burden.

In the following we discuss difficulties related to manuscripts becoming printed copy, while meeting publisher requirements (Section 2). In Section 3 we provide a statistical overview of about 90 000 STM (scientific, technical and medical) L^AT_EX papers prepared for publishing in the last 7 years. In Section 4 some final remarks will be given.

2 From manuscript to printed copy

A manuscript prepared for publication eventually becomes printed copy, meeting journal style requirements. It also contains enriched structure, which is converted into XML, valid for a publisher-specific DTD. Requirements directed to the meaning of content are the most difficult to fulfil and we always search for some ways to ease this process. In the following we discuss most common obstacles for producing a valid XML structure from L^AT_EX documents.

L^AT_EX is used to display the content in the desired way, very often forgetting about the meaning and consistency. Broken math formulas (see the upper part of Fig. 1) or a single phrase split across several cells in a table (see L^AT_EX code and its output in the lower part of Fig. 1) are good examples.

L^AT_EX can be used to change the appearance of some content or to create some symbols in many different ways, but often such code has no equivalent in the XML (see examples in Fig. 2). Such structures have to be replaced with a Unicode symbol or converted to pictures.

Publisher requirements for XML usually state to use MathML⁴ for mathematical content. A graphic made from a formula is not very pleasant to the reader's eye; it does not scale so smoothly as a MathML object and most importantly it contains no constituent information, and is not editable.

Some content enrichment is done having in mind world wide databases, identifying authors by ORCID (Open Researcher and Contributor ID) or another code and connecting them to their papers, counting paper citations and determining journal impact factors, connecting research sponsors to grant numbers. Therefore frontmatter and backmatter are crucial parts of the paper. Depending on the publisher, requirements for, e.g., the bibliography references and their citation links are very strict and structure-difficult. One of the ways to fulfil these requirements is to create a hooked version of some standard TeX distribution package (such as `natbib` or `hyperref`), which generates necessary additional XML-oriented

¹ eXtensible Markup Language.

² Document Type Definition.

³ We rarely encounter manuscripts written in plain TeX, so we use L^AT_EX concepts throughout this article.

⁴ Mathematical Markup Language.

```

'$R=\{x|x$ is real $\}$' instead of '$R=\{x|x \mbox{ is real } \}$'

***

\begin{tabular}{ccccc}
\hline
Sample & Depth (cm) & Weight of Sample & CRS & Pb-210 age \\
Number & & Counted (g) & sediment accumulation & (year AD) \\
& & & rate (g/cm2/yr)a & \\
\hline
...
\end{tabular}

```

Sample Number	Depth (cm)	Weight of Sample Counted (g)	CRS sediment accumulation rate (g/cm ² /yr) ^a	Pb-210 age (year AD)
...				

Figure 1: L^AT_EX code examples of a broken math formula and table headings with phrases split into separate cells.

```

\raisebox{.2em}{\n$}\big/\raisebox{-.2em}{\m$} \implies n/m

***

${\longrightarrow\hspace*{-3.1ex}{\circ}\hspace*{1.9ex}} \implies \rightarrow

***

$1\!/\!1$ \implies 1 instead of \usepackage{dsfont}$\mathds{1}$ \implies 1

***

\newcommand{\forkindep}[1] [] {%
  \mathrel{\mathop{\vcenter{\hbox{\oalign{
    \noalign{\kern-.3ex}
    \hfil$\vert$\hfil\cr\noalign{\kern-.7ex}$\smile$\cr
    \noalign{\kern-.3ex}
  }}}\displaylimits_{#1}}}
}

```

Figure 2: Examples of symbols created using L^AT_EX: on the left-hand side is the source code, on the right-hand side, its output.

content without changing the user output. Such actions are extremely dependent on stability of packages and T_EX distributions.

On our side, as typesetters, there are few L^AT_EX to XML converters being used (like T_EX4ht or the one described in [1]). Also there are some thoughts to explore LuaT_EX-based converter possibilities. Each of them, theoretical or practical, have flaws different than others and one thing they all have in common — in order to produce an XML valid for a publisher-provided DTD, the source content has to be prepared, either changing the T_EX source directly or using available T_EX distribution tools.

Author creativity can often make the manuscript processing a very hard task. Looking at L^AT_EX document content, from a typesetter’s point of view

there are a few important highlights: document class and style packages declaring the formatting of the paper and locally defined macros. Manuscripts constructed with a *heavy* and deep understanding of T_EX structures require special accuracy — in order to create an XML which meets publisher requirements some of these structures are dismantled and replaced throughout a corpus (in other words, expanded), and others are converted into pictures while producing an XML. On the other hand, manuscripts using only *light* macros, such as defining repeatedly appearing phrases, measurement units etc., and packages found in one of the main T_EX distributions (such as T_EX Live or MiK_T_EX) or CTAN,⁵ require very little intervention, mainly oriented toward contex-

⁵ Comprehensive T_EX Archive Network.

tual enrichment for XML production (e.g., author information, funding related information connection to appropriate databases).

If the author uses a publisher-provided template, few changes are noticeable, whereas author-created formatting usually results in a completely different layout from the prepared printed copy, which makes it difficult to notice some unintended mismatch to original output content. Also the author, having put so much work into creating the desired layout, often feels disappointed by the outcome.

Manuscripts written using mostly unstructured plain text usually do not change much from the layout point of view, but the corpus must be given a contextual meaning. Also, strange combinations of primitive \TeX command sequences, where usually some widely-known standard coding should have been applied, typically need to be replaced with the standard coding, but a human has to decide whether this is the case. Such manuscripts require reading the author’s mind to some extent, e.g., where the theorem or its proof ends — especially difficult if these structures are nested, i.e., a proof contains theorems and proofs of its own, whether the two letters combined into a single glyph should be replaced with an appropriate \LaTeX command sequence, or if this is some field-related denotation and should be left untouched (for XML a picture should be generated from this symbol), etc.

3 Manuscript content analysis

Data description This article provides a statistical overview of about 90 000 STM (scientific, technical and medical) \LaTeX papers which have been prepared for 252 journals of well-known publishing houses such as Elsevier, Springer, Mattson Publishing Services, BioMed Central, IOS Press, International Press, and others. The data covers the years 2010 to 2016. Manuscripts have not been sorted in any way, therefore they include random nationalities, institutions, science fields, etc.

The provided results were gathered by analyzing manuscript source files (\TeX), which show what researchers use for writing STM content. In order to see what is used overall, `.fls` and `.log` files have been analyzed. Only a small number of manuscripts are sent with compilation output files attached. In the current set `.log` files were found for 6% of manuscripts. Therefore compilation output files must have been produced by recompiling gathered manuscripts. For this purpose three distributions of \TeX Live, released in 2010, 2014 and 2016, were at our disposal. The following engines have been used for successful compilation of about 90% of manuscripts: `pdf \TeX`

Table 1: Formats used by researchers for manuscript compilation.

Format	Manuscript count
<code>pdflatex</code>	2962
<code>latex</code>	2489
<code>platex</code>	54
<code>xelatex</code>	52
<code>tex</code>	11
<code>amstex</code>	4
<code>pdftex</code>	4
<code>platex-sjis</code>	4
<code>lualatex</code>	3
<code>eplatex</code>	1
<code>mpost</code>	1
<code>uplatex</code>	1

(`latex`, `pdftex`, `pdflatex`), Lua \LaTeX (`lualatex`), X \TeX (`xelatex`). In order to generate the `.fls` files, the option `-recorder` was used.

Of course, we encounter only a portion of papers produced worldwide. Therefore, in most cases concrete numbers have no meaning here, and only percentages will be provided. All of the statistical data presented here can be accessed at github.com/vtex-soft/statistics.tex-manuscripts, and interested readers are encouraged to explore further.

One of the main interests in analyzing this data is to get a picture of which \TeX family tools are most popular from a researcher’s point of view and how it changes (if it does) over the years.

\TeX tools used As noted above, only 6% of manuscripts were provided with their compilation `.log` files. Additionally, 20% of cases had PDF files matching the `.tex` filename. Table 1 shows compilation formats used by researchers, extracted from `.log` file content and PDF metadata. While the most commonly used engine is `pdf \TeX` , 10 252 (48%) manuscripts were compiled to DVI first instead of directly producing a PDF file.

Most researchers use the latest \TeX distribution version. But as one can see from Fig. 3, there are a number of authors who compile their manuscripts with \TeX distributions up to ten years old.

Further analysis has been split into two main parts, separating the document classes and packages used.

Document classes Throughout the papers, 366 unique document classes were found. Only 15% are in \TeX Live distributions since 2010, 2 are in the current CTAN file list (namely `svjour` and `smfart`) and other classes are distributed by publishing houses or created by authors (see Table 2).

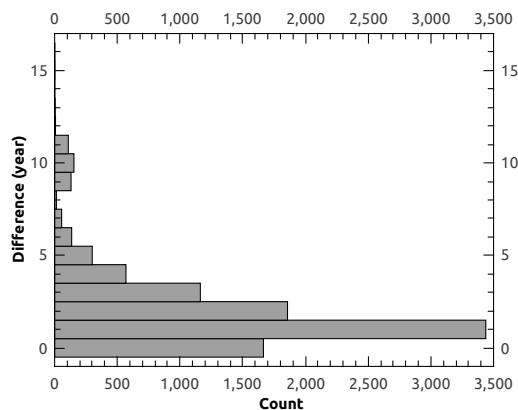


Figure 3: Age frequency of \TeX distribution originally used by manuscript authors.

Table 2: Counts of unique document classes and packages in the analyzed manuscripts.

	Classes count	Packages count
Total	366	1847
Since 2015	143	1023
In \TeX Live 2010–2016	55	996
In \TeX Live 2016	48	970
In CTAN	2	66

Most commonly, manuscripts were provided using the `article`, `elsarticle`, `amsart`, `svjour3` and `revtex4` classes (see Fig. 4). All of these, except for the `svjour3` class, are included in TL2016. The `svjour3` class is provided by *Springer*. Over the years it has replaced `svjour` and `svjour2` classes, but only the first version, `svjour`, is currently found on CTAN.

While the `article` class is used independent of publishing houses, for other classes in Table 3 one can see a relationship with the publishing house by which a manuscript has been accepted. This relationship between class and publisher is natural, because the latter usually promotes certain classes for particular journals or groups of journals, and provides templates for authors to use. Such publisher-oriented manuscripts are more easily transferred into publication-ready papers and require less intervention, and therefore there are fewer typesetting errors and layout changes. Sadly, this is the case for only a relatively small number of papers. The total number of different classes used, shown in Table 3, shows that a substantial number of manuscripts are written using rare classes, or classes normally used for another publisher’s journal.



Figure 4: Most common document classes used in the manuscripts. Word size reflects frequency.

\LaTeX 2 ϵ was introduced over 20 years ago, but we still encounter substantial use of the outdated \LaTeX 2.09 version. Over 1000 of the analyzed papers were formatted using `\documentstyle` command (the most often loaded class is `article`).

Packages Only 5% of manuscripts do not contain packages loaded in addition to the document class. Nearly 2000 unique packages were used throughout the manuscripts (see Table 2). More than half of them were found in \TeX Live, 66 more are in the current CTAN file list (e.g., `psfig`, `axodraw`, `picins`, etc.; 10 of them are obsolete) and other files are distributed by publishers or created by authors.

The most common packages are shown in Fig. 5 and Table 4. The top of the list is stable throughout the entire time span analyzed: the most commonly used are the American Mathematical Society (AMS) packages, then there are `graphicx`, `color`, `hyperref`, `inputenc`, `mathrsfs` and `epsfig`. Some packages have become more popular in the last two years, notably `hyperref` and `tikz`. It is interesting to note that, while according to CTAN the `graphicx` package is preferred over `epsfig`, use of the latter is decreasing only slowly.

The data shows that a few packages have been used only with certain document class: `fix-cm` with `svjour3` class in 98% of cases, `spr-astr-addons` with `aastex` class in 100% of cases. Packages like the AMS bundle are more likely to be used with any class. Fig. 6 shows how in the last two years used packages are related to the most common `article` class: mostly it is used in combination with styles related to layout formatting, such as `fullpage`, `fancyhdr`, `indentfirst`, etc.

Among the less frequently used packages are a few new styles:⁶ `mathpartir` (21 uses, on CTAN

⁶ Here a style is called new if it is included in \TeX Live 2016, but not in TL2014.

Table 3: The distribution of document classes according to the publishing house to which a manuscript has been submitted, as percentages.

Class	Last known source	BMC	DUP	Elsevier	International Press	IOS Press	Mattson	Springer
aastex ^a	TL2014							6
aicom2e	IOS Press					17		
amsart	TL2016	22	74	19	2	1	11	8
article	TL2016	42	19	29	15	16	37	29
bmc_article ^a	BMC	10						
bmcart ^a	BMC	15						
elsarticle ^a	TL2016	3		36				1
imsart ^a	IMS				1		46	
ios-book-article	IOS Press					4		
iosart2c	IOS Press					28		
ipart ^a	Intl. Press				78			
jaise2e	IOS Press					9		
svjour3	Springer	4						34
<i>Total (%)</i>		96	93	84	96	75	94	77

^a Document class uses `article` as parent, therefore the `article` class is used in 59% of all cases.

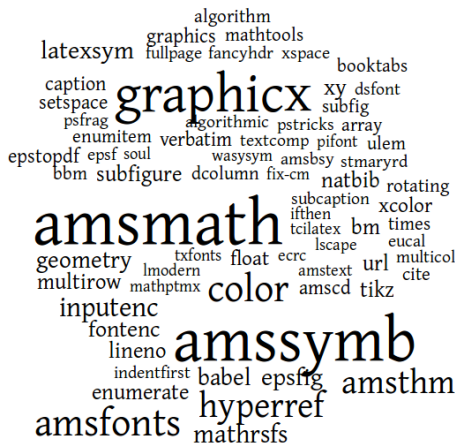


Figure 5: The 70 most common packages, extracted from manuscripts since 2015. Word size reflects frequency.

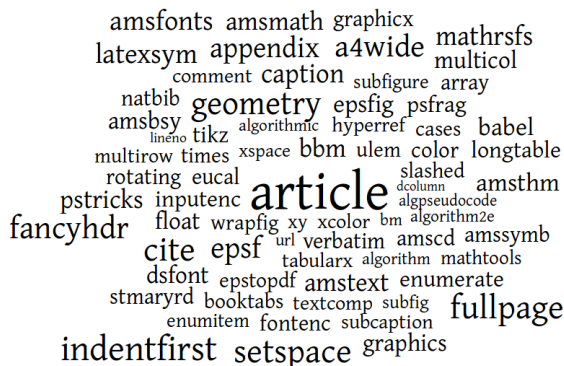


Figure 6: `article` class in relation to the packages shown in Fig. 5. Word size reflects frequency.

since 2016-02-26), `pgfornament` (3 uses, on CTAN since 2016-03-09), `prftree` (1 use, on CTAN since 2014-12-02), and `pstring` (1 use, on CTAN since 2017-01-05). The `todonotes` package appeared on CTAN in 2008-09-06 and its usage in the gathered manuscripts has steadily increased over the years. The package `subcaption` is on CTAN since 2002, but only in 2013 does this package appear among those used, with an increasing frequency since then.

A comparison of the packages loaded directly by the user (extracted from `.tex` files) and those loaded indirectly (extracted from `.fls` files) shows that many packages are bundles of files, and where the user loads only, e.g., the `graphicx` package, by default the `graphics` and `keyval` files are also loaded. It is interesting to note that in 40% of cases the `natbib` and `url` packages are not loaded directly.

AMS packages also dominate when comparing manuscripts with publisher-ready papers. Packages like `url`, `fontenc`, `bm` in the latter set of papers are used with 96%–99% frequency instead of 6%–7%. The `xcolor` package is used 4% more frequently than `color`, but this package overall is used at half the rate than in original manuscripts. Packages rarely used by authors such as `etoolbox` (94%), `ifthen` (18%), `dcolumn` (16%), `array` (16%), `afterpackage` (12%), `atveryend` (7%) are often used in publisher-ready paper preparation.

Package options 85% of the manuscripts’ used packages were loaded without additional options. 1453 of the unique packages were never given an

Table 4: Most common packages, split into the time ranges 2010–2014 and 2015–2016.

Package (2010–2014)	Usage (%)	Package (2015–2016)	Usage (%)
amsmath	52	amsmath	59
amssymb	51	amssymb	56
graphicx	51	graphicx	46
amsfonts	22	amsfonts	28
color	19	color	28
amsthm	14	hyperref	23
epsfig	13	amsthm	21
hyperref	13	inputenc	14
latexsym	11	mathrsfs	12
inputenc	9	epsfig	11
mathrsfs	8	latexsym	11
babel	8	babel	10
natbib	8	geometry	10
url	7	fontenc	9
fontenc	7	xy	9
subfigure	7	enumerate	8
bm	6	url	8
graphics	6	tikz	8
multirow	5	multirow	8
xy	5	lineno	8
geometry	5	natbib	8
enumerate	5	bm	7

option, while 109 of them always had at least one option specified. Among the most frequently used packages, `hyperref` and `inputenc` were given options in 50% and 99% of cases, respectively. The most common options for `hyperref` were: `colorlinks` (23%), `citecolor` (23%), `urlcolor` (19%), `linkcolor` (15%), `breaklinks` (6%), `bookmarks` (6%). The most common options for `inputenc` were `latin1` (40%) and `latin9` (10%). The `graphicx` package was rarely given an option, but the most common was `dvips`.

Fonts While there were some manuscripts compiled with Lua \TeX , no OpenType fonts were used. The `fontspec` package was used only two times and `.fls` files show that fonts were loaded using TFM files, Type 1, and virtual fonts. The most common font families are shown in Table 5: mostly the default Computer Modern family is used, unless `amsfonts` package is loaded, from which the `symbols`, `cmextra`, `euler` fonts are used.

4 Reflections

In this article we have briefly presented some statistical data gathered from about 90 000 STM manuscripts. The data shows that researchers most often use stable well-known packages and document classes,

Table 5: Most common font families. Data extracted from `.fls` files.

Font family	Usage (%)
cm	95
amsfonts	92
rsfs	13
symbol	11
zapfding	10
times	10
ec	8
xypic	7
txfonts	6
stmaryrd	3
courier	3
wasy	2
helvetic	1
bbm	1
cm-super	1
doublestroke	1
esint	1
lm	1
bbold	1

while new packages are promoted very slowly. At the same time, authors tend to create and use their own little \TeX tools.

Creating non-standard structures or formatting of the look of the manuscript, without consideration of the final structure-based product, creates many obstacles for processing author manuscripts into to-be-published papers. Such actions imply that the knowledge of the good \LaTeX practices is not spread widely enough, or authors are simply not familiar with publishing-related processes. On the other hand, this creativity shows that \TeX and its features have been found useful and popular among the worldwide scientific community.

References

- [1] V. Kriauciukas, L. Razinkovas, and L. Žamoitinaitė. Parsing \LaTeX for \LaTeX . In *BachoTeX 2015 proceedings*, pages 31–36. GUST, 2015.
- [2] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. *The Not So Short Introduction to $\LaTeX 2_{\epsilon}$* , 2015. <https://ctan.org/pkg/lshort>.

◇ Lolita Tolénė
 \TeX
Mokslininkų 2a
Vilnius LT-08412, Lithuania
lolita.tolene (at) vtex dot lt

Ten years of work in *Wiadomości Matematyczne*—an adventure with L^AT_EX and Emacs hacking

Marcin Borkowski

Abstract

Since 2007 I have been working for the “*Wiadomości Matematyczne*” journal (<http://wydawnictwa.ptm.org.pl/index.php/wiadomosci-matematyczne>), where I am responsible for—among other things—L^AT_EX-based typesetting. This is enough time to form some habits, and also to make some predictions. I would like to share them with my T_EX friends.

1 Introduction

This year marks the tenth anniversary of my work in *Wiadomości Matematyczne*, a (sort of) newsletter of the Polish Mathematical Society. (“Sort of” meaning it appears twice a year, so the “news” in “newsletter” is sometimes more like “olds.”) Together with a friend, we were appointed “secretaries”, which (at varying points in time) meant everything from handling email, to designing the look-and-feel of the printed issues, to hacking together L^AT_EX classes to accomplish that design, to proofreading and typesetting papers, making corrections suggested by the authors (or argue why they cannot be made), to actually driving to the post office to send out freshly printed issues to people.

Needless to say, using L^AT_EX (but also other tools, most notably *Emacs*) is a large part of this undertaking. In the present paper, I would like to share some experiences and thoughts on the matter.

This paper—or tale, if you will—is organized as follows. First, I explain some of the assumptions and policies we set up at the very beginning (surprisingly, many of them did not change). Then I proceed to what is perhaps the most interesting for the typesetting-oriented readers: the L^AT_EX classes we developed to prepare our pdfs. For the most part I abstain from quoting actual source code; an interested reader may find it on my website at <http://mbork.pl/wiadmatfiles.zip>, so that this paper does not get too T_EXnical. Next, I describe some *Emacs* functions which we use in our editing work; some of them are general enough to be of use for the wider public. Finally, I try to summarize my experience with a few general rules of thumb I found out to be useful.

In the paper I sometimes say “I” and sometimes “we”. The former means the author; the latter usually means the author together with the other secretary, Paweł Mleczko.

2 Assumptions and policies

When we started working on *Wiadomości Matematyczne*, we did not know much about what we are exactly expected to do. We knew that proofreading would be our primary duty. Soon it turned out that it was quite frustrating when the proofreaders and the typesetter did not remain in strict contact; in fact, the person responsible for typesetting lived in another city. We quickly figured out that if *we* did the typesetting (which we were confident we were capable of, and which we both liked to do), things would go much more smoothly. We also suggested to the editors that a visual overhaul would not be a bad idea.

That meant that we needed to do a few things, even before we dived into coding our L^AT_EX classes. One of them was deciding how we are going to keep all the incoming files in order. To this end, we adopted a very strict set of rules. It turned out that this was a good idea (and that an even stricter one would be even better!). First, I estimated the overall number of papers we may be dealing with over the course of several years; my estimation was that the number should not exceed a thousand within foreseeable future. Therefore we settled on 4-digits identifiers, just in case. (As of this writing, we are at 521.) Each paper was to get an identifier of the form `art-0000-name`, with the right number in place of the zeros (so we started with `art-0001-...`), and `name` was to be the family name of the (alphabetically) first author, folded to pure ASCII and in lowercase. That way, the identifiers are more-or-less human-readable, too—we humans are not as good as computers in remembering much numerical stuff.

Further, this means that each paper lands in a separate directory called `art-0000-name`, and this directory should contain a L^AT_EX file (unsurprisingly called `art-0000-name.tex`) and possibly other files, like `art-0000-name-photo-1`, etc. Also, each of these directories is a repository for a version control system (we settled on Mercurial). Finally, alongside those directories, we create directories named like `wm-53-1`, containing issue 1 of volume 53 of *Wiadomości Matematyczne*, with an appropriately named (and version-controlled) L^AT_EX file inside. With the help of a web-savvy friend we set up a server for all these repositories, so that we could easily pull each other’s changes and push our own. After some time, I also wrote a few shell scripts: one for executing some action (like *pull* or *update*) on all (appropriately named) repositories in a local directory, another one for cloning all repositories not present on my computer, etc.

Having that (and a few other things, like deciding on the encoding system— we started with `cp-1250` and at some point moved to `UTF-8`), we started to think about what we would need our \LaTeX classes to do. (It turned out much later that many of our assumptions were wrong.) The initial list looked more or less like this.

- Each paper should be typesettable separately, but we need a way to typeset the whole issue, too.
- We need to collect a very specific set of metadata about the paper and the authors, much beyond \LaTeX 's default trio of title, author's name and date.
- We really want grid typesetting, so we need e.g. heavily customized sectioning, enumerate and theorem-like commands.
- We want \LaTeX to do as much as possible for us, but we want a way to influence things manually if needed.

With those (and a few other) things in mind, I started developing the classes. It took maybe a few weeks to get some working prototypes/proofs of concept, and soon we had working \LaTeX classes. Obviously, in the course of actually using them, it turned out that they were not exactly ideal. After about five years we decided that our technological debt had risen to an unacceptable level and I decided to rewrite the classes (almost) from scratch. That turned out to be a good decision—the “new” classes, while still fairly complex, are much better to handle. The main goal when writing the “new” classes was simplicity. It turns out that if some obscure case comes up once in, say 20 or 50 articles (or even 10), coding a dozen or more lines of code to cater for that was a mistake. It's much better to deal with such rare cases manually. Of course, if we aimed at total automation of typesetting, the situation would be different; but since we carefully proofread each and every article ourselves anyway, it's more effective to have a simpler, more manageable codebase with clear ways to manually override the default behavior instead of some clever way for \TeX to do it itself without any way to influence its decisions.

3 \LaTeX classes

As I said, I will not go through the code of our classes *in extenso*; they are shy of 1800 lines of \LaTeX code, and it would be too boring anyway. Instead, I am going to highlight a few issues I think are interesting, in a kind of broad view perspective. Anyone wishing to see the nitty-gritty details is invited to look at the class code. It should be available on CTAN at some

point; meanwhile, I uploaded all code discussed here on my personal web page.

3.1 Documenting classes

From the very beginning I knew that I'd like my classes well-documented. The first version was written using the `gmdoc` class by Grzegorz Murzynowski. It later had some issues with `pdf \LaTeX` (as opposed to `X \LaTeX`), so I dropped it in favor of the classical `ltxdoc`. This turned out to be a not-so-good decision; `doc` is rather unwieldy with the four spaces before `\end{macrocode}`, etc. I would gladly return to `gmdoc` at some point in time.

3.2 One or more?

At first, I decided to have one class for each type of article (a regular paper, an obituary, a book review, etc.) and one for the whole issue. This turned out to be a bad decision. (I was so eager to try out the `docstrip`'s selective inclusion of various source file fragments in various resulting files that I apparently didn't think that through well enough ...) The benefits of this approach were infinitesimal (in fact, I can think of one only: compiling individual articles must have been faster by a fraction of a second), and the resulting complexity was very difficult to handle. In the “new” classes, I reduced the number of classes dramatically, and instead decided to select the article type with a class option, which works much better—especially since all the code specific to any type of article must be present in the whole-issue class anyway.

3.3 Packages we use

We rely heavily on a number of packages. Here's what they are with a short explanation/rationale for using each of them.

- `xparse`, which helps define commands with complicated syntax,
- `etoolbox`, which makes `\expandafter` and company almost unnecessary,
- `amsmath`, which is fairly obvious for the mathematical content. We use the `intlimits` option with it to preserve Polish typesetting tradition,
- `mathtools`, which I find essential (in fact, I don't know why it's not part of `amsmath`!),
- `pgfkeys` and `pgfopts`, which help define the class options,
- `amsrefs`, which is nicer to use for bibliographies than `BIB \TeX` or `BIB \LaTeX` : it lets us keep the bibliography in the same file as the rest of the paper, and more importantly, changing the look of the bibliography with it is very easy. We use

the `nobysame` and `initials` options, which for some strange reason are not the default,

- `MinionPro`, since this is the font we use,
- `polski`, since we typeset in Polish,
- `geometry`, which is a pretty obvious choice,
- `ifpdf`, so that the journal logo in eps format can be used when producing a dvi file (does anyone still use dvi, by the way?)
- `graphicx`, since we often include pictures, and `tikz`, since we often create them ourselves,
- `multicol`, since book reviews are typeset in two columns,
- `fancyhdr`, which (again, for some strange reason) is not included in L^AT_EX itself,
- `enumitem`, for obvious reasons (and more on that later),
- `booktabs`, for obvious reasons,
- `adorn`, since we want some decorations,
- `microtype`—very useful, especially that the pages are rather narrow,
- `upref`, for obvious reasons,
- `nicefrac`, which is occasionally useful,
- `url`—we need urls in bibliographies and sometimes elsewhere,
- `pdfpages` and `hyperref`, which are needed to prepare pdfs with separate articles for the website (they are made from the whole issue’s pdf).

As you can see, the list is quite impressive. This really shows that bare-bones L^AT_EX is not extremely useful without a host of packages; I feel that many of them should in fact be part of the L^AT_EX core.

3.4 Docstrip guards

As I mentioned, in the first iteration of the classes we used `docstrip` to generate separate classes for various article types. Currently, however, we have basically two classes: `wm-art` (for typesetting a single article) and `wm-issue` (for typesetting the whole issue). (There is one more, but we will cover that later.) We use `article` and `issue` docstrip guards to differentiate the code suitable for only one of those. Most of the code is shared between the two.

3.5 Class options

I decided to use `pgfopts` for options support. Even though we don’t actually use key/value type options, it’s nice to use `pgf` even for simple options which should just execute some piece of code when used. It turns out that option handling with `pgf` is pretty clever, and even though the learning curve is a bit steep, it is definitely worth the effort.

Each article type has its own option, setting a relevant Boolean switch. There are also other class options, like `proof` (which enables cropmarks), `ebook` (which sets the page geometry with minimal margins), etc.

3.6 Macros for typesetting the whole issue

We define two hooks named `\AtBeginArticle` and `\AtEndArticle`. In the case of a single article, they just fall back to `etoolbox`’s `\AfterEndPreamble` and `\AtEndDocument`; in the case of the whole issue, they add their argument to macros `\everybeginarticle` and `\everyendarticle` (using `etoolbox`’s `\appto`). Also, we define two macros named `\ArticleOnly` and `\IssueOnly`, each accepting one argument and expanding to that argument or nothing in respective situations.

Next, we define (only in `wm-issue`) commands like `\Year` and `\Volume` so that we can typeset these in each article’s header.

Now comes the fun part. We cannot assume that two articles will not have the exact same `\labels`, so we have to do something about this. We use `\let` to save the original meaning of `\label` and `\ref` and define our versions, using a prefix `\subjobname`. That prefix identifies the article (it is set in the `wm-issue` class to be the article filename sans extension). Also, we make sure that the references use lining (i.e., non-oldstyle) numerals. Of course, we also handle `\pageref`, `\eqref` and `\cite`, all in a similar manner. (The citations are slightly more difficult due to some `amsrefs` quirks.)

Since at the beginning of each article we want to typeset a header containing (among others) the page range for this article, we need the number of its *last* page. This amounts (more or less) to

```
\AtEndArticle{\origlabel{\subjobname:end}}
```

Again, in reality this is slightly more complicated, since the `\origlabel` must be put somewhere else for reviews (they are typeset in two columns).

Next up are the macros facilitating loading articles into the whole issue. This is a bit tricky, since each article has its own `\documentclass` and `\begin{document} ... \end{document}`. (Nowadays, we have things like `docmute` and `combine`, which help with such issues. When I was writing the classes for *Wiadomości Matematyczne*, they didn’t exist, or at least I didn’t know about them.)

First we define the macros `\wmdocumentclass`, `\wmdocument`, `\wmenddocument` and `\wmusepackage`, which will be substituted for the respective built-ins. The `\wmdocumentclass` must set up the article type, reset all the counters like `section`, `footnote`, etc., and make sure we start a new page. (The

new page thing again is quite tricky, since one of the quirks of *Wiadomości Matematyczne* is that an article can start on the same page as the previous one's end. We achieve this by setting a Boolean switch `newpage` to `true`, but only for those articles that actually need it.) Since we usually need `\DeclareMathOperator` (which is normally defined as a preamble-only command) in individual articles, we reset it to the previously remembered value. (Preamble-only commands work by being set to a special command telling the user that this can only be used in the preamble. To mitigate that, one needs to save the original command to something (with `\let`), then — after the preamble — `\let` the command back to what it was saved to.)

The commands `\wmdocument`, `\wmenddocument` and `\wmusepackage` are pretty simple. The first two just typeset everything saved with `\AtBeginArticle` and `\AtEndArticle` respectively; `\wmusepackage` is just a no-op (articles actually needing external packages are a very rare thing, and in such cases we `\usepackage` them in the preamble of the whole issue manually).

As mentioned, one serious complication arises from the fact that there are some “article groups” with individual articles *not* starting on a new page. We handle that by means of a `Group` environment and two Boolean switches, whose names are self-explanatory: `\ifingroup` and `\iffirstingroup`.

The next big thing is the table of contents. This is basically a big mess (just like in \LaTeX itself, although I redid it from scratch instead of trying to coerce the original to work my way). One reason is that we have to cater for the article groups. Another one is that we actually want *two* tables of contents: a Polish one and an English one. Of course, the title version in the ToC may be different than the one in the article and/or its running head. Yet another complication is connected with the so-called “vacats”. These are empty pages before an article (they arise naturally when each new article starts on the right, i.e., odd-numbered page, and when the previous article had an odd number of pages). A long-standing tradition in our journal is to put various things on those pages, ranging from pictures of mathematicians' monuments to conference posters to funny quotations. They appear in the ToC, at its end, under a “Miscellanea” section, with all the page numbers put together. Finally, since we may have a lot of UTF-8-only characters in the title, and we do not want `inputenc`'s macro expansions to get into the `.toc` file, we have to make sure that the `@title` macros and their like are expanded exactly once. All that means a pile of `\expandafters`, `\unexpanded`,

etc., which could probably be simplified a lot — but it ain't broken, so I'm rather hesitant to fix it. (I assume \LaTeX 3 might help *a lot* with these issues, but I was not brave enough to use it.)

3.7 Gathering metadata

In standard \LaTeX , you have the `\title`, `\author` and `\date` macros. This is far from enough in *Wiadomości Matematyczne*. While we actually do not need the date, we need a lot of additional stuff. For “regular” articles, we need the English title (and also we occasionally want to differentiate between the “normal” title, the title in the running head and the version in the ToC). Also, we have many other types of articles, such as obituaries (where we need the name of the late person, their date of birth and death, a picture and a scan of their signature), articles about prize laureates (which is more or less similar — without the dates, of course, and the signature, but with the prize name instead) and book reviews (with lots of data about the book itself, including a scan of the cover). Since we need a lot of similar `\title`-like macros, I decided to write a macro to write them for me:

```
\newcommand{\DefineDataGrabber}[1]{%
  \csdef{#1}##1{\csdef{@#1}{##1}}}
```

Now, issuing a command like `\DefineDataGrabber{title}` is more or less equivalent to `\def\title#1{\def@title{#1}}`, which is kind of cool (and quite Lispy, in fact).

Another interesting thing about gathering article metadata is the collection of author names. Obviously, we need support for more than one author, but how their data are typeset may differ in various places. We need at least four ways of doing that. At the beginning of each article, we just list the author names together with their *cities* (another tradition of the journal). If any author has a nonempty `\authornote`, we include a footnote mark here, too. Also on the first page we want to actually put the author footnotes. At the very end of each article we typeset their names, institutions and emails. Finally, we need the author names (in their “short” form) in the ToC.

The way we handle this is as follows. We define a command `\makeauthorlist`, which (when run) creates an auxiliary macro `\authorlist`, containing first an invocation of `\firstauthor` (with nine parameters corresponding to the author's data), then (if needed) subsequent invocations of an analogous macro `\nextauthor`. Then, `\makeauthorlist` is called `\AtBeginArticle`. When we want to typeset something for each author, we define `\firstauthor`

and `\nextauthor` to do what we need (e.g., typeset the names and the cities, or typeset the author footnotes, etc.) and run the `\authorlist` command.

3.8 Design and implementation

With the design, I figured that the hard part was the design itself. When that is done, the \LaTeX side of things is usually rather easy.

Well, I was wrong.

The first thing is the font choice. We typeset with *Minion Pro*, but we make our class available to authors, who don't necessarily have that font installed. Hence we check (using `\IfFileExists` for the existence of the file `MinionPro.sty`, and only use that font if this file is present.

We use oldstyle numerals in text, but references are typeset with lining numerals and sometimes their “tabular” version, so we need commands to turn them on when needed. Also, we use Polish-style inequalities (with slanted lines).

We redefine quite a few of \LaTeX 's skips, like `\baselineskip`. (For some reason, \LaTeX redefines the skips `\AtBeginDocument`, so we need to give them twice: once within that command and once without it, since we sometimes need to typeset some material *before* `\begin{document}`.) We redefine `\smallskip`, `\medskip` and `\bigskip` to be equal to a quarter, a half and a whole `\baselineskip`. We also define `\smallskipneg`, `\medskipneg` and `\bigskipneg` — just in case — and much more seldom used “stretch” and “shrink” variants (i.e., zero-length skips with possibility of stretching or shrinking the same amounts). Finally, we redefine skips around displayed equations to be much smaller than the defaults. Again, this must be done `\AtBeginDocument`.

Actually, vertical skips were one of the hardest parts of our classes. You may ask, what is difficult with that? Well, we try to typeset on a grid. This means that we take some care for many things to have the height equal to some multiple of `\baselineskip`. This includes section titles (moderately easy), theorems (rather easy), figures (difficult); when we have displayed equations or quotations, we drop the grid requirement on that page. Since many articles in *Wiadomości Matematyczne* do not contain a lot of math, this works quite well. Unfortunately, vertical skips, page layout and page breaking are one of the darker \TeX corners, and I have to admit that \TeX behavior in that regard is often a mystery to me.

The next thing in the design department is the page layout. This is accomplished with the help of the `geometry` package. A small complication arises from the fact that — depending on the class options — we want the options to be slightly different. It turned

out that a simple `\ifbool` *within the options* works well. (For clarity, I avoid plain \TeX 's `\ifs`, using `etoolbox`'s `\ifbool`'s instead.)

Each article has a special “header” on its first page. It contains the journal logo, journal name, volume and issue numbers and page numbers for the article. This is all simple: page numbers are there thanks to `\labels` at the beginning and end, the logo is in a pdf file (we use `\IfFileExists` so that the authors using our class who do not have that file can use the class anyway), and absolute positioning on the page is done by the wonderful `tikz` package.

The title and authors are a bit more work, but this work is not difficult, only tedious: we put all that stuff in a box of fixed height (equal to `24\baselineskip`) so that the grid won't be disturbed. One nice touch is that “author footnotes” are distinguished from the usual footnotes by the numbering system: they are “numbered” with Greek letters. Here we also make use of the `\authorlist` mentioned earlier. One thing worth noting is that we need to test if some data (like the author footnotes, for example) is undefined or empty; `etoolbox`'s `\ifdefvoid` is very useful for that.

For articles about prize laureates, new professors and book reviews we want to wrap an image with text. This is notoriously difficult in \TeX , but we went the easy way and decided that one paragraph will always be long enough to fill the space around the (rather small) picture. This way, we just set `\hangafter` and `\hangindent` to suitable values, and put the picture in a zero-height `\vbox` within `\vadjust` (taking care of vertical dimensions for everything). If the first paragraph is extremely short (which can happen from time to time), we have a simple remedy:

```
\newcommand{\fakepar}
{\leavevmode\nobreak\hfil\break}
```

(Notice the lack of `\indent`; this is intentional. Since the `\fakepar` needs to be inserted manually anyway, I wanted it to be general enough to support unindented paragraphs, like sections, etc.)

Finally, `\AtEndArticle` we typeset the author information (it looks slightly different among the various article types, but that is trivial to accomplish).

Some additional care needs to be taken when typesetting section and subsection titles. A section title takes up space equal to three lines (or more in case of long section titles, which we discourage and which seem to never happen anyway). We want `1.5\baselineskip` above and a half below. However, if a section begins at the top of a page, we do not want any vertical skips above it (and hence also below, because ... grid!). This is hard or impossible to achieve

automatically, so we have the macro `\attoppage`, setting a suitable switch to true. Also, the `\section` macro has to behave differently at the very beginning of the article: the skip above is then smaller (because we don't need it anyway). It is similar with subsections: normally, a subsection has one full `\baselineskip` above, but not if it directly follows a `\section`. All this is accomplished through special values of `\penalty`s and checks for `\lastpenalty`. (In the initial version of the classes, this mechanism was used much more often, e.g., with theorems; in the current iteration of the class, I decided that was too tricky and decided to go for a simpler solution, with a possibility of easy inserting manual skips, both positive and negative.)

Running headers is the next thing. No surprises here — we use `fancyhdr`, we set up the pagestyles for the first page of the article and for the rest of them, and we define macros `\theleftrunninghead` and `\therightrunninghead` so that the user can easily override them manually.

Theorem-like environments are a much more complicated business. For various reasons I was not satisfied with the L^AT_EX defaults. In my own papers I usually use `amsthm`, but for *Wiadomości Matematyczne* I decided to go my own way and do all the theorems from scratch. One reason is grid typesetting. A more important one is that by default, the *theorem*'s optional argument is typeset in a T_EX box, so its spaces are fixed. With rather short lines this tends to look ugly if the rest of the line happens to be very *loose* in T_EX's terms, i.e., its spaces are much wider than usual.

Since I did theorems from scratch anyway, I decided to do them my way. For starters, `\newtheorem` always creates a numbered and a non-numbered (starred) variant. Another thing is the handling of the optional argument. Oftentimes it consists only of a call to `\cite` or `\citelist`; in such a case, L^AT_EX puts the bracketed output of `\cite` in parentheses, which I don't like. In our case, if the optional argument begins with `\cite`, the parentheses are dropped. Again, this can be manually overridden by using the `\relax` command (which is a no-op, but is different than `\cite`, etc.). Since it is conceivable that someone might want to drop the parentheses for a different reason, we provide a `\noparen` command (a no-op again, but also recognized by the theorem environment, along with `\cite` and `\citelist`). Finally, we define a slew of theorem-like environments by default.

The next thing is enumerations, which are easy: I employ the great `enumitem` package. One unorthodox thing we do in *Wiadomości Matematyczne* is

the following. We strongly discourage more than one level of enumerations (and totally forbid more than two), and instead we use various item styles to distinguish between various semantics. For instance, if the items form a conjunction, they are marked with arabic numerals in parentheses; if they form an alternative, they are marked with lowercase Roman numerals with a dot afterwards, etc. I'm not sure whether anyone notices, but I like it that way.

The next topic is bibliographies. The code responsible for them is quite large, but this is mainly because we need to define a lot of bibliography types. As I mentioned, we use `amsrefs`, which I like a lot. One of the greatest things about this package is that defining a new bibliography style is so easy. One of less great things is that it messes internally with the catcode of the apostrophe, which conflicts with the usage of the `\'` macro. We need to resort to some `\xdef` hackery because of that when defining the `coauthor`. (We have a special kind of a bibliography in *Wiadomości Matematyczne*: a list of publications by one person. In such a case, the `author` field is not typeset, although we introduce a `coauthor` field, which is typeset at the very end in the form of “(coauthor: ...)” or “(coauthors: ...)”. The internal macro containing the `coauthor` field is called `\bib'coauthor`, with the apostrophe as a letter; on the other hand, we need the `\'` control symbol to typeset the Polish word for “coauthor”, “współautor”.) Finally, we define an environment `bibliography` with *two* optional arguments: the first one is the title of the bibliography (the default dependent on the article type) and the second being a *prefix*, so that we can have two bibliographies in one article, the first one with entries numbered [1], [2], etc., and the second one with entries numbered e.g. [A1], [A2], etc. (this is sometimes needed). Anyway, the main takeaway here is: `amsrefs` is very nice to use and quite hard to hack on.

The last *difficult* thing is inserting figures and tables. We do not use floats, since we prefer to have full manual control over the placement of “floating” material. Therefore we redefine the `figure` and `table` environments. While we are at it, we provide some machinery to control the captions: the `figure*` environment makes them unnumbered, and the figure prefix (like “Fig.”) empty by default, but redefinable through `\renewcommand`. We put the figure (with caption) into a `\vbox` of depth zero, measure its height, round it up to the nearest multiple of `\baselineskip` and repackage it into another `\vbox` of the computed size. This way we can retain grid typesetting. If the `figure` environment (or similar) starts in horizontal mode, we use `\adjust`.

The rest is, happily, much easier. We slightly modify the default design of footnotes, we define a custom `quotation` environment, a simple modification of equation numbers (we want them in tabular numerals, and we provide an `\eqrefr` macro for equation ranges, like “(1–3)”), we define some Polish-specific dashes, etc. One interesting thing is that we have `\emergencystretch` set to 1pt. This is very useful for narrow columns. We also define some very narrow horizontal skips (half of `\`, and its negative counterpart). We also have a few last-emergency macros for influencing the typesetting. These are `\manualshortenthispage` (expanding by default to `\enlargethispage{-1\baselineskip}`), `\manuallooser` (basically, setting `\looseness`) and the following two:

```
\newcommand{\manualfillpar}
  {\setlength{\parfillskip}{0pt}\par}}
\newcommand{\manualindentfillpar}
  {\setlength{\parfillskip}{\parindent}%
  \par}}
```

which allow us to fight very short or very long last lines of a paragraph.

3.9 Making pdf files for individual articles

After typesetting, printing and sending out the whole issue, when the dust settles, we need to prepare a pdf file of every article to put on the website and send to the authors. This is not as easy as typesetting every article separately, since we want the page numbers to be exactly like in the printed issue; also, that would not work in case of articles beginning in the middle of a page. Hence we use `pdfpages` to include the relevant pages from the pdf of the issue.

Our solution is as follows: when the issue is typeset with a special option, `generatefiles`, we invoke a special macro `\generatefile` for each article (using `\AtBeginArticle`). This macro takes the page numbers from the labels and writes out a file named `wm-11-2-333-444.tex`, where 11 stands for the volume number, 2 for the issue number and 333 and 444 for the begin and end pages of the article. This file is very short and consists of setting the pdf metadata and including the relevant pages from `wm-11-2.pdf`. (Also, it uses a special, very simple third class generated from the dtx file.)

There are two main difficulties here. One is that the second issue each year has page numbers resuming where the first one ends, so we need to take care of page number arithmetic. Another is that we don’t want to have e.g. ties in pdf title, so we redefine a few standard commands to generate their ASCII equivalents (most notably, both `~` and `\` expand to a space in this context). Last but not

least, we `\usepackage{hyperref}` so that the page numbers in the pdf file matches the ones printed on each page (instead of starting from 1).

4 Emacs editing functions

L^AT_EX classes and general workflow is one thing. Actually editing files is another. We use *Emacs* to get the full editing power available to humanity. Even though it has its flaws, it is an extremely flexible tool. Customizing *Emacs* to work better for *Wiadomości Matematyczne* is an ongoing effort; currently, I am writing functions to automatically generate article templates, help with filling them with some metadata and upload them to our Mercurial repository, all with minimal manual intervention.

In this section, I would like to briefly describe the *Emacs* tools I’ve made so far, which turn out to be quite general and possibly useful for others.

4.1 Automatic replacement of strings

The first thing is automatic replacement of strings. There are some things that just need to be changed to reasonable defaults. One of them is Polish diacritical signs. We clearly do not want our source code to be littered with things like `\.z\’o\l{~}w` instead of proper UTF-8 “zółw”. Another is dollar signs (single and double), which we want to be converted to `\(`, `\)`, `\[` and `\]`. Yet another is `\-`, which should be just deleted everywhere, or ties, which need to be inserted after one-letter words and in a few other places and deleted from math mode.

For this, I developed an *Emacs* command called `mrr-auto-replace`. It is configured by means of a list of lists. Each of these lists consists of a regex, optionally followed by a predicate (i.e., a function returning a Boolean value) and by one or more strings. This works as follows: *Emacs* walks through the entire file (buffer, to be more precise) looking for the given regexen, and if the predicate is satisfied in any of the places found, the part matching the regex is replaced by one of the strings given. The strings are cycled, so we can e.g. have the regex `\$\$` (matching two dollar signs) replaced alternately by `\[` and `\]`. The predicate option is useful for distinguishing between math and text modes; AUCT_EX (which is an *Emacs* package for interacting with T_EX and friends) has a function called `texmathp`, returning a true value if the cursor (“point”, in *Emacs*-speak) is in math mode.

It seems simple (and so it is — the source code for `mrr-auto-replace` is less than 20 lines), but it is extremely useful. It is usually one of the first things we run on any L^AT_EX file received.

4.2 Semi-automatic replacement of strings

Sometimes, however, we cannot trust a machine to do the right thing. For such cases, we have the `mrr-replace-mode` *Emacs minor mode* (i.e., something we can turn on or off in a given buffer). It is configured by means of another list (with a similar structure as previously, though a bit more elaborate here). This time, however, whenever we find an occurrence of any of the regexen from our list, we stop, highlight it and give the user a chance to *select* one of the possible replacements. The use-case should be obvious — one of the possibilities is changing things like `++` to one of `-`, `--` or `---`. Indeed, we have more than forty such replacement possibilities, and while going through the file using this utility is tedious, it is much less so than if we did that manually. Also, this was much more complicated to code; it takes up more than a hundred lines of code (and as of writing this, it also contains a few minor bugs).

Being a minor mode has the additional advantage that this is *non-modal* in a sense: after turning `mrr-replace-mode` on, only a few keys behave in a special way: `TAB` cycles through the possible replacements (including the original version), `RET` (*Enter* on modern keyboards) looks for the next place to replace, and `C-g` (*Control-g*) quits the mode. This way, if we decide that we have to make some edits other than the ones defined in our list of potential replacements, there is nothing to stop us without exiting the replacement mode.

4.3 Various small hacks

Apart from the two bigger things mentioned above, we have a few smaller tools. For instance, I noticed that if I insert a tie, I almost always delete any space at that point first. Hence I bound the tilde key to a custom *Emacs* command I wrote which does exactly this. Another thing I often do when editing files (as opposed to writing them from scratch) is inserting commas and dashes (the latter often in pairs!). Therefore, I modified the comma-inserting command so that if I type it when *after* a space, *Emacs* inserts it *before* that space anyway. It's very simple, but a very nice time-saver.

For dashes, I have something special. Since usually, when I insert a dash, I need to remove any punctuation in that place (usually a comma), I defined another command to do just that. But more often than not, I want to enclose a fragment of text in dashes. Therefore, when I first select some text and then invoke my command, two dashes are inserted around the selection (“region” in *Emacs* language). Yet another simple command (11 lines of code) making editing much nicer.

4.4 Plans for the future

This is of course not everything *Emacs* can do for us. I noticed that there are numerous repetitive activities I perform when working on articles for *Wiadomości Matematyczne*. One of them is sending emails to the rest of the editors with e.g. pdfs for proofreading. Since I use *Emacs* as my email client (obviously!), I plan to write a command to prepare such an email (with a template text and the pdf attached) automatically. Another one I plan to do one day is a command which would walk across all the articles we are working on and display a summary with each article's status (like “after converting to our class, but before proofreading” or “after sending to the author for proofreading/confirmation”). The possibilities are vast, *Emacs* Lisp is a nice language to work with — it is only a question of time to mold *Emacs* into a system customized to this particular journal's workflow.

5 Summary

As can be seen from this tale, working on a journal is a complicated (but rewarding!) business. From the \TeX nic standpoint, there seem to be a few general recurring themes here. The most important (at least for \TeX nicians) is that there is no point in trying to force \TeX to do everything automatically; it is much better to cover, say, 90% of cases automatically and have facilities for manual override for the remaining 10%. Also, when writing classes for a journal, good knowledge of \TeX is very useful. Expansion control and vertical mode are especially important. Moreover, it is usually a good idea to use packages instead of reinventing the wheel whenever possible. And when we have full control over whatever comes into the journal (i.e., we heavily edit all incoming files), redefining even basic \LaTeX macros and environments (like, say, the `document` or `figure` environments or the `\usepackage` macro) should not scare anyone away. On the other hand, to minimize the editing effort, it is probably a good idea not to mess around with \LaTeX guts. The results will be less appealing aesthetically, though, since authors often make horrible design decisions.

Two paths I *didn't* follow — and which seem worth trying — are using $\text{\LaTeX}3$ and restricting what authors can do. The former is obvious — I would expect $\text{\LaTeX}3$ to reduce the need for things like `etoolbox` or plain old `\expandafers` and friends. The latter might be useful, since some authors use \LaTeX in an extremely, shall we say, *creative* way (like numbering all footnotes manually or defining dozens of macros making the source file slightly shorter and totally unreadable, or using the very same $\text{\LaTeX}2.09$

style preamble with lots of unnecessary stuff and cargo-cult coding artifacts for all their documents, etc.). Since our authors are mathematicians, they are unfortunately accustomed to using L^AT_EX; if that were not the case, we might prefer *Markdown* or something similar to restrict the authors' freedom to break things.

From the point of view of an editor who works with text files written by someone else, the obvious takeaway is that you need to use a serious text editor. Which one of the two you choose may be less important: while Vim is difficult to beat in terms of using as few keystrokes as possible to achieve a given transformation of text (a sport known as *vimgolf*), *Emacs* shines in the flexibility/programmability department, and also offers a more comprehensive environment, such as email clients, shell buffers, and a time-tracking/organizational application (the famous *Org-mode*).

In any case, being a secretary of a journal and using L^AT_EX and *Emacs* is an ongoing adventure that I hope to last for at least another decade.

◇ Marcin Borkowski
Faculty of Mathematics
and Computer Science
Adam Mickiewicz University
ul. Umultowska 87
61-614 Poznań, Poland
mbork (at) amu dot edu dot pl
<http://mbork.pl>

Production notes

Karl Berry

This seems an opportune place to say a few words about *TUGboat* production. In general, our process is nothing like as regularized as that described by Marcin.

One immediate difference is that *TUGboat*, by its nature, has to handle articles using any T_EX engine. We use pdf(L^A)T_EX by default, which can handle the majority of articles, but it's typical and reasonable for an article about LuaT_EX to require LuaT_EX, etc.

So, we can't create an entire *TUGboat* issue in one run. Instead, each article is processed separately into its own PDF. We then concatenate the individual PDFs to make the full-issue PDF to be uploaded to our printer.

To do the concatenation, we've used a variety of tools, most commonly Ghostscript and `pdfjam` (ctan.org/pkg/pdfjam) of late. ConT_EXt and `pdftk` have also been useful. Different tools are needed as years go by and software and systems change (for no convincing reason).

The same tools can select PDF pages when splicing two articles together, that is, when one article ends and another begins on the same page. We try to avoid this, partly because of the extra production trouble, but pri-

marily because it is better for readers to find new articles starting on new pages. But content must dictate form, so we make it work out when it's needed. (Incidentally, another PDF check is for all fonts being embedded, using `pdffonts` from Xpdf, foolabs.com/xpdf/.)

The trickiest part of producing the whole issue as a concatenation is the page numbering. We have a control file which lists all the articles in the order in which they will appear, as well as the beginning page number for the issue. Then each article writes its beginning and ending (`\AtEndDocument`) page numbers into external files, where the next article can read them. The two tables of contents use the same external files, so as to ensure consistency of the page numbers.

Unfortunately, nothing comparable keeps titles and authors consistent among the tables of contents and articles. Partly this is due to inertia, partly because it would be hard to implement in full generality, and partly because sometimes there are intentional differences among the three places—forced line breaks, abbreviations, etc.

Back to issue production: the compilation of each article, and the overall process, is done with GNU Make, via a single included Makefile fragment which defines nearly all needed actions. The per-article Makefiles merely give the name of the file, the engine to use (if not `pdflatex`), etc.; the goal being, naturally, to eliminate redundancy wherever possible.

We use GNU Aspell (gnu.org/s/aspell) with some `sed` preprocessing to do spell checking: `aspell list \ --mode=tex --add-extra-dicts='pwd'/.dict.pws \ | sort -fu`. The idea being that a given article can have a `.dict.pws` file with the spelling exceptions needed that don't make sense to add to the global exception list (unusual proper names, one-off neologisms, etc.).

Besides spell checking, we've implemented several custom checks across an entire issue, again done in the central Makefile: doubled words (math.utah.edu/~beebe/software/file-tools.html#dw), lowercase letters inside `\acro`, tripled letters ("eee"), etc. More globally, we check that the tables of contents aren't missing an article processed in the central control file. Of course, besides the automated checks, humans review each and every word, line, and page that goes out.

Character encodings are an unending hassle. We receive many articles in UTF-8 these days, often with confusion or incorrect usage of accents, dashes, etc., or garbled in transmission. Other articles still use Latin-1 or similar. For articles which have only a few "special" characters, we strongly recommend taking advantage of T_EX's inherent capability, and sticking to 7-bit ASCII.

One final point is that all production work is done on Unix (CentOS 7 these days), using T_EX Live. Thanks to the well-known portability of T_EX documents, there is rarely a problem with an author obtaining different results than the production run, with one glaring exception: when fonts are found by X_YT_EX or LuaT_EX via system lookup, instead of by filename. This makes the document immediately and completely unportable—so I implore everyone, please don't do this in *TUGboat* articles!

Streszczenia

Slajdy i inne powiązane materiały dla wielu prezentacji znajdują się pod adresem <http://tug.org/tug2017>.

— * — —

Justować czy nie justować?

Leila Akhmadeeva, Boris Veytsman

W naszej prezentacji podczas konferencji TUG'16 porównaliśmy prędkość czytania oraz zrozumiałość tekstów justowanych, z zastosowaniem przenoszenia wyrazów, oraz niejustowanych, bez przenoszenia. Do porównań używane były dwa czynniki: czy teksty były justowane i czy stosowano przenoszenie wyrazów.

Na szczęście polecenia

```
\usepackage[none]{hyphenat}
```

oraz `\sloppy` pozwalają (kiepsko) justować bez przenoszenia wyrazów. W prezentacji przedstawimy nową konfigurację eksperymentów oraz ich wstępne wyniki.

Odpluskwanie plików \LaTeX owych – nie daj się draniom

Barbara Beeton

Każda użytkowniczka \LaTeX a przynajmniej raz w karierze napotkała dojmujący problem, gdy kompilacja zakończyła się z jakiegoś nieznanego powodu. Sposoby radzenia sobie z prostszymi problemami są dość dobrze znane, ale bywają sytuacje, gdy stare, sprawdzone metody zawodzą.

W tej prezentacji zostaną przedstawione strategie oraz taktyki radzenia sobie z wieloma rodzajami problemów, jakie ujawniły się w trakcie mojego wieloletniego udziału w zespole technicznego wsparcia AMS-u, odpowiadania na pytania zadawane przez autorów oraz przez pracowników redakcji. Przedstawione zostaną usterki typowe i nietypowe – dla każdego coś.

Czego może nauczyć \TeX ownika dziesięć lat pracy w redakcji

Marcin Borkowski

Od 2007 roku pracuję jako sekretarz redakcji „Wiadomości Matematycznych”, gdzie zajmuję się m.in. składem czasopisma przy użyciu \LaTeX a. To wystarczająco dużo czasu, aby nabrać przyzwyczajenia oraz móc poczynić pewne przewidywania. W referacie chciałbym podzielić się nimi z przyjaciółmi \TeX owymi.

Irytacje \TeX owe – co stoi na drodze do zupełnego otoczenia produkcyjnego

Paulo Ney de Souza

Istnieje kilka niewielkich (i denerwujących) kwestii, które stoją \TeX owi na drodze, aby mógł się on stać kompletnym środowiskiem produkcyjnym. Przyjrzymy się najważniejszym z nich i temu, jak niektóre z nich można pilnie potraktować, i zbadamy, co stoi na przeszkodzie do rozwiązania pozostałych kwestii.

Produkcje \TeX owe – ePub nowym celem

Paulo Ney de Souza

Najpierw formatem wynikowym \TeX a było DVI, potem nastąpiło przejście na PS i PDF, a teraz zbliża się wielka

zmiana: ePub. W trakcie prezentacji przeanalizujemy dotychczasową drogę i to, czego możemy się nauczyć z tego, jak działa Open Source'owy ekosystem \TeX a.

Warsztat intrologatorski: portfolio

Willi Egger

Powoli warsztaty kaligraficzne i intrologatorskie stają się \BachTeX ową tradycją. Na tegorocznych warsztatach będziemy robili portfolio, solidną teczkę do przenoszenia ważnych lub cennych dokumentów. Części przednia i tylna będą wyklejone papierem. Uczestników namawiam do przyniesienia własnego papieru, jeśli taki masz i chęć, aby był częścią ich dzieła. Taki papier nie powinien być ani zbyt cienki ani zbyt gruby. Będą Wam potrzebne 2 arkusze o wymiarach ok. 35 na 21 cm.

Con \TeX Text: kurs/warsztaty (dla początkujących Con \TeX Textualistów)

Willi Egger

Szczególnie dlatego, że tegoroczny \BachTeX jest wydarzeniem połączonym z TUG, zapraszamy wszystkich użytkowników \TeX a na wprowadzenie do Con \TeX Texta. Jak każdy system składu oferujący możliwość wykonania wirtualnie każdego przedsięwzięcia, Con \TeX Text jest olbrzymim systemem. Podczas warsztatu możemy tylko nieco uchylić zasłonę. Warsztat będzie sesją praktyczną, podczas której rozpoczniemy zabawiać się podstawowymi elementami składającymi się na dokument. W ostatniej jego części będzie można popracować nad małym przedsięwzięciem – jednostronicowym dokumentem zawierającym wszystkie elementy rachunku. Jestem zadowolony z możliwości poprowadzenia warsztatu i cieszę się na spotkanie z zainteresowanymi.

Nie na temat (całkowicie): Wiele twarzy (i gatunków) piwa

Michał Gasewicz

Zapraszam do świata smakowitego piwa! Czy jesteś gotowy poznać różnorodność i bogactwo najstarszego napoju alkoholowego na świecie? Przestań kojarzyć piwo z powszechnym, tanim, nieabsorbującym napojem niewymagającym refleksji. Powąchaj, posmakuj i baw się dobrze! Odważ się spróbować piw, których smak i aromat znacznie odbiegają od koncernowych jasnych lagerów. Będzie trochę piwowarstwa nowofalowego, reprezentującego obecne trendy, ale oczywiście tradycyjne polskie style również będą obecne.

Niestety, liczba uczestników musi być ograniczona do nie więcej niż dziesięciu. Loteria, przy większej liczbie chętnych ...

Dzieci \TeX a

Hans Hagen

Dla kogoś przywiązanego do składu matematyki naturalnym punktem wyjścia do tematu konferencji (przesłanki, przyzwyczajenia, przewidywania) wydaje się to, czym się zajmuje i czego używa użytkownik: matematyka. Ale można też poszukać odpowiedzi w aktualnych osiągnięciach nauk biologicznych oraz w historii. W trakcie prezentacji postaram się przedstawić te tematy z mniej

technicznej perspektywy. Mimo iż zapewne nie uda mi się udzielić poprawnych odpowiedzi na te pytania, to przynajmniej mam nadzieję zapoczątkować dyskusję.

Zmienne fonty

Hans Hagen

Pod koniec roku 2016 do standardu OpenType 1.8 została włączona idea fontów zmiennych. Jako że spodziewam się nacisków ze strony zapalonych użytkowników ConTeXta, aby zapewnić obsługę tej sztuczki, zdecydowałem się rozszerzyć mechanizm ładowania fontów o takie fonty. Oczywiście jest to eksperymentalne i takim pozostanie przez jakiś czas, bo po prostu na razie można testować zaledwie kilka (w pewnym sensie zaślepkowych) fontów. Miło jednak znów zobaczyć, że T_EX wciąż radzi sobie z nowościami.

Kolorowe fonty, aktualizacja i spojrzenie w przyszłość

Hans Hagen, Taco Hoekwater

Współprezenterzy: Lorien Otten, Lara Brandligt and Teun Otten.

Dzieci komunikują się za pomocą języka zwartego oraz obrazków, takich jak emotikony (emoji). Obrazki te często nie mają wielu detali, co jest zgodne z niedawnymi badaniami w Holandii, które wykazały, że dziecięce obrazki nie są bogate w detale. Polscy guru odpowiedzieli za wolne fonty lm oraz gyre nie pokwapili się do dostarczenia dzieciom ich ulubionych piktogramów, tak więc cel ten trzeba osiągnąć inaczej.

Spółeczność ConTeXta dysponuje tak zwanymi fontami Cowfonts, obecnie dostępnymi w wersji kolorowej. Po dziesięciu latach font „koeileters” doczekał się unowocześnienia. Nowa wersja używa technologii OpenType do połączenia istniejących czterech fontów PostScript Type1 w jeden font typu truetype.

Kontynuacją tego będzie zbiór emoji Duane’a Bibby’ego. Taco przekształcił (zgrubne) rysunki we właściwe kolorowe fonty obwiedniowe, Hans zapewni, aby to zadziałało w T_EXu, a (pierwszy) zestaw znaków zostanie wybrany przez naszych przyszłych użytkowników: dzieci.

W trakcie prezentacji Taco najpierw wprowadzi technologię (do czego jako przykładu użyje najnowszej wersji Cowfonts), potem Hans powie krótko o obsłudze fontów kolorowych w LuaT_EXu, a na koniec Lorien, Teun oraz Lara poproszą publiczność o ocenę, które z małych obrazków mają sens.

Historia znaków chromatycznych (akcydencji) w muzyce

Jean-Michel Huffle

[Patrz strona 147.]

MLBIBT_EX od teraz rozumie Unicode

Jean-Michel Huffle

[Patrz strona 245.]

DocVar: gospodarowanie zmiennymi dokumentu

Zumbeltz Izaola, Paulo Ney de Souza

Pakiet docvar pomaga w zarządzaniu zmiennymi dokumentu. Są to informacje o dokumencie (dotyczy w zasa-

dzie książek), które są wspólne dla zbiorów dokumentów (na przykład serii książek), chociaż różnią się w każdym konkretnym przypadku. Może to na przykład być: tytuł książki, nazwisko autora, podtytuł, ... Omawiany pakiet pomaga definiować nowe zmienne oraz ich używać. Wśród planowanych funkcji jest na przykład dziedziczenie wartości, gdy docvar nie zdefiniowano, i przekształcanie wartości zmiennej, gdy docvar użyto. Przedstawione zostaną podstawowe idee pakietu oraz powstająca implementacja.

Parametryczny font z symbolami matematycznymi

Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski

Zgodnie ze specyfikacją „Unicode Technical Report, #25. Unicode Support for Mathematics” (autorstwa B. Beeton, A. Freytaga i M. Sargenta III; <http://unicode.org/reports/tr25/>) font matematyczny „unikodowy” jest w istocie kompozytem wielu fontów: powinien zawierać pismo szeryfowe i bezszeryfowe (proste, pochylone i pogrubione), frakturę, pismo kaligraficzne, pismo tablicowe (blackboard bold), przy czym niektóre z pism powinny zawierać alfabet grecki; oczywiście „jądrem” fontu matematycznego jest zestaw znaków matematycznych: symboli, operatorów, nawiasów, symboli geometrycznych i in.

Taki „kompozytowy” font ma tę zaletę, że – dzięki temu, że spełnia wymogi standardu – może być używany w przez rozmaite programy działające pod różnymi systemami operacyjnymi. Wadą takiego rozwiązania jest to, że jego fontów składowych nie da się w łatwy sposób podmienić.

Użytkownicy T_EXa są w lepszej sytuacji – mogą za pomocą skryptu T_EXowego skonstruować font z wybranego zestawu, w szczególności mogą podmienić font podstawowy. Niestety, w niektórych przypadkach takie podejście zawodzi, gdyż okazuje się, że znaki z pozoru czysto geometryczne, takie jak strzałki czy nawiasy, mogą nie pasować optycznie do wybranego alfabetu podstawowego.

W trakcie naszej prezentacji chcielibyśmy przedstawić ideę generowania fontu sterowanego parametrycznie, zawierającego tylko niezbędne symbole matematyczne. Spodziewamy się, że dzięki parametryczności powinno być możliwe dopasowanie detali wizualnych symboli do wybranego zestawu fontów i że to dopasowanie – aczkolwiek niekoniecznie trywialne – nie powinno być nazbyt pracochłonne. Mamy nadzieję, że zestawianie w pełni funkcjonalnego fontu matematycznego z fontów składowych za pomocą skryptu T_EXowego okaże się zadaniem istotnie prostszym niż konstruowanie fontu matematycznego ogólnego zastosowania za pomocą edytora fontów.

Konfigurator dla T_EX Live

Siep Kroonenberg

Ponieważ T_EX Live można uruchamiać ze współdzielonego nośnika sieciowego, w trybie wyłącznie do odczytu, potrzeba włożyć trochę pracy, aby zaspokoić oczekiwania użytkowników systemu Windows: T_EXa należy włączyć do Start Menu, pliki T_EXowe powinny się uruchamiać

w edytorze L^AT_EXa na skutek dwukrotnego kliknięcia, i jeszcze użytkownicy nie powinni się matwić o ustawienie ścieżki wyszukiwania searchpath.

W wypadku instalacji osobistych zarówno T_EX Live, jak i MiK_TE_X dbają o sprawy podstawowe. Jeśli chodzi o instalację z sieci, powinno się dawać uzyskać konfigurację w Windows oddzielnie, dla użytkownika bądź stacji roboczej, bez powtarzania żmudnej procedury instalacyjnej.

Zajmuję się instalowaniem T_EX Live w Rijksuniversiteit Groningen, co obejmuje również kilka programów zewnętrznych. Na przestrzeni lat używałam różnych rozwiązań tego problemu.

Obecne rozwiązanie opiera się na programie uruchomieniowym, który zapewnia dostęp do programów zewnętrznych oraz do różnych komponentów samego T_EX Live. Na starcie czyta on swoje menu i przyciski z pewnego pliku ini.

W trakcie pierwszego uruchomienia program wykonuje niezbędną konfigurację Windows, również zdefiniowaną w pliku ini. Program oferuje różne sposoby usunięcia (zapomnienia) tej konfiguracji.

Zamierzam opowiedzieć o możliwych funkcjach systemu uruchomieniowego, o takim programie używanym na uniwersytecie w Groningen i o niektórych problemach, na jakie dotychczas się natknęłam.

T_EX w szkołach średnich – pomysł do podjęcia przez GUST

Anna Beata Kwiatkowska, Jerzy Ludwichowski

Przedstawimy pomysł Anny: GUST miałby dostarczyć na swojej witrynie zestaw materiałów T_EXowych takich, które mogłyby być używane przez uczniów szkół średnich.

Początkowo materiały miały być wykorzystywane przez uczniów Gimnazjum i Liceum Akademickiego, eksperymentalnego zespołu szkół średnich będących pod opieką Uniwersytetu Mikołaja Kopernika w Toruniu, jednego z najlepszych zespołów szkół średnich w Polsce. Anna naucza tam informatyki, jednocześnie będąc pracownikiem Wydziału Matematyki i Informatyki UMK.

Mamy nadzieję na wzbudzenie dyskusji o tym, jak podejść do takiej grupy docelowej.

Aktualne przedsięwzięcia fontowe e-foundry GUST

Jerzy Ludwichowski

Przedstawię krótko listę przedsięwzięć fontowych GUST-owego zespołu e-foundry

Składanie bibliografii w L^AT_EXu zgodnie ze standardem ISO 690

Dávid Lupták

Przygotowanie odwołań bibliograficznych i cytowań zgodnie z międzynarodowym standardem ISO 690 jest wymagane przez wiele instytucji i nie jest ograniczone tylko do czeskich i słowackich instytucji akademickich. Niestety, składanie zgodne z tym standardem nie jest wspierane przez system składania dokumentów L^AT_EX. Pakiet bibl_{at}ex-iso690 został przejrzany i ulepszony tak, aby

był całkowicie zgodny z wymaganiami standardu międzynarodowego, i znacznie upraszcza składania bibliografii wszystkich typów źródeł informacji. Prezentacja skupia się na nowej wersji pakietu bibl_{at}ex-iso960, który wprowadza możliwe przykłady użycia i w ten sposób również dostarcza skrócony ogląd standardu ISO 690.

Automatyczne budowanie binariów dla T_EX Live

Mojca Miklavec

Binaria T_EX Live są budowane raz na rok dla około 20 różnych platform przez kilku ochotników i w ciągu roku nigdy nie są aktualizowane. To dobry kompromis pomiędzy wymaganymi przez użytkowników w miarę aktualnymi binariami, stabilnością i obciążeniem wolontariuszy budujących binaria i przygotowujących paczki dystrybucyjne.

Z drugiej strony, społeczność ConTeXa jest mocno zależna od dostępności najnowszych binariów luaTeXa. Niekiedy, po zaimplementowaniu nowych własności, wymagane są najnowsze binaria X_YTeXa.

Zbudowaliśmy ostatnio infrastrukturę, które potrafi po każdej aktualizacji automatycznie zbudować binaria T_EXowe dla pewnej liczby platform, wysłać powiadomienia pocztą elektroniczną przy niepowodzeniach tego procesu, udostępnić raporty deweloperom i binaria użytkownikom.

Przedstawimy rozwiązanie umożliwiające nam swobodę znacznie częstszego przeprowadzania kompilacji, co z kolei umożliwia wczesną detekcję problemów i znacznie szybsze dostarczanie binariów użytkownikom.

Reguła jedyna, by złamać wszystkie

Mojca Miklavec, Arthur Reutenauer

Od prawie dziesięciu lat jesteśmy odpowiedzialni za repozytorium wzorców przenoszenia wyrazów w T_EXu, mamy do czynienia ze wszystkimi sprawami technicznymi oraz prawnymi związanymi z obsługą wzorców w pakietach makr oraz z włączaniem ich do dystrybucji. Jak dotąd jednak niewiele uwagi poświęciliśmy ogólnym zasadom dzielenia wyrazów w różnych obsługiwanych językach i to zagadnienie właśnie zamierzamy przedstawić i zwieńczyć rzeczą udzieleniem definitywnej odpowiedzi na wielkie pytanie: jedna reguła, by złamać je wszystkie.

Po drugiej stronie lustra – i co Alicja tam znalazła...

Frank Mittelbach

Dalsza podróż w poszukiwaniu automatycznego znajdowania optymalnego stronicowania dokumentów prowadzi nas do bajkowej krainy funkcji obiektowych, ograniczeń wywołań, wzorców układów graficznych i innych mistycznych stworów i Królowej, która krzyczy „Szybciej! Szybciej! Szybciej!” bo „na pozostanie w jednym miejscu zużyjesz cały dostępny Ci czas. Jeśli chcesz się jeszcze dostać gdzieś indziej, to musisz bieć przynajmniej dwa razy szybciej niż dotychczas!”

GM-Scenariusze po dwóch latach. Zupełny odjazd. Ale – czy zupełny w sensie Turinga? Czyli: Jak z ducha l3expan-owego począłem i zrodziłem potwora.

Grzegorz Murzynowski

Artykuł przedstawia aktualny stan mechanizmu GM-Scenariuszy, mini-esolangu jednoznakowych instrukcji, który pokrywa i rozszerza funkcjonalności pakietu l3expan, należącego do zestawu expl3.

W sensie (teorii) automatów skończonych, mechanizm GM-Scenariuszy wydaje się być deterministycznym automatem ze stosem (DPDA), zaś język przezeń akceptowany – językiem bezkontekstowym. Podaję argumenty, nie całkiem formalne, na poparcie tej tezy.

Zamieszczam diagram automatu, który, jak ufam, faktycznie zaimplementowałem, oraz gramatykę formalną języka GM-Scenariuszy w jego obecnym kształcie.

W uwagach końcowych odnoszę się do uwag, jakie otrzymałem po prezentacji GM-Scenariuszy na TUG@BachoT_EX 2017. W większości – w stylu „Przyjmuję do wiadomości i wykonania”.

Używanie Markdown wewnątrz dokumentów T_EXowych

Vít Novotný

Markdown to lekki język oznaczania tekstów, który ułatwia pisanie dokumentów o prostej strukturze w składni jasnej i bezpośredniej. Chociaż istnieją narzędzia do renderowania dokumentów w Markdownie za pośrednictwem T_EXa, to jednak zazwyczaj są one budowane z użyciem T_EXa jako podstawy, nie zaś w samym T_EXu.

Praca zwięźle przedstawia grupę istniejących narzędzi, a następnie wprowadza pakiet makr dla formatów T_EXa opartych na plainie, który działa na innej zasadzie. Możliwość umieszczania w dowolnym dokumencie T_EXowym kawałków zapisanych w Markdownie, jak też wykesponowanie makr T_EXowych do zarządzania renderowaniem elementów Markdownowych, to wygodna metoda włączania Markdownu do istniejących T_EXowych ścieżek przetwarzania.

fntutil and updmap – przeszłe i przyszłe zmiany (lub: sprzątanie bałaganu)

Norbert Preining

Dwa podstawowe programy pomocnicze w dowolnej instalacji T_EXa to: fntutil, odpowiedzialny za odbudowywanie formatów, i updmap, odpowiedzialny za tworzenie map fontów dla różnych programów.

Przez wiele lat w użyciu pozostawały zacne skrypty shellowe Thomasa Essera, dodające nowe funkcje i po hakiersku obchodzące ograniczenia. W ciągu ostatnich dwóch lat zostały one przepisane na Perla. Zmieniło to znacząco operacyjne zachowanie obu programów, mianowicie teraz czytane są wszystkie pliki konfiguracyjne, a nie tylko jeden.

Wraz z przygotowywaną wersją T_EX Live 2017 zamierzamy postąpić krok dalej i pozbyć się największego źródła zamieszania: mieszaniny skryptów -sys i nie-sys. Wielu osób przypadkowo – albo zwiedzionych wynikami wyszukiwania w Google’u – wywołuje updmap albo

fntutil bez części -sys, tworząc osobiste kopie plików konfiguracyjnych, co szybko prowadzi do chaosu.

T_EX Live 2017 przełączy się na inny układ: programu podstawowego (fntutil, updmap) nie można zawołać bezpośrednio, bo trzeba najpierw zdecydować, czy ma to nastąpić w trybie użytkownika, czy w trybie systemowym – przekazując tę informację albo w linii poleceń, albo w dedykowanym skrypcie.

W trakcie prezentacji przypomnę zakres funkcjonalny obecnych wersji programów fntutil i updmap, włącznie z wyjaśnieniem organizacji różnych plików konfiguracyjnych. W dalszej kolejności zapowiem planowane zmiany do TL2017 i na później. Na koniec przedstawię listę najlepszych praktyk, jakimi powinni się kierować użytkownicy, aby najbezpieczniej osiągnąć cel.

Uwolnione dźwięki

Maciej Rychły

[Patrz strona 118.]

Obrazowanie semantyki za pomocą subtelnej typografii i interpunkcji

Kumaran Sathasivam, S.K. Venkatesan, Yakov Chandy

Semantyka języka posiada głęboko zagnieżdżoną strukturę, którą typografia ujawnia przez używanie do składania hierarchii akapitów czcionek o różnej wielkości i stylu. Na poziomie akapitu, z powodu szczupłości środków typograficznych do ujawnienia semantyki używa się obficie interpunkcji. Akapity są dzielone na mniejsze jednostki semantyczne, takie jak zdania, z użyciem kropek na końcach i wielkich liter na początkach. Zdania ulegają dalszemu podziałowi na jeszcze mniejsze części, z użyciem średników, dwukropków, przecinków i myślników. Odstępy z kolei służą do dzielenia języka na najmniejsze atomy semantyczne, mianowicie słowa bądź frazy. W niniejszej pracy przyglądamy się nowym urządzeniom, zarówno typograficznym, jak i interpunkcyjnym, które usuwają niejednoznaczności i odkrywają głęboko zagnieżdżoną strukturę semantyczną.

MFLua 0.8

Luigi Scarso

Wersja MFLua 0.8 dodaje nowe odwołania i nowy podstawowy skrypt do uruchamiania kodu Lua z wnętrza MetaFonta. Wyczyszczone i uproszczone zostało jądro kodu. Praca przedstawia przykład wytworzenia prostego fontu OTF za pomocą zarówno MetaFonta, jak i FontForge’a w taki sam sposób jak mf2pt1.

Restauracja L^AT_EXowa

Przemysław Scherwentke

To recenzja książki „L^AT_EX, książka kucharska.” Pokazujemy, dlaczego jest to dobra książka kucharska. Zatrzymujemy się przy niektórych recepturach. Sugerujemy pewne zmiany wystroju.

TeX w szkołach? Oczywiście: przypadek użycia w Uniwersytecie Masaryka, Brno, Czechy

Petr Sojka, Vít Novotný

TeX jest używany w uczelniach, takich jak Uniwersytet Masaryka, do wielu celów: do pisania rozpraw doktorskich, esejów i publikacji przez studentów, uczenia publikowania elektronicznego, programowania piśmiennego, pisania prac naukowych, sprawdzianów, slajdów, jak też do generowania dokumentów i stron internetowych z uczelnianych baz danych przez system informacyjny uczelni i jej kadre.

TeX i związane z nim technologie są od ponad dwóch dekad systematycznie wspierane i udostępniane przez Wydział Informatyki. W tej pracy podsumowujemy wsparcie i projekty, które zrealizowaliśmy do tej pory, oceniamy wyniki, i omawiamy możliwe przyszłe wdrożenia technologii związanych z TeXem. Opierając się na danych z użycia fithesis3 na Uniwersytecie Masaryka podajemy argumenty, dlaczego odpowiedź na powracające pytanie z tytułu jest pozytywna, przynajmniej w takich uczelniach jak nasza.

Hackaton: Dokumentacja pakietów L^ATeXa

Damien Thiriet

Celem tego warsztatu jest ulepszenie dokumentacji pakietów L^ATeXa. Uczestnicy będą mogli poprawić lub rozszerzyć CTANową dokumentację swojego ulubionego pakietu (lub takiego, który uważają za szczególnie użyteczny).

Będziemy również zachęcali uczestników do przygotowania nowych wpisów w ramach projektu pakietomat.wordpress.com (w języku polskim). Mamy też nadzieję, że znajdą się chętni do wspólnego testowania, poprawiania i rozszerzania już opublikowanych wpisów.

Nawyki użytkowników TeXa a wymagania wydawców

Lolita Tolené

Składacze zawsze balansują na cienkiej linii między nieograniczoną kreatywnością autorów a ścisłymi wymaganiami wydawców, aby tworzyć pełnotekstowy XML. W trakcie tej prezentacji chcielibyśmy przedstawić obie strony. Konstrukcja TeXa sprawia, że do oczekiwanego celu można dotrzeć na wiele sposobów. To dlatego istnieje ogromny zbiór pakietów stworzonych na przestrzeni lat. A na co dzień są też używane makra lokalne. Przedstawimy, które pakiety TeXowe są powszechnie używane w pracach naukowych i jaka część z nich pochodzi ze standardowego źródła, jak CTAN lub TeX Live. Przyjrzymy się zwyczajom autorów używających TeXa do zapisu treści naukowych. Mając zaś na uwadze XML, przedyskutujemy, jak i dlaczego te zwyczaje są ważne dla składaczy w trakcie przygotowywania rękopisów do publikacji.

Xdvipsk: dvips gotowy na fonty OpenType i więcej typów graficznych

Sigitas Tolušis, Arūnas Povilaitis, Valentinas Kriaučiukas

Przedstawiamy dwa rozszerzenia programu dvips. Jedno z nich pozwala zrecznie włączać obrazy bitmapowe i zo-

stało zaimplementowane z użyciem biblioteki FreeImage. Drugie rozszerzenie rozwiązuje całkiem stare zadanie: dodaje do dvipsa obsługę fontów OpenType. Jeśli chodzi o zarządzanie fontami OpenType, to rozszerzony przez nas program dvips – xdvipsk – podąża drogą luatexową: pracuje na plikach DVI skompilowanych przez LuaTeXa i spodziewa się znaleźć niezbędne unikodowe pliki mapowań będące skutkami ubocznymi kompilacji. Istnienie plików mapowań jest zapewniane przez specjalny pakiet L^ATeXa.

Przykład humanistycznej książki naukowej

Andrzej Tomaszewski

Daaaawno temu w Bachotku na czarno-białych przezroczach miotanych z rzutnika slajdów prezentowałem książkę, która opuściła drukarnię równo przed dwudziestu laty. Na życzenie bachotkowej gawiedzi o projektowaniu naukowego wydania poematu Owidiusza „Haliutica” mądrym dla memoriału, idiotom dla nauki, politykom dla praktyki, melancholikom dla rozrywki opowiem de novo przy kolorowych pedeeefach.

[Editor's note: Zachęcamy do zwrócenia szczególnej uwagi na slajdy; zawierają one wiele pięknych ilustracji z fontami i układami stron z prezentowanej książki.

<http://www.gust.org.pl/bachotex/2017-pl/presentations/atomaszewski-1-2017.pdf>]

Zmienne i kolorowe czcionki OpenType: wyzwania i szanse

Adam Twardoch

W marcu 2015 roku opublikowano wersję 1.7 specyfikacji formatu fontów OpenType. Nowa edycja formatu wprowadziła wiele znaczących nowości: dodano specyfikację tablicy „MATH” stosowanej w składzie matematycznym, a także trzy formaty zapisu glifów wielokolorowych: „COLR/CPAL”, w którym tradycyjne monochromatyczne glify wektorowe nakładane są na siebie z użyciem palety zdefiniowanych w czcionce barw, „CBDT/CBLC”, w którym glify wielokolorowe opisywane są obrazami rastrowymi PNG, a także „SVG”, w którym do opisu wielokolorowych glifów stosowany jest pełen arsenał wektorowo-bitmapowego języka Scalable Vector Graphics, umożliwiającego stosowanie np. gradientów kolorystycznych, konturów i półprzezroczystości.

W przeciągu roku 2016 grupa robocza, składająca się z pięciu dużych firm (Apple, Adobe, Google, Microsoft i Monotype) oraz kilku zaproszonych ekspertów (John Hudson, Erik van Blokland, Adam Twardoch) pracowała nad kolejnym nowym aspektem cyfrowego formatu fontów, o nazwie OpenType Font Variations. Wyniki tych prac zostały przedstawione we wrześniu 2016 na konferencji ATyPl w Warszawie: nowa wersja 1.8 specyfikacji OpenType umożliwia opisywanie zmienności kształtu i układu glifów przy użyciu dwu mechanizmów: tablicy „gvar”, technicznie zgodnej ze stworzonym w 1993 roku przez Apple, lecz nigdy nie rozpowszechnionym, rozszerzeniem formatu fontów TrueType GX Variations, oraz tablicy „CFF2”, zastępującej wcześniejszą „CFF” opisującą kształt glifów postscriptowo. Dodano również tablicę

„sbix”, która – podobnie jak „CBDT” – umożliwia opis glifów wielokolorowych za pomocą obrazów rastrowych PNG.

W ciągu ostatnich dwóch lat OpenType znacznie się zmienił. Adam Twardoch przedstawi w swoim referacie najważniejsze zmiany w cyfrowym formacie fontów OpenType i w jego bliźniaczej normie międzynarodowej ISO/IEC 14496-22, a także odniesie się do wyzwań i szans związanych z adaptacją opentypowych nowości w świecie \TeX a.

STIX, Fira, Noto i przyjaciele: piękne nowe kroje pism na licencjach otwartych

Adam Twardoch

Ostatnie kilka lat obfitowało w premiery rodzin krojów pism, które powstały w profesjonalnych domach typograficznych jak Monotype, Tiro Typeworks czy Huerta Tipográfica, na zlecenie m.in. firm Google, Mozilla i Adobe. Fonty z tymi krojami wydane zostały na licencji otwartej SIL Open Font License, która w dziedzinie „wolnych fontów” stała się licencją de facto standardową. W przeszłości fonty opensource tworzyli zwykle lingwiści, hobbisci czy członkowie rozmaitych organizacji technicznych, natomiast w ostatnich kilku latach na rynek opensource trafiła wiele fontów wybitnych pod względem zarówno projektowym jak i technicznym, stworzonych przez zawodowych projektantów krojów pism.

Adam Twardoch dokona subiektywnego wyboru rodzin krojów pism, które nie tylko znakomicie wyglądają w składzie, ale jednocześnie zawierają ambitne repertuary znaków i można je bezpłatnie stosować w dowolnym zakresie. Autor w szczególności uwzględni zastosowania w typografii akademickiej i naukowej.

ZBOWID! Zespołowa Błyskawiczna Otwarta Wydawniczość Internetowej Dokumentacji

Adam Twardoch

Firma FontLab Ltd. od 2011 roku pracuje nad programem FontLab VI, przebudowanym od podstaw narzędziem do cyfrowego tworzenia krojów pism. Jednym z zadań Adama Twardocha jako dyrektora produktów tej firmy jest zarządzanie procesem tworzenia i publikowania kolosalnych rozmiarów dokumentacji technicznej, która będzie towarzyszyć gotowemu produktowi.

Przez dwa ostatnie lata Adam badał różne procesy, które umożliwiłyby zespołowi FontLab wspólne pisanie treści dokumentacji, śledzenie postępów prac, a także tworzenie ostatecznej publikacji HTML i PDF w łatwy w rozszerzaniu i zarządzaniu, przejrzysty i zautomatyzowany sposób. Ten proces powinien unikać narzędzi i formatów zamkniętych, powinien dawać pierwszeństwo formatom

i technikom otwartym, dobrze opisanym i wspieranym przez dużą, stabilną liczbę użytkowników. Proces powinien umożliwiać „integrację ciągłą” na komputerach lokalnych i serwerach. Osoba zarządzająca procesem powinna mieć możliwość ręcznej ingerencji w dowolny krok procesu.

Po wielu próbach Adam zbudował proces produkcyjny dokumentacji oparty na formacie źródłowym Markdown, na serwisie Github jako powłoce redakcyjnej i zarządczej, na pakiecie MkDocs, używającym implementacji Python Markdown do konwersji wikipodobnego zbioru plików Markdown na wielostronicowy serwis HTML, oraz na komercyjnym programie Prince XML tworzącym PDF. W tym czasie Adam napisał i udostępnił na otwartej licencji kilka narzędzi wspierających ten proces.

Referat będzie studium przypadku. Adam Twardoch przedstawi najważniejsze wymogi procesu oraz wady i zalety wieloformatowej, wielojęzycznej, wielonarzędzowej konfiguracji, którą stworzył. Autor opowie też, dlaczego nie skorzystał dotąd z \TeX a, zastanowi się, gdzie mógł być \TeX a użyć i zapyta czy może w jakiś sposób sięgnąć po \TeX a w przyszłości.

Przygotowanie pakietu ltxsparklines: Wycieczka CTANowca do świata CRAN.

Boris Veytsman

Wśród naukowców popularne stają się dynamiczne dokumenty tworzone przez R i \TeX a. Ta praca przedstawia wiążącą się z tym ścieżkę przetwarzania i sposoby wykrzesania czegoś więcej z tych dokumentów.

Sparklines to niewielkie rysunki (wysokie na około jedną linię tekstu), spopularyzowane przez Edwarda Tufte. Omówię mój pakiet ltxsparklines do R i lekcje wyńsione w trakcie jego pisania i dostarczania do CRAN.

Zamierzam też zwięźle porównać CTAN, CPAR oraz CRAN z punktu widzenia kontrybutora.

10 lat rozwoju fontów matematycznych OpenType

Ulrik Vieth

Rozwój fontów zawsze stanowił jeden z głównych tematów konferencji Bachotkowych. Zwieńczeniem było zbudowanie fontów Latin Modern i \TeX Gyre w formacie OpenType, których mogą używać silniki unikodowe, jak Lua \TeX i X \LaTeX .

Przez ostatnie 10 lat, od czasu pojawienia się standardu OpenType, prace ogniskowały się na fontach matematycznych, uzupełniających istniejące fonty tekstowe. W trakcie prezentacji omówię dotychczasowe osiągnięcia i co jeszcze zostaje do zrobienia.

TUG@BachTeX 2017 abstracts

Editor’s note: Slides and other related information for many of the talks are posted at <http://tug.org/tug2017>.

— * —

TeX annoyances — what is in the way to a full production environment

Paulo Ney de Souza

Several minor (and annoying) issues stand in the way of TeX to be a complete production environment. We will go over the most important ones, discuss how some of them should be addressed soon, and explore some directions on the way to solve the rest.

TeX Production — ePub, the new target

Paulo Ney de Souza

DVI was once the output of TeX, we have since moved to PS, PDF, and now on the verge of a big change — ePub. The talk will explore how we got here and what we can learn from the way the open source TeX echo system works.

ConTeXt: tutorial/workshop (for ConTeXt beginners)

Willi Egger

Especially since this year’s BachTeX was a joint event with TUG, we wanted to invite all TeX users to an introduction to ConTeXt. As with any typesetting system offering possibilities to handle virtually any project, ConTeXt is a huge system. During the workshop we can only lift the veil a little bit. The workshop will be a hands-on session in which we will start playing with basic elements to create a document. Towards the end of the workshop, there will be a chance to work on a small project — a single-sided document containing all the elements to build an invoice. I am glad to lead this workshop, and look forward to meeting everyone who is interested.

Colorful fonts, an update and peek into the future

Hans Hagen, Taco Hoekwater

Co-presenters: Lorien Otten, Lara Brandligt and Teun Otten.

Kids communicate in compact language and pictures like emoticons (emoji). These pictures are often also not that detailed, which suits recent studies in The Netherlands showing that drawings that kids make themselves become less detailed. The Polish font gurus responsible for the free lm and gyre fonts never got to providing kids their beloved pictograms so that goal has to be achieved differently.

The ConTeXt community has the so-called cow-fonts, now available as a color font. After ten years, the “koeileters” font is ready for an update. The new version uses OpenType technology to combine the existing four PostScript Type1 fonts into a single TrueType font.

A follow up on this will be an emoji set designed by Duane Bibby. Taco will convert the drawings (roughs) to proper color outline fonts, Hans will make sure they work well in TeX, while the (first) subset will be chosen by our future users: kids.

During this presentation Taco will first introduce the technology (for which he will use the latest cow fonts as an example), then Hans will quickly tell a bit about how the color font technology is supported in LuaTeX, and then Lorien, Teun and Lara will challenge the audience to tell them which little pictures make sense.

DocVar: Manage document variables

Zunbeltz Izaola, Paulo Ney de Souza

The package `docvar` helps to manage DOCUMENT VARIABLES. Those are pieces of information about a document (mostly books) that are common to a collection of documents (a book series), but different in each particular case. They may be, for example, the title of the book, the name of the author, the subtitle, . . . This package helps to define new variables and use them. Planned features include the inheritance of the value when a docvar is not defined and transformation of the variable value when the docvar is used. We present the main ideas of the package and its ongoing implementation.

TeX at secondary schools — an idea to be taken up by GUST

Anna Beata Kwiatkowska, Jerzy Ludwichowski

We will present an idea floated by Anna: GUST should provide on its web site a collection of TeX helper materials that could be used at secondary schools.

Initially the site would be targeted at the pupils studying at the Liceum i Gimnazium Akademickie, under the care of Nicholas Copernicus University of Toruń, one of the best secondary schools in Poland. Anna teaches there in computing and is also a staff member at the NCU’s Faculty of Mathematics and Computer Science.

We hope to spur a discussion on how to tackle such a specific group.

Automating binary building for T_EX Live*Mojca Miklavc*

T_EX Live binaries are built once per year for about 20 different platforms by a number of volunteers and never get updated during the year. This is a good compromise between users' demand for reasonably new binaries, stability and the burden on volunteer builders and packagers.

The ConT_EXt community on the other hand strongly depends on the availability of the latest LuaT_EX binaries at any given time. There are also occasional requests for the latest binaries of X_YT_EX when new features get implemented.

We have recently set up a build infrastructure that can automatically build T_EX binaries after every commit for a number of platforms, send emails when builds break, show reports and make the binaries available to users.

We will present our solution which gives us the freedom to run the builds much more frequently, to detect build problems earlier and to distribute newer binaries to users much faster.

One rule to break them all*Mojca Miklavc, Arthur Reutenauer*

For almost ten years we've been in charge of the repository of hyphenation patterns for T_EX, dealing with all technical and legal matters connected with their support by macro packages and their inclusion in distributions. Little consideration, however, has been so far given to the general principles of hyphenation for the different languages that are supported, and that's what we now would like to present, by finally giving the definitive answer to the great question: one rule to break them all.

Through The Looking Glass — and what Alice found there ...*Frank Mittelbach*

Continuing the quest for automatically finding optimal pagination of documents the journey takes us now to the fairy land of objective functions, call-out constraints, layout templates and other mystical creatures and a Queen that cries "Faster! Faster!" because "... it takes all the running YOU can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"

We will explore how fast we must run to enter that world.

L^AT_EX Restaurant*Przemysław Scherwentke*

This is a review of the book „L^AT_EX, książka kucharska.” (“L^AT_EX, a Cookbook.”) We show why this is a good cookbook. We look at some recipes. We suggest some changes in the decor.

Hackaton: Documenting L^AT_EX packages*Damien Thiriet*

The aim of this workshop is to revise, correct or extend the CTAN documentation of your favoured L^AT_EX package.

Participants could also prepare an entry for a L^AT_EX package for the `pakietomat.wordpress.com` project (in Polish). We would also appreciate collective testing, improving and extending of already existing entries.

An example of a humanist scholarly book*Andrzej Tomaszewski*

A long, long time ago, during a BachoT_EX, throwing black-and-white slides from a slide projector at the public I presented a book which left the printing house exactly 20 years ago. In reply to BachoT_EX goers' demand I'll again describe the design process for the scholarly edition of Ovid's poem "Haliutica", this time using color PDFs.

[Editor's note: We'd like to draw special attention to the slides from this talk, which include many beautiful images of fonts and layouts from the book. <http://www.gust.org.pl/bachotex/2017-p1/presentations/atomaszewski-1-2017.pdf>]

Variable and color OpenType fonts: chances and challenges*Adam Twardoch*

In March 2015, the OpenType font format specification version 1.7 was released. This was a major extension of the spec, which added support for the "MATH" table for mathematical typesetting, as well as support for three storage formats for multi-color glyphs: "COLR/CPAL" which combines pre-existing monochrome outline glyphs into multi-color glyphs, "CBDT/CBLC", which uses PNG bitmaps to store multi-color glyph images, and "SVG", which stores multi-color glyphs as a mixture of complex vector graphics (with gradients, strokes and transparencies) as well as bitmaps.

Throughout 2016, a working group consisting of five large companies (Apple, Adobe, Google, Microsoft and Monotype) and a few invited experts (John Hudson, Erik van Blokland, Adam Twardoch) worked on another major extension to the standardized font format: OpenType Font Variations. In

September 2016, the results were presented at the ATypeI Warsaw conference: OpenType version 1.8 added support for variable glyphs and metrics via the “gvar” table, backwards-compatible to the TrueType GX Variations extension which was introduced in 1993 by Apple but never gained any traction, and the “CFF2” table, which provides a similar mechanism for PostScript-flavored fonts and replaces the previous “CFF” table. The “sbix” table was also added, which uses PNG bitmaps for multi-color glyphs, much like the “CBDT” table.

Thus, in the last two years, OpenType has changed massively. In this talk, Adam Twardoch will present the new additions to the OpenType font format and its sibling ISO/IEC 14496-22 international standard, and will comment on both the chances and the challenges in introducing those changes into the \TeX world.

STIX, Fira, Noto and friends: beautiful new open source fonts

Adam Twardoch

In the last few years, Google, Mozilla and Adobe have worked with a number of professional font foundries large and small, including Monotype, Tiro Type-works, Huerta Tipográfica and others, to bring a flurry of high-quality, professional font families available under the SIL Open Font License, the de-facto standard license for open source fonts. While in the past, most open source font projects were created by technical organisations, non-designers, linguists and hobbyists, the 2010s saw the birth of quality type designs never previously seen in the open source realm.

In this talk, Adam will present a selection of his personal highlights of font families that not only look good, but are free for use on any project, and include ambitious character sets or typographic extensions, making them suitable for academic and scientific typesetting.

CORDIDA! Collaborative Opensource Rapid Digital Internet Documentation Authoring

Adam Twardoch

Since 2011, FontLab Ltd. has been working on a complete rewrite of the company’s main commercial software product, FontLab VI—a font editor for professional type designers. As FontLab’s Director

of Products, Adam Twardoch has been managing the authoring and publishing of the massive technical documentation that will accompany the released app.

Adam has spent the last two years researching various workflows that would allow the FontLab team to write content in a collaborative way, oversee the progress of the content creation, and have an automated, transparent, extensible and manageable process of creating the final documentation in HTML and PDF formats. The workflow should avoid proprietary formats and tools, and should favor well-documented formats and techniques that have a large and stable user base. The workflow should allow continuous integration on local machines and remote servers, and should allow the person responsible for the production to “intervene” at any point of the production process.

In the end, Adam has settled on a workflow that involves Markdown as the source format, Github as the authoring and collaborative front-end, the MkDocs package that uses the Python Markdown implementation to produce a multi-page HTML website from a set of wiki-like Markdown documents, as well as the commercial Prince XML engine to produce PDF. Adam has written and open-sourced a number of tools that help in the process.

In this case study talk, Adam will present his key requirements for the process and the strengths and weaknesses of a multi-format, multi-language, multi-tool setup that he has adopted. In particular, the author will show why he didn’t use \TeX so far, theorize on where he could have used portions of \TeX , and ask whether he still can use \TeX in some way.

10 years of OpenType math font development

Ulrik Vieth

Font development has always been a major topic at Bacho \TeX conferences, culminating in the development of Latin Modern and \TeX Gyre fonts in OpenType format for use with Unicode engines such as Lua \TeX and X \TeX .

In the past 10 years, ever since it was introduced, the focus has been on developing OpenType math fonts complementing existing text fonts. In this talk, we will review what has been achieved and what remains to be done.

Die T_EXnische Komödie 2/2017

Die T_EXnische Komödie is the journal of DANTE e.V., the German-language T_EX user group (dante.de). (Non-technical items are omitted.)

STEPHAN LUKASCZYK, Tagungsbericht Frühjahrstagung 2017 [The Annual Dante Spring Meeting in Zeuthen]; pp. 27–33

Report on DANTE's annual meeting in Zeuthen/Berlin.

THOMAS HILARIUS MEYER, L^AT_EX für Geisteswissenschaftler — Ein Projekt zur zielgruppenspezifischen Dokumentation von T_EX & Co. [L^AT_EX for humanists — a project for target-specific documentation of T_EX & Co.]; pp. 34–38

Since November 2016 a small group consisting of Lukas C. Bossert, Axel Kielhorn, Thomas H. Meyer, Craig Parker-Feldmann, Philipp Pilhofer, Christine Römer, Martin Sievers and Uwe Ziegenhagen has been working on an introductory text on L^AT_EX that is specifically targeted to humanists. In this article an overview of the project is given.

EBERHARD W. LISSE, Arztstempel mit L_YX/L^AT_EX erstellt [Creating medical stamps with L_YX/L^AT_EX]; pp. 39–47

As an obstetrician and gynecologist in private practice in Windhoek, Namibia, I am not required (unlike in Germany) to affix a stamp onto a prescription, but in particular for sick leave certificates, employers often demand this and therefore my patients ask for it. I conduct all my written correspondence with L_YX and have decided to develop a stamp in L^AT_EX and then tie it into L_YX, with the main obstacle being a lack of motivation on my part. As in Germany, such software stamps are permitted and the results can be adapted with little effort. This solution could be used in practices working with L_YX or L^AT_EX, possibly even in other professions required to use similar stamps.

THOMAS HILARIUS MEYER, Bibelstellen mit L^AT_EX verarbeiten und durch Register erschließen [Processing Bible passages]; pp. 48–60

L^AT_EX is deeply loved by STEM scientists, but humanists can profit from its features as well. One example is the convenient processing of Bible passages including the creation of an index.

GERD NEUGEBAUER, CTAN ist bei 2.0 angekommen — Neue Möglichkeiten beizutragen und mehr [CTAN 2.0 — New Features and ways to contribute]; pp. 60–72

[Published in *TUGboat* 38:1.]

HERBERT VOSS, Lucida-Schriften als OpenType von der TUG [OpenType Lucida fonts from TUG]; pp. 73–91

The Lucida font family is part of many operating systems, but supported here are only Lucida Bright, Lucida Sans and Lucida Sans Typewriter. In cooperation with Bigelow & Holmes, TUG has created a font bundle that provides a complete OpenType font family for the typesetting of text and math at a special price.

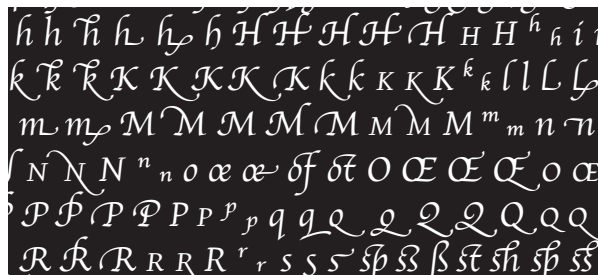
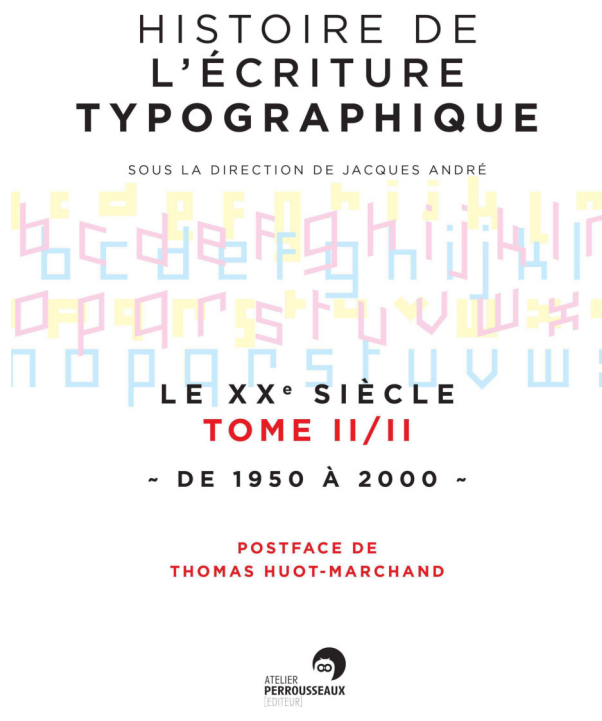
[Received from Herbert Voß.]

Review and summaries: *The History of Typographic Writing — The 20th century Volume 2* (ch. 1–5), from 1950 to 2000

Charles Bigelow

Histoire de l'Écriture Typographique — le XXI^{ème} siècle; tome II/II, de 1950 à 2000. Jacques André, editorial direction. Atelier Perrousseau, Gap, France, 2016, ISBN 978-2-36765-006-7, <http://tinyurl.com/ja-xxieme-ii>. 364 pp., 391 figures (illustrations, photos, diagrams, etc.), illustrated end papers. Also available as an ebook. The book is in French. Volume 1 (reviewed in *TUGboat* 38:1) covers the years 1900 to 1950.

Occasional commentary below by the reviewer is placed in square brackets; the main text summarizes the original writing.



End paper (fragment): Apple Chancery typeface and typography both by Kris Holmes.

Charles Bigelow

Jacques André: Introduction

This is the last volume in the series created by Yves Perrousseau, on the “History of Typographic Writing” from its beginning to the end of the 20th century.

In the 20th century, the powers of social and informational functions of writing, previously distinguished in part by their modes of production — for example, public inscriptions and signage, book and news publishing, and personal handwriting — were expanded by technological advances. Commercial, governmental, political, and educational institutions used typographic media to ever-greater extent and effect, although individual expression remained, for a time, limited to handwriting and typewriting. By the end of the century, however, new technologies of typography vastly enhanced the power, extent, and graphical range of personal written expression.

This second volume of the history of 20th century typography is intended for general readers interested in the history, art, and technology of the century, as well for specialists and students in the field. It has been written by ten different authors and thus reflects as many different perspectives and styles. In addition to text and copious illustrations, it includes an extensive bibliography.

1. Alice Savoie: Typography transformed: the era of photocomposition (*La typographie en pleine mutation: l'ère de la photocompositions*)

“Photocomposition before 1945: false starts and early experiments.”

In the early decades of the 20th century, several inventions applied photography to type setting. Despite clever mechanisms and novel names, the Bawtree, Photoline, Rotofoto, Thothomic, and Uher-type proto-phototypesetters proved less efficient, less economical, and lower in quality than established hot-metal composing machines and hence failed to become commercially successful. This first phase of photocomposition was followed by the so-called “first generation” photocomposers — the Intertype Fotosetter and the Monophoto, which adapted hot-metal machines by replacing the casting unit with a photo unit. These machines produced commercially adequate output, but were not widely used.

“Second generation” photo-electronic systems, especially the pioneering Lumitype invented in France in the 1940s by Moyroud and Higonnet but developed in the U.S. as the Photon (sold in France as the Lumitype), revolutionized text composition in the 1960s and 1970s. Third generation phototypesetters were based on cathode ray tube (CRT) imaging and computer control, and fourth-generation machines were based on laser imaging.

The phototypesetting revolution was not merely technical but also social. Fast typing abilities on QWERTY keyboards (AZERTY in France) coupled with quick learning of computer mark-up codes and commands replaced the mechanical skills learned from long apprenticeship in hot-metal type technology. “Photocomposition enabled the type-compositor to trade the blue collar laborer’s shirt and noisy, heavy machines, for the white collar office shirt and precision knives and photochemical processes.”

[CB: Thus began a trend toward higher education and social mobility for typographers, women and men, reflected academically, first in the awarding of Bachelor’s, then Master’s, and most recently, Ph.D. degrees in typography, supplanting the exclusively masculine apprenticeships of older generations of typographers.]

2. Alice Savoie: The creation of new typefaces for photocomposition (*Concevoir de nouveaux caractères pour la photocomposition*)

The designs of Adrian Frutiger and Ladislav Mandel.

Phototypesetting machines transformed not only the process of composing texts but also the process of making type. Type fonts ceased to be miniature metal sculptures and instead became abstract photographic images, requiring new techniques and often, new designers.

In 1953, Charles Peignot, director of the Deberny & Peignot foundry in Paris, hired a young Swiss designer, Adrian Frutiger, and assembled a team that included Ladislav Mandel and Lucette Girard, to produce high-quality photo fonts for the Lumitype photo-typesetter. The team first adapted popular metal faces like Garamond, Baskerville, and Times Roman to the strictures and distortions of high-speed optical imaging, but then Frutiger persuaded Peignot to support development of a totally new family of sans-serif types based on Frutiger’s student studies at the Zurich School of Arts and Crafts [where he was taught by Walter Käch and Alfred Willima].

The result in 1957 was the astonishing Univers family. In the metal type era, extensive font families like those of P.-S. Fournier and M.F. Benton had been cut incrementally in various sizes and styles over years or decades, but Univers burst forth from Deberny & Peignot all at once in 21 variations of weight, width, roman and italic, and all photographically scalable to many sizes. Typography would never be the same again.

[Univers was enthusiastically embraced by modernist graphic designers and over ensuing decades, its basic concepts were adopted by later generations

of type designers. There is hardly a new family of sans-serif types today that does not owe a debt to Univers, whether overt or unacknowledged.]

As phototype achieved commercial success in the 1960s and 1970s, more firms commissioned and developed original typefaces for photocomposition. At Monotype, John Dreyfus commissioned new photo text faces by Frutiger, Jose Mendoza, and Chris Brand. At Linotype, Mike Parker commissioned new script faces by Matthew Carter and Hermann Zapf, as well as new types for Arabic, Hindi, Hebrew, Greek, and other non-Latin alphabets.

Foreseeing typeface piracy in the photo era, Charles Peignot, with Stanley Morison, Jan van Krimpen, Hermann Zapf, and others, formed the International Typographic Association (l’Association Typographique Internationale, ATypI) to promote intellectual and artistic property protection for typeface designs. Several American photocomposing machine manufacturers prospered by developing cheaper and faster machines but plagiarizing typefaces, relying on lack of American copyright for type designs [still the case] as well as weak or absent protections in other countries.

Beginning in the 1970s, the International Typeface Corporation commissioned new types and modernized versions of traditional types for photocomposition. New ITC types by designers Ed Benguiat, Hermann Zapf, and others were licensed by many photo and digital composing machine manufacturers and found wide popularity, especially in advertising and display typography.

Christian Laucou: First interlude: Classification of typefaces and cataloging of fonts (*Première pause: classification des caractères et catalogage des fontes*)

As typeface variations multiplied, type classification became a perennially fascinating intellectual exercise. Classification systems were proposed by, among others: Thibaudeau in 1921; Audin in 1929; Duvillé in 1931; Tschichold in 1951; Vox in 1952; Turner, Berry & Johnson in 1953; and the German DIN standard in 1962. Most of these shared, to varying degrees, a small set of core classes denoting text typefaces of historical eras, supplemented by stylistic variations mainly produced in the 19th century. Differences between classification systems were partly due to lumping or splitting of a few classes, like the gothic scripts, the numerous sans-serifs, and multitudinous “fantasy” display faces.

The Vox classification was adopted by ATypI in 1962 and remains widely used and useful, but new classifications continued to be proposed, in part

because increasing multiplicity of type forms rendered older classifications incomplete, and partly because perceived flaws in the logic or concepts of previous systems spurred new efforts. Bringhurst, in 1992 and later, utilized art historical nomenclature as well as biological taxonomy to articulate aesthetic-conceptual relationships of type forms. In commercial type sales, marketing and advertising, categories based on usage, context, and emotion have appeared in type catalogs, specimens, and web sites. Classification of non-Latin typefaces, such as Chinese or Arabic, posed additional difficulties because of cultural and historical distinctions not always shared with Latin typography.

[In the classification systems cited above, the number of different classes ranges between 5 and 22, with average and median both around 10. Because of the vast proliferation of type forms in the digital era and type usage by billions of computer and smart phone users, type classification has become a nexus of modern Internet culture, inviting further analyses of font features and classes, whether logical, semantic, or pragmatic.]

3: Jacques André: Office typography: typewriters, printers, and “strike-on” fonts (*Vers la typographie de bureau: machine à écrire, imprimantes et caractères à impact*)

Following their invention in the 19th century, typewriters proliferated in the 20th century. Keyboard layouts varied by manufacturer until standardization of a few layouts according to country or language, like QWERTY in the U.S. and AZERTY in France. For ease of use and mechanical simplicity, typewriter typography was graphically simplified. Most typewriters had monospaced fonts and a single type size. Only a few sizes were available.¹

When a key was struck, a character image on a moving type bar impacted an ink-impregnated ribbon and squashed the character image onto paper.

Because of wear on type from the very high number of repeated impacts and coarsening of letter images from ribbon squash, typewriter typefaces were usually monoline and based on sturdy designs, particularly slab-serif faces. Typewriters became so

¹ [CB: Also called “fixed-pitch” or “fixed-width” fonts, as used in this footnote. All characters have identical widths, for example, the letter ‘i’ in its space has the same set width as the letter ‘m’ in its space. In the U.S., the most popular standard size was traditional English “pica”, with a height of six lines per inch when single spaced vertically, and 10 characters per inch horizontally. The smaller “elite” size set at 12 characters per inch but usually at the vertical “pica” line spacing.]

popular that traditional type foundries created printing typefaces to imitate the typewritten look. The popular Courier, designed for IBM electric typewriters at IBM in 1956 by Howard Kettler, was based on geometric slab-serif printing types. Sans-serif, italic, and all-capital typewriter faces were also produced.

A deficiency of the typewriter was that it produced “one-off” documents that were not easily reproducible. A few carbon paper copies of lesser quality than the original could be made while typing, but mimeography, offset lithography, and photocopying were used to reproduce typewritten documents in greater quantities.

The ubiquity of the typewriter, its conceptual simplicity, its standardized keyboard, and its vast number of users led to adoption of typewriter-like input for other systems including Telex, Teletype, Varityper, and Justewriter, as well as computer input using paper tape perforated by keyboard typing. Computer output also predictably produced typewriter-like printing. When CRT monitors and keyboards began to be used for computer input, the dot-matrix characters displayed on screens resembled, more or less, monospaced and monoline typewriter fonts. Thus, the typewriter became one of the earliest, longest enduring, and most important paradigms in human-computer interaction.

Adoption of typewriter-like computer input also spurred standardizations of the numerical computer codes corresponding to letters and characters, resulting in ASCII (American Standard Code for Information Interchange), European ISO Latin, and IBM EBCDIC character encodings. Stringent technical limitations and typographic simplicity did not, however, totally suppress artistic ingenuity. Typewriter and “ASCII art”, made with monospaced typewriter or computerized typewriter-like characters, included a plethora of often playful and ingenious images and patterns.

Christian Laucou: Second interlude: Games with letters (*Deuxième pause: Jouons avec les lettres*)

In the Latin, Hebrew, and Arabic writing traditions [Chinese and Japanese could be included], scribes often played with the arrangement and shaping of letters to make pictorial, ornamental, or scholarly arrangements of text. This tendency continued into European typography with the Hypnerotomachia Poliphili printed by Aldus (1499), an edition of Calimachus by Henri II Estienne (1577), and the polyglot Bible by Christophe Plantin (1572). Renditions of pictorial typography include the mouse’s tail

CRT monitors, which displayed simple dot-matrix characters, began to be widely used for computer data input and programming. When this screen technology was adopted for broad public usage in the French Minitel system in conjunction with the telephone service, tens of millions of customers began to read dot-matrix characters on screens.

The limitations of low resolution digital letter imaging prompted some designers, such as Wim Crowel, to devise rectilinear and polygonal letterforms adapted to the restrictions of then-current computer technology, but these novel experiments were soon supplanted by more traditional-looking letter forms as digital resolutions increased.

In the 1960s, in the fields of computer-aided design and manufacturing, there was pioneering research and development of mathematical descriptions and renderings of curves for computer graphics. In France, Pierre Bézier at Renault and Paul de Casteljaou at Citroën adopted cubic splines for the description and rendering of curved lines and surfaces.

The decade of the 1970s was rich in exploration of digital letter forms. Peter Karow at the URW firm in Hamburg developed the Ikarus digital type system, which encoded contours of letters with cubic splines that could be output to computer plotters to cut photo-masks for photo-optical typesetters, and could be software scan-converted to rasters, run-lengths, and bitmaps for different kinds of digital typesetting equipment. Also in the 1970s, Philippe Coueignoux at MIT and Patrick Baudelaire at Xerox PARC independently used mathematical curves and splines to define letter contours for typography. At Stanford University, Donald Knuth developed his Metafont system for font creation and digitization, using cubic splines. Also in the 1970s and early 1980s, a few digital typesetting machines, especially for newspapers, used outline formats — some based on straight-line vectors and others on circular arcs — optimized for fast output.

Karow's Ikarus system gained commercial success among digital typesetting manufacturers and font developers. Moreover, URW itself digitized hundreds of very high resolution fonts in the Ikarus spline format, and those, along with fonts from manufacturers using Ikarus, became the basis for a substantial subset of the PostScript and TrueType fonts produced in the 1980s and 1990s by Adobe, Apple, and Microsoft for personal computers and laser printers.

The Xerox corporation played a major role in the development of digital typography in the 1970s. At the Xerox Palo Alto Research Center (PARC), bitmap fonts were developed for the screens of personal computers — the “Altos” — to display approx-

imations of some traditional typefaces, and spline-defined letterforms were developed by Patrick Baudelaire. The xerographic laser printer was invented at Xerox by Gary Starkweather in 1969 and was commercially developed for high-speed xerographic printing systems by 1977.

In the mid-1980s, Xerox's innovations were imitated and popularized in products like the Apple Macintosh computer and LaserWriter printer. Xerox had also developed software for computer interchange and output of type and pages on laser printers. Adobe Systems, founded by alumni of Xerox PARC, developed the PostScript page description language, which used outline fonts of cubic Bézier curves as part of a general imaging model, to solve the problem of device-independent page interchange and rendering. The first commercial PostScript printer was the Apple LaserWriter launched in 1985.

The spline-defined font outlines of Ikarus, PostScript, and similar systems had several advantages, including: economy in computer file size and memory utilization, scalability to arbitrary sizes, ease of rotation and modification. The raster scan-conversion of abstract mathematical outlines to arrays of discrete pixels on monitor screens or page bitmaps of laser printers raised difficult technical and aesthetic issues at low resolutions. Technical issues involved tracing pixels along the edges of characters and filling the edge-defined shapes, with the goals of increasing computational speed and efficiency. [These were mainly solved by improved rendering algorithms as well as by the increases in computing speed and memory described by Moore's Law.]

The aesthetic problems, however, proved more difficult because they involved aspects of human vision, mechanisms of reading, and expectations of the appearance of text, all less amenable to algorithmic analysis and hardware advances. At the laser printer resolutions of the 1980s, all below 600 dots per inch, simple scan conversion produced letterforms in which irregularities of stem weights, horizontal alignments, letter spacings, and traditional detailing produced texts that failed to conform to reader expectations. The outputs were accordingly judged inferior, and there was a scramble to ameliorate perceived type quality. Karow was the first to address this problem; in the late 1970s and early 1980s, the Ikarus system used software distortion of master outlines to conform to digital grids before scan-conversion. This was done off-line to produce bitmap fonts.

To its PostScript Type 1 fonts, Adobe added data to mark stems, curved bowls, vertical alignments, and other features, and those data were used to locally distort the outlines of characters prior to

rasterization in order to impose greater regularity when the characters were rasterized for the digital grids of printers. Adobe termed these declarative data “hints” but kept their implementations as trade secrets. Adobe’s advance over Ikarus was that PostScript hints were applied on-the-fly during rasterization in the printer, instead of off-line to produce fonts in bitmap and raster formats.

The success of PostScript and its fonts engendered competitors, of which the most successful was TrueType, invented at Apple and later licensed to Microsoft. TrueType used quadratic B-splines instead of cubic Bézier splines, and procedural instructions for fitting outline shapes to raster grids.

[The concept of procedural hinting had previously been developed in the late 1980s by the Folio corporation for its F3 font technology and disclosed to Apple early in the design of TrueType. Sun Microsystems acquired the Folio F3 technology but did not strive to promote it as a standard in competition to PostScript or, later, TrueType.]

In 1989, Microsoft licensed TrueType technology for its Windows operating system, igniting a years-long commercial battle popularly known as the “Font Wars”, in which the combatants made rival claims of technical and artistic superiority for their font technologies. A partial cease-fire in the Font Wars came in the 1990s when former combatants Microsoft and Adobe agreed on an expanded format named OpenType, in which character outlines could be implemented in either PostScript or TrueType form, and which included data to support alternative and context-sensitive forms and glyphs required in certain non-Latin writing systems like Arabic and the Indic scripts. OpenType, however, was promoted by the Adobe-Microsoft pair against a similar, earlier font technology, TrueType GX that had been previously released by Apple, so the font wars were not entirely over with the announcement of OpenType.

Between 1985 to 2000, some of the aesthetic problems of digital type were ameliorated in two ways. First, for computer screens, the algorithmic adjustment of pixel intensities along character edges, called “gray-scaling” or “anti-aliasing” reduced the perceptibility of jagged pixels along curves and diagonals. [This depended on the pixel resolutions of screens. At resolutions below (approximately) 120 pixels per inch, gray scaled edges looked smoother but blurrier and were not as acceptable as manufacturers hoped. Screen resolutions above 220 and 300 pixels per inch after the year 2000 effectively resolved the problem of jaggedness and irregularity of text on screen, obviating the need for hints.]

Second, for printers, doubling of resolutions from 300 to 600 dots per inch reduced the more egregious irregularities in text rendering, while techniques for decreasing intensity of laser beams along character edges to reduce apparent jaggedness of curves and diagonals, similar to anti-aliasing in which spot size was analogous to screen gray-scaling) made hinting less necessary or unnecessary. [Limitations on electrostatic printing limit the effective resolutions that can be achieved for mass-market devices.]

As computerized typography and document layout advanced, leaders in the computer document industry faced the problem of exchanging electronic documents across networks, computers, and devices, which required standardization of computer character encodings beyond the American ASCII and European ISO Latin standards. Begun by Xerox in the 1980s and supported by Apple, Microsoft, and other firms later in that decade or in the 1990s, a 16-bit character encoding standard called “Unicode” was developed with the goal of eventually encompassing all the world’s written languages. A similar encoding project was begun in Europe as the ISO-10646 standard. These parallel projects were merged in the early 1990s as the Unicode standard. Among many other benefits, Unicode brought computer character standardization to many of the non-Latin and non-European orthographies and writing systems that had encountered obstacles to efficient computerization, thus spurring development of computer-aided document production and distribution.

[CB: Because of the length needed for the above review of the information-packed Chapter 5 on digital fonts, the remaining chapters of the book will be covered in the third and final part of this review. For reference, the remaining titles and authors are:

- “The first commercial digital fonts”, by Frank Adebaiye;
- “Interlude: On the revival of typefaces”, by Franck Jalleau;
- “Everyday working fonts from 1985 to 2000”, by Olivier Jean;
- “Hybridization, (de-)construction, and quotation in typography from 1985 to 2000”, by Hervé Aracil;
- “Interlude: On the preservation of typographic heritage”, by Alan Marshall; and
- “Postface — the metamorphosis of typography”, by Thomas Huot-Marchand.

Ending with an extensive bibliography and index.]

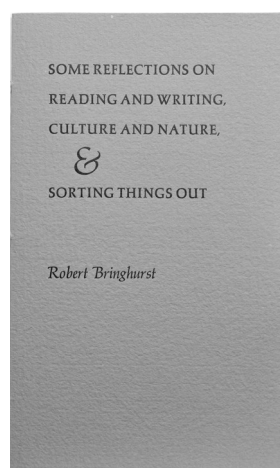
◇ Charles Bigelow
<http://lucidafonts.com>

**Book reviews: *What Is Reading For?*
and *Some Reflections on Reading and Writing,
Culture and Nature, & Sorting Things Out*
by Robert Bringhurst**

Boris Veytsman

Robert Bringhurst, *What Is Reading For?* RIT Cary Graphic Arts Press; Rochester, 2011, Softcover, 40pp. US\$29.95. ISBN 978-1-933360-53-9.

Robert Bringhurst, *Some Reflections on Reading and Writing, Culture and Nature, & Sorting Things Out.* RIT Cary Graphic Arts Press; Rochester, 2016, Hand sewn softcover, 12pp. US\$20.00.



Typography is an unusual art. While, like other arts, it appeals to our senses and emotions, it is also intimately related to our reason and thoughts, since typography is intended to be a medium for the latter. Thus it is not coincidental that Robert Bringhurst is not just a master typographer, but also a poet and a deep thinker. His essays are always worth careful study. This review is about two books by this author.

The book *What is Reading For?* is based on the talk Bringhurst gave in 2010 at the symposium *The Future of Reading* at the Rochester Institute of Technology. It is a small brochure densely packed with thoughts, ideas and observations. I have re-read it several times, and every time found something new and important to ponder.

This is not just a book about books. Rather, Robert Bringhurst took the occasion to talk about reading in the wide context of human history. I cannot help but quote one of the first paragraphs, rather controversial (or should we say, prescient?):

I've heard some very interesting things at this symposium, and even some things I

would describe as hopeful. I regard myself broadly as an optimist, and to prove it I will tell you that, despite all signs to the contrary, it is very likely that there is such a thing as a future. I even think that reading, very broadly and deeply defined, may be part of that future. Whether *Homo unsapiens* will be involved in any way is another question. Human activity over the last few centuries has been, on the whole, so short-sighted and self-centered that it has become very difficult to defend the proposition that our species *deserves* a future. But of course, what you do not deserve isn't always what you do not get. There are plenty of individual exceptions, and I don't doubt there are many in this room. But what we do in the aggregate, as a species, is sit at the top of the food chain, gorging ourselves. What future can anyone see in that?

I need to stop here, suppressing the urge to quote the whole book from the beginning to end.

Bringhurst talks about the past of the book the reading, and then goes on to the future and digital books. He compares Western and Eastern writing systems and the deep influence they had on the traditions of reading, typography and calligraphy. He talks about oral cultures and how careful listening is a form of reading.

For me of special interest were his thoughts about the double role of the book — as a piece of art and as an embodiment of the author's thoughts that are by themselves more important than the book itself:

The physical book, as Richard Lanham said the other day, can have a talismanic value, and that's important. But whenever you deal with talismans, you have to keep track of the difference between the talisman and the spirit it represents. *Moby Dick* is a book, and some of us love it so much that we want to honor it by setting it in magnificent type and printing it really well, on really good paper, perhaps with a few magnificent wood engravings of ships, harpoons, and whales for graphic relief, and then sewing and casing it really well, and displaying it like an icon. That's fine thing to do. But if the costume gets too grand, it may defeat its purpose. Books, whether written or oral, are and have to be utilitarian objects. They have to be used, like shoes and socks. That is to say, you have to read them — and to make them

worthwhile, you have to read them yourself. Machines can't do it for you, and no one else can do it for you. Someone else could certainly read it aloud while you listen, but you still have to read it too, using your ears instead of your eyes.

Bringhurst continues the comparison between books and utilitarian objects by quoting a poem by Pablo Neruda about a pair of knit socks made for him by a loving and loved woman. The poet resists the temptation to put the socks “in a gilded cage”. He pays the tribute to the socks by treating them as socks:

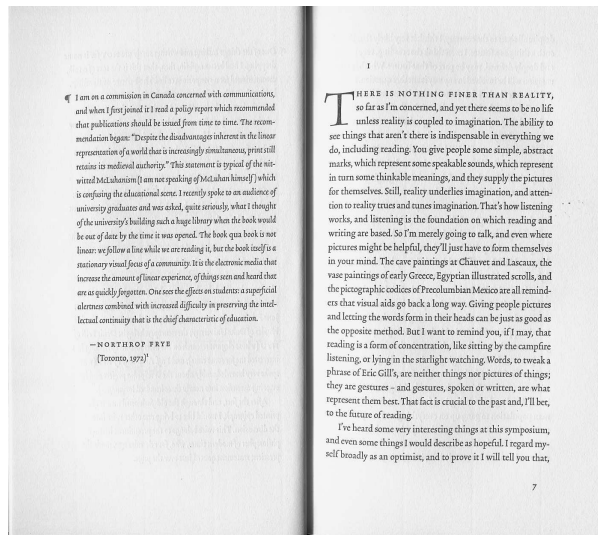
I stuck out my feet
and put them on:
the lovely
socks
and then my shoes.

Bringhurst talks about the cheap editions of his childhood: “a paper brick whose pages would stay together just about long enough to get the book home from the store, and whose pages, falling out of the brittle glue, would stiffen and then crumble into flakes of brown snow within a few decades.” Still, he concludes, “it worked.” Young Bringhurst bought Kant’s *Critique of Pure Reason* in a \$1.45 edition of 1961, separated it into sections several millimeters thick and read these sections on a bus or subway. This act, a sacrilege for a book fetishist, makes a lot of sense from Bringhurst’s point of view: Kant’s thought is much more important than the material embodiment of it in a cheap paperback. This point of view can be traced in Bringhurst’s own typography: the book art is always secondary to the book contents. While this approach is well known to readers of *The Elements of Typographic Style*, it is evident in this brochure as well: one of the requirements Bringhurst makes for digital books is, as a matter of course, “[a] few bells and whistles as possible”.

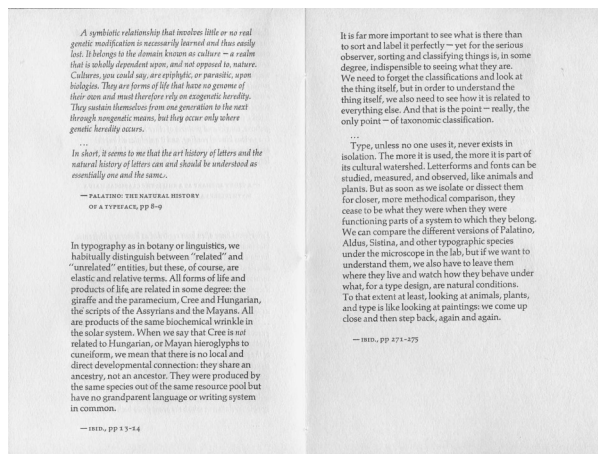
What Is Reading For? has many astute observations, such as a note about the celebration of immaturity in the scribal culture in European monasteries: I immediately recalled the halls of certain high-tech companies in the modern US. Bringhurst easily travels between poetry, music and art (a very interesting section compares a 17th century painting by Gerard ter Borch to a 19th century painting by Jean-Baptiste Corot). It is a pleasure to read.

The book is also a pleasure to hold, confirming Bringhurst’s thought that it is a fine thing to honor the book by “setting it in magnificent type and printing it really well, on really good paper”. The

book is typeset by letterpress with beautiful fonts: Trinité for text, Rialto for titling, New Hellenic for Greek and Parmenides for archaic Greek type, and Kaiti, Mincho and Kazuraki for Han type.



A second book I’d like to mention here is *Some Reflections on Reading and Writing, Culture and Nature, & Sorting Things Out*, a small pamphlet of Bringhurst’s quotations, handset and printed by Amelia Fontanel for the occasion of Bringhurst being presented with the 2016 Frederic W. Goudy Award. The book may be especially significant for the T_EX community because it is set in Hermann Zapf’s fonts. There are just 100 copies of this book (mine is No. 73), so if you want it for your collection, you should hurry. Below is a spread:



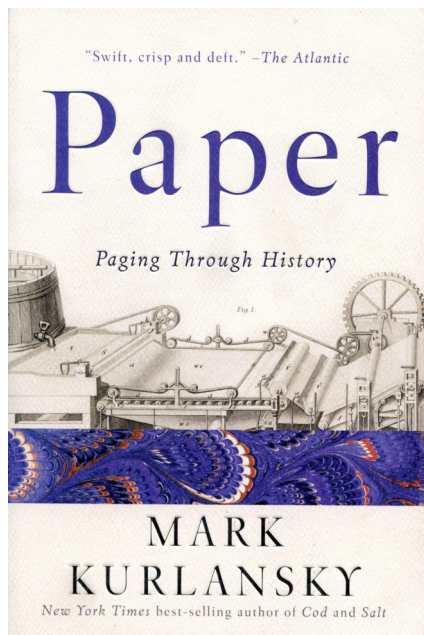
These books are a great tribute to the master of typography, poet and thinker Robert Bringhurst.

◇ Boris Veytsman
<http://borisv.lk.net>

Book review: *Paper: Paging Through History*, by Mark Kurlansky

David Walden

Mark Kurlansky, *Paper: Paging Through History*. W.W. Norton & Company, 2017, xx+389 pp.¹ Paperback, US\$16.95. ISBN 978-0-393-35370-9.



Mark Kurlansky is well known as the author of several books on seemingly narrow topics, such as salt² and the codfish,³ but which actually range far and wide. *Paper: Paging through History* is another of these. For instance, this book addresses the histories of writing, printing, and paper, with the thesis that more people were becoming literate, thus requiring a practical writing material (paper), and getting all those words onto the paper requiring printing.

The book also discusses the histories of these topics in the Far East, Middle East, Europe, and America, starting in 3500 BCE. The author also brings in his beliefs about the ways of innovation: new technology doesn't enable societal changes so much as societal changes require new technology, and new technology doesn't drive out old technology but rather the old continues to exist beside the new. There is no doubt considerable truth to Kurlansky's assertions, but I see them as a little too black and white.

Since Kurlansky has written about 17 nonfiction books on lots of different topics, he can hardly be an expert on all of them and thus I doubt readers can assume the book is precisely accurate at all times. Also, he was a journalist for 15 years, and his books lie somewhere between journalistic and academic studies.⁴ However, even if Kurlansky glosses over

some things or occasionally makes a mistake, I see the book's value as being an easy to read, integrated *introduction* to the histories of (a) paper and the methods of making it and its myriad uses, (b) writing, materials upon which to write, and substances for writing leading up to ink, and (c) printing. It makes a lot of sense to treat these three areas together.

The book's presentation is generally chronological. However, with such a broad scope and so much to share from his research — “Littered with facts”, is a blurb on the book's half title page by Lily Rothman in *Time*) — the organization of the content into chapters seems a little forced. Also, the chapter titles (e.g., “5: Europe between two felts”, “15: Invitation from a wasp”) don't help the reader understand the structure of the book; they make some sense after one has read a chapter — mostly not before. Nevertheless, it is lots of fun to read all those tidbits and to see the logical connections the author tries to make.

Here are some example bits of information.

- Paper making came late to Europe (1140–1400). Paper from 250 BCE has been found in China; paper making was spreading westward through Central Asia and all the way to Egypt in the millennia prior to reaching Europe; the Aztecs were writing on something very much like paper when the Spanish reached what is now Mexico. In China paper was used for wrapping before it was used for writing.
- By the 13th century, high quality paper was being made in the region of Fabriano, Italy. Fabriano had long been a center for pounding wool into felt, and it was easy to convert a felt mill's wool-pounding hammers into rag-beating hammers for paper making.
- In 1719 René Antoine Ferchault de Réaumur's study of American Wasps, which make their homes from a wood-based paper-like substance, kicked off the eventual move to primarily making paper from certain woods rather than from old rags (which were getting scarce and expensive). [The source of chapter 15's title is revealed.]
- In the Revolutionary War between the original U.S. colonies and Britain, paper was in short supply in the colonies, but was necessary for paper plugs to separate the gunpowder in its firing chamber from bullets that shot out of the colonists' muskets. One historian to whom Kurlansky refers has reported that “most of the 3,000-copy press run of Saur's 1776 German Bible was used to fire American muskets.”
- In the early 20th century, book-length “wordless books” were in fashion with the story told completely with woodcuts.



The Stanford Bunny in origami, from *Origamizer: A Practical Algorithm for Folding Any Polyhedron*, by Erik D. Demaine and Tomohiro Tachi, erikdemaine.org/papers/Origamizer_SoCG2017 (CC-BY). See also www.cc.gatech.edu/~turk/bunny/bunny.html and [youtube.com/watch?v=GAnW-KU2yn4](https://www.youtube.com/watch?v=GAnW-KU2yn4).

- In the 21st century, more than two millennia after paper was first made in China, China is again the largest producer of paper in the world. It wanted to be the biggest and became the biggest despite having to import much of the material from which paper is made. More than half of the world's pulp for paper is imported by China, and half of the world's paper for recycling is imported by China.
- The thin seaweed sheets used to wrap sushi are made the same way paper is made and even fit the “randomly-woven-fibers” definition of paper.

I had read a lot about the history of printing and a book on the history of writing before reading this book about paper; this book was a good reminder of that prior knowledge while also telling me the fascinating story of paper. I definitely recommend the book to anyone who has not previously studied the history of paper and its relationship to writing and printing, including young people.

An aside. In these days of increasing home schooling I think this book could be used to show

the child and parent the ways of innovation and how societal need and technology are interleaved, as well as of course conveying the specific histories of writing, paper, and printing. The book also touches on environmental and sustainability issues. Also, given the topics, hands-on craft projects would be a natural adjunct to studying the book. Certainly studying this book would have been a lot more interesting than the standard curriculum that I went through in my K-to-12 days in California public schools.

The thought about home schooling came to mind because late in the book the author mentions contemporary users of paper including a move from origami as craft to origami as sculpture (as shown at left). In this context the author also mentions Eric Demaine, whose Ph.D. thesis was a ground-breaking contribution to the field of computational origami. Demaine was home schooled: news.mit.edu/2003/demaine-0226.

The type note at the end of the book says the book was set in Dante, designed by Giovanni Mardersteig in 1954.

UNDERNEATH THAT EXPLANATION, THE NOTE also says the historical-initials-like images (a copy of one starts this paragraph) at the beginning of the first paragraph of each of the book's almost twenty chapters are based on an alphabet designed by Albrecht Dürer in 1525 and cut into linoleum by the book's author. Missing from this colophon is anything about the paper with which the pages and cover of the book are made: weight, surface finish, source mill or company, and so forth. Having read this book by Kurlansky, I now believe colophons should routinely talk about a book's paper as well as its type.

Still, as befits a book on the history of paper, the paperback cover is stylish. The title and author's name are nicely debossed into attractively rough cover paper. I decided to buy the book the instant I glanced at it on the bookstore's new arrivals rack — and, having read it, I am very glad I did.

Notes

¹The book's page count includes three useful auxiliary resources — an 8-page timeline, a 6-page bibliography, and a 36-page index.

²*Salt: A World History*.

³*Cod: A Biography of the Fish that Changed the World*.

⁴*Paper* is the book by Kurlansky which I have found most interesting among the other books I have read by him. (*The Basque History of the World* was the next most interesting.)

◇ David Walden
walden-family.com/texland



The Treasure Chest

This is a selection of the new packages posted to CTAN (ctan.org) from March–July 2017, with descriptions based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. More information about any package can be found at ctan.org/pkg/pkgname. A few entries which the editors subjectively believe to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

We hope this column and its companions will help to make CTAN a more accessible resource to the T_EX community. See also ctan.org/topic. Comments are welcome, as always.

◇ Karl Berry
tugboat (at) tug dot org

biblio

lbt2bib in **biblio**
Convert `amsrefs .lbt` to `BIBTEX .bib`.

dviware

dviinfox in **dviware**
Dump information about DVI and XDV files.

fonts

algolrevived in **fonts**
Revival of Frutiger’s font designed in 1963 for code blocks in the Algol manual.

alkalami in **fonts**
Font from SIL for Arabic-based writing systems in Nigeria and Niger.

jfmutil in **fonts/utilities**
Process JFM and VF files used in (u)pT_EX.

shobhika in **fonts**
OpenType Devanagari font designed for scholars.

graphics

ladder in **graphics/pgf/contrib**
Simple ladder diagrams in TikZ.

mptrees in **graphics/metapost/contrib/macros**
Probability trees in MetaPost.

pst-geometrictools in **graphics/pstricks/contrib**
Protractor, ruler, compass, etc. symbols in PSTricks.

pst-poker in **graphics/pstricks/contrib**
Customizable poker cards in PSTricks.

pst-rputover in **graphics/pstricks/contrib**
Place text over PSTricks objects without obscuring background colors.

graphics/pstricks/pst-spinner

pst-spinner in **graphics/pstricks**
Drawing fidget spinner toys in PSTricks.

pst-vehicle in **graphics/pstricks/contrib**
Rolling vehicles on mathematically-defined curves.

spectralsequences in **graphics/pgf/contrib**
Spectral sequence diagrams in TikZ.

tikzcodeblocks in **graphics/pgf/contrib**
Code blocks like `scratch`, `NEPO`, and `PXT` in TikZ.

info

***biblatex-cheatsheet** in **info**
Cheat sheet for BIBL_AT_EX/Biber.

studies-lm in **info/german**
Interactive L_AT_EX course material, in German.

language/japanese

bxjaprnind in **language/japanese/BX**
Support Japanese typesetting conventions for open parentheses.

pxufont in **language/japanese**
Emulate non-Unicode Japanese fonts using Unicode fonts.

macros/generic

hlist in **macros/generic**
Horizontal and columned lists.

ifptex in **macros/generic**
Check if engine is pT_EX or a derivative.

macros/latex/contrib

actuarialsymbol in **macros/latex/contrib**
Actuarial symbols for life contingencies and financial mathematics.

biochemistry-colors in **macros/latex/contrib**
Define the standard colors used in biochemistry.

bredzenie in **macros/latex/contrib**
Polish variant for “lorem ipsum” sample text.

bxcalc in **macros/latex/contrib**
Extends `calc` for user-defined units and usability in more contexts.

bxorigcapt in **macros/latex/contrib**
Retain document class’s names `\chaptername`, `\today`, etc., overriding `babel`.

childdoc in **macros/latex/contrib**
Compile `\include` files directly.

cje in **macros/latex/contrib**
Support for the *Canadian Journal of Economics*.

correctmathalign in **macros/latex/contrib**
Better horizontal spacing for some math expressions in alignments.

currency in **macros/latex/contrib**
Format currencies consistently, using `siunitx`.

draftfigure in **macros/latex/contrib**
Replace figures with a (customizable) white box.

easyformat in `macros/latex/contrib`
Markup italic, bold, bold italic with underscores in source.

gotoh in `macros/latex/contrib`
Implementation of the Gotoh sequence alignment algorithm (see article in this issue).

invoice2 in `macros/latex/contrib`
Intelligent invoices with L^AT_EX3.

knowledge in `macros/latex/contrib`
Display, link, index concepts throughout a document.

latex-mr in `macros/latex/contrib`
Guide to L^AT_EX and Polyglossia for Marathi and other Indian languages.

lmi in `macros/latex/contrib`
Official class for *Lecture Notes in Informatics*.

lucida-otf in `macros/latex/contrib`
Support for the OpenType Lucida Bright fonts, including math (tug.org/lucida).

marginfit in `macros/latex/contrib`
Fix margin notes on the wrong side or off the bottom of the page.

mathpunctspace in `macros/latex/contrib`
Control space after comma and semicolon in math.

mcexam in `macros/latex/contrib`
Create randomized multiple choice exams.

minidocument in `macros/latex/contrib`
Subdocuments inside a L^AT_EX document.

modular in `macros/latex/contrib`
Relative sectioning commands.

numnameru in `macros/latex/contrib`
Spelled-out numbers in Russian.

pdfpreview in `macros/latex/contrib`
Annotate PDF files with marginal notes.

poetry in `macros/latex/contrib`
Facilities for typesetting poetry and poetical structure.

rutitlepage in `macros/latex/contrib`
Title pages for Radboud University.

scratch in `macros/latex/contrib`
Draw programs like Scratch (scratch.mit.edu).

scratchx in `macros/latex/contrib`
Include Scratch programs in documents.

sesstime in `macros/latex/contrib`
Add timing marks to lecture notes, and other time management tools.

thaienum in `macros/latex/contrib`
Thai numerals or characters as `enumerate` labels.

typoid in `macros/latex/contrib`
Measure alphabet lengths, em and ex values.

uhhassignment in `macros/latex/contrib`
Typeset homework assignments.

xsim in `macros/latex/contrib`
Improved exercise sheets, succeeding `exsheets`.

zebra-goodies in `macros/latex/contrib`
Handy macros for paper writing.

`macros/latex/contrib/babel-contrib`

azerbaijani in `m/l/c/babel-contrib`
Babel style for Azerbaijani.

`macros/latex/contrib/beamer-contrib/themes`

beamerthemetamu in `m/l/c/beamer-contrib/themes`
Beamer theme for Texas A&M University.

`macros/latex/contrib/biblatex-contrib`

biblatex-enc in `m/l/c/biblatex-contrib`
Style for École nationale des chartes (Paris).

biblatex-oxref in `m/l/c/biblatex-contrib`
Styles inspired by the *Oxford Guide to Style*.

biblatex-shortfields in `m/l/c/biblatex-contrib`
Use fields `shortseries` and `shortjournal` if defined; compatible with `biblatex-claves`.

`macros/luatex`

combofont in `macros/luatex/latex`
Experimental package to add NFSS declarations to combined fonts.

luamesh in `macros/luatex/latex`
Compute and draw 2D Delaunay triangulations.

luapackageloader in `macros/luatex/generic`
Load packages from the standard Lua package.
`path` and `package.cpath`.



Cartoon by John Atkinson (<http://wronghands1.com>).

Calendar

2017

- Jul 30 – SIGGRAPH 2017, “At the ♥ of
Aug 3 Computer Graphics & Interactive
Techniques”, Los Angeles, California.
s2017.siggraph.org
- Jul 31 – Balisage: The Markup Conference,
Aug 4 Rockville, Maryland. www.balisage.net
- Aug 8–11 Digital Humanities 2017, Alliance of
Digital Humanities Organizations,
“Access/accès”, McGill University,
Montréal, Canada. dh2017.adho.org
- Aug 23–27 TypeCon 2017, “Counter!”,
Boston, Massachusetts. typecon.com
- Sep 1 *TUGboat* **38:3** (regular issue),
submission deadline.
- Sep 4–7 17th ACM Symposium on Document
Engineering, Valetta, Malta.
www.doceng2017.org
- Sep 11–17 11th International ConT_EXt Meeting,
“ConT_EXt Gardening”,
Maibacher Schweiz, Germany.
meeting.contextgarden.net/2017
- Sep 12–16 Association Typographique Internationale
(ATypI) annual conference, “Atypique”,
Montréal, Canada. www.atypi.org
- Sep 17–22 XML Summer School, St Edmund Hall,
Oxford University, Oxford, UK.
xmlsummerschool.com
- Sep 23 DANTE 2017 Herbsttagung and
57th meeting,
VHS, Mönchengladbach, Germany.
www.dante.de/events.html
- Sep 28 – Ladies of Letterpress +
Oct 1 Print Week, St. Louis, Missouri.
www.letterpressconference.co
- Oct 6 American Printing History Association’s
42nd annual conference,
“Good, Fast, Cheap Printed Words
and Images in America before 1900”,
Worcester, Massachusetts
printinghistory.org

- Oct 21 GuIT Meeting 2017,
XIII Annual Conference, Mestre, Italy.
www.guitex.org/home/en/meeting
- Oct 23 Award Ceremony: The Updike Prize
for Student Type Design,
Speaker: Nina Stössinger,
Providence Public Library,
Providence, Rhode Island.
www.provlib.org/updikeprize

2018

- Mar 2 *TUGboat* **39:1** (regular issue),
submission deadline.
- Apr 4–6 DANTE 2018 Frühjahrstagung and
58th meeting, Passau, Germany.
www.dante.de/events.html
- Apr 29 – BachoT_EX 2018, 26th BachoT_EX
May 3 Conference, Bachotek, Poland.
www.gust.org.pl/bachotex
- May 1 **TUG 2018** deadline for abstracts
for presentation proposals.
tug.org/tug2018
- Jun 25–29 SHARP 2018, “From First to Last: Texts,
Creators, Readers, Agents”. Society
for the History of Authorship, Reading
& Publishing. Sydney, Australia.
www.sharpweb.org/main
- Jun 25–27 **Practical T_EX 2018**, Rensselaer
Polytechnic Institute, Rome, New York.
tug.org/practicaltex2018
- July 1 **TUG 2018** deadline for preprints for
printed program. tug.org/tug2018

TUG 2018 (satellite conference to the
International Congress of Mathematicians)
Rio de Janeiro, Brazil.

- Jul 20–22 The 39th annual meeting of the
T_EX Users Group.
tug.org/tug2018
-

Status as of 30 July 2017

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568. e-mail: office@tug.org). For events sponsored by other organizations, please use the contact address provided.

User group meeting announcements are posted at lists.tug.org/tex-meetings. Interested users can subscribe and/or post to the list, and are encouraged to do so.

Other calendars of typographic interest are linked from tug.org/calendar.html.

T_EX 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 tug.org/consultants.html. If you'd like to be listed, please see there.

Aicart Martinez, Mercè

Tarragona 102 4^o 2^a

08015 Barcelona, Spain

+34 932267827

Email: [m.aicart \(at\) ono.com](mailto:m.aicart@ono.com)

Web: <http://www.edilatex.com>

We provide, at reasonable low cost, L^AT_EX or T_EX page layout and 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.

Dangerous Curve

+1 213-617-8483

Email: [typesetting \(at\) dangerouscurve.org](mailto:typesetting@dangerouscurve.org)

We are your macro specialists for T_EX or L^AT_EX fine typography specs beyond those of the average L^AT_EX macro package. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical T_EX and L^AT_EX typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a T_EX book.

de Bari, Onofrio and Dominici, Massimiliano

Email: [info \(at\) typotexnica.it](mailto:info@typotexnica.it)

Web: <http://www.typotexnica.it>

Our skills: layout of books, journals, articles; creation of L^AT_EX classes and packages; graphic design; conversion between different formats of documents.

We offer our services (related to publishing in Mathematics, Physics and Humanities) for documents in Italian, English, or French. Let us know the work plan and details; we will find a customized solution. Please check our website and/or send us email for further details.

Latchman, David

2005 Eye St. Suite #4 Bakersfield, CA 93301

+1 518-951-8786

Email: [david.latchman \(at\) textnical-designs.com](mailto:david.latchman@textnical-designs.com)

Web: <http://www.textnical-designs.com>

L^AT_EX consultant specializing in the typesetting of books, manuscripts, articles, Word document conversions as well as creating the customized packages to meet your needs.

Call or email to discuss your project or visit my website for further details.

Peter, Steve

+1 732 306-6309

Email: [speter \(at\) mac.com](mailto:speter@mac.com)

Specializing in foreign language, multilingual, linguistic, and technical typesetting using most flavors of T_EX, I have typeset books for Pragmatic Programmers, Oxford University Press, Routledge, and Kluwer, among others, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. In addition, I've helped publishers write, maintain, and streamline T_EX-based publishing systems. I have an MA in Linguistics from Harvard University and live in the New York metro area.

Sofka, Michael

8 Providence St.

Albany, NY 12203

+1 518 331-3457

Email: [michael.sofka \(at\) gmail.com](mailto:michael.sofka@gmail.com)

Personalized, professional T_EX and L^AT_EX consulting and programming services.

I offer 30 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in T_EX and L^AT_EX: Automated document conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in T_EX or L^AT_EX, `knitr`.

If you have a specialized T_EX or L^AT_EX need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

T_EXtnik

Spain

Email: textnik.typesetting@gmail.com

Do you need personalised L^AT_EX class or package creation? Maybe help to finalise your current typesetting project? Any problems compiling your current files or converting from other formats to L^AT_EX? We offer +15 years of experience as advanced L^AT_EX user and programmer. My experience with other programming languages (scripting, Python and others) allows building systems for automatic typesetting, integration with databases, ... We can manage scientific projects (Physics, Mathematics, ...) in languages such as Spanish, English, German and Basque.

Veytsman, Boris

46871 Antioch Pl.
Sterling, VA 20164
+1 703 915-2406
Email: [borisv \(at\) lk.net](mailto:borisv@lk.net)
Web: <http://www.borisv.lk.net>

T_EX and L^AT_EX consulting, training and seminars. Integration with databases, automated document preparation, custom L^AT_EX packages, conversions and much more. I have about two decades of experience in T_EX and three decades of experience in teaching & training. I have authored several packages on CTAN, Perl packages on CPAN, R packages on CRAN, published papers in T_EX related journals, and conducted several workshops on T_EX and related subjects.

Webley, Jonathan

2/4 31 St Andrews St
Glasgow, G1 5PB, UK
07914344479
Email: [jonathan.webley \(at\) gmail.com](mailto:jonathan.webley@gmail.com)

I'm a proofreader, copy-editor, and L^AT_EX typesetter. I specialize in math, physics, and IT. However, I'm comfortable with most other science, engineering and technical material and I'm willing to undertake most L^AT_EX work. I'm good with equations and tricky tables, and converting a Word document to L^AT_EX. I've done hundreds of papers for journals over the years. Samples of work can be supplied on request.

TUG Institutional Members

TUG institutional members receive a discount on multiple memberships, site-wide electronic access, and other benefits:

tug.org/instmem.html

Thanks to all for their support!

American Mathematical Society,
Providence, Rhode Island

Association for Computing
Machinery, *New York, New York*

Center for Computing Sciences,
Bowie, Maryland

CSTUG, *Praha, Czech Republic*
Institute for Defense Analyses,
Center for Communications
Research, *Princeton, New Jersey*
Maluhu & Co., *São Paulo, Brazil*

Marquette University,
Milwaukee, Wisconsin

Masaryk University,
Faculty of Informatics,
Brno, Czech Republic

MOSEK ApS,
Copenhagen, Denmark

New York University,
Academic Computing Facility,
New York, New York

Overleaf, *London, UK*

ShareLaTeX, *United Kingdom*

Springer-Verlag Heidelberg,
Heidelberg, Germany

StackExchange,
New York City, New York

Stockholm University,
Department of Mathematics,
Stockholm, Sweden

T_EXFolio, *Trivandrum, India*

TNQ, *Chennai, India*

University College, Cork,
Computer Centre,
Cork, Ireland

Université Laval,
Ste-Foy, Québec, Canada

University of Ontario,
Institute of Technology,
Oshawa, Ontario, Canada

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

Reports and notices

- 110 TUG 2017 conference information
- 114 *Jean-Michel Hufflen* / TUG@BachO_TE_X 2017
- 264 Streszczenia
 - all abstracts, in Polish
- 270 TUG@BachO_TE_X 2017 abstracts (de Souza, Egger, Hagen, Hoekwater, Izaola, Kwiatkowska, Ludwiczowski, Miklavc, Mittelbach, Reutenauer, Scherwentke, Thiriet, Tomaszewski, Twardoch, Vieth)
- 273 From other T_EX journals: *Die T_EXnische Komödie 2/2017*
- 280 *Boris Veytsman* / *What Is Reading For?* and *Some Reflections on Reading and Writing, Culture and Nature, & Sorting Things Out* by Robert Bringhurst
 - review of these two thought-provoking works by Bringhurst
- 282 *David Walden* / *Paper: Paging Through History*, by Mark Kurlansky
 - review of this book on the intertwined history of writing, printing, and paper
- 285 *John Atkinson* / Word on the street
- 286 Calendar
- 287 T_EX consulting and production services
- 288 Institutional members

Introductory

- 157 *Willi Egger* / Bookbinding workshop: Making a portfolio
 • instructions for constructing an art folio, with diagrams
- 145 *Michał Gasewicz* / Off topic (completely): Many faces (and types) of beer
- 126 *Hans Hagen* / Children of T_EX
 • premises, predilections, predictions for T_EX, with reference to many books
- 116 *Janusz Nowacki* / Calligraphy by Barbara Galińska
 • a few words about and samples of the work by this superlative artist
- 118 *Maciej Rychły* / Released sounds
 • paintings, music, stories
- 165 *K. Sathasivam, S.K. Venkatesan, Y. Chandy* / Revealing semantics using subtle typography and punctuation
 • punctuation history, redundancies, ambiguities, and resolutions
- 141 *Luigi Scarso* / MF_Lua 0.8 — Prologue
 • philosophical reflections on society and the T_EX software family
- 125 *Boris Veytsman* / The state of T_EX
 • address at the conference of the incoming TUG President

Intermediate

- 185 *Takuto Asakura* / Implementing bioinformatics algorithms in T_EX — the Gotoh package, a case study
 • a sequence alignment algorithm in primitive T_EX, optionally with `texshade`
- 263 *Karl Berry* / Production notes
 • the rather helter-skelter *TUGboat* production process
- 284 *Karl Berry* / The treasure chest
 • new CTAN packages, March–July 2017
- 274 *Charles Bigelow* / Review and summaries: *The History of Typographic Writing — The 20th century*, Volume 2 (ch. 1–5)
 • second of three installments; chapter-by-chapter summaries for vol. 2 (1950–2000), ch. 1–5
- 255 *Marcin Borkowski* / Ten years of work in *Wiadomości Matematyczne* — an adventure with L^AT_EX and Emacs hacking
 • L^AT_EX and Emacs policies, workflow, and macros for the journal *Wiadomości Matematyczne*
- 193 *Siep Kroonenberg* / T_Launch, the T_EX Live Launcher for Windows
 • T_EX Live configuration (editors, viewers, ...) in a multi-user Windows installation
- 212 *L^AT_EX Project Team* / L^AT_EX news, issue 27, April 2017
 • ISO 8601 date format; TU encoding improvements; Hyphenation; Default language; Line spacing in parboxes
- 202 *Jerzy Ludwichowski* / GUST e-foundry current font projects
 • brief overview of current GUST OpenType projects, mostly math-related
- 214 *Vít Novotný* / Using Markdown inside T_EX documents
 • generic support for Markdown input inside T_EX, via Lunamark and Lua
- 175 *Petr Sojka, Vít Novotný* / T_EX in Schools? Just Say Yes: The use of T_EX at the Faculty of Informatics, Masaryk Univ.
 • historical and current use at a large university in the Czech Republic
- 249 *Lolita Tolén* / T_EX user habits versus publisher requirements
 • study of programs and packages used in practice, and XML translation issues
- 171 *Boris Veytsman and Leila Akhmadeeva* / To justify or not to justify? Why bad typography may be harmful
 • experimental results of slower reading of `\sloppy` text than `\raggedright`
- 173 *Boris Veytsman* / Making `ltxsparklines`: The journey of a CTAN contributor into the world of CRAN
 • an R package to support Tufte sparklines (word-sized graphics) in L^AT_EX

Intermediate Plus

- 159 *Barbara Beeton* / Debugging L^AT_EX files — Illegitimi non carborundum
 • editing, tracing, diagnosing, testing, puzzling
- 203 *Hans Hagen* / Variable fonts
 • supporting variable fonts in LuaT_EX and ConT_EXt
- 245 *Jean-Michel Huppen* / M_LB_ET_EX now handles Unicode
 • per-bib file specification of encodings, and supporting Unicode
- 208 *Bogusław Jackowski, Piotr Strzelczyk, Piotr Pianowski* / Parametric math symbol fonts
 • assembling OpenType math fonts from a text font and existing math symbols, with LuaT_EX
- 238 *Dávid Lupták* / Typesetting bibliographies compliant with the ISO 690 standard in L^AT_EX
 • the `biblatex-iso690` package and comparison with existing approaches
- 213 *Frank Mittelbach* / L^AT_EX table columns with fixed widths
 • a convenient interface for fixed-width columns in L^AT_EX tables
- 188 *Norbert Preining* / `updmap` and `fmtutil` — past and future changes (or: cleaning up the mess)
 • new usage, with new per-tree configuration and persisting across reinstallations

Advanced

- 147 *Jean-Michel Huppen* / History of accidentals in music
 • usage and typesetting of sharps, flats, etc., from ancient to modern music
- 218 *Grzegorz Murzynowski* / GMS, the “General Meta-Scenarios”: A proper extension to the `l3expan` package of the `expl3` bundle and language, two years later
 • a method for avoiding any redundant and verbose code
- 197 *S. Tolušis, A. Povilaitis, V. Kriaučiukas* / Xdvipsk: Dvips ready for OpenType fonts and more image formats
 • extending Dvips to support more bitmap formats and OpenType fonts, in a LuaT_EX workflow