Opening words

Christina Thiele
President, TEX Users Group

Well, we finally made it into your mailbox! The last issue for 1994! It's been a tough year for our publications. Tough year for the editors, too. With delays due to technical problems and just plain scheduling problems, it has been a difficult year for our membership to feel that they are still connnected to their user group. And what's so frustrating in all these delays is that there is so much that has gone on this past year that we want to tell you about.

We've seen the introduction of the new standard LATEX, no longer specifying it as '2_ε'; we've seen the genesis of such TEX developments as the Omega project and NTS, the New Typesetting System; the appearance of colour on the typeset page has provoked a revitalisation in the DVI Standards Committee; the explosion of work being done to bring TEX onto the World Wide Web via HTML — to name but a few which spring to mind.

Information from all over!

You will have noticed the great increase in the distribution of all manner of TEX-related information, from CDs to CTAN, from meetings to publications. There is so much information now available in almost all forms of media currently in place, that it's hard to keep track of it all. Hard to know what to read or buy or attend first!

Being a member of TUG — indeed, being a member of any user group — already provides you with some 'navigating tools' — publications, meetings, information posted to archives are some of the ways in which the wealth of information and assistance can be made manageable.

TUG publications, in particular the proceedings issue of *TUGboat*, give members a permanent record of what's happening, what people are doing, information and background on new applications, advice and examples of how use TEX better, and generally provide that sense of community which membership helps foster. Add to that the publications from other user groups, and you start to feel overwhelmed by how much information there is, how much work individuals in the TEX community are putting into their research — and then into their writing. How to keep up with it all?

One navigating 'tool' you might not think of is the abstract, a short text which often starts off a paper. These are a required element in the proceedings issues of *TUGboat*, but a lot of regular-issue articles also carry them. As well, *TUGboat*'s editor, Barbara Beeton, makes a concerted effort to get abstracts or summaries of material which appears in publications such as DANTE's *Komödie* or the *Cahiers* from GUTenberg. This issue here also has the abstracts from the 1994 EuroTEX meeting.

So, in a few short reading sessions, you can be on top of what's happening right now: some 37 abstracts from the 1994 TUG meeting, 3 from *Cahier* 18 and 9 from no. 19, rounded off with 19 abstracts from the 1994 EuroTEX meeting. Even if that's all you read out of this issue of *TUGboat*, at least you'll be aware of the work being done by a broad range of members of the TEX community, and when the time comes that you need that information in all its detail, you'll know where to look. An abstract sure is handy!

1994 also saw TUG begin to provide information and materials on-line via its subdirectory on CTAN; we hope to see frequently requested items from the office made available electronically, allowing anyone to get basic information directly, rather than mailing or phoning the TUG office. In addition to administrative items such as membership forms, we also have a slowly expanding set of 'info-sheets', short 1- or 2-page documents which can provide handy information in a summary form. Conference information for the upcoming meeting in Florida will also be found there. Use ftp to your nearest CTAN site and go to `tex-archive/usergrps/tug`.

And what about 1995?

Well, there's an election coming up! Nominations are needed for five board members — and one president. My term will come to an end with this summer's annual meeting in Florida (*another* event that's coming up fast), and after having been on the board since 1988, it's time to move over. Information on nomination procedures will appear in *TTN*.

There's just enough room left to squeeze in an advance reminder for the 1995 Annual Meeting, in Florida — where they do believe in air-conditioning! So plan now to come to the TradeWinds hotel; information will be appearing in *TTN*, on CTAN, and on the Web (see the poster for details).

◇ Christina Thiele
15 Wiltshire Circle
Nepean, Ontario
K2J 4K9 Canada
cthiele@ccs.carleton.ca

Editorial Comments

Barbara Beeton

TEX meetings in 1994

This year's TUG meeting was held in Santa Barbara on the University of California campus, on a peninsula overlooking the Pacific Ocean. I won't go into details, except to invite you to see for yourself in the proceedings (published as *TUGboat* **15** no. 3) the wide variety of topics on the menu. Some faces from the past reappeared — Leslie Lamport and Chuck Bigelow are two whose names should be familiar to all *TUGboat* readers. A most enjoyable meeting.

I also attended EuroTEX '94 — an even greater adventure. Held in Sobieszewo, Poland (a seaside town near Gdańsk), it was in a totally new region for me, and I'm still thankful that Włodek Bzyl and his wife were so kind to collect me at the airport. (Thank you ever so much, Włodek!)

As always at TEX meetings, there were many new faces to attach to names I already knew from e-mail correspondence, and old friends with whom to catch up on gossip. The accommodations and the meeting rooms were in a holiday resort, just a few hundred meters from the Baltic shore.

Włodek Bzyl and Tomek Przechlewski, editors of the GUST bulletin, were the meeting organizers. Hanna Kołodziejska, the President of GUST, also joined in making everyone feel warmly welcome. A full report appears later in this issue, so let me just take this opportunity to thank everyone who made the experience so memorable.

A new, expanded TEX FAQ

With Robin Fairbairns in command, a UKTEXUG working party has produced a new, reorganized, much expanded TEX FAQ ("Frequently Asked Questions" list, with answers); the results can be seen in *Baskerville* Vol. 4 No. 6, December 1994.

The original FAQ was compiled as an adjunct to the Usenet newsgroup `comp.text.tex` and maintained by Bobby Bodenheimer of Caltech. (Thanks, Bobby, for all your efforts.) The new edition has been created with his knowledge, and it is intended to feed it back to `c.t.t` via Bobby's regular posting.

The new FAQ is available electronically from CTAN, in `tex-archive/usergrps/uktug/faq`. The source (for LATEX2$_\varepsilon$) is there as well as several predigested versions:

- `letterfaq.ps`, for U.S. letter-size paper
- `newfaq.*`, for A4 paper; `.dvi`, `.ps` and `.pdf` versions, using PostScript fonts, are available
- `newfaq-cm.dvi`, for those who have only Computer Modern fonts

Thanks to Robin and his colleagues for a fine job.

`UKTeX Digest` ceases to exist

With the distribution of Volume 94, Issue 48, the `UKTeX Digest` ceased to exist. This digest was a regular visitor to my e-mail in-box, and I learned many useful things from its contents.

Originally created in July 1987 by Peter Abbott to announce developments of the UK TEX Archive at Aston University (another of Peter's creations), it was moderated and edited by Peter for its first few years of existence. Since mid-1990, David Osborne has been the editor. Through the seven and a half years of its existence, 336 digests were produced.

With the emergence of CTAN (the Comprehensive TEX Archive Network), the need for a UK-oriented digest has disappeared. However, for some time David has also been editing the `TeXhax Digest`, the model for `UKTeX`, and `TeXhax` will continue, absorbing contributions that would have been addressed to `UKTeX`.

There is still a need for such an electronic distribution list as a question, answer and announcement forum for the global TEX community, especially for people without access to news, or who do not have the time to read the full contents of `comp.text.tex`.

In any event, this is an appropriate occasion to publicly express our appreciation to Peter and David for their efforts through the years, with special thanks to David for the statistics that I have shamelessly copied from his notice in the final issue.

Well done!

Miscellaneous gossip

Raman's thesis, "An Audio System for Technical Reading (ASTeR)", has won the ACM Dissertation Award for 1994. The official presentation will be in March 1995.

Anyone who attended the 1992 TUG meeting in Portland will remember Raman's presentation, a preliminary report on his research, and his friendly black Labrador retriever, Aster, to whom the work is dedicated. Another look at the article in the proceedings (*TUGboat* 13, No. 3) will be worth your time, and increase your awareness that there is an audience for TEX that isn't interested only in how the output looks on paper.

Congratulations and best wishes to both Raman and Aster.

From David M. Jones, creator of the (now sadly outdated) TEX index: "Reviving the TEX index will be my New Year's resolution, but I'm still shy of making any promises, since my record for keeping those over the last couple of years has been dismal. :-("

(Actually, David has been occupying his spare time with other useful TEX-related pursuits; he has contributed ideas and code to the display math environments of $\mathcal{AMS}$-LATEX, and his name now appears among the acknowledgements for that package. David also attended his first TUG meeting this past summer in Santa Barbara, demonstrating that he really does exist.)

The question keeps arising on `comp.text.tex` and other electronic forums, where is an electronic version of *TUGboat*?

Contrary to what is apparently the general belief, *TUGboat* does not lend itself easily to electronic distribution, requiring, as it does, nonstandard (and sometimes proprietary) fonts, nonstandard versions of macro packages, multiple files for some articles, both plain-based and LATEX processing, and other varieties of special handling. Moreover, since *TUGboat* is just completing its 15^{th} volume, the archives contain a variety of input styles and require several different versions of (LA)TEX for processing, including the no-longer-available TEX78.

However, the TUG Publications Committee is looking into ways to get around these problems, so as to be able to provide at least some of the most important material in electronic form. Stay tuned.

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

The TUG94 Proceedings — Apologia

Michel Goossens, Sebastian Rahtz and Barbara Beeton

The TUG94 Annual Meeting took place at the beginning of August 1994, and the deadline for (revised) compuscripts was the end of August. It was decided to produce the final camera-ready copy on a typesetter in Europe, to give the editor and his assistant a chance to check it before sending to America. After a last check of the articles during September, PostScript files for a Linotronic typesetter at 1270 dpi resolution were generated at the beginning of October in Geneva. These files were transferred by Sebastian Rahtz to a machine connected to the Linotronic typesetter of the Computing Centre of the University of London, where Philip Taylor had kindly volunteered to help produce the bromides.

Now the problems began to get more interesting. The first problem to emerge was with memory on the Linotronic. Initially, we tried to send 20 page batches, but almost all failed. Cutting this down to fewer and fewer pages per section, numerous trials over several weeks resulted in almost no output, as the large collection of Lucida fonts being downloaded with each job consistently ran the poor typesetter out of memory. Using the fine-tuning of *dvips* to specify the exact memory available in the typesetter did not help. Since nothing useful was being accomplished, we decided to stop including Type-1 Lucida fonts in the output, and instead generate PK format bitmaps at 1270dpi using *ps2pk*.

This resulted in jobs that were much more acceptable to the typesetter, and rolls of bromide started to appear. However, several pages, especially those of the Haralambous papers, used complex METAFONT fonts which again exhausted the memory of the typesetter (see table 1). So a few of the font examples of the Tiqwah (yannisT) and Ω (yannisO) papers had to be regenerated at 600 dpi and embedded as EPS pictures in the text.

Note: The table of fonts used in each paper is not included in this file.

Still not everything was coming out (we were running now in batches of 10 pages), with about 40 pages causing problems. These were finally run one by one through the typesetter. The resulting monstrous heap of rolls of bromide was cut up, collated and posted off to Barbara Beeton, who (inevitably) found some pages missing; each of these (seven of them) was transmitted as an individual PostScript file and printed at the AMS.

The color pages were completely done in the United States. They fell into two batches. Those that were available as PostScript files with CMYK color information had their 4-colour separations done and films produced at the AMS, while the *BM2FONT* color images of Sowa (Colour Examples 10 to 13) were scanned at the printer, Cadmus.

It had been planned to send the colour PostScript files direct to the printer, and have them separated there, but the *dvips* output seemed to defeat Cadmus, who claimed to find no colour on the pages! By this time, help from heaven was badly needly, and it duly arrived in the shape of a new RIP in the typesetter at AMS which could do colour separations directly, without need for a preliminary processing phase. Fingers crossed, Barbara threw the TUG94 papers at it, and got back a pile of correctly separated negatives; the only problem then was that she had to work out by eye which was which for each page out of cyan, magenta, yellow and black...

Some other problems, born of incompatibilities between the (fixed) resolution at which some graphics had been prepared and the higher resolution of the typesetter, required additional reruns and falling back on the tried and true cut-and-paste makeup for several pages, again at AMS. Finally, the time spent by Barbara on the *Proceedings* issue was cut from her time scheduled to work on the next regular issue (15 #4, the issue you are now looking at).

All these unforeseen problems, and the fact that several people in different locations had to deal with them, meant that the issue was not completely ready until the end of January 1995. It is only thanks to the perseverance of Sebastian Rahtz that the issue finally arrived on your desk. We want to apologize for this delay and sincerely hope that this year the *Proceedings* of the Florida Annual Meeting will be available before the end of 1995. Moreover for the following year we propose to change the whole schedule and have the *Proceedings* ready *at* the Conference, i.e., in July.

Michel Goossens
Sebastian Rahtz
Barbara Beeton

EuroTEX'94

Julita Bolland (text)
Toni Walter (drawings)
Włodek Bzyl (the prize founder)

Introduction. What and where Sobieszewo is, probably only a few people in Gdańsk know without checking the map of Poland. The name of Gdańsk may or may not be more familiar. If not, then for all those who are not particularly good at geography I'll give some reference information, which cannot be as detailed as I would like it to be.

Gdańsk, an old Baltic seaport, is situated just at the mouth of the Wisła river (the longest river in Poland). Sometime in the past Gdańsk became a rich merchant city with an autonomous government. Merchants were rich enough to gain self-government and too wise to break relations with the Polish kingdom.

The incredible wealth of the merchants created the shape of the city with many granaries, magnificent houses and monumental churches. "Once upon a time" there was even the second Elizabethan Theatre in Europe; unfortunately nothing was left for us.

Knowing vaguely the approximate position of Gdańsk we can trace the path to Sobieszewo. Somewhere in the mouth of the Wisła river, in between its arms there is a piece of land where houses stand in groups — that's Sobieszewo, the very place where the Eighth European TEX Conference was held this year on 26–30 September. It may not sound as exciting as Hawaii or Malaysia, ..., but here the world was typeset with thick northern forests, grey, cold Baltic sea and quick-dunes. Or even more, just imagine spotting unknown mushrooms among bushes near your foot like notes left by Nature and little, yellow, transparent dots of Baltic sea honey* on the beach. But as the Chinese saying goes:

百 闻 不 如 一 见 .

The Conference started on Monday 26th September and ended on Friday 30th September; however, some of us came the day before. Approximately 60 participants from Western, Middle and Eastern Europe attended the Meeting together with special guests Barbara Beeton and Christina Thiele.

The organisers were GUST (DeGUSTibus non est disputandum) and Gdańsk University. The event was sponsored by grants from the Polish Ministry of Education, Addison-Wesley, Springer-Verlag, GUTenberg, DANTE, UKTug, NTG. Special thanks goes to Phil Taylor for his invaluable help.

There was evening and there was morning, one day. Although technically it was the first day (though not the very first one, since the official opening was celebrated the following day), nonetheless we were busy visiting Gdańsk in the afternoon, not to mention that some of us started this day file with 'a walk along an unguarded beach' in a group hunting for amber.

The more down-to-TEX part of the programme, supposed to begin on the 27th of September, was divided into three topics: Principles, Practice, Progress. The presented papers have been published in the Proceedings of the Eighth European TEX Conference.* But let's not anticipate events which don't belong to this sequence or else we lose control.

There was evening and there was morning, a second day. The official opening took place in the morning starting with Phil Taylor's speech and with a few warm words said by Włodek Bzyl and Hanna Kołodziejska. At this very moment the programme of the Conference started to run interactively.

Although talks were planned as the main part of all activities, in a couple of days coffee breaks (or coffee joints?) took over the schedule. Exchanging

* In case of misunderstanding check The Oxford Dictionary under the entry 'amber'.

* The Proceedings are available at the bargain price of 15 DM. If you want a copy please contact us at the address: matwb@halina.univ.gda.pl

views, chatting, etc. occurred just at the time of the coffee breaks (joints?). The organisers, in order to help us get acquainted with each other, gave a 'diplomatic' (standing) party. Champagne, (...), beer and a little something kept us alive for a few hours and probably no one was underfull or overfull. Bogusław 'Jacko' Jackowski and Phil Taylor gave a special treat whose 'specialité de la maison' were two guitars and the most popular songs.

Evening passed and morning came, that was the third day. Talks, talks and coffee breaks. The performers did their best to excite the audience. The subjects varied from LaTeX2ε through Yannis Haralambous Ω to METAFONT. The end of the day was marked by the Babel discussion.

There was evening and there was morning, a fourth day. On this day the first \bye-s appeared; however there were still many talks and at the end of the whole Conference the Cathy Booth prize was awarded. In Sobieszewo the best paper hasn't been chosen arbitrarily, but in democratic and secret voting.

The winners were Bogusław Jackowski and Marek Ryćko who got the prize twice before: Karlsruhe 1989, Praha 1992, and (a reminder) in Aston 1993 they gave the keynote talk.

There was evening and there was morning, fifth, sixth, seventh day. Tutorials and courses were organized after the Conference. It should be noted that the most popular one was 'Book Design and Typography' by Marek Ryćko and Phil Taylor.

Conclusion. In a last few words on atmosphere, I would like to quote Erik Frambach:

> *You can think of a conference as a concert for a symphony orchestra. The quality of the concert depends on the individual skill of the players but even more on their ability to cooperate. There was much enthusiasm for any tune regardless of its significance. Because of this attitude the players were all at ease, which encouraged personal communication very much. Very soon social intercourses became a major occupation for many.*

In this way talks, tutorials, courses became headlines and footlines for the conference shape.

GOD BLESS YOU MR KNUTH.

The Participants. The conference gathered 59 people from all over Europe. Portraits of some of them are displayed below. Dear reader, do you recognize some of them? If yes, please drop an email to `matwb@halina.univ.gda.pl`. The individual with the highest number of correct guesses will be given a bottle of *Gdańsk Gold Wasser* at the next EuroTeX meeting — to be held in the Netherlands.*

Julita Bolland (text)
Toni Walter (drawings)
Włodek Bzyl (the prize founder)

---

* Editor's note: The editor regrets the delay in *TUGboat* publication that prevents most readers from participating in this contest. We will discuss with Włodek the possibility of a substitute contest.

Head 1.

Head 3.

Head 2.

Head 4.

Head 5.

Head 6.

Head 7.

Head 8.

Head 9.

Head 10.

Minutes of the $\mathcal{NTS}$ meeting held at Lindau on October 11/12th 1994

Philip Taylor,
Technical Director, $\mathcal{NTS}$ project

Present: Philip Taylor, Jiří Zlatuška, Bernd Raichle, Friedhelm Sowa, Peter Breitenlohner, Joachim Lammarsch.

It was **agreed** that no progress had apparently been made on the 'canonical TEX kit' project, and that no progress was likely to be made unless and until an active proponent of the project emerged within, or was recruited to, the group; accordingly the project was officially placed on ice.

It was **agreed** that in the absence of adequate funding for the $\mathcal{NTS}$ project proper, no serious work could be carried out; several possible sources of funding remained to be explored, and the group were hopeful that this project would see the light of day before too long.

It was **agreed** that the ε-TEX project was both feasible and very worthwhile, and that all efforts should initially be concentrated on achieving progress in this area. With the benefit of hindsight it was **agreed** that the original proposal to issue new releases at six-monthly intervals had been over-optimistic, and that a more realistic timescale would involve new releases once per year. It was also **agreed** that the first release should be accomplished as soon as possible, consistent with the need to ensure that the code released was both bug-free and unlikely to require more than a minimum of re-thinking in the light of experience.

The group attempted to identify as many ideas as possible which either have already been proposed for incorporation in ε-TEX, or which were natural consequences of (or alternatives to) ideas already proposed. The remainder of this document lists the various ideas mooted, and discusses their intention and implementation.

Proposals

`\horigin`, `\vorigin` (dimen registers, default = 1 in)

These two registers, requested by Phil, will serve to make explicit for the first time the canonical (1″, 1″) origin decreed by DEK in the definition of the DVI format, and on which all formats are currently predicated. Phil explained that his college, amongst others, had eschewed this convention right from the outset, and has instead adopted the more logical (0″, 0″) origin, requiring all drivers to be configured in a non-standard manner. Providing the origin registers within ε-TEX would allow all drivers to be reconfigured to the standard, whilst existing practices could be maintained simply by local initialisation of the registers to (0″, 0″). As ε-TEX might eventually require the adoption of a new version of the DVI format (to encompass, for example, colour), that might also be the appropriate time at which to propose universal adoption of a (0″, 0″) origin.

\⟨enhancement⟩`state` (internal integer registers, one for each enhancement)

A unified mechanism is proposed for all enhancements [1] whereby an internal integer register is associated with each, the name of the register being derived from the concatentation of the name of the enhancement and the word 'state'; such registers are read/write, and if their value is zero or negative the associated enhancement is disabled. [2] If a positive non-zero value is assigned to any such register, then the associated enhancement shall be enabled, and if the register is interrogated then a positive non-zero value shall indicate that the associated enhancement is enabled. It is possible that in a future release differing values assigned to or returned by such registers may indicate the revision-level of enhancements, and therefore it is initially recommended that only the values zero or one be used.

`\TeXXeTstate`, `\MLTeXstate` (internal integer registers)

These are the only two enhancements currently under consideration, although Bernd Raichle also has a proposal for an alternative ligaturing mechanism which would probably of necessity form an enhancement if adopted. MLTEX is not proposed for incorporation in the first release, but may be incorporated in the second. The group acknowledges the generosity of Michael Ferguson in allowing the incorporation of his work on MLTEX.

`\interactionmode` (internal integer register)

Allows read/write access to the present `\scrollmode`, `\nonstopmode`, etc., family of primitives; the values will be a monotonic sequence of period one, and descriptive names will be associated through the ε-plain (and ε-LATEX?) formats. [3]

⟨additional `\tracing...` detail⟩

Peter has implemented augmented semantics for some of the `\tracing` commands whereby increasingly positive values given increasingly detailed output.

`\protected` (new prefix for macro definitions)

Analogous to `\long`, `\outer`, etc., causes the associated macro to be non-expanding in contexts

where such behaviour is likely to be undesirable (in particular in `\write`s and `\edef`s); an explicit `\expandafter \empty` may be used to force expansion in these circumstances.

`\bind` (new prefix for macro definitions)

Proposed by Phil, this was intended to allow macros to be bound to the current *meaning* of embedded control sequences rather than to their names, in a manner analogous to PostScript's 'bind def'. However the group were unconvinced of the merits of this proposal, and it was classified as 'more work needed' (MWN).

`\evaluate {`⟨arithmetic expression⟩`}`

Intended for use on the r-h-s of `\count`, `\dimen` and `\skip` assignments, it would allow the use of infix arithmetic operators such as `+`, `-`, `*` and `/`; the type of the result would, in general, be the type of the simplest operand forming a part of the expression, and the normal semantics of TEX would allow this to be further coerced where necessary. Parenthesised sub-expressions would be allowed. [4]

`\contents` ⟨box #⟩

Proposed by Jiří, this is intended to allow the TEX programmer access to the sort of information normally available only via the log file as a result of a `\show`; in principle it would generate the simplest list of TEX tokens which would generate the box specified, assuming that each token generated still had its canonical meaning. MWN.

⟨anchors⟩

Proposed by Jiří, an "anchor point" would be in some senses analogous to a mark, but rather than recording textual information it would instead record the co-ordinates of itself, relative to the reference point of the smallest surrounding box. Additional new primitives would be required to return the co-ordinates of a specified anchor point. MWN.

`\scantokens {`⟨balanced text⟩`}`

Allows an existing token-list to be re-input under a different catcode regime from that under which it was created; as it uses all of TEX's present `\input` mechanism, even `%%ff` notation will be interpreted as if `\input`. Causes an 'empty filename' to be input, resulting in '`( )`' appearing in the log file if `\tracingscantokens` (q.v.) is strictly greater than zero. If the token list represents more than one line of input, and if an error occurs, then `\inputlinenumber` will reflect the logical input line from the token list rather than the current input line number from the current file.

`\unexpanded {`⟨balanced text⟩`}`

An alternative to `\protected`, for use when a whole brace-delimited token list ('balanced text') is to be protected from expansion. Intended to be used in `\write`s and `\edef`s.

`\every`⟨whatever⟩

The group discussed many possibilities of implementing additional `\every` primitives in ε-TEX; most were classified as MWN, but one (`\everyeof`) is being considered for ε-TEX version 1.

`\futuredef` ⟨cs⟩ ⟨tok⟩ ⟨tok⟩

Analogous to `\futurelet`, but the ⟨cs⟩ will be expandable, and expand to the next token encountered (or to the next balanced text if the next token is of catcode 1). MWN.

`\futurechardef` ⟨cs⟩ ⟨tok⟩ ⟨char-or-tok⟩

A combination of `\futurelet` and `\chardef`, will allow the next character to be inspected and its character code returned iff it has not yet been tokenised. If tokenisation has already taken place, will return −1. Intended to allow the catcode of the next character to be changed based on its value.

`\ifdefined` ⟨cs⟩

Allows direct testing of whether or not a given ⟨cs⟩ is defined.

`\ifcsname ... \endcsname`

Ditto, but for a sequence of ⟨tokens-expanding-to-characters⟩; this also avoids wasting of hash table space.

`\unless` ⟨boolean-if⟩

Inverts the sense of the following boolean-if; particularly useful in conjunction with `\ifeof` in `\loop ... \ifeof ... \repeat` constructs, but also of use with (say) `\ifdefined` and `\ifcsname`.

`\TeX`⟨whatever⟩`state`

More work needed! A mechanism whereby a TEX document can ask TEX some questions about the current state of its digestive tract. For example it would be nice to know if TEX was currently involved in an assignment, and if so which part of the assignment was currently being elaborated.

`\marks` ⟨integer⟩

Allows, for the first time, a whole family of marks rather than just the one provided by TEX; will also require analogous `\topmarks` ⟨integer⟩, etc. The group propose to provide 16 such marks, but are interested to know if the (LA)TEX community consider this sufficient. A related `\markdef` primitive may be provided to simplify mark allocation, in a manner analogous to the existing `\...def` primitives.

`\deferred \special` (or perhaps `\deferredspecial`)

At the moment, only `\writes` are deferred; there are cases when it would be desirable for other things, too, to be expanded only during `\shipout`, and `\special`s are one of these.

`\textcode` ⟨integer⟩

Could provide a text-mode analogy to TEX's `\mathcode`. MWN.

`\middle` ⟨delimiter⟩

Analogous to `\left` and `\right`, allows delimiters to be classed as `\middle`, and their spacing thereby adjusted.

`\filename`

Would allow access to the name of the file currently being `\input`. Lots of discussion on just how much or how little should be returned. MWN.

`\OSname`

Very contentious. Would provide the name of the operating system, and thereby allow documents to behave differently on different systems. Deprecated on that basis, and will not be provided unless/until a `\system` primitive is also provided.

`\system {`⟨balanced text⟩`}`

Definitely not proposed for ε-TEX version 1. Would allow operating system calls to be made, and their status and result(s) returned in some way. A lot MWN.

`\tracingscantokens` (internal integer register)

See `\scantokens`.

⟨smarter discretionaries⟩, e.g. `\discretionarylefthyphenmin`

Hyphenation after an implicit hyphen is sometimes highly desirable, and the group are investigating mechanisms whereby this could be both provided and parameterised. MWN.

`\everyhyphen` (token list register)

Would allow TEX's present hard-wired behaviour of placing an empty discretionary after every explicit hyphen to be modified. However, there are potentially problems of recursion, and perhaps even a need to remove the hyphen. MWN.

`\clubpenalties`, `\widowpenalties`

A start at improving TEX's penalty system by making it more flexible. These two penalty 'arrays' would allow a different penalty to be associated with one-line widows, two-line-widows, etc. [5]

`\ifenhanced`

A boolean-if which would return `true` iff any enhancement is enabled. Would allow a ε-TEX document to check if it is being processed in 'extended' more or 'enhanced' mode. Phil argues for this one but the group are unconvinced: the advice of the TEX community is to be sought.

`\lastnodetype`

Would allow, for the first time, the unambiguous identification of the type of the last node of one of TEX's internal lists, removing (for example) the ambiguity when `\lastpenalty` returns 0 (which can indicate no penalty node, or a penalty node with value 0). Would return one of a monotonic series of integers of period one. Meaningful names would be assigned to these through the ε-series formats. [3]

`\unnode`

Would allow the removal of *any* node from the end of one of TEX's internal lists.

`\lastnode`

Perhaps analogous to `\contents` (q.v.), or perhaps quite different, would allow access to the value of the last node of one of TEX's internal lists. Generalises TEX's present mechanism whereby only a subset of nodes can be accessed. MWN.

`\readline` ⟨integer⟩ `to` ⟨cs⟩

Allows a single line to be read from an input file as if each character therein had catcode 12. [6] Intended to be used for verbatim copying operations, in conjunction with `\scantokens`, or to allow error-free parsing of 'foreign' (non-TEX) files.

`\everyeof {`⟨balanced text⟩`}`

Provides a hook whereby the contents of a token list register may be inserted into TEX's reading orifice when end-of-file is encountered during file reading. Would not be invoked if the file indicated logical e-o-f through the medium of `\endinput`. Proposed by Phil to allow clean processing of file-handling code which requires a (sequence of characters yielding) `\else` or `\fi` to be found in a file, where no such sequence can be guaranteed.

`\listing` (internal integer register)

Would allow the generation of a listing containing (for example) TEX's analysis of current brace depth, macro nesting, etc. Different positive values would allow different amounts of information to be generated. Would the TEX community like such a feature?

`\defaultextension`

Would allow TEX's present hard-wired behaviour of appending `.tex` to a filename not possessing an explicit extension to be modified, allowing an alternative extension to be specified. Would this be of use to the L2ε/L3 team, and/or to the TEX world in general?

⟨fixed point arithmetic⟩

Several of the above ideas cannot be implemented at the moment, as they would allow access to the 'forbidden area' of machine-dependent arithmetic. If TEX's present floating point calculations were replaced by Knuth's fixed-point arithmetic proposals, then there would no longer be a forbidden area and all such ideas could, in principle, be implemented.

Notes:

[1] 'Extensions' are basically new primitives which have no effect on the semantics of existing TEX documents, except insofar as any document which tests whether such a primitive is, in fact, undefined, will clearly obtain opposite results under TEX and ε-TEX; 'enhancements' are more fundamental changes to the TEX kernel which may affect the semantics of existing TEX documents even if no new primitive is used or even tested. Such changes may be, for example, differences in the construction of TEX's internal lists, or perhaps different hyphenation or ligaturing behaviour.

[2] It is currently proposed that all enhancements be disabled by ε-IniTEX immediately prior to the execution of `\dump`. This decision was taken based on the advice of Frank Mittelbach.

[3] Question: should there, in fact, be an ε-plain (or ε-LATEX) format, or should there simply be an `e-plain.tex` file which can be loaded by a user document? Peter votes for an `e-plain.tex` file that will `\input plain.tex` but no hyphenation patterns.

[4] Should ε-TEX allow access to more powerful operators than just `+`, `-`, `*` and `/`?

[5] 'Arrays' are not very obvious in TEX at the moment, although there are, for example, `\fontdimen`s and such-like. But should these have fixed bounds (as in 256 count registers, for example), or arbitrary upper bounds (as in font dimens, if the 'extra' elements are assigned as soon as the font is loaded). Or should they be finite-but-unbounded, as in `\parshape`, wherein the first element indicates the number of elements which follow? These questions are applicable to marks as well as to penalties...

[6] Should spaces have catcode 10 for this operation? Peter thinks so, but based on existing simulations of this operation, Phil is more inclined to think they should have catcode 13.

◇ Philip Taylor
The Computer Centre, RHBNC
University of London, U.K.
<P.Taylor@Vax.Rhbnc.Ac.Uk>

Tools

TEX innovations at the Louis-Jean printing house

Maurice Laugier and Yannis Haralambous

Abstract

In this paper we will present several TEX innovations, conceived, or currently under development, at ILJ (the Louis-Jean printing house). ILJ was founded in 1804, at Gap (Southern French Alps) and employs 130 people. Due to its specialization in typesetting and printing of scientific books, ILJ has been using TEX since the late eighties. Currently about 30% of ILJ's book production is done with TEX. New developments in the TEX area sponsored or financed by ILJ are described in this paper.

— — * — —

In exactly ten years ILJ (Imprimerie Louis-Jean) will celebrate its bicentennial. Needless to say, this printing house has followed closely all developments of the printing industry: leaden types from the early XIXth century until 20 years ago, photocomposition in the seventies, and since the early eighties, the computer. Almost everything has changed: authors have changed, publishers have changed, even the product a printing house produces is not the same anymore: some years ago one was making books, now they are more and more often accompanied (and perhaps will be replaced in a few years) by those small silver disks, called CD-ROMs.

In the old days, the author would most probably supply a manuscript. Often one had to rival Champollion's skills to decipher these manuscripts, in order to be able to compose them. Later on one used to receive manuscripts in typed form; no deciphering was necessary any more, but this implied the work of an intermediate person, usually the author's secretary, or the author him/herself.

In the last decenny authors have bought personal computers; together with these engines they bought programs that make them believe they can typeset a book. More and more authors send "ready-to-print" books on floppies, which most of the time are "ready-to-throw-away". Since it is not possible to teach authors the rudiments of typography, one has to invest time and energy in *getting the most* out of these files, and finally be able to print the book the author had in mind. For this reason, one has to be able to offer the whole spectrum of services, starting with text input, and finishing with industrial printing.

Typesetting a book in TEX is an even bigger challenge, since the printing process requires also a strong know-how in programming: one has to know TEX sufficiently well to either write the TEX code for a book, or modify the code supplied by the author; one has to know SGML if the text is received marked up in that language and has to be converted into TEX, or inversely, if the author and/or publishier wishes to have the book in SGML form; one has to have some knowledge of PostScript in case something goes wrong at the color separation or flashing stage, and so on.

It follows that a minimum number of services ILJ has to supply are:

1. Processing of TEX and LATEX files, at the input, DVI or PostScript level, that is:
   - writing TEX or LATEX code, or converting Word, WordPerfect, Mathor, etc., documents into decent TEX/LATEX code;
   - correcting it;
   - checking the page setup;
   - incorporating illustrations;
   - coloring it;
   - providing TEXnical assistance on the development of the LATEX style file;
   - producing an SGML representation of it;
   - selecting or creating if necessary the fonts required for it.
2. Making high resolution films (1200 to 2400 dpi).
3. Industrial printing, binding, routing.

To be able to solve reasonably quickly the problems arising in a process as complicated as the one just described, ILJ had to develop a certain number of tools. The fact that TEX is an open system with no commercial maintenance was a risk to take; it also gave ILJ the opportunity of developing auxiliary tools which it would be impossible to make in conjunction with closed "ready-to-use" systems such as PageMaker or Quark XPress.

1 Oriental scripts

The second author, while working at the Institute of Oriental Languages and Civilizations in Paris, wrote a typesetting system for Oriental languages, based on TEX. ILJ has contributed to these projects both technically (by providing the necessary back end for Oriental typesetting and printing), and financially.

Together with John Plaice (Université de Laval, Québec), the second author is also developing Ω an extension of TEX internally based on ISO/IEC 10646/UNICODE. ILJ is ready to adopt ISO/IEC 10646/UNICODE as the fundamental encoding for text processing, in order to solve once and for all the problem of

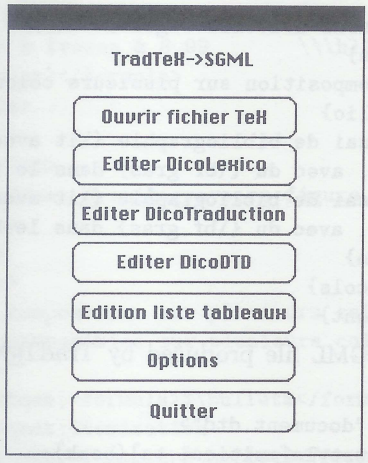


Figure 1: The generic menu of TradTEX→SGML

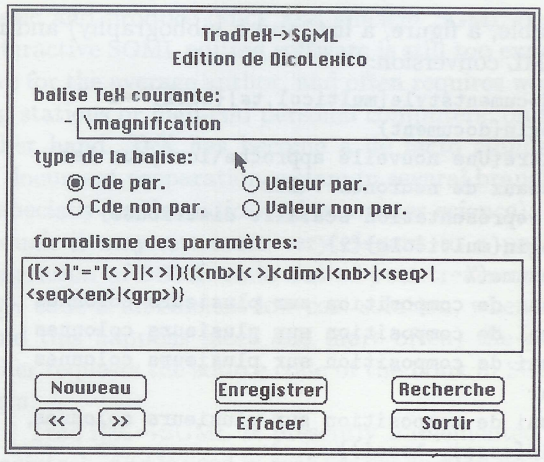


Figure 2: Editing the TradTEX→SGML database

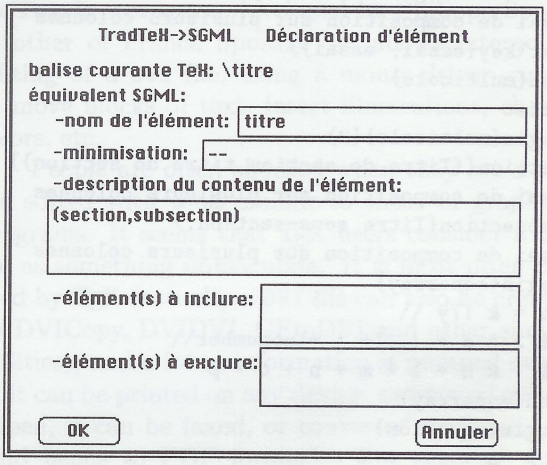


Figure 3: Declaring a TEX command in SGML

encoding ambiguities, a problem which can be very painful for texts with special needs (texts using symbols and/or non-Latin characters).

2 The TradTEX→SGML program

2.1 The principle

The TradTEX→SGML program, developed by Franck Spozio, is an assistance to conversion of TEX or LATEX files into SGML. The user of TradTEX→SGML must have a fairly good knowledge of both TEX and SGML, since the process of conversion involves a stage of analysis.

TradTEX→SGML creates a database containing all TEX codes it has encountered as well as their structure; this database file is accumulating information on TEX commands, from several runs: primitives, standard macros, or user-defined ones. Furthermore it checks the syntactic validity of these commands, in case the same command is defined with different structures in different files or contexts.

The file "DicoLexico" contains the TEX commands and their structures. As the reader can see in fig. 2, this database file can be edited and modified on the fly. The file "DicoTraduction" contains the SGML entities corresponding to the TEX commands, as well as their structure. The TEX codes which have to remain unchanged in the SGML file, in a `<NOTATION>` entity (for example, those describing math formulas) are stored in a special file, called "liste tableaux".

When reading the TEX code, TradTEX→SGML analyzes all TEX tokens; whenever a token has not been defined or contradicts the description contained in "DicoLexico", the user is prompted to specify the structure of the command (by a dialog as in fig. 3); the data requested are (a) the number of arguments, (b) the nature of delimiters, (c) if it is acting on one or more tokens coming after or before, (d) special characters, etc. Whenever the TEX command affects the formatting of the text, the corresponding informations are stored into the file "DicoDTD"; they will be used to create the DTD file which will mimic the original formatting specifications of the file.

Once this informations is stored, the user is prompted for the translation of the token in SGML (names of elements, attributes, processing instructions, entities, etc.). This information is then stored in file "DicoTraduction".

Once all tokens have been read and verified, the translation of the TEX file into SGML begins, using information from all four database files mentioned above; at the same time a DTD file is created. An example follows of a LATEX file (with two abstracts, a section and a subsection, an array of equations,

a table, a figure, a list, and a bibliography) and its SGML conversion:

```
%\documentstyle[multicol,ts]{book}
\begin{document}
\titre{Une nouvelle approche\lcr pour les
réseaux de neurones :\lcr
la représentation scalaire distribuée}
\begin{multicols}{2}
\resume{%
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
\par
Essai de composition sur plusieurs colonnes
\cle{essai, essai}}
\abstract{%
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
\par\key{essai, essai}}
\end{multicols}
\filet
\begin{multicols}{2}
\section{{Titre de section titre de section}}
Essai de composition sur plusieurs colonnes
\subsection{Titre sous-section.}
Essai de composition sur plusieurs colonnes
\begin{eqnarray}
x & = & 17y \\
y & > & a + ... + j +\nonumber\\
  &   & K + l + m + n + o + p
\end{eqnarray}
\begin{equation}
y  >  a + ... + j +
\end{equation}
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
\begin{tabular}{||l|lr||}\hline
\haut gnats & gram & \$13.65 \bas\\\cline{2-3}
            & each &  .01        \\\hline
gnu & stuffed & 92.50 \\\cline{1-1}\cline{3-3}
emur&         & 33.33 \\ \hline
armandillo & frozen & 8.99       \\\hline
\end{tabular}
\end{multicols}
\begin{figure}
\vglue 3cm
\caption{Essai premi\'ere figure}
\end{figure}
\begin{multicols}{2}
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
\begin{itemize}
\item[$\bullet$] Essai d'itemisation
\item[$\bullet$] Essai d'itemisation
\end{itemize}
\begin{figure}
\vglue 8cm
\caption{Essai nouvelle figure}
\end{figure}
Essai de composition sur plusieurs colonnes
\begin{biblio}
\bib{1}{Essai de bibliographie {\it avec} de
l'italique, avec du {\bf gras} dans le texte}
\bib{2}{Essai de bibliographie {\it avec} de
l'italique, avec du {\bf gras} dans le texte}
\end{biblio}
\end{multicols}
\end{document}
```

The SGML file produced by TradTEX→SGML is:

```
<!DOCTYPE "document.dtd">
--\documentstyle[multicol,ts]{book}--
<doc>

<titre id=???>Une nouvelle approche<?\lcr>
 pour les r&eacute;seaux de
 neurones&espace;&colon;<?\lcr> la
 repr&eacute;sentation scalaire
 distribu&eacute;e
</titre>
<deuxcols>
<resume>
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
<par>
Essai de composition sur plusieurs colonnes
<cle>essai&comma; essai</cle></resume>
<abstract>
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
<par><key>essai&comma; essai</key></abstract>
</deuxcols>
<?\filet>
<deuxcols>
<section>Titre de section titre de section
</section>
Essai de composition sur plusieurs colonnes
<subsect>Titre sous-section</subsect>
Essai de composition sur plusieurs colonnes
<formula>\begin{eqnarray}
x & = & 17y \\
y & > & a + ... + j +\nonumber\\
  &   & K + l + m + n + o + p
\end{eqnarray}</formula>
<formula>\begin{equation}
y  >  a + ... + j +
\end{equation}</formula>
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
<formula>\begin{tabular}{||l|lr||}\hline
\haut gnats & gram & \$13.65 \bas\\\cline{2-3}
            & each &  .01        \\\hline
gnu & stuffed & 92.50 \\\cline{1-1}\cline{3-3}
```

```
emur&          & 33.33 \\ \hline
armadillo & frozen & 8.99        \\\hline
\end{tabular}</formula>
</deuxcols>
<figure>
<?\vglue 3cm>
<legende>Essai premi&egrave;re figure
</legende>
</figure>
<deuxcols>
Essai de composition sur plusieurs colonnes
Essai de composition sur plusieurs colonnes
<itemize>
<?\item>&lqsb;<formula>$\bullet$</formula>]
 Essai d&quot;itemisation
<?\item>&lqsb;<formula>$\bullet$</formula>]
 Essai d&quot;itemisation
</itemize>
<figure>
<?\vglue 8cm>
<legende>Essai nouvelle figure</legende>
</figure>
Essai de composition sur plusieurs colonnes
<biblio>
<bib>1</bib>Essai de bibliographie
<it> avec</it> de l&quot;italique&comma;
avec du <bf> gras</bf> dans le texte
<bib>2</bib>Essai de bibliographie
<it> avec</it> de l&quot;italique&comma;
avec du <bf> gras</bf> dans le texte
</deuxcols>
<par>
</doc>
```

And finally the DTD file:

```
<!NOTATION TeX SYSTEM "NotationTeX">
<!ELEMENT formula #NOTATION TeX #CURRENT>
<!ELEMENT par -O >
<!ELEMENT doc -- >
<!ELEMENT titre -- (section,subsection)>
<!ATTLIST titre id ID #IMPLIED>
<!ELEMENT deuxcols -- >
<!ELEMENT resume -- >
<!ELEMENT abstract -- >
<!ELEMENT cle -- >
<!ELEMENT subsect -- >
<!ELEMENT figure -- >
<!ELEMENT legende -- >
<!ELEMENT itemize -- >
<!ELEMENT biblio -- >
<!ELEMENT bib -- >
<!ELEMENT it -- >
<!ELEMENT bf -- >
<!ELEMENT section -- >
<!ELEMENT key -- >
<!ELEMENT eitemize -- >
```

When using a consistent and fairly stable set of TEX/LATEX macros throughout several documents, TradTEX→SGML can become more and more automatic and produce quick and efficient SGML code. Interactive SGML editing software is still too expensive for the average author, and often requires working stations or high-end personal computers; on the other hand, TEX has become a de facto standard of document preparation system in several branches (especially mathematics and computer science): although it may not seem very elegant to a purist, generation of SGML code out of post-treatment of TEX code is an efficient low-cost solution, whenever (and this happens more and more often) the publisher requests the source code of the book in SGML form.

TradTEX→SGML is presently implemented on Macintosh, and is used in real-life production by ILJ.

3 eDVItor

eDVItor is a program developed by Philippe Spozio (brother of Franck Spozio). It allows interactive editing of a DVI file, using a mouse-driven cursor to move blocks of text, insert illustrations, change colors, etc.

People are often shocked when they see a DVI file edited with tools similar to those of graphical programs. It seems that TEX users consider a DVI file as something immaculate. It is most often created by TEX (actually a DVI file can also be created by DVICopy, DVIDVI, GFtoDVI and other similar utilities), and a lot of information is pumped out of it; it can be printed on any device, previewed on any screen, it can be faxed, or converted to PostScript (and hence to PDF format). But none of these drivers and utilities change the text formatting in a DVI file; DVIDVI will perhaps change the order of pages, DVICopy will replace characters by other characters with the same metrics, and drivers do not modify a DVI file.

There is a reason behind this: according to TEX ideology, TEX does the ultimate text formatting, it would be vain to modify it manually. This of course is true, if we consider TEX's line breaking algorithm, or the typesetting of math formulas.

But what about titles, figures, or horizontal lines? We are forced to admit that these depend on the taste of the... human typesetter, rather than on TEX's skill. After all, to place a horizontal line or an illustration, we give millimetric instructions to TEX, concerning both the size of the object and the size of the surrounding white space. And these instructions can very well be wrong, or *slightly* wrong.

In the best of all worlds, one would run TEX on a file as many times as necessary, until the file is perfect from all points of view. In a real-life production world this is unfortunately not possible: a 600-page

book can be run only a limited number of times. eDVItor allows us to make "those small last-minute changes", directly on the (otherwise perfect) DVI file.

Of course, the same changes have to repeated every time a DVI file is created anew (unless the corrector is smart enough to report the changes also to the TEX code file). In fig. 4, the reader can see a DVI file progressively modified by dragging blocks of text around.

eDVItor has been implemented both on DOS and on Macintosh. It allows also insertion of illustrations, by simple copy and paste operations, coloring of text and color separation of the whole document. A second version of this program, currently under α-testing, executes commands placed into the DVI file by the means of \special commands. Here are some commands which can be executed by eDVItor v2:

- Color processing:

```
\special{Color color1}
text in color 1
\special{Color color2}
text in coulor 2
\special{Color color1+color2}
texte in coulor 1+2
```

- Vertical column adjustment:

```
\special{Post Equalize
        Column1,Column2,column3
        Height=25cm Pages 17-27}
```

  This command will spread paragraphs so that columns 1, 2 and 3 of pages 17–27 will have the same height, namely 25cm. To use this functions one has to specify first the parameters of stretching and shrinking of interparagraph blank space.

```
\special{Post Expand sup=1pc inf=-2mm}
```

  This command will modify interline space.

- Positioning and moving around figures:

```
\special{Post InsertAt x=10cm y=2,4cm
Pages 1,[76] illustration figure.eps}}
```

  EPSF figure figure.eps is placed at coordinates x =10cm, y =2.4cm on page 1 (folio 76).

- Inclusion of graphical commands: These commands allow framing of a block of text, or positioning of simple geometrical figures (rectangles, circles, ellipses), eventually filled with a certain color or gray density.

```
\special{Post SetFillDensity 30}
\special{Post FilledZoneFrame}
```

- Inclusion of external DVI files:

```
\special{Post Import fichier.dvi}
```

  The DVI file fichier.dvi will be merged into the current DVI file. Suppose you are writing a book on TEX and want to include an example of TEX file output: for example a beginning of chapter page. Up to now there were two solutions: either simulate the beginning of chapter by writing the corresponding TEX code, or take the real TEX file you want to show, produce a DVI file, run dvips with the -E option and obtain an EPSF file, and include the latter in the original DVI file as an illustration. Of course the latter solution has the disadvantage that it is not device independent anymore: the EPSF file unavoidably contains bitmap fonts in a fixed resolution. eDVItor allows you to include a DVI file into another DVI file.[1]

It should be noted that modifications not involving PostScript are applied to the DVI file when processed by eDVItor: in that way, (a) one obtains a new DVI file, modified according to the \special commands, and processable by any DVI driver, (b) since the \special commands are in the TEX code, these modifications are automatically applied whenever eDVItor processes the DVI file (so one can produce new DVI files without fear of losing precious information added manually, as in the case of version 1 of eDVItor).

It is the hope of the authors that eDVItor will be the first step to a "WYSIWYG" TEX, in the sense of more effective DVI file manipulation and possibility of last-minute changes.

◇ Maurice Laugier
General Director of ILJ,
Imprimerie Louis-Jean, B.P. 87,
05003 Gap Cédex, France
Email: louijean@cicg.grenet.fr

◇ Yannis Haralambous
187, rue Nationale, 59800 Lille,
France.
Email: haralambous@univ-lille1.fr

[1] The second author always wondered how D.E. Knuth did volume E of Computers & Typesetting, where GFtoDVI printouts are mixed with TEX code.

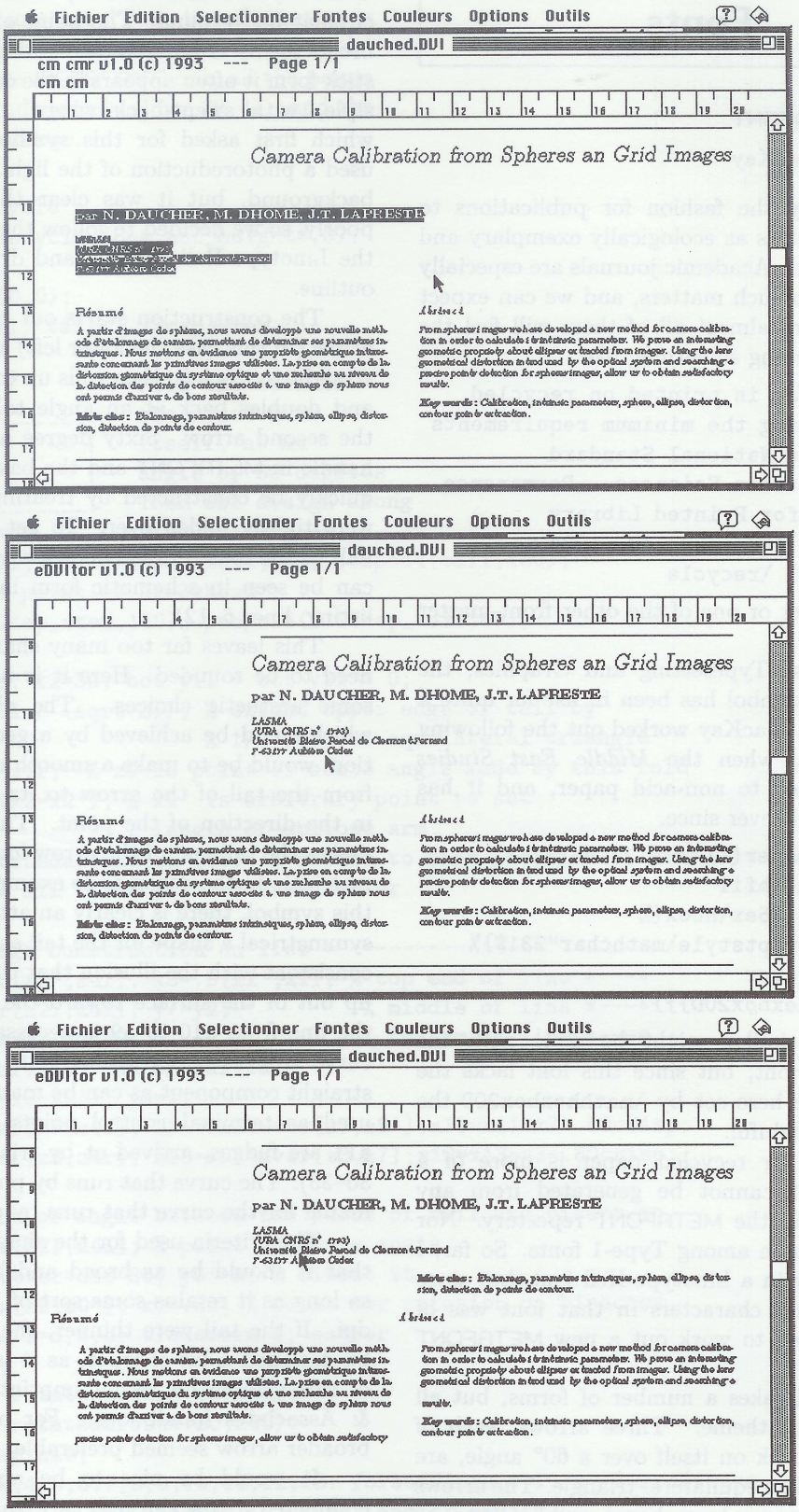


Figure 4: Editing a DVI file with eDVItor

Recycled METAFONT

Pierre A. MacKay

It is increasingly the fashion for publications to advertise themselves as ecologically exemplary and archivally antacid. Academic journals are especially under scrutiny in such matters, and we can expect that in the future almost all of them will feel the necessity of including something like

```
This journal is printed on recycled
paper meeting the minimum requirements
of American National Standard
for Information Sciences---Permanence
for Paper for Printed Library
Materials, ANSI Z39, 48--1984.
\goodpaper\ \recycle
```

on the inner cover or one of the other front-matter pages.

At Humanist Typesetting and Graphics, the ⊛ `\goodpaper` symbol has been in use for quite a while. Theodora MacKay worked out the following some years ago, when the *Middle East Studies Bulletin* converted to non-acid paper, and it has proved serviceable ever since.

```
\def\goodpaper{{%
  \ooalign{\hfil
    \raise.25ex\hbox{%
      $\scriptstyle\mathchar"231$}%
    \hfil\crcr
    \mathhexbox20D}}}
```

Something similar might be done with the Type-1 Symbol font, but since this font lacks the large open circle here set by `\mathhexbox20D` the effort would be painful.

The image for recycled paper is more of a problem since it cannot be generated from any existing glyphs in the METAFONT repertory. Nor is it as yet common among Type-1 fonts. So far, I have seen it only in a Linotype-Hell font, but since none of the other characters in that font was of interest, I decided to work out a new METAFONT version.

The symbol takes a number of forms, but all with a consistent theme. Three arrows, each of which is rolled back on itself over a 60° angle, are arranged to form an equilateral triangle. The arrows may be broad, (paper bags, boxes, publications) or narrow (several types of plastic). The broad form is usually shown in outline, or in white on a dark circular background. The point of the arrow in its broad forms is quite blunt, though in the narrow stick form it often appears in the conventional sharp style, with swept-back wings. The publication which first asked for this symbol had previously used a photoreduction of the light image on a dark background, but it was clear that this took ink poorly, so we decided to follow the general model of the Linotype-Hell symbol, and draw the arrows in outline.

The construction starts out simply. The basic arrow (in this case the lower left) starts out heading straight to the left, then rolls up towards the viewer and doubles back at an angle to meet the tail of the second arrow. Sixty degree angles are easy to handle in METAFONT and the basic framework can quickly be constructed by treating the tail as if it were tightly folded over. To get the rolled effect, half of the folded area is cut away (on a line which can be seen in schematic form in the source code listing, lines 6–12).

This leaves far too many sharp corners, which need to be rounded. Here it is necessary to make some aesthetic choices. The obvious approach, which could be achieved by a geometric construction, would be to make a smooth curve carry round from the tail of the arrow to its eventual heading in the direction of the point. This, unfortunately, makes the tail end of the arrow look grossly unsymmetrical. In the dozen or so examples I have seen of this symbol, there is clearly an attempt to retain as symmetrical a shape for the tail as can be managed, consistent with the illusion that the arrow is rolling up out of the surface toward the viewer. Because the line from `a10` to `a9` is necessarily straight, the curve from `a8` to `a10` needs to have as large a straight component as can be managed. The values used as terminal control points for `a8`, `a10` and `a11` are fudges, arrived at by trial and error (lines 30–33). The curve that runs by point `a3` is a precise match for the curve that runs by point `a7`.

The criteria used for the shape of the head are that it should be as broad and blunt as possible, so long as it retains some sort of point even at 300 dpi. If the tail were thinner, the arrowhead could be designed a bit sharper, as it is in the rendition of this symbol on recent imprints of the O'Reilly & Associates handbooks. For our purposes, the broader arrow seemed preferable.

It would be nice to be able to rotate the completed arrow through two transforms and have the three arrows set in place that way, but the basic picture is far too complex, and produces METAFONT's plaintive message "That transformation is too hard." So, with careful attention to the model on page 138 of the METAFONTbook, all the points are transformed individually (lines 39–47), and the picture is twice redrawn.

```
cmchar "Recycle";
beginchar(Recycle,18u#,asc_height#,0);
italcorr 0;
adjust_fit(0,0);
numeric tail; tail=2.25cap_stem;
%   * /
%    /     /         Fold the tail of
%   /     /______    the arrow over
%   \    /       |   itself, at an
%    \  /        |   angle of 60 degrees
%     \/_________|   Then cut design along
%         *          line *--------*
penpos1(tail,90); penpos2(tail,90); penpos4(tail,150);
% wings and point of arrowhead
penpos5(3.0cap_stem,150); penpos6(2.25cap_hair,60);
%
x1=.5w-.25u; x2=3u; bot y1l = bot y2l = 0;
x2'=x2-(tail / (sqrt 3)); % extend upper edge of tail by
                          % 1/2 base of equilateral triangle
y1'=y9=y2'=y1r;  % z9 is point of acute angle made by this fold
x1-x1'=1.5(x2-x2'); % x1' is arbitrary point to set
                    % length of oblique arm
z4r=z1' rotatedaround(z2',60); % rotate to find location of pen4
z6=z5=z4; % all three pens on same center.
%
% Cut through construction on line *---------------*
z3=whatever[z2',z4r]; x3=.5[x2',x2]; % top end of line *---*
z7=whatever[z2l,z4l]; y7=y2;         % middle of line *---*
x8'-x2=x2-x2'; y8'=0;                % bottom of line *---*
% Fudge start and end of curves.
x8=x8'+.35(x2-x2'); y8=0;
z10=whatever[z7,z4l]; z11=whatever[z7,z8'];x10=x11=x7+.80crisp;
z12=whatever[z3,z4r]; z13=whatever[z3,z7];x12=x13=x3+.80crisp;
%
% Find point of angle between two parts of tail and bisect it
z9=whatever[z2l,z4l]; % point of acute angle
z9-z0'=whatever*dir 30; y0'=h; % bisect it.
z0=whatever[z9,z0']; x0=.5w; % point for rotation on bisector
forsuffixes $=a,b,c: transform $; endfor
a=identity;
b=identity rotatedaround(z0,120);
c=identity rotatedaround(z0,240);
pickup crisp.nib;
for n=1,2,3,4,5,6,7,8,9,10,11,12,13: forsuffixes e=l,,r: forsuffixes $=a,b,c:
  z$[n]e = z[n]e transformed $; endfor endfor endfor
forsuffixes $=a,b,c:
  z$8' = z8' transformed $;
  draw z$9--z$1r--z$1l--z$8{z$8-z$1l}..z$11{z$7-z$8'}..{z$10-z$7}z$10;
  draw z$10--z$4l--z$5l--z$6r--z$5r--z$4r--
       z$12{z$3-z$4r}..{z$7-z$3}z$13--z$7{z$8'-z$7}..{z$1l-z$8}z$8;
  labels($1,$1l,$1r,$2,$3,$4,$5l,$5r,$6r,$7,$8,$9,$10); endfor
labels(2',8'); endchar;

```

"Recycle" is the first and, as yet, the only character in the HTG pi font. It is mapped to the uppercase "R".

```
\font\htgpi=htgpi10 at 12pt
\def\recycle{{\htgpi R}}
```

The parameter file `htgpi10.mf` is a straight steal from `cmsy10` with only the initial comment lines and the final line changed. The driver file `pifont.mf` is modeled on `symbol.mf`, even to the point of retaining the early half of the file for slanted characters.

```
% pifont.mf Driver file for new
% characters. Parameters based on cmsy
font_coding_scheme:="Font dependent";
mode_setup; font_setup;
font_slant slant;
font_x_height x_height#;
font_quad 18u#
  if not monospace:+4letter_fit# fi;
% Slanted symbols here as needed
% Remaining characters unslanted
slant:=mono_charic#:=0;
currenttransform:=identity
  yscaled aspect_ratio
  scaled granularity;
Recycle:=ASCII"R";
input recycle
bye.
```

Who knows what other characters may be added in the future.

◇ Pierre A. MacKay
Department of Classics DH-10
Department of Near Eastern
Languages and Civilization
(DH-20)
University of Washington
Seattle, WA 98195 U.S.A.
Phone: 206-543-2268;
FAX:206-543-2267
mackay@cs.washington.edu

Indica, an Indic preprocessor for TEX
A Sinhalese TEX System

Yannis Haralambous

Abstract

In this paper a two-fold project is described: the first part is a generalized preprocessor for Indic scripts (scripts of languages currently spoken in India—except Urdu—, Sanskrit and Tibetan), with several kinds of input (LATEX commands, 7-bit ASCII, CSX, ISO/IEC 10646/UNICODE) and TEX output. This utility is written in standard Flex (the GNU version of Lex), and hence can be painlessly compiled on any platform. The same input methods are used for all Indic languages, so that the user does not need to memorize different conventions and commands for each one of them. Moreover, the switch from one language to another can be done by use of user-defineable preprocessor directives.

The second part is a complete TEX typesetting system for Sinhalese. The design of the fonts is described, and METAFONT-related features, such as metaness and optical correction, are discussed.

At the end of the paper, the reader can find tables showing the different input methods for the four Indic scripts currently implemented in *Indica*: Devanagari, Tamil, Malayalam, Sinhalese. The author hopes to complete the implementation of Indic languages into *Indica* soon; the results will appear in a forthcoming paper.

– – ∗ – –

1 *Indica*

1.1 Introduction

Many Latin-alphabet native writers find the Greek and Cyrillic alphabets exotic (not to mention African and phonetic characters). Actually this shouldn't happen, since —at least for the upper case— Greek, Cyrillic and Latin types can have the same design: they have the same roots, have evolved more-or-less in the same way, and the same principles of Occidental type design can be applied to them. There are even common glyphs to the three ('A', 'B', 'E', 'H', 'M', 'O', 'P', 'T', 'X') which will appear only once in case one wishes to have a big "Greco-Cyrillico-Latin" font.

The situation is completely different in the case of Indic languages. Once again all of their scripts have the same roots, but instead of keeping the same style and being complementary to each other, they all have the same set of letters, in the same order, but with (often very) different shapes. Every child in India learns the same alphabet "ka-kha-ga-gha-..." but depending on the region, the letter shapes can be very different: क ख ग घ ङ च छ... in Devanagari script, ಕ ಖ ಗ ಘ ಙ ಚ ಛ... in Malayalam, ක ඛ ග ඝ ඞ ච ඡ... in Sinhalese, etc.[1]

This justifies the choice of a common transliteration scheme for all Indic languages. But why is a preprocessor necessary, after all?

A common characteristic of Indic languages is the fact that the short vowel 'a' is inherent to consonants. Vowels are written by adding diacritical marks (or smaller characters) to consonants. The beauty (and complexity) of these scripts comes from the fact that one needs a special way to denote the *absence of vowel*. There is a notorious diacritic, called "virāma", present in all Indic languages, which is used for this reason. But it seems illogical to *add* a sign, to specify the *absence* of a sound. On the contrary, it seems much more logical to remove something, and what is done usually is that letters are either brought very near (in Sinhalese) or written one over another (Malayalam), or written together while losing some parts (Devanagari, Bengali, ...). In this way we obtain those hundreds of beautiful ligatures which make the charm of Indic scripts.

When typesetting with TEX, the preprocessor will have to indicate to TEX all the necessary ligatures which can be either constructed from character parts (as in the case of Velthuis's Devanagari), or spread in several 256-character tables (as in the case of the Sinhalese font described in the second part of this paper). Also, it often happens that a vowel is written in front of a group of consonants, although phonologically it comes after the group; and since the transliteration is always phonetic, the preprocessor will take the vowel from where it belongs phonetically and place it where it belongs graphically.

Finally the preprocessor is needed for the simple task of inserting \- commands (discretionary hyphens) at the appropriate locations: since characters and ligatures are often constructed from other characters, or belong to several font tables, there is little hope for getting efficient hyphenation patterns so that TEX can hyphenate as it does for Western languages.

1.2 The interna of *Indica*

The preprocessor *Indica* is written in a special way, allowing easy changes and expansions, thanks to the use of **Flex**. Flex is a lexical analyzer, released under GNU copyleft; it generates C code out of simple pattern matching instructions. The advantage of

[1] One could compare this situation to the existence of Antiqua, old German, and Irish types for the same alphabet (a differentiation sadly missing from the ISO/IEC 10646/UNICODE encoding).

Flex is that without being a good programmer one can make powerful and error-free C programs.

How does it work? The minimal Flex file is of the form

```
%{
%}
%%
...lines of code...
%%
main()
{
yylex();
}
```

where the *lines of code* are of the form

```
xyz   { do_this(); do_that(); }
```

`xyz` is a pattern which may appear in the input file, and `do_this();`, `do_that();` are arbitrary C commands, executed whenever the pattern is matched in the input file. This scheme is extremely powerful, since patterns can be arbitrary regular expressions. Suppose, for example, that you want to write a program which finds all TEX commands followed by a blank and adds an empty group to them, if needed (to avoid getting TEXis beautiful, as most TEX users did at least once in their lives): `\TeX␣` shall be replaced by `\TeX{}` and so on, for every command followed by a blank. You can with the following single line of Flex code:

```
"\\"[a-zA-Z]+/" " { ECHO; printf("{}"); }
```

The double quotes indicate verbatim mode, the double backslash is the usual C notation to obtain a backslash in a string, `[a-zA-Z]+` is a regular expression meaning "one or more lowercase/uppercase letters" and finally `/" "` means "this pattern should be matched only if followed by `" "` (a blank)". The `ECHO;` command transmits the input pattern to the output, and `printf{}` adds the `{}` string.

The reader may now have realized the power and ease of use of Flex. Moreover, the generated C code is automatically optimized for the platform on which Flex is run so that one can be sure that the code will compile without problems into a quick and smooth executable.

Indica is written in Flex. To obtain an executable, you will have to run Flex first and then C. The necessary steps are explained in section 1.3.1. Having read the excellent book *lex & yacc* by Levine, Mason and Brown (1992) the user will be able to adapt *Indica* to his/her personal needs, if these are not already covered by the broad range of *Indica*'s input encodings.

1.3 Guidelines for the use of *Indica*

1.3.1 How to install *Indica*

Indica is written in Flex, the GNU version of the standard UNIX utility Lex.[2] On the server you will find executables for Macintosh and MS-DOS. If you are on some other platform, or if you want to make changes to the `indica.lex` file, you will have to compile it again. This operation consists of the following (relatively straightforward) steps:

1. run Flex on `indica.lex`, with the `-8` option:

   ```
   flex -8 indica.lex
   ```

2. Flex will create the file `lex.yy.c` (`LEX_YY.C` on MS-DOS); this is a machine-generated, C++ compatible, ANSI C code file. Run your favourite C-compiler on it, and link the result with the standard ANSI C libraries.

After having fetched or compiled your own executable of *Indica*, you can use it. For this you must prepare your document using the syntax explained in section 1.5, and run *Indica* to produce a regular TEX or LATEX file. *Indica* uses the standard C input and output streams, so you have to type `<` and `>` to redirect these streams to your files:

```
Indica < foo.inp > foo.tex
```

where `foo.inp` is the document you prepared and `foo.tex` is the TEX file *Indica* will create for you.

In this way *Indica* can be used as a filter for piping operations: if your operating system allows piping and your TEX implementation uses the standard input stream, you can systematically write `Indica < foo.inp | TeX` to pre-process `foo.inp` and run TEX on the result, avoiding thereby the creation of an intermediate TEX file.

1.4 *Indica* input schemes

TEX can handle only 8-bit fonts (fonts with 256 characters at most). This seems more or less sufficient for the needs of a certain number of Western European languages, but is definitely unsuitable for Oriental scripts like the Sinhalese one.[3] Hence, the use of a preprocessor is unavoidable. *Indica* will allow the use of the same input scheme(s) for all Indic languages: one will be able to write multilingual Indic documents without changing the input conventions, whenever a language switch occurs. There are

[2] Actually it uses a very important feature of Flex which is not part of the POSIX Lex standard, namely *exclusive states*. *Indica* has to be compiled on a Lex version with this feature; see Levine, Mason, and Brown (1992) for more details.

[3] The TEX extension Ω (Plaice, 1994; Haralambous and Plaice, 1994) will solve these (and many more) problems by using internally the UNICODE encoding, and 16-bit virtual fonts for the output.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.																
1.																
2.																
3.	Usual 7-bit ASCII (ISO 646)															
4.																
5.																
6.																
7.																
8.							ä*								Ä*	
9.																ṟ
A.					ñ*	Ñ*	l̃	ṁ	ă̄	ī̆	ū̆			ṉ		
B.						á̄	à̄	í̄	ì̄	ĕ*	Ĕ*	ŏ*	Ŏ*	ú̄	ù̄	
C.	ē*	Ē*	ō*	Ō*	ǟ*	Ǟ*	ṛ́	ṛ̀	n̆*	N̆*	m̆*	M̆*				ṝ́
D.	ã	ĩ	ũ	ẽ	õ	ĕ	ŏ	ḻ				Ẽ*	Õ*			Ḻ*
E.	ā	ß	Ā	ī	Ī	ū	Ū	ṛ	Ṛ	ṝ	Ṝ	ḷ	Ḷ	ḹ	Ḹ	ṅ
F.	Ṅ	ṭ	Ṭ	ḍ	Ḍ	ṇ	Ṇ	ś	Ś	ṣ	Ṣ	√	ṃ	Ṃ	ḥ	Ḥ

Table 1: The CSX 8-bit input encoding

Code positions followed by * are extensions of CSX proposed hereby by the author. The gray square ■ denotes positions which have not yet been determined.

The author would like to point out that even if certain characters are usually not used in uppercase form, they could very well appear inside all-caps text; so, IHHO, all characters should be included in the table in lowercase and uppercase form. Uppercase letters missing from the table are: Ṟ, Ĩ, Ṁ, Ă̄, Ī̆, Ū̆, Ṉ, Á̄, À̄, Í̄, Ì̄, Ú̄, Ù̄, Ṛ́, Ṛ̀, Ṝ́, Ã, Ĩ, Ũ, Ĕ and Ŏ (a total of 21 codes).

four possible input schemes, common to Hindi, Sanskrit, Bengali, Tamil, Telugu, Malayalam, Kannada, Oriya, Gujarati, Gurmukhi, Sinhalese and Tibetan:

1. `SEVENBIT`, a 7-bit (ISO 646) encoding scheme, based on Frans Velthuis' Hindi/Sanskrit transcription. Some extensions were necessary for Sinhalese, but also for other Indic languages, to obtain the character set of the (Indic part) of UNICODE/ISO 10646-1 standard (ISO, 1993).
2. `CSX`, the *Classical Sanskrit Extended* encoding, an 8-bit extension of ISO 646, proposed by an ad hoc committee, at the 8th World Sanskrit Conference, in Vienna 1990 (Wujastyk, 1991) (Table 1). For Sinhalese and other Indic languages some necesssary extensions were included in the character set of the (Indic part) of UNICODE/ISO-IEC 10646-1 standard (ISO, 1993).
3. `LATEX`, a standardized form of LaTeX commands (for example, only `\d{m}` is valid for 'ṃ', and not `\d m` or `\d{}m` or `\def\foo{\d{m}}\foo`, etc.), describing the "standard" transliteration of Indic languages.
4. `UNICODE`, the 16-bit version of ISO/IEC 10646-1 (see ISO, 1993), with an anticipated Sinhalese encoding by the author (since Sinhalese is not yet part of ISO 10646).[4]

[4] Although there is not a broad choice of UNICODE-compatible software yet (Windows NT is the most popular case of such software), the author believes that UNICODE is already now the ideal solution for *document storage* and *transmission*, especially when used in conjunction with a markup language like SGML.

The reader will find a complete table of equivalences between (1), (2) and (3), applied to Sinhalese, in Table 4.

1.5 The *Indica* syntax

Three kinds of predefined *Indica* commands exist:

1. commands affecting the input mode:
   ```
   #SEVENBIT
   #CSX
   #LATEX
   #UNICODE
   ```
   as described in 1.4.
2. commands determining the current (Indic) language:
   ```
   #BENGALI
   #GUJARATI
   #GURMUKHI
   #HINDI
   #KANNADA
   #MALAYALAM
   #ORIYA
   #SANSKRIT
   #SINHALESE
   #TAMIL
   #TELUGU
   #TIBETAN
   #NIL
   ```
   the last one being used to return for arbitrary non-Indic text to non-preprocessed mode.
3. the
   ```
   #ALIAS
   ```
   command, which allows creation of new names for the commands listed above.

Here are the rules you have to follow when using these commands:

- the "escape character" for *Indica* commands (or should I say "directives"?) is `#`. A command name consists of this character, followed by at most 32 *uppercase letters* or *8-bit characters* (in the range `0x80`–`0xff`). It follows that you can write, for example, '`#NIL;`' or '`#NILthis`', but *not* '`#NILYannis`'; in the latter case you can either leave a blank space ('`#NIL␣Yannis`') or insert an empty group ('`#NIL{}Yannis`') or apply any other similar TEXtrick.
- TEX and LATEX commands are not affected by the preprocessor. Be careful, though, because command *arguments* will nevertheless be preprocessed: if you write

  ```
  #HINDI mohan \TeX\ raake"s
  \begin{center} mis paal
  ```

  then, `\TeX` and `\begin` will be left unchanged by the preprocessor, while `center` will produce चेनतेर and `\begin{चेनतेर}` is hardly something standard LATEX would accept. In these cases it is advised to write

  ```
  #HINDI mohan \TeX\ raake"s
  #NIL\begin{center}#HINDI mis paal
  ```

- *Indica* commands are *not* nested: if you switch to Bengali and then Hindi, you will have to type `#BENGALI` once again to return to the former language (there is no "group closing" command, bringing you back to the state you were before, as in TEX for example).
- Input mode switching commands (`#SEVENBIT`, `#CSX`, etc.) can appear anywhere in the text. They don't produce any immediate effect when in `NIL` language; the corresponding input mode is stored and applied on forthcoming Indic text. Default settings (applied automatically at the begining of every file) are the `NIL` language, and `SEVENBIT` input mode.
- The `ALIAS` command has the following syntax:

  ```
  #ALIAS SINHALESE FOO
  ```

  which has to be written *at the beginning of a line*. The first argument is the command name for which we want to create an alias; the second argument is the alias itself. After the definition above, you can use `#FOO` instead of `#SINHALESE`.

  You can use *uppercase* Latin alphabet letters, or *8-bit characters* in aliases. For example, you could define

  ```
  #ALIAS MALAYALAM M
  #ALIAS NIL N
  ```

  and afterwards type only `#M` to switch to Malayalam, and `#N` to switch back to `NIL` language. Or, you could define

  ```
  #ALIAS MALAYALAM മലയാളം
  ```

  provided your platform has a graphic interface allowing Sinhalese screen display (Macintosh, Windows, X-Window...) and provided the encoding you use places Malayalam characters in the upper 8-bit range.

  Numbers cannot be part of aliases, so the usual TEX operators `#1`, `#2`, `##1`... are not affected by *Indica*. More generally, whenever *Indica* encounters a hash mark followed by an unknown string (not a predefined command name or previously defined alias), it leaves both the hash mark and the string untouched.[5]
- *Indica* does not take TEX comment marks into consideration. If you write

  ```
  % This is a TeX comment
  %#TIBETAN
  % etc etc
  ```

  unlike TEX, *Indica* will read these lines and switch to Tibetan language.
- *Indica* will read only the files you ask it to read; it will not interpret (LA)TEX `\input` commands.[6] On the other hand, a file already processed by *Indica* does not contain any *Indica* commands any more, so that you can re-process it an arbitrary number of times without altering it. It follows that you could write a batch file to run *Indica* on all files of your working directory, just to be sure that no file has been left unprocessed.

1.6 Simultaneous text and transcription

If you write your Sinhalese text in `LATEX` input mode, you can copy and paste it to some other part of the document and run it in `NIL` language mode; it will produce the "standard" Latin transcription of the same text. The only precaution you need to take is to include Christina Thiele's TEX macro `\diatop` (see Thiele, 1987), in the preamble of your document. This macro typesets characters with double or triple diacritization (like $\bar{\bar{a}}$, $\bar{\bar{\dot{r}}}$, etc.)

[5] !'Cuidado! If you mispell an *Indica* command, you will end up with a hash mark and the misspelled string in your (LA)TEX code and should prepare yourself to get a very mean (LA)TEX error message: (LA)TEX just hates useless hash marks.

[6] This feature could be implemented in *Indica*, but would result in a loss of portability: every TEX implementation has its own environment variables for file path searching. The same environment variables should be included into *Indica*'s code, so that exactly the same files may be found and opened.

Here is the expansion of this macro:

```
\def\diatop[#1|#2]{{\setbox1=\hbox{{#1{ }}}%
        \setbox2=\hbox{{#2{ }}}%
        \dimen0=\ifdim\wd1>\wd2\wd1\else\wd2\fi%
        \dimen1=\ht2\advance\dimen1by-1ex%
        \setbox1=\hbox to1\dimen0{\hss#1\hss}%
        \rlap{\raise1\dimen1\box1}%
        \hbox to1\dimen0{\hss#2\hss}}}%
%e.g. of use:
%  \diatop[\'|{\=o}] gives o macron acute
```

An example of simultaneous text and transcription (illustrating the use of aliases as well) is:

```
#LATEX
#S s\d{r}tuva #N (s\d{r}tuva) season,
#S aitih\={a}sika #N (aitih\={a}sika)
historical, #S au\d{s}adha #N (au\d{s}adha)
medicines, #S \d{n}aya #N (\d{n}aya) loan,
#S ko\b{l}a\u{m}ba #N (ko\b{l}a\u{m}ba)
Colombo, \ldots
```

and (after preprocessing by *Indica*) TEX will typeset

ඍතුව (sṛtuva) season, ඓතිහාසික (aitihāsika) historical, ඖෂධ (auṣadha) medicines, ණය (ṇaya) loan, කෝළඹ (koḻaṁba) Colombo, ...

2 Sinhalese TEX

2.1 Introduction to the Sinhalese writing system

Sinhalese is one of the two major languages spoken in Sri Lanka (Ceylon), the second being Tamil. Sinhalese script is a South-Indian script, close to Malayalam and Kannada. The alphabet consists of 18 vowels and 35 consonants. It is a syllabic script: the basic consonant glyph form denotes the consonant followed by the (inherent) short vowel 'a': ක is 'ka', ඛ is 'kha' etc. To obtain a consonant not followed by a vowel, one uses a special diacritic called *virāma*. Hence, ක් is 'k', ඛ් is 'kh', etc. In contrast to Hindi, a virāma is used in all circumstances, *even* at the end of a word.

Here are the 35 Sinhalese consonants (with inherent short 'a' vowel):

ක ඛ ග ඝ ඩ ච ඡ ජ ඣ ඤ ට ඨ
ඩ ඪ ණ ත ථ ද ධ න ප ඵ බ භ
ම ය ර ල ළ ව ශ ෂ ස හ ෆ

There are also 6 nasalized consonants:

ඟ ඦ ඬ ඳ ඹ ඥ

The vowels have full-size glyphs when they appear at word beginning:

අ ආ ඇ ඈ ඉ ඊ උ ඌ ඍ ඎ
ඏ ඐ එ ඒ ඓ ඔ ඕ ඖ

A vowel following a consonant (or a series of consonants) is denoted by a special stroke, or certain auxiliary characters appearing on the right or on the left of the consonant. Here are the letters ක (ka) and ද combined with each one of these vowels:

ක කා කැ කෑ කි කී කු කූ කෘ
කෲ කෙ කේ කෛ
කො කෝ කෞ
ද දා දැ දෑ දි දී දු දූ දෘ දෲ දෙ දේ
දෛ දො දෝ දෞ

Special care must be taken in three cases:

1. When a consonant without vowel is followed by ර (r): the virāma sign of the consonant and the 'r' character are replaced by a special stroke under the consonant. For example, instead of ක්‌ර we will write ක්‍ර.

   If the *consonant* + *'r'* combination is followed by a vowel, then special rules apply. Here are the letters ක and ද (as above) combined with ර and each one of the vowels above:

   ක්‍ර ක්‍රා ක්‍රැ ක්‍රෑ ක්‍රි ක්‍රී ක්‍රෘ ක්‍රෲ
   ක්‍රෙ ක්‍රේ ක්‍රෛ ක්‍රො ක්‍රෝ ක්‍රෞ
   ද්‍ර ද්‍රා ද්‍රැ ද්‍රෑ ද්‍රි ද්‍රී ද්‍රෘ ද්‍රෲ ද්‍රෙ ද්‍රේ
   ද්‍රෛ ද්‍රො ද්‍රෝ ද්‍රෞ.

2. When a consonant without vowel is followed by ය (y): the virāma sign of the consonant disappears, and ය is replaced by the pseudo-letter ්‍ය. For example, instead of ක්‌ය we will write ක්‍ය.

   If the *consonant* + *'y'* combination is followed by a vowel, then special rules apply. Here are the letters ක and ද (as above) combined with ය and each one of the vowels above:

   ක්‍ය ක්‍යා ක්‍යැ ක්‍යෑ ක්‍යි ක්‍යී
   ක්‍යු ක්‍යූ ක්‍යෘ ක්‍යෲ ක්‍යෙ ක්‍යේ
   ක්‍යෛ ක්‍යො ක්‍යෝ ක්‍යෞ

ශ්‍රා ක්‍රෝ ක්‍රෞ

ද්‍ය ද්‍යා ද්‍යෑ ද්‍යෑ ද්‍යි ද්‍යී ද්‍යු ද්‍යූ ද්‍යෘ
ද්‍යෲ ද්‍යෟ ද්‍යෳ ද්‍යෙ ද්‍යේ ද්‍යෛ
ද්‍යො ද්‍යෝ ද්‍යෞ.

3. A third special case occurs, when a consonant [except 'r' itself??] with inherent short 'a' vowel is *preceded* by 'r'. In that case the 'r' is not written and a spiral-like stroke is added on top of the consonant. For example, instead of ර්ල we will write ල්. This phenomenon does not occur when the consonant is followed by some other vowel than 'a'. Here are all consonants with 'r' spiral strokes:

ක් ඛ් ග් ඝ් ඞ් ච් ඡ් ජ් ඣ් ඤ් ට්
ඨ් ඩ් ඪ් ණ් ත් ථ් ද් ධ් න් ප් ඵ්
බ් භ් ම් ය් ල් ළ් ව් ශ් ෂ් ස් හ්
ෆ්

Beside the special cases enumerated above, frequently ligatures occur between consonants. A ligature between two consonants implies that the first one is not followed by a vowel; the virāma sign is omitted in that case. Here are some examples:

ජ් + ඤ = ඥ ක් + ව = ක්ව
ක් + ෂ = ක්ෂ න් + ධ = න්ධ
න් + ථ = න්ථ න් + ද = න්ද
ත් + ථ = ත්ථ

Finally there are two special signs: *anusvara* (ṃ) written ◌ං and *visarga* (ḥ) written ◌ඃ. Sinhalese punctuation follows the English rules. Hyphenation is done between syllables, i.e., *after a vowel*.

2.2 Design of the Sinhalese font

Because of the nature of Sinhalese syllables, most combinations of consonants and vowels had to be drawn separately (the reader can find a complete table of consonant/vowel combinations in Table 3). This brought the total number of distinct character positions to more than 460, placed in three 256-character tables. Despite the large number of characters, the design of a Sinhalese font does not require a superhuman effort; in fact, the shapes of many Sinhalese letters are *modular*, and can be produced by assembling elementary strokes in different ways.

To illustrate this feature of Sinhalese letters, here is a selection of such elementary strokes[7]:

1. on the *left* side of the letter: (α) the left stem of ඝ, (β) same as α, but with an horizontal bar, as in ඩ, (γ) the left stem of ය, (δ) a lowered closed loop, as in ඹ;
2. the *middle* part of the letter: (κ) a simple baseline stroke, as in ප, (λ) the same with a pinch, as in ඩ, (μ) the same with a "bridge" as in ඝ;
3. on the *right* part of the letter: (χ) a short stroke with a rounded loop, as in ඝ, (ψ) a somewhat higher stroke with a triangular loop, as in ඵ, (ω) a high and round stroke without loop, as in ච.

Out of the combinations of these four left parts, three middle parts and three right parts we will make a table to see how many of them actually exist (NE = "does not exist"):

	α			β			γ			δ		
	κ	λ	μ	κ	λ	μ	κ	λ	μ	κ	λ	μ
χ	ප	ඝ	NE	NE	ඝ	ඝ	NE	ය	NE	NE	NE	NE
ψ	ඵ	NE	NE	ඵ	ඝ	NE	ථ	ය	NE	NE	NE	NE
ω	ච	ඩ?	NE	ච	ඩ	NE	ට	ධ	NE	ම	ඹ	ඹ

As we see, more than half of the entries represent extant characters. Similar phenomena occur for other groups of Sinhalese letters. And of course there are also some isolated cases, which have to be drawn separately (like ණ, ඔ, ළ and so forth).

This modularity of Sinhalese forms makes the choice of METAFONT for the realization of a Sinhalese font even more interesting. The Sinhalese font, as presented in this paper, was commissioned from the author by the Wellcome Institute for the History of Medicine, following a proposal by Dominik Wujastyk (to whom the author would like to express his gratitude).[8] The character forms were inspired by the font of Godakumbura (1980), compared to the forms of Disanayaka (a modern Sinhalese script method; 1993), Clough (a classical 19th century dictionary with many ligatures, 1892) and Белькович (the Russian "official" Sinhalese dictionary, 1983), the last one having the most beautiful type, in the author's humble (and non-Sinhalese native) opinion. Useful information was also found in Lambert (1983), a study of south Indian scripts, and the catalogues of writing systems of the world (Nakanishi, 1980 and Faulman, 1880).

[7] Unfortunately the author does not know the original names of these strokes.

[8] See Somadasa (1994) for the first book printed using this Sinhalese system.

	8 pt		9 pt		10 pt		12 pt	
`FX`	.369 pt	+6.25%	.401 pt	+2.777%	.434 pt	0%	.510 pt	−2.08%
`FY`	.347 pt	0%	.391 pt	0%	.434 pt	0%	.521 pt	0%
`shthin`	.217 pt	+12.21%	.217 pt	+10.96%	.217 pt	0%	.217 pt	−15.79%
`shfat`	.906 pt	+10%	.972 pt	+6.66%	.998 pt	0%	1.106 pt	−6.67%
`usual_left`	.406 pt	+10%	.422 pt	+5%	.434 pt	0%	.495 pt	−5%
`usual_right`	.406 pt	+10%	.422 pt	+5%	.434 pt	0%	.495 pt	−5%

Table 2: Scaling of font parameters for optical correction

2.2.1 Optical scaling

As we all know, one of the big advantages of METAFONT drawn characters is optical scaling, that is scaling of characters in a non-linear way, to correct certain optical effects. This technique has been applied by D.E. Knuth, in the Computer Modern fonts, the first realistic example of a font family drawn in METAFONT.

The same technique has been used for Sinhalese. Here are the (technical) details: Sinhalese characters have been designed using 6 main parameters:

1. `FX`, horizontal basic unit;
2. `FY`, vertical basic unit; (in the Computer Modern fonts the same basic unit is used horizontally and vertically, namely `u`). In cases where a length/width had to be defined independently of its orientation, we have used `.5[FX,FY]` (the mean value).
3. `shthin`, the width of thin strokes;
4. `shfat`, the width of a certain number of fat strokes; (in fact, for intermediate cases the variable quantity λ`[shthin,shfat]`, with $\lambda \in [0,1]$ has been used).
5. `usual_left`, the standard left sidebearing;
6. `usual_right`, the standard right sidebearing.

Optical correction consisted in scaling these parameters differently for 8, 9 and 12 points, as in table 2 (the reader can see in the second column the percentage of deviation from the hypothetical linearly scaled value).

As the reader can see, the value of `shthin` remains the same from 8 to 12 points; this guarantees that thin strokes will not disappear in small pointsizes (and makes letters look more elegant in large pointsizes, as in Roman Bodoni fonts). The horizontal basic unit `FX` gets (proportionally) bigger in small sizes: letters become up to 6.25% wider; `FX` also gets slightly smaller at 12 points: letters become 2.08% narrower. The same tactic is applied to sidebearings.

The following sample of text illustrates optical correction. The same text (taken from Бельковичъ, 1983), is typeset in 8, 9, 10 and 12 point sizes.

රැසියන් සිංහල ශබ්ද කොෂය සකස්කිරීමෙහි ලා දෙස්තර දූදිගම වි. රැද්‍රිගු ගේන් ලැබුණු විශාල සහාය ගැන ඔහුට සම්පාදක වරයා සෘතඥතාවය පුද කරයි.

ශ්‍රී ලංකාව රැසියන් බස හදරන සිංහල ජනතාවටත්, සෝවියත් සංගමයෙහි සිංහල බස හදරන රැසියන් ජනතාවටත් මෙම ශබ්ද කොෂය ප්‍රයෝජනවත් වෙතියි සම්පාදක වරයා ප්‍රාර්ථනා කරයි.

රැසියන් සිංහල ශබ්ද කොෂය සකස්කිරීමෙහි ලා දෙස්තර දූදිගම වි. රැද්‍රිගු ගේන් ලැබුණු විශාල සහාය ගැන ඔහුට සම්පාදක වරයා සෘතඥතාවය පුද කරයි.

ශ්‍රී ලංකාව රැසියන් බස හදරන සිංහල ජනතාවටත්, සෝවියත් සංගමයෙහි සිංහල බස හදරන රැසියන් ජනතාවටත් මෙම ශබ්ද කොෂය ප්‍රයෝජනවත් වෙතියි සම්පාදක වරයා ප්‍රාර්ථනා කරයි.

රැසියන් සිංහල ශබ්ද කොෂය සකස්කිරීමෙහි ලා දෙස්තර දූදිගම වි. රැද්‍රිගු ගේන් ලැබුණු විශාල සහාය ගැන ඔහුට සම්පාදක වරයා සෘතඥතාවය පුද කරයි.

ශ්‍රී ලංකාව රැසියන් බස හදරන සිංහල ජනතාවටත්, සෝවියත් සංගමයෙහි සිංහල බස හදරන රැසියන් ජනතාවටත් මෙම ශබ්ද කොෂය ප්‍රයෝජනවත් වෙතියි සම්පාදක වරයා ප්‍රාර්ථනා කරයි.

රැසියන් සිංහල ශබ්ද කොෂය සකස්කිරීමෙහි ලා දෙස්තර දූදිගම වි. රැද්‍රිගු ගේන් ලැබුණු විශාල සහාය ගැන ඔහුට සම්පාදක වරයා සෘතඥතාවය පුද කරයි.

ශ්‍රී ලංකාව රැසියන් බස හදරන සිංහල ජනතාවටත්, සෝවියත් සංගමයෙහි සිංහල බස හදරන රැසියන් ජනතාවටත් මෙම ශබ්ද කොෂය ප්‍රයෝජනවත් වෙතියි සම්පාදක වරයා ප්‍රාර්ථනා කරයි.

2.3 "Do I need BigTEX for all those macros?"

Sorry to disappoint you, but there are no macros. *Indica* does all the work for you and its output is rather unreadable for a human—but quite readable for TEX. With LATEX 2_ε and the `T1` (Cork) encoding you only need to place the files `T1sinha.fd`,

`T1sinhb.fd`, `T1sinhc.fd` in the same place as your other FD files, and write

```
\newcommand{\SHa}{\fontfamily{sinha}%
                  \selectfont}
\newcommand{\SHb}{\fontfamily{sinhb}%
                  \selectfont}
\newcommand{\SHc}{\fontfamily{sinhc}%
                  \selectfont}
```

in the preamble of your file. If you wish to install the Sinhalese fonts in a more formal manner, recognizing the encoding of the font as being different from `T1` (we call it `SH1`), then you only need to place files `SH1sinha.fd`, `SH1sinhb.fd`, `SH1sinhc.fd` together with the other FD files you use, and use the package `sinhala.sty` when you run LaTeX 2ε. So you would begin your document like this:

```
\documentclass{article}
\usepackage{sinhala}
\begin{document}
...
```

This method is not recommended, however, if you switch frequently from Latin to Sinhalese and your machine is not very powerful: LaTeX 2ε reads a file (called `nfsh1.def`) everytime you switch encodings; even if this file is very short, the open/close operations may slow down TeX. The author hopes that this problem will be solved in future releases of LaTeX 2ε.

If you are not working with LaTeX 2ε then you have to define the fonts manually, remembering that they always come in triplets, like

```
\font\SHa=sinha10
\font\SHb=sinhb10
\font\SHc=sinhc10
```

The available point sizes are 8, 9, 10 and 12. Please contact the author if you need other point sizes, or scale the ones you have linearly. There is no bold or slanted style yet (although it would be straightforward to obtain them out of the METAFONT code), because the author has never seen such forms. Any information on Sinhalese typographical traditions and aesthetics would be most welcome.

References

A.A. Белькович රුසියානුසිංහල ශබ්දකෝෂය *(Русско-Сингальский Словарь)*. Русский Язык, Москва, Россиа, 1983.

Rev. B. Clough. සිංහල ඉංග්‍රීසි අකාරාදිය *(Sinhalese-English Dictionary)*. Wesleyan Mission Press, Kollupitiya, Sri Lanka, 1892, facsimile edition by Asian Educational Services, New Delhi, 1982.

J.B. Disanayaka. *Let's read and write Sinhala*. Pioneer Lanka Publications, London, 1993.

C. Faulman. *Das Buch der Schrift, enthaltend die Schriftzeichen und Alphabete aller Zeiten und aller Völker des Erdkreises*. Druck und Verlag der kaiserlich-königlichen Hof- und Staatsdruckerei, Wien, 1880.

C.E. Godakumbura. *Catalogue of Ceylonese Manuscripts*. The Royal Library, Copenhagen, 1980.

Y. Haralambous and J. Plaice. "First Applications of Ω: Greek, Arabic, Khmer, Poetica, ISO 10646/UNICODE, etc.". In *Proceedings of the 15th TeX Users Group Annual Meeting (Santa Barbara). TUGboat*, **15** (3), pp. 344-352, 1994.

ISO. *Information technology — Universal Multiple-octet Coded Character Set*. ISO/IEC 10646-1:1993(e) edition, 1993.

H.M. Lambert *Introduction to the Scripts of South India and Ceylon, manuscript prepared as a companion to: Introduction to the Devanagari Script, for Students of Sanskrit, Hindi, Marathi, Gujarati and Bengali*. Oxford University Press, 1983.

J. Levine, T. Mason, and D. Brown. *lex & yacc*. O'Reilly & Associates, Inc., Sebastopol, California, 1992.

A. Nakanishi. *Writing systems of the World*. Charles E. Tuttle Company, Tokyo, 1980.

J. Plaice. "Progress in the Ω Project". In *Proceedings of the 15th TeX Users Group Annual Meeting (Santa Barbara)*. 1994. *TUGboat*, **15** (3), pp. 320-324, 1994.

K.D. Somadasa. *Catalogue of the Sinhalese Manuscripts in the Wellcome Institute for the History of Medicine*. Wellcome Institute, London, 1994.

C. Thiele. "TeX, Linguistics and Journal Production". In *TeX Users Group Eighth Annual Meeting, Seattle, August 24–26, 1987*. 1987.

D. Wujastyk. "Standardization of Romanized Sanskrit for Electronic Data Transfer and Screen Representation". *Sesame Bulletin*, **4**(1), 27–29, 1991.

◇ Yannis Haralambous
187, rue Nationale
59800 Lille, France.
Email: `haralambous@univ-lille1.fr`

Table 3: Sinhalese consonants and vowel combinations

Part a. Without vowel, and vowels 'a'–'r̥̄'											
		a	ā	ä	ǟ	i	ī	u	ū	ṛ	ṝ
		ෙ	ො	ෙැ	ෙෑ	ි	ී	ු	ූ	ෘa	ෘaa
ka	ක්	ක	කා	කැ	කෑ	කි	කී	කු	කූ	කෘa	කෘaa
kha	ඛ්	ඛ	ඛා	ඛැ	ඛෑ	ඛි	ඛී	ඛු	ඛූ	ඛෘa	ඛෘaa
ga	ග්	ග	ගා	ගැ	ගෑ	ගි	ගී	ගු	ගූ	ගෘa	ගෘaa
gha	ඝ්	ඝ	ඝා	ඝැ	ඝෑ	ඝි	ඝී	ඝු	ඝූ	ඝෘa	ඝෘaa
ṅa	ඞ්	ඞ	ඞා	ඞැ	ඞෑ	ඞි	ඞී	ඞු	ඞූ	ඞෘa	ඞෘaa
ca	ච්	ච	චා	චැ	චෑ	චි	චී	චු	චූ	චෘa	චෘaa
cha	ඡ්	ඡ	ඡා	ඡැ	ඡෑ	ඡි	ඡී	ඡු	ඡූ	ඡෘa	ඡෘaa
ja	ජ්	ජ	ජා	ජැ	ජෑ	ජි	ජී	ජු	ජූ	ජෘa	ජෘaa
jha	ඣ්	ඣ	ඣා	ඣැ	ඣෑ	ඣි	ඣී	ඣු	ඣූ	ඣෘa	ඣෘaa
ña	ඤ්	ඤ	ඤා	ඤැ	ඤෑ	ඤි	ඤී	ඤු	ඤූ	ඤෘa	ඤෘaa
ṭa	ට්	ට	ටා	ටැ	ටෑ	ටි	ටී	ටු	ටූ	ටෘa	ටෘaa
ṭha	ඨ්	ඨ	ඨා	ඨැ	ඨෑ	ඨි	ඨී	ඨු	ඨූ	ඨෘa	ඨෘaa
ḍa	ඩ්	ඩ	ඩා	ඩැ	ඩෑ	ඩි	ඩී	ඩු	ඩූ	ඩෘa	ඩෘaa
ḍha	ඪ්	ඪ	ඪා	ඪැ	ඪෑ	ඪි	ඪී	ඪු	ඪූ	ඪෘa	ඪෘaa
ṇa	ණ්	ණ	ණා	ණැ	ණෑ	ණි	ණී	ණු	ණූ	ණෘa	ණෘaa
ta	ත්	ත	තා	තැ	තෑ	ති	තී	තු	තූ	තෘa	තෘaa
tha	ථ්	ථ	ථා	ථැ	ථෑ	ථි	ථී	ථු	ථූ	ථෘa	ථෘaa
da	ද්	ද	දා	දැ	දෑ	දි	දී	දු	දූ	දෘa	දෘaa
dha	ධ්	ධ	ධා	ධැ	ධෑ	ධි	ධී	ධු	ධූ	ධෘa	ධෘaa
na	න්	න	නා	නැ	නෑ	නි	නී	නු	නූ	නෘa	නෘaa
pa	ප්	ප	පා	පැ	පෑ	පි	පී	පු	පූ	පෘa	පෘaa
pha	ඵ්	ඵ	ඵා	ඵැ	ඵෑ	ඵි	ඵී	ඵු	ඵූ	ඵෘa	ඵෘaa
ba	බ්	බ	බා	බැ	බෑ	බි	බී	බු	බූ	බෘa	බෘaa
bha	භ්	භ	භා	භැ	භෑ	භි	භී	භු	භූ	භෘa	භෘaa
ma	ම්	ම	මා	මැ	මෑ	මි	මී	මු	මූ	මෘa	මෘaa
ya	ය්	ය	යා	යැ	යෑ	යි	යී	යු	යූ	යෘa	යෘaa
ra	ර්	ර	රා	ඍ	ඎ	රි	රී	රු	රූ	රෘa	රෘaa
la	ල්	ල	ලා	ලැ	ලෑ	ලි	ලී	ලු	ලූ	ලෘa	ලෘaa
va	ව්	ව	වා	වැ	වෑ	වි	වී	වු	ෟ	වෘa	වෘaa
śa	ශ්	ශ	ශා	ශැ	ශෑ	ශි	ශී	ශු	ශූ	ශෘa	ශෘaa
ṣa	ෂ්	ෂ	ෂා	ෂැ	ෂෑ	ෂි	ෂී	ෂු	ෂූ	ෂෘa	ෂෘaa
sa	ස්	ස	සා	සැ	සෑ	සි	සී	සු	සූ	සෘa	සෘaa
ha	හ්	හ	හා	හැ	හෑ	හි	හී	හු	හූ	හෘa	හෘaa
ḷa	ළ්	ළ	ළා	ළැ	ළෑ	ළි	ළී	ළු	ළූ	ළෘa	ළෘaa
fa	ෆ්	ෆ	ෆා	ෆැ	ෆෑ	ෆි	ෆී	ෆු	ෆූ	ෆෘa	ෆෘaa

Part b. Vowels ‘ḷ’–‘au’, anusvara, visarga										
	ḷ	ḹ	e	ē	ai	o	ō	au	aṃ	aḥ
	ඔ	ඕ	එ	ඒ	ඓ	ඔ	ඕ	ඖ	ං	ඃ
ka	කෘ	කෲ	කෙ	කේ	කෛ	කො	කෝ	කෞ	කං	කඃ
kha	ඛෘ	ඛෲ	ඛෙ	ඛේ	ඛෛ	ඛො	ඛෝ	ඛෞ	ඛං	ඛඃ
ga	ගෘ	ගෲ	ගෙ	ගේ	ගෛ	ගො	ගෝ	ගෞ	ගං	ගඃ
gha	ඝෘ	ඝෲ	ඝෙ	ඝේ	ඝෛ	ඝො	ඝෝ	ඝෞ	ඝං	ඝඃ
ṅa	ඞෘ	ඞෲ	ඞෙ	ඞේ	ඞෛ	ඞො	ඞෝ	ඞෞ	ඞං	ඞඃ
ca	චෘ	චෲ	චෙ	චේ	චෛ	චො	චෝ	චෞ	චං	චඃ
cha	ඡෘ	ඡෲ	ඡෙ	ඡේ	ඡෛ	ඡො	ඡෝ	ඡෞ	ඡං	ඡඃ
ja	ජෘ	ජෲ	ජෙ	ජේ	ජෛ	ජො	ජෝ	ජෞ	ජං	ජඃ
jha	ඣෘ	ඣෲ	ඣෙ	ඣේ	ඣෛ	ඣො	ඣෝ	ඣෞ	ඣං	ඣඃ
ña	ඤෘ	ඤෲ	ඤෙ	ඤේ	ඤෛ	ඤො	ඤෝ	ඤෞ	ඤං	ඤඃ
ṭa	ටෘ	ටෲ	ටෙ	ටේ	ටෛ	ටො	ටෝ	ටෞ	ටං	ටඃ
ṭha	ඨෘ	ඨෲ	ඨෙ	ඨේ	ඨෛ	ඨො	ඨෝ	ඨෞ	ඨං	ඨඃ
ḍa	ඩෘ	ඩෲ	ඩෙ	ඩේ	ඩෛ	ඩො	ඩෝ	ඩෞ	ඩං	ඩඃ
ḍha	ඪෘ	ඪෲ	ඪෙ	ඪේ	ඪෛ	ඪො	ඪෝ	ඪෞ	ඪං	ඪඃ
ṇa	ණෘ	ණෲ	ණෙ	ණේ	ණෛ	ණො	ණෝ	ණෞ	ණං	ණඃ
ta	තෘ	තෲ	තෙ	තේ	තෛ	තො	තෝ	තෞ	තං	තඃ
tha	ථෘ	ථෲ	ථෙ	ථේ	ථෛ	ථො	ථෝ	ථෞ	ථං	ථඃ
da	දෘ	දෲ	දෙ	දේ	දෛ	දො	දෝ	දෞ	දං	දඃ
dha	ධෘ	ධෲ	ධෙ	ධේ	ධෛ	ධො	ධෝ	ධෞ	ධං	ධඃ
na	නෘ	නෲ	නෙ	නේ	නෛ	නො	නෝ	නෞ	නං	නඃ
pa	පෘ	පෲ	පෙ	පේ	පෛ	පො	පෝ	පෞ	පං	පඃ
pha	ඵෘ	ඵෲ	ඵෙ	ඵේ	ඵෛ	ඵො	ඵෝ	ඵෞ	ඵං	ඵඃ
ba	බෘ	බෲ	බෙ	බේ	බෛ	බො	බෝ	බෞ	බං	බඃ
bha	භෘ	භෲ	භෙ	භේ	භෛ	භො	භෝ	භෞ	භං	භඃ
ma	මෘ	මෲ	මෙ	මේ	මෛ	මො	මෝ	මෞ	මං	මඃ
ya	යෘ	යෲ	යෙ	යේ	යෛ	යො	යෝ	යෞ	යං	යඃ
ra	රෘ	රෲ	රෙ	රේ	රෛ	රො	රෝ	රෞ	රං	රඃ
la	ලෘ	ලෲ	ලෙ	ලේ	ලෛ	ලො	ලෝ	ලෞ	ලං	ලඃ
va	වෘ	වෲ	වෙ	වේ	වෛ	වො	වෝ	වෞ	වං	වඃ
śa	ශෘ	ශෲ	ශෙ	ශේ	ශෛ	ශො	ශෝ	ශෞ	ශං	ශඃ
ṣa	ෂෘ	ෂෲ	ෂෙ	ෂේ	ෂෛ	ෂො	ෂෝ	ෂෞ	ෂං	ෂඃ
sa	සෘ	සෲ	සෙ	සේ	සෛ	සො	සෝ	සෞ	සං	සඃ
ha	හෘ	හෲ	හෙ	හේ	හෛ	හො	හෝ	හෞ	හං	හඃ
ḷa	ළෘ	ළෲ	ළෙ	ළේ	ළෛ	ළො	ළෝ	ළෞ	ළං	ළඃ
fa	ෆෘ	ෆෲ	ෆෙ	ෆේ	ෆෛ	ෆො	ෆෝ	ෆෞ	ෆං	ෆඃ

Table 4: Table of Devanagari, Tamil, Malayalam and Sinhalese characters and the different input modes

NAME	D	T	M	S	CSX	SEVENBIT	LATEX
ANUSVARA	ं	◌	ം	ං	m, M	.m, M	\d{m}, \d{M}
VISARGA	ः	◌ः	ഃ	ඃ	h, H	.h, H	\d{h}, \d{H}
VOW. A	अ	அ	അ	අ	a, A	a	a, A
VOW. AA	आ	ஆ	ആ	ආ	ā, Ā	aa, A	\={a}, \={A}
VOW. AAA	-	-	-	ඇ	ä, Ä	*"a*, *.A*	\"{a}, \"{A}
VOW. AAAA	-	-	-	ඈ	ā̈, Ā̈	*"aa*, *.AA*	\diatop[\=|\"a], \diatop[\=|\"A]
VOW. I	इ	இ	ഇ	ඉ	i, I	i	i, I
VOW. II	ई	ஈ	ഈ	ඊ	ī, Ī	ii I	\={\i}, \={i} \={I}
VOW. U	उ	உ	ഉ	උ	u, U	u	u, U
VOW. UU	ऊ	ஊ	ഊ	ඌ	ū, Ū	uu, U	\={u}, \={U}
VOW. VOC. R	ऋ	-	ഋ	ඍ	ṛ, Ṛ	.r	\d{r}, \d{R}
VOW. VOC. RR	ॠ	-	ൠ	ඎ	ṝ, Ṝ	.R	\diatop[\=|\d r], \diatop[\=|\d r]
VOW. VOC. L	ऌ	-	ഌ	ඏ	ḷ, Ḷ	.l	\d{l}, \d{L}
VOW. VOC. LL	ॡ	-	ൡ	ඐ	ḹ, Ḹ	.L	\diatop[\=|\d l], \diatop[\=|\d L]
VOW. CANDRA E	ऍ	-	-	-	ĕ	??!!	\u{e}
VOW. SHORT E	ऎ	எ	എ	එ	ĕ, Ĕ	^e	\v{e}, \v{E}
VOW. E	ए	ஏ	ഏ	ඒ	e, E	e	e, E
VOW. AI	ऐ	ஐ	ഐ	ඓ	ai, Ai, AI	ai, E	ai, Ai, AI
VOW. CANDRA O	ऑ	-	-	-	ŏ	??!!	\u{o}
VOW. SHORT O	ऒ	ஒ	ഒ	ඔ	ŏ, Ŏ	^o	\v{o}, \v{O}
VOW. O	ओ	ஓ	ഓ	ඕ	o, O	o	o, O
VOW. AU	औ	ஔ	ഔ	ඖ	au, Au, AU	au, O	au, Au, AU
CONS. KA	क	க	ക	ක	k, K	k	k, K
CONS. KHA	ख	-	ഖ	ඛ	kh, Kh, KH	kh, K	kh, Kh, KH
CONS. GA	ग	-	ഗ	ග	g, G	g	g, G
CONS. GHA	घ	-	ഘ	ඝ	gh, Gh, GH	gh, G	gh, Gh, GH
CONS. NGA	ङ	ங	ങ	ඟ	ṅ, Ṅ	"n	\.{n}, \.{N}
CONS. CA	च	ச	ച	ච	c, C	c	c, C
CONS. CHA	छ	-	ഛ	ඡ	ch, Ch, CH	ch, C	ch, Ch, CH
CONS. JA	ज	ஜ	ജ	ජ	j, J	j	j, J
CONS. JHA	झ	-	ഝ	ඣ	jh, Jh, JH	jh, J	jh, Jh, JH
CONS. NYA	ञ	ஞ	ഞ	ඤ	ñ, Ñ	~n	\~{n}, \~{N}
CONS. TTA	ट	ட	ട	ට	ṭ, Ṭ	.t	\d{t}, \d{T}
CONS. TTHA	ठ	-	ഠ	ඨ	ṭh, Ṭh, ṬH	.th, .T	\d{t}h, \d{T}h, \d{T}H
CONS. DDA	ड	-	ഡ	ඩ	ḍ, Ḍ	.d	\d{d}, \d{D}
CONS. DDHA	ढ	-	ഢ	ඪ	ḍh, Ḍh, ḌH	.dh, .D	\d{d}h, \d{D}h, \d{D}H
CONS. NNA	ण	ண	ണ	ණ	ṇ, Ṇ	.n	\d{n}, \d{N}

NAME	D	T	M	S	CSX	SEVENBIT	LATEX
CONS. TA	त	த	ത	ත	t, T	t	t, T
CONS. THA	थ	-	ഥ	ථ	th, Th, TH	th, T	th, Th, TH
CONS. DA	द	-	ദ	ද	d, D	d	d, D
CONS. DHA	ध	-	ധ	ධ	dh, Dh, DH	dh, D	dh, Dh, DH
CONS. NA	न	ந	ന	න	n, N	n	n, N
CONS. NNNA	ऩ	ண	-	-	**??!!**	??!!	??!!
CONS. PA	प	ப	പ	ප	p, P	p	p, P
CONS. PHA	फ	-	ഫ	ඵ	ph, Ph, PH	ph, P	ph, Ph, PH
CONS. BA	ब	-	ബ	බ	b, B	b	b, B
CONS. BHA	भ	-	ഭ	භ	bh, Bh, BH	bh, B	bh, Bh, BH
CONS. MA	म	ம	മ	ම	m, M	m	m, M
CONS. YA	य	ய	യ	ය	y, Y	y	y, Y
CONS. RA	र	ர	ര	ර	r, R	r	r, R
CONS. RRA	ऱ	ற	റ	-	**??!!**	*"r*	??!!
CONS. LA	ल	ல	ല	ල	l, L	l	l, L
CONS. LLA	ळ	ள	ള	ළ	l̲	L	\b{l}, \b{L}
CONS. LLLA	ऴ	ழ	ഴ	-	**??!!**	*"l*	??!!
CONS. VA	व	வ	വ	ව	v, V	v	v, V
CONS. SHA	श	ஶ	ശ	ශ	ś, Ś	"s	\'s, \'S
CONS. SSA	ष	-	ഷ	ෂ	ṣ, Ṣ	.s	\d{s}, \d{S}
CONS. SA	स	ஸ	സ	ස	s, S	s	s, S
CONS. HA	ह	ஹ	ഹ	හ	h, H	h	h, H
CONS. FA	फ़	-	-	ෆ	f, F	f	f, F
CONS. NAS. GA	-	-	-	ඟ	n̆g, N̆g, N̆G	*Ng*	\u{n}g, \u{N}g, \u{N}G
CONS. NAS. CA	-	-	-	ඤ	n̆c, N̆c, N̆C	*Nc*	\u{n}c, \u{N}c, \u{N}C
CONS. NAS. DDA	-	-	-	ඬ	n̆ḍ, N̆ḍ, N̆Ḍ	*N.d*	\u{n}\d{d}, \u{N}\d{d}, \u{N}\d{D}
CONS. NAS. DA	-	-	-	ඳ	n̆d, N̆d, N̆D	*Nd*	\u{n}d, \u{N}d, \u{N}D
CONS. NAS. BA	-	-	-	ඹ	m̆b, M̆b, M̆B	*Nb*	\u{m}b, \u{M}b, \u{M}B
CONS. NAS. JA	-	-	-	ඦ	n̆j, N̆j, N̆J	*Nj*	\u{n}j, \u{N}j, \u{N}J
CONS. QA	क़	-	-	-	q	q	q
CONS. KHHA	ख़	-	-	-	k̲h	.kh, .K	\b{k}\b{h}
CONS. GHHA	ग़	-	-	-	g̲	.g	\b{g}
CONS. ZA	ज़	-	-	-	z	z	z
CONS. DDDHA	ड़	-	-	-	r̲	R	\b{r}
CONS. RHA	ढ़	-	-	-	r̲h	Rh	\b{r}h
CONS. YYA	य़	-	-	-	**??!!**	*"y*	??!!

NOTES:

— Columns D, T, M, S, stand respectively for Devanagari, Tamil, Malayalam and Sinhalese. The fonts used in this paper for the first three scripts have been made by Frans Velthuis (velthuis@rc.rug.nl), Thomas Ridgeway (165 McGraw Street, Seattle, WA 98109 USA), Jeroen Hellingman (jhelling@cs.ruu.nl) and the author. Some of them are still under β-status, so please contact their respective authors for more information on their availability.

— SEVENBIT column: Entries in slanted style are extensions to Frans Velthuis' transcription, proposed by the author.

The EAN barcodes by TEX

Petr Olšák

Abstract

In this article, we describe the algorithm for the transformation from the EAN 13 code (13-digit number) to the barcode (the sequence of bars and spaces) and we show the implementation of this algorithm to the macro language of TEX. The drawing of the bars is realized by the TEX primitive `\vrule`. Some data from the standard for the EAN barcodes (tolerances and so on) are presented too. The corresponding TEX macro is available on CTAN in `tex-archive/macros/generic/ean`.

– – ∗ – –

I have prepared my first book about TEX written in Czech (Olsak, 1995). My interest in preparing the book didn't end with sending the manuscript or the type matter to the publisher because the publisher is our $\mathcal{CS}$TUG (the Czechoslovak TEX users group). I made the cover design of the book, I worked on the distribution problems like getting the ISBN, and so on.

When I got the ISBN (International Standard Book Number), I converted it to EAN 13 (European Article Numbering) and I took concern about the barcode for this number, because it is commonly used on book covers. I found out that it would be very expensive to have commercial firms make the barcode. On the other hand, using TEX to produce the barcode is a very natural application of this program because of its high accuracy and its algorithmic macro language. To find the description of the conversion algorithm with 13 digits as the input and the barcode metrics as the output was the only problem. This algorithm is described in (Benadikov et al., 1994).

The transformation from ISBN to EAN is simple. The ISBN is a 10-digit number. The dashes between digits divide the ISBN into the fields "country-publisher-number-checksum" and (essentially) can be ignored. First we write three new constant digits (978) at the front of the ISBN. Next we compute a new checksum digit (the last one). The algorithm for computing the ISBN checksum is different from the one for computing the EAN checksum. For the EAN, first we need to compute the sum of digits on the even positions. Let the sum be e. Next we compute the sum of digits on the odd positions (without the checksum digit). Let the sum be o. We evaluate the expression $3 \times e + o$. The difference between the result and the next modulo-10 number is the checksum digit. For example, *The TEXbook* book hard cover (Knuth, 1986) has its ISBN 0-201-13447-0 (0: country USA; 201: publisher Addison Wesley; 13447: internal book number assigned by publisher; 0: the checksum digit). We write the three constant digits at the front and remove the checksum digit to obtain 978020113447?. Now $e = 7{+}0{+}0{+}1{+}4{+}7 = 19$ and $o = 9+8+2+1+3+4 = 27$. The difference between $3 \times 19 + 27 = 84$ and 90 is 6 and this is the checksum digit. We can divide the result by dashes into 6 digit fields (only for easier reading) and the result is 9-780201-134476.

The transformation from the EAN number to the barcode metric is more complicated. The leftmost digit (the 13th position) is 9 for books, but it is different for other kinds of goods. This digit doesn't have its own field in the barcode but it influences the algorithm for the transformation of the following digits to their fields. The widths of the bars or white spaces between the bars are multiples of the basic so-called "module X". The size of module X varies for different SC standards (see below), but the basic size is 0.33 mm. Each digit from positions 12 to 1 is transformed to a field of module width 7X. This field by definition contains two bars and two white spaces. The "start mark" of module width 3X (1X bar, 1X space and 1X bar) is appended before the digit from the 12th position. The same "stop mark" (module width 3X) is appended after the last digit and a so-called "separator mark" of module width 5X (1X space, 1X bar, 1X space, 1X bar and 1X space) is placed between digits on the 7th and 6th positions. The "mark" bars are 5X longer than the bars from the digits.

It is easy to see that the total length of the EAN barcode is 95X. We also have to consider the 11X white space to the left of the code and the 7X white space to the right of the code. These are the minimal white "margins", which are important for the barcodes on a color background. The total number of bars is 30.

Each digit is transformed into two bars in its 7X size field according to one of the tables (A, B and C) shown in Table 1.

Zero in the table stands for the white module (of size 1X) and one means the black module. For example, the digit 4 is converted to 1X space, 1X bar, 3X space and 2X bar by table A and to 2X space, 3X bar, 1X space and 1X bar by table B. Notice that all tables convert the digit into exactly two spaces and two bars and that the converted field starts with the space if table A or B is used, and with the bar if table C is used.

The digits in positions 6 to 1 are transformed by table C under any circumstance. The digits in positions 12 to 7 are transformed by table A or B.

	tab. A	tab. B	tab. C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1010000
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Table 1: Tables A, B and C

13th digit	12	11	10	9	8	7
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Table 2: Dependence on the 13th digit

The choice depends on the value of the digit in the 13th position, as described in Table 2.

For example, if the digit in position 13 is 9 (our case for books), the digits in positions 12, 9 and 7 are transformed by table A and the ones in positions 11, 10 and 8 are transformed by table B.

Now we can show the TEX macro. First we load the special OCRb font for printing out the EAN code in human-readable form. This printout is appended to the barcode. The METAFONT sources of these fonts are available on CTAN; they were created by Norbert Schwarz. I had to make one little correction in these sources: the command "`mode_setup;`" was added to the beginning of the file `ocrbmac.mf`. Starting at line 4 some "variables" are declared.

```
1 \message{The EAN-13 barcodes macro. Copyright (C) Petr Olsak, 1995}
2 \font\ocrb=ocrb9       % for EAN in ``number form''
3 \font\ocrbsmall=ocrb7  % for ISBN
4 \newcount\numlines \newcount\nummodules  % number of bars and of modules.
5 \newcount\numdigit \newcount\evensum \newcount\oddsum  % internal variables
6 \newdimen\X            % the module size X,
7 \newdimen\bcorr        % the bar correction (see below).
8 \newdimen\workdimen \newdimen\barheight   % internal variables
```

The main macro `\EAN` (line 11) converts the 13-digit EAN number to the internal 60-digit number `\internalcode`. Each digit of the `\internalcode` represents the multiple of the X module size for either the white space or the bar. The order of digits is the same as the order of spaces and bars in the code. The odd positions in the `\internalcode` (from the left) stand for the white spaces and the even ones for the bars.

The usage of the macro is `\EAN 9-780201-134476`, for example. The presence of the "-" signs has no significance. The macro reads 13 digits and saves them in `\firstdigit`, `\frontdigits` and `\enddigits`. At this point, the macro converts the input into `\internalcode` using macros `\settables`, `\usetabAB`, `\insertseparator` and `\usetabC`. The `\testchecksum` macro (line 25) checks for the correctness of the last (check-sum) digit of the EAN.

```
9  \def\internalcode{0111}             % Begin mark at start
10 \def\frontdigits{}                  % 12--7 digit of EAN
11 \def\EAN{\begingroup\EANscan}
12 \def\EANscan#1{\if#1-\let\next=\EANscan \else
13     \advance\numdigit by1
14     \ifnum\numdigit<13
15       \ifodd\numdigit \advance\oddsum by #1 \else \advance\evensum by #1 \fi
16       \let\next=\EANscan
17       \ifnum\numdigit=1 \settables#1\def\firstdigit{#1}\else
18       \ifnum\numdigit<8 \usetabAB#1\edef\frontdigits{\frontdigits#1}\else
19       \ifnum\numdigit=8 \insertseparator \A \usetabC #1\def\enddigits{#1}%
```

```
20      \else             \usetabC#1\edef\enddigits{\enddigits#1}%
21      \fi\fi\fi
22    \else \testchecksum#1\usetabC#1\edef\enddigits{\enddigits#1}%
23      \let\next=\EANclose
24    \fi\fi \next}
25 \def\testchecksum#1{\multiply\evensum by3 \advance\evensum by\oddsum
26    \oddsum=\evensum
27    \divide\oddsum by10 \multiply\oddsum by10 \advance\oddsum by10
28    \advance\oddsum by-\evensum \ifnum\oddsum=10 \oddsum=0 \fi
29    \ifnum#1=\oddsum \else
30    \errmessage{The checksum digit has to be \the\oddsum, no #1 !}\fi}
```

At the time of the `\EANclose` expansion (line 31), we close the `\internalcode` by the `\insertendmark`; next we write to the `log` the EAN number in the 13-digit form and in the internal 60-digit representation. The last action is to "run" the macro `\EANbox`, which makes the box with the barcode. The input parameter to this macro is the 60-digit `\internalcode`.

How was the `\internalcode` made? The macros starting at line 35 answer this question. These macros are the tables mentioned above rewritten in the macro language of TEX.

There is no need to define table C in the macro, because table C is the exact "inverse" of table A. When we insert the separator (line 52 of the macro), the odd number of digits (namely 5) is appended to the `\internalcode`. This implies that the parity of the black/white order is changed. Using table `\A` is therefore sufficient for the transformation of the digits in positions 6 to 1 (see line 19).

```
31 \def\EANclose{\insertendmark
32    \wlog{EAN: \firstdigit\space\frontdigits\space\enddigits}%
33    \wlog{EANinternal: \internalcode}%
34    \expandafter\EANbox\internalcode..\endgroup}
35 \def\A{\def\0{3211}\def\1{2221}\def\2{2122}\def\3{1411}\def\4{1132}%
36   \def\5{1231}\def\6{1114}\def\7{1312}\def\8{1213}\def\9{3112}}
37 \def\B{\def\0{1123}\def\1{1222}\def\2{2212}\def\3{1141}\def\4{2311}%
38   \def\5{1321}\def\6{4111}\def\7{2131}\def\8{3121}\def\9{2113}}
39 \def\settables#1{\ifnum#1=0 \def\tabs{\A\A\A\A\A\A}\fi
40                  \ifnum#1=1 \def\tabs{\A\A\B\A\B\B}\fi
41                  \ifnum#1=2 \def\tabs{\A\A\B\B\A\B}\fi
42                  \ifnum#1=3 \def\tabs{\A\A\B\B\B\A}\fi
43                  \ifnum#1=4 \def\tabs{\A\B\A\A\B\B}\fi
44                  \ifnum#1=5 \def\tabs{\A\B\B\A\A\B}\fi
45                  \ifnum#1=6 \def\tabs{\A\B\B\B\A\A}\fi
46                  \ifnum#1=7 \def\tabs{\A\B\A\B\A\B}\fi
47                  \ifnum#1=8 \def\tabs{\A\B\A\B\B\A}\fi
48                  \ifnum#1=9 \def\tabs{\A\B\B\A\B\A}\fi}
49 \def\usetabAB#1{\expandafter\scantab\tabs\end \usetabC #1}
50 \def\scantab#1#2\end{#1\def\tabs{#2}} % The tab #1 is activated and removed
51 \def\usetabC#1{\edef\internalcode{\internalcode\csname#1\endcsname}}
52 \def\insertseparator{\edef\internalcode{\internalcode 11111}}
53 \def\insertendmark{\edef\internalcode{\internalcode 111}}
```

Now comes the most important part of our macro: creating the bars using the TEX primitive `\vrule`. The internal macro `\EANbox` (line 54) does this job. This macro reads the 60-digit `\internalcode` (ended by two dots) as its parameter. It scans two digits per step from the parameter (first digit: the white space; second digit: the black bar) and puts in the appropriate kerns and rules. Each kern/rule pair is corrected by a so-called "bar correction". The standard recommends making each rule thinner than what is exactly implied by the multiple of the X size. This recommendation is due to the ink behavior during the actual printing. For example, for offset process technology, it is recommended to reduce the bar width by 0.020 mm. If the bar width is reduced, the white space must be enlarged by the same amount in order to preserve the global distance between bars.

The bars 1, 2, 15, 16, 29 and 30 have nonzero depth (5X) because these are the lines from the start, the separator and the stop marks. The height of the bars is 69.24 X in the normal case but it may be reduced, if the ISBN is appended to the top of the code. If the `\barheight` is zero, than the implicit height is used. Otherwise the `\barheight` is used. This feature gives the user the possibility to set the bar height individually.

```
\def\EANbox{\vbox\bgroup\offinterlineskip
  \setbox0=\hbox\bgroup \kern11\X\EANrepeat}
\def\EANrepeat#1#2{\if#1.\let\next=\EANfinal \else\let\next=\EANrepeat
  \advance\numlines by1
  \advance\nummodules by#1 \advance\nummodules by#2
  \workdimen=#1\X \advance\workdimen by \bcorr \kern\workdimen
  \workdimen=#2\X \advance\workdimen by-\bcorr \vrule width\workdimen
     \ifdim\barheight=0pt height 69.24242424\X \else height\barheight \fi
     \ifnum\numlines=1    depth5\X\else  % the start mark
     \ifnum\numlines=2    depth5\X\else
     \ifnum\numlines=15   depth5\X\else  % the separator mark
     \ifnum\numlines=16   depth5\X\else
     \ifnum\numlines=29   depth5\X\else  % the end mark
     \ifnum\numlines=30   depth5\X\else depth0pt \fi\fi\fi\fi\fi\fi
  \fi\next}
```

The `\EANfinal` macro checks for the correctness of the scanned `\internalcode`. The number of the digits must be 60 and the sum of digits must be 95 (since 95X modules is the total). If the check fails, the `\internalerr` macro is activated. However this situation should never occur. This error indicates that some internal tables are wrong and/or the consistency of the macro is broken.

The `\vbox` is completed by `\EANfinal`. The natural depth of the internal `\hbox` with the bars is 5X because that is the depth of the mark rules. We overwrite this depth by zero and append the human-readable EAN number using the font `\ocrb`.

If the user writes the ISBN number using the macro `\ISBN` (`\ISBN 0-201-13447-0` for example), these data are appended to the top of the barcode and the height of the bars is reduced.

Finally, lines 81 and 82 define the X module size and the bar correction.

```
\def\EANfinal{\testconsistence
  \kern7\X\egroup
  \hbox{\ocrbsmall \kern10\X \ISBNnum}\kern1\X
  \dp0=0pt \box0 \kern-1\X
  \hbox{\ocrb\kern2\X\firstdigit\kern5\X \frontdigits\kern5\X \enddigits}
  \egroup \global\barheight=0pt \gdef\ISBNnum{}}
\def\testconsistence{\ifnum\numlines=30\else\internalerr\fi
  \ifnum\nummodules=95\else\internalerr\fi}
\def\internalerr{\errmassage{Sorry, my internal tables are wrong, may be.}}
\barheight=0pt
\def\ISBNnum{}
\def\ISBN #1 {\def\ISBNnum{ISBN #1}\barheight=45.151515\X\relax}
\X=.33mm       % Basic size 100%, SC2 code
\bcorr=.020mm  % Bar-correction for offset process
\endinput
```

If the macro was stored in file `ean13.tex` then it can be run with plain TEX as shown below:

```
\input ean13
\nopagenumbers
\ISBN 80-901950-0-8  \EAN 978-80-901950-0-4
\end
```

The output looks like:

X size	standard	scaled	size incl. margins
0.264	SC0	0.800	29.83 × 21.00
0.270	SC0	0.818	30.58 × 21.53
0.281	SC0	0.850	31.70 × 22.32
0.297	SC1	0.900	33.56 × 23.63
0.313	SC1	0.950	35.43 × 24.94
0.330	SC2	1.000	37.29 × 26.26
0.346	SC2	1.050	39.15 × 27.58
0.363	SC3	1.100	41.02 × 28.29
0.379	SC3	1.150	42.88 × 30.20
0.396	SC4	1.200	44.75 × 31.51
0.412	SC4	1.250	46.61 × 32.82
0.429	SC5	1.300	48.48 × 34.14
0.445	SC5	1.350	50.34 × 35.45
0.462	SC5	1.400	52.21 × 36.76
0.478	SC5	1.450	54.07 × 38.08
0.495	SC6	1.500	55.94 × 39.39
0.511	SC6	1.550	57.80 × 40.70
0.528	SC7	1.600	59.66 × 42.01
0.544	SC7	1.650	61.53 × 43.33
0.561	SC7	1.700	63.39 × 44.64
0.577	SC7	1.750	65.26 × 45.96
0.594	SC8	1.800	67.12 × 47.26
0.610	SC8	1.850	68.99 × 48.58
0.627	SC8	1.900	70.85 × 49.90
0.643	SC8	1.950	72.72 × 51.20
0.660	SC9	2.000	74.58 × 52.52
0.700	SC10	2.120	79.05 × 55.67

Table 3: Various sizes of the X module

The macro also works with LaTeX (both LaTeX 2.09 and LaTeX 2ε).

At the end of this article we compare the tolerances described in the standard, the TeX accuracy and the possibilities of some output devices.

The X module size can vary. The macro above makes EAN barcodes for the basic X module size of 0.33 mm. This size is described in the SC2 variant of the standard as the basic 100% size code. However the standard also allows some other sizes of the X module. One can change the parameter `\X` to obtain the other size of EAN code. Of course, then the size of the OCRb font must be changed too.

The allowed sizes of the X module are described in Table 3.

The small sizes of the X module are recommended for high quality output devices while the large sizes of X allow the possibility to make the barcodes even on a low resolution output device.

Depending on the width of the X module, the standard specifies three tolerance parameters. The parameter a specifies the tolerance for the bar width,

X size	$\pm a$	$\pm b$	$\pm c$
0.26	32	38	75
0.28	52	41	81
0.30	72	44	87
0.32	92	47	93
0.33	101	49	96
0.34	105	50	99
0.36	115	53	104
0.38	124	56	110
0.40	134	59	116
0.42	143	62	122
0.44	152	65	128
0.46	162	68	133
0.48	171	71	139
0.50	181	73	145
0.52	190	76	151
0.54	199	79	157
0.56	209	82	162
0.58	218	85	168
0.60	228	88	174
0.62	237	91	180
0.64	246	94	186

Table 4: The tolerances

the parameter b specifies the tolerance for the distance between edges (either left or right ones) of two consecutive bars, and the parameter c specifies the tolerance for the width of the field for one digit (therefore for width 7X). The Table 4 describes these tolerances in micrometers (μm). I don't know why the table column "X size" doesn't match with the column "X size" of the previous table. Sorry, standards are mysterious.

Now we can compare. Consider the basic 100% size (the X module is 0.33 mm). The tolerance for the width of the bar is 101 μm, the TeX (in)accuracy is 0.0054 μm, the pixel size of the phototypesetter at 2400 dpi is approximately 10 μm and the recommended bar correction for the offset process is 20 μm. If we use the phototypesetter at 1200 dpi, the inaccuracy of its output is comparable to the bar correction for the offset process.

Depending on the inner `dvi` driver algorithm the high TeX accuracy may be lost and the tolerance parameters may be overcome. The `dvi` driver algorithms include one of two possible approaches to the "rounding" problem. The first approach is to position and round each rule from `dvi` individually. In the second approach, the `dvi` driver works only with rounded values (one pixel = one unit) *before* making the queue of kern, rule, kern, rule... In this case, the roundoff error can accumulate and the

parameter c can be overcome. But it seems to me that the barcode scanners can read the code better if the metrics of the consecutive bars and spaces are preserved instead of the global width.

As I observed, the `dvi` drivers usually round the rule width *up* to the pixel units and never *down*. The consequence of this feature is that spaces tend to be one pixel smaller than the rules of (presumably) the same width. Therefore I recommend to add one half of the pixel size to the bar correction, namely to the `\bcorr` register.

I have heard that EAN barcodes are successfully read from stickers printed by matrix printers with a very low resolution at module X size of 0.33 mm or comparably small. That would imply that the tolerances of the barcode scanners are usually much higher than those required by the standard.

References

Adriana Benadiková, Štefan Mada and Stanislav Weinlich. *Čárové kódy, automatická identifikace (Barcodes, the Automatic Identification)*. Grada 1994, 272 pp., ISBN 80-85623-66-8.

Donald Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ix+483 pp. Hard cover ISBN 0-201-13447-0.

Petr Olšák. *Typografický systém TEX (Typesetting System TEX)*. CSTUG 1995, 270 pp., ISBN 80-901950-0-8.

◇ Petr Olšák
Department of Mathematics
Czech Technical University in Prague
Czech Republic
olsak@math.feld.cvut.cz

ance parameters may be overcome. The dvi driver algorithms include one of two possible approaches to the "rounding" problem. The first approach is to position and round each rule from dvi individually. In the second approach, the dvi driver works only with rounded values (one pixel = one unit) *before* making the queue of kern, rule, kern, rule... In this case, the roundoff error can accumulate and the parameter c can be overcome. But it seems to me that the barcode scanners can read the code better if the metrics of the consecutive bars and spaces are preserved instead of the global width.

As I observed, the dvi drivers usually round the rule width *up* to the pixel units and never *down*. The consequence of this feature is that spaces tend to be one pixel smaller than the rules of (presumably) the same width. Therefore I recommend to add one half of the pixel size to the bar correction, namely to the \bcorr register.

I have heard that EAN barcodes are successfully read from stickers printed by matrix printers with a very low resolution at module X size of 0.33 mm or comparably small. That would imply that the tolerances of the barcode scanners are usually much higher than those required by the standard.

References

Adriana Benadiková, Štefan Mada and Stanislav Weinlich. *Čárové kódy, automatická identifikace (Barcodes, the Automatic Identification).* Grada 1994, 272 pp., ISBN 80-85623-66-8.

Donald Knuth. *The TEXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, Reading, MA, USA, 1986. ix+483 pp. Hard cover ISBN 0-201-13447-0.

Petr Olšák. *Typografický systém TEX (Typesetting System TEX).* CSTUG 1995, 270 pp., ISBN 80-901950-0-8.

◇ Petr Olšák
Department of Mathematics
Czech Technical University in Prague
Czech Republic
Email: olsak@math.feld.cvut.cz

Letter

An open letter to the TUG Board

It has been suggested by those whose opinion I respect that my somewhat emotive statements concerning the plan to split *TUGboat* from the automatic benefits of TUG membership whilst retaining *TTN* was not calculated to lead to sympathy for my cause. Whilst I accept this, I also know that once an idea has become entrenched it requires more than moderate words to cause a re-consideration of the situation, and I therefore feel that the force with which I put my points was not out of proportion. However, as many board members were not present at the meeting, and as it will no doubt take some time for the minutes to be circulated, I would like to briefly re-iterate my point of view and to explain it.

The proposal, as put at the meeting, is to reduce the membership fee by $20.00, and in addition to make *TUGboat* available as an 'optional extra' for a further $15.00; *TTN* will continue to be sent to all, and its content will be enlarged and improved. The arguments adduced in favour of this are that (1) *TUGboat* is of interest to only a minority of members, whilst (2) *TTN* is of interest to all.

Clearly (2) is at best debatable, and at worst totally flawed: several speakers emphasised that they did *not* want to receive *TTN*. Proposition (1) is interesting, and the obvious question is "on what is this assertion based?" When were the members polled to ask if they valued *TUGboat* or not? How much of this argument is based on hearsay and rumour, and how much on fact?

When I received my first *TUGboat*, it was almost completely incomprehensible to me: I had no idea what the majority of articles meant. But then the same was true for *The TEXbook*, and for the Algol-60 and -68 reports: they were initially complete gibberish, expressed in a language that only the congnoscenti could possibly understand. But I did not give up: I persevered. And each time that I re-read *TUGboat*, or *The TEXbook*, or the Algol-6X reports, I learned a little more. And when the next issue of *TUGboat* arrived, I was able to read a small amount of it without excessive effort, although other parts remained a mystery. But again I did not give up; again I persevered; and with each issue of *TUGboat* my understanding grew.

But as my understanding grew, so did my love of TEX; each article was a further insight into the

brain of its designer and creator, Don Knuth; and the more I learned, the more I wanted to learn.

And finally the day came when I felt confident enough to propose an article of my own; not too long, not too technical, but my first faltering step as an *active* member of the TEX community, no longer completely passive and totally dependent on others for my every TEX need.

And all of this came about through the magic of *TUGboat*, under the inspired editorship of the irreplaceable Barbara.

Then, several years later, I received my first copy of *TTN*. I was horrified: it was almost completely filled with the disgusting details of the infighting which had taken place before, during and after the 'night of the long knives'. It was about *TUG*, not about TEX.

But I did not join TUG to learn about TUG; TUG is simply a vehicle, not an independent entity with a de facto right to existence. It exists for one purpose and one purpose only: to propagate the word about TEX.

And in that last sentence I summarise what I believe is at the heart of this somewhat heated debate: TUG should be about TEX, not about TUG.

Now you can argue that (a) TUG members need to know about TUG activities, and (b) that *TTN* carries TEX-related matter. With (a), I can take no exception: *of course* we need to know about TUG activities. But we do not need a whole magazine/journal/w-h-y devoted to the subject: a few pages in *TUGboat*, ready-prepared by the present/incoming *TTN* team so as to minimise the load on Barbara, is all that is required. But with (b), I take great exception: there is *no need* for a second TUG publication about TEX; it already has a first-class publication in *TUGboat*, and anything which competes with it simply serves to diminish its value. When I have read a useful snippet in *TUGboat*, and need to find it again at some point in the future, I can either consult the on-line indexes, or scan the back covers (modulo the ragged-right, "aren't we clever designers?", Vol. 7, no. 1) to find the article of interest. I do not *want* to have to remember whether it was in *TTN* or *TUGboat* — I want to *know* that it was in *TUGboat*, the definitive TEX journal.

But yes, I agree; more articles for beginners are needed. But their place, too, is in *TUGboat*, at the very beginning; and each article thereafter should be a little more complex, until the final one is of the level demanded by Joachim Schrod and others of his intellect. And by structuring it in this way, the reader will be gently led, just beyond the limits of their own abilility. And with each issue their ability will grow, until they, too, are *contributors* to the wonderful world of TEX, not merely passive users.

And therein, dear Board, surely lies the flaw of your proposal: you seek to divide the TEX community into providers and users, thereby emulating all that is awful about the appalling world of Word Imperfect and Quark SlowBoat. Please ask yourselves: is this really what you want?

Three final points, purely concerned with finance. (1) Do you really believe that there exists an organisation stupid enough to pay $60.00 for *TUGboat* alone, when it can pay $55.00, throw away its copy of *TTN*, and still receive *TUGboat* for $5.00 less? (2) Do you really intend to raise your prices to students, by charging them $5.00 per year *more* than they are presently paying? $20.00, being 50% of the proposed TUG subscription, plus $15.00 for *TUGboat*, is $35.00; (these were the exact figures given at the business meeting in response to Anita's query, although seemingly no-one present noticed this); at the moment a student pays 50% of $60.00 ($30.00) and receives *TUGboat* as an integral part. And (3), what about economies of scale? As the number of copies of *TUGboat* which you produce diminishes, the unit cost increases; is it your covert intention to make *TUGboat* so expensive to produce that you can eventually justify ceasing its publication altogether?

Yours very sincerely,

Philip Taylor,
The Computer Centre
Royal Holloway and Bedford New College
University of London
Egham, Surrey TW20 0EX,
England

Typesetting Commutative Diagrams

Gabriel Valiente Feruglio
University of the Balearic Islands
Mathematics and Computer Science Dept.
E-07071 Palma de Mallorca (Spain)
dmigva0@ps.uib.es

Abstract

There have been several efforts aimed at providing TEX and its derivatives with a suitable mechanism for typesetting commutative diagrams, with the consequent availability of several macro packages of widespread use in the category theory community, and a long debate about the best syntax to adopt for commutative diagrams in LATEX3 has taken place during 1993 in the CATEGORIES discussion list. From the user's point of view, however, there is not much guidance when it comes to choosing a macro package, and even after a decision is made, the conversion of diagrams from the particular conventions of a macro package to another macro package's conventions may prove to be rather hard.

Typesetting commutative diagrams is a surprisingly difficult problem, in comparison with TEX macro packages for other purposes, as judged by the amount of code needed and years of development invested. The existing macro packages for typesetting commutative diagrams are reviewed in this paper and they are compared according to several criteria, among them the capability to produce complex diagrams, quality of the output diagrams, ease of use, quality of documentation, installation procedures, resource requirements, availability, and portability. The compatibility of the different macro packages is also analyzed.

– – ∗ – –

1 Introduction

Commutative diagrams are a kind of graph that is widely used in category theory, not only as a concise and convenient notation but also as a powerful tool for mathematical thought.

A diagram in a certain category is a collection of nodes and directed arcs, consistently labeled with objects and morphisms of the category, where "consistently" means that if an arc in the diagram is labeled with a morphism f and f has domain A and codomain B, then the source and target nodes of this arc must be labeled with A and B respectively.

A diagram in a certain category is said to *commute* if, for every pair of nodes X and Y, all the paths in the diagram from X to Y are equal, in the sense that each path in the diagram determines through composition a morphism and these morphisms are equal in the given category. For instance, saying that the diagram[1]

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle g'} \\ C & \xrightarrow[f']{} & D \end{array}$$

commutes is exactly the same as saying that

$$g' \circ f = f' \circ g.$$

As a notation, the graphic style of presentation inherent to commutative diagrams makes statements and descriptions involving categories more clear and manageable than textual presentations. For instance, consider the definition of an equalizer. A morphism $e : X \to A$ is an *equalizer* of a pair of morphisms $f : A \to B$ and $g : A \to B$ if $f \circ e = g \circ e$ and for every morphism $e' : X' \to A$ satisfying $f \circ e' = g \circ e'$ there exists a unique morphism $k : X' \to X$ such that $e \circ k = e'$.

An equivalent definition is that e is an equalizer if the upper part of the diagram

$$\begin{array}{ccccc} X & \xrightarrow{e} & A & \overset{f}{\underset{g}{\rightrightarrows}} & B \\ {\scriptstyle k}\uparrow & \nearrow_{e'} & & & \\ X' & & & & \end{array}$$

commutes and, whenever the lower part of the diagram also commutes, there is a unique k such that the whole diagram commutes.

As a tool for thought, proofs involving properties that are stated in terms of commutative diagrams can often be given in a "visual" way, in what has been called *diagram chasing*. For instance, the proposition that if both inner squares of the following diagram commute, then also the outer rectangle commutes,

$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{g} & C \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle b} & & \downarrow{\scriptstyle c} \\ A' & \xrightarrow[f']{} & B' & \xrightarrow[g']{} & C' \end{array}$$

[1] All the diagrams in this paper have been typeset using the XY-pic macro package, unless otherwise stated. The reader should not infer any preference by the author for that particular macro package, but should understand that some macro package is needed for the examples in the paper. Sample diagrams typeset with the other macro packages are given in Appendix I.

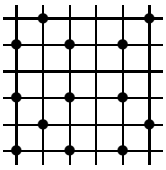
can be proven as follows:

$$\begin{aligned}
(g' \circ f') \circ a &= g' \circ (f' \circ a) && \text{(associativity)}\\
&= g' \circ (b \circ f) && \text{(commutativity of left square)}\\
&= (g' \circ b) \circ f && \text{(associativity)}\\
&= (c \circ g) \circ f && \text{(commutativity of right square)}\\
&= c \circ (g \circ f) && \text{(associativity).}
\end{aligned}$$

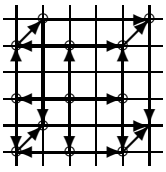
Commutative diagrams[2] range from simple, rectangular matrices of formulae and arrows to complex, non-planar diagrams with curved and diagonal arrows of different shapes.

2 Constructing commutative diagrams

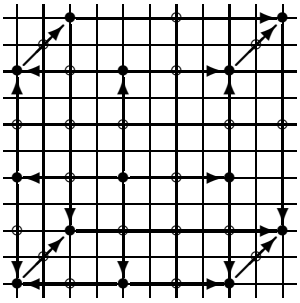
Commutative diagrams are constructed in most cases as rectangular arrays, as Donald Knuth does in Exercise 18.46 of [4]. The objects or vertices are set much like a `\matrix` in TEX or an `array` environment in LATEX,



and the morphisms or arrows are set either right after the vertex where they start,



or in a cell on their own,



depending on the macro package being used, where the grids correspond to the sample diagram presented in Appendix I. (Sketching a commutative diagram on such a grid on paper may prove to be a mandatory step before typing the actual diagram, at least for all but the simplest diagrams.) This gives a first distinction,

- one object and all departing morphisms in each non-empty cell, or
- either one object or one or more morphisms in each non-empty cell.

Whether they belong together with their source object in a cell or they use a cell on their own, morphisms are specified by the address of their target cell. Such addresses can be implicit, absolute or relative to the source cell, and they can be either arbitrary or limited by the available diagonal slopes.

Moreover, some macro packages even support symbolic addresses, by which nodes are tagged with identifier names and arrows are specified by making reference to the names of their source and target nodes. This is a step forward in the sense of the LATEX principle of emphasizing structural descriptions, and in fact it is of great help for designing complex diagrams because it divides the task into two separate subtasks, the one of producing a correct and elegant arrangement of nodes and the other of laying out the correct arrows and positioning their labels.

3 Evaluation guidelines

The following aspects are considered in the next section for each of the macro packages in turn. The spirit of these guidelines is to give the potential user a feeling of what to expect from a macro package for typesetting commutative diagrams, and they are based on the experience of the author during the last few years, as user of some of the macro packages.

3.1 Arrow styles

The arrows used in commutative diagrams often are of different shapes, in order to distinguish different kinds of morphisms such as monomorphisms, epimorphisms, isomorphisms, and inclusions, to name just a few, and sometimes they have a shaft other than a solid line, for instance dashed or dotted, to indicate that it is the existence of the corresponding morphisms which is being characterized.

A collection of built-in arrow shapes and shafts is included in every macro package, and some macro packages even provide facilities for defining new arrow styles, for instance by defining a new control sequence name and choosing a particular combination of tail (the piece that appears at the source end), head (the piece that appears at the target end), and shaft, from a predefined pallette of possible heads, tails, and shafts.

[2] The epithet "commutative" is traditional and originates in the fact that diagrams may be used to display equations such as the commutative and associative laws. Although not all such diagrams which people draw commute in the formal sense given, this paper adheres to tradition and all such diagrams are called commutative diagrams herein.

3.2 Automatic stretching

Most of the macro packages provide for the automatic stretching of arrows to meet their source and target nodes, where meeting a node means to get as close to the (rectangular) box enclosing the node as dictated by some predefined parameter.

While this may be appropriate for most horizontal and vertical arrows, in the case of diagonal arrows it may leave the arrow too far from the node, and extra diagram fine-tuning (see below) is needed in such cases in order to get the arrow closer to the node. The macro package by John Reynolds, however, incorporates basic facilities for associating a hexagon, octagon, or diamond to a node, instead of the usual rectangle, although it does not exploit them in the macros for commutative diagrams.

3.3 Diagram fine-tuning

Given a correct description of the structure, a macro package has the task of choosing the best possible arrows to produce the commutative diagram. Sometimes the best choice may not seem good enough, because only a limited number of slopes may be available for the arrows, because arrows may cross, and because arrow labels may superimpose. Manual fine-tuning belongs therefore to producing complex commutative diagrams.

Arrow stretching can be regarded as automatic fine-tuning. Manual fine-tuning facilities, on the other hand, include moving labels around, moving arrows around, modifying their size, changing the distance from the source node to the beginning of the arrow, as well as from the end of the arrow to the target node, and setting spacing parameters such as the gap between columns and between rows. Some macro packages provide the facility to adjust these gaps to different values between specific rows or columns, which is essential in order to get the proper perspective of a three-dimensional diagram. Otherwise, empty rows and columns have to be added to the diagram to get the desired perspective. Appendix III shows the degree of automatic stretching provided by each of the macro packages.

3.4 Installation

None of the macro packages requires a complex installation procedure, and in most cases the only requirement in order to get the package running is to drop a single macro or style file somewhere in the TEX search path. Some macro packages, however, have accompanying special fonts to get better diagonal lines and arrows, that is, they provide more diagonal slopes and a wider variety of arrow heads and tails to choose from.

In such a case, installation can get more complicated. METAFONT is not as easy to drive or as familiar to the user as TEX or LATEX; many implementations do not make it available, and on others only the system administrator is able to install fonts. A ready-to-use collection of the additional fonts at standard magnifications is distributed, however, with some macro packages.

3.5 Documentation

This ranges from small text files to comprehensive user guides, and even to book chapters.

3.6 User support

The authors of the different macro packages have been receptive to comments and willing to provide user support. Almost all of the macro packages remain under development and are open to suggestions from users. Moreover, further development of the XY-pic macro package by Kristoffer Rose and Ross Moore is being funded by three different sources.

3.7 Ease of use

The relative ease of use of a macro package is a subjective matter, depending to a large extent on previous experiences in using similar macro packages. Nevertheless, there are at least two characteristics of a macro package for typesetting commutative diagrams that are worth mentioning.

The way in which the array of cells underlying a commutative diagram has to be conceived is of most importance. The requirement, found in some macro packages, of extra cells for morphisms makes the macro package much more difficult to use, because the user has to add many spurious rows and columns only to hold these morphisms and to get proper spacing, and the code for the diagrams gets bigger and more obscure (compare the last two grids in the previous section).

Orthogonal to the conception of the array of cells is the way in which coordinates for the source and target nodes of the arrows have to be specified. While such addresses are implicit in the name of the arrow in some macro packages, they are absolute coordinates, coordinates relative to the cell where they are declared, or even symbolic coordinates in other macro packages.

The other aspect is the degree of manual fine-tuning needed to achieve a readable commutative diagram. Even when the macro package provides enough facilities, fine-tuning a complex commutative diagram may take more time and effort than

conceiving, designing, and coding the whole diagram. Some of the macro packages require visual or measured adjustment by the user of the size and position of every node, arrow, and label, whereas for others most diagrams may be input as easily as any other mathematical formula in TEX and they are typeset nicely without any manual adjustment at all.

3.8 Resource requirements

It is well known that TEX has been designed to support high-quality typesetting of mathematical text, and that it does not offer much built-in support when it comes to drawing and performing arbitrary computations. Because most of the macro packages are built on top of TEX, they are forced to resort to indirect ways of performing computations and to produce large diagrams by juxtaposition of small line and arrow segments. Therefore, a complex diagram may take up lots of computations, line segments, words of TEX memory, and time to typeset. Appendix IV compares resource requirements for the different macro packages, showing the main file size together with statistics of both total time and marginal time. The statistics are based on sample runs to typeset the sample diagrams presented in Appendix I with the different macro packages.

3.9 Availability

All the macro packages reviewed in this paper can be found in the CTAN archives, and either are in the public domain or are free software, subject to the terms of the GNU General Public Licence as published by the Free Software Foundation. They are listed in Appendix V.

3.10 Compatibility

Converting a commutative diagram among different macro packages is no straightforward task, not only because of the different approaches to constructing a diagram mentioned in the previous section, but also because of differences in naming conventions and in the available arrow styles and slopes. Converting the sample diagram in Appendix I has taken the author many hours of careful work, and in some cases building the diagram again from scratch for another macro package has proven to be the most efficient solution.

The macro packages are therefore highly incompatible. Nevertheless, the macro package by Paul Taylor provides some initial facilities for emulating other macro packages. Maybe a common, agreed-upon syntax for commutative diagrams (see the last section below) would provide a suitable framework for solving these incompatibilities. Moreover, although it may seem rather natural that the macro packages are not compatible with each other, because the idioms are under development and none of the authors is, in principle, under any obligation to the users of the other macro packages, the adoption of a common standard would have the advantage to the whole user community that the diagrams which have already been drawn with one macro package could be pasted into a document using another macro package.

3.11 TEX format requirements

While it would be desirable to be able to typeset a commutative diagram under any derivative of TEX, some macro packages can only run on LATEX because they borrow the `picture` environment and one or more of the special fonts `line10`, `linew10`, `circle10`, and `circlew10`. Other macro packages require $\mathcal{AMS}$-TEX or the `amsmath` package in LATEX. The other way round, some macro packages run on TEX but do not run when used in a LATEX document.

3.12 Output quality

This is perhaps the most subjective aspect in these guidelines, and therefore it is left for the reader to evaluate. See the sample diagrams in Appendix I, and make a guess at which of the macro packages has been used in Valiente (1994).

4 Macro packages

The different macro packages are listed in turn in the following, under the name of the respective author, and they are analyzed according to the evaluation guidelines presented in the previous section. No attempt has been made to put them in chronological order of development, and the list is sorted by author name.

4.1 American Mathematical Society

$\mathcal{AMS}$-TEX includes some commands for typesetting commutative diagrams, which are also available in $\mathcal{AMS}$-LATEX as a separate option. Only horizontal and vertical arrows are supported, and therefore $\mathcal{AMS}$-TEX can only handle "rectangular" commutative diagrams. Moreover, only "plain" arrows can be used within commutative diagrams, although $\mathcal{AMS}$-TEX provides about 30 different arrow shapes, and arrows do not automatically stretch to their source and target vertices. Commutative diagrams are specified as an array of cells, with either one object or one or more morphisms in each non-empty cell, although unlike matrices, no column separator is needed (a special delimiter has to be used,

however, in place of missing arrows). Arrow coordinates are implicit in the name of the arrow and only the four basic directions are available, where arrows can only extend to the adjacent row and/or column in the array. The only fine-tuning facilities provided are a stretching command to force arrows in the same column to be set to the same length (actually, to the width of the longest label in that column), which does not suffice in order to achieve appropriate arrow stretch when the vertices have different width (this manual stretching facility requires the whole `amsmath` package to be loaded in $\mathcal{AMS}$-LaTeX), and a command to change the minimum arrow width in a diagram, for instance to get it to fit on a page. Documentation is as scarce as the facilities the package provides, only four pages in [11] and one page in [9].

4.2 Barr

Instead of using a matrix notation, commutative diagrams are specified in the macro package developed by Michael Barr by composing more elementary diagrams, using primitive shapes such as squares and triangles. Arrow coordinates are implicit within these shape macros. Additional arrows can be specified by giving the absolute address, within an implicit `picture` environment, of their source node, together with the relative address of their target node as a slope and a length, but stretching is not automatic in these cases. It supports diagonal arrows only in the usual LaTeX slopes, and only a few different arrow shapes are available. There are no facilities for diagram fine-tuning. It only runs on LaTeX. Documentation consists of a 10-page document [1] which explains the principles and gives detailed examples.

4.3 Borceux

In the macro package developed by Francis Borceux, commutative diagrams are specified as an array of cells, with one object and all departing morphisms in each non-empty cell. There are facilities for introducing one object and one morphism, or two *crossing* morphisms, in each non-empty cell, but at most two items may belong to the same cell. The delimiter for columns is, unlike the `&` character used in *all* the other macro packages, the special character § that is not even available in many keyboard layouts. It supports diagonal arrows of different shapes and in many different, although not arbitrary, slopes, and it also supports parallel and adjoint (counter-parallel) arrows, some curved arrows, and automatic stretching. Arrow coordinates are implicit in the name of the arrow for the 32 principal directions. Different facilities for diagram fine-tuning are provided. It only runs on LaTeX. Documentation consists of a detailed 12-page document [2]. Two restricted macro files are distributed for small TeX implementations, one that only allows for plain arrows and another one that also provides parallel and adjoint (counter-parallel) plain arrows. A further macro file is distributed with the package that provides additional triple, quadruple, and quintuple arrows, parallel and disjoint.

4.4 Gurari

Unlike the case of most of the other macro packages, Eitan Gurari has developed a general drawing package on top of TeX. It supports diagonal arrows of different shapes and arbritrary slopes, curved arrows and loops, automatic stretching, and symbolic addressing. Arrow coordinates can be symbolic, because of the possibility of naming any location within a drawing, but they are relative in the sample diagrams presented in the appendices because the macros used are the ones given in page 160 of [3]. It runs on both TeX and LaTeX. The macros are well documented in the book, with several basic chapters and one chapter devoted to general grid diagrams, but there is only one page describing commutative diagrams and there is only one sample diagram in the whole book.

4.5 Reynolds

John Reynolds has developed a macro package consisting of a collection of general macros for producing a wide variety of diagrams and another collection of macros, which depend on the general macros, for producing commutative diagrams. It supports diagonal arrows only in the usual LaTeX shapes and slopes, because the macros depend on the LaTeX picture facilities to draw lines, arrows, and circles, although it also supports parallel and adjoint (counter-parallel) arrows, loops, and it provides automatic stretching. Commutative diagrams as specified by giving the absolute coordinates for each node and for the source and target node of each arrow, an approach close to symbolic addressing. Excellent facilities for diagram fine-tuning are provided. It only runs on LaTeX. Documentation consists of a rather cryptic 12-page ASCII file [5] describing the macro package, together with a LaTeX input file that produces a 7-page document of sample diagrams.

4.6 Rose

A macro package has been developed by Kristoffer Rose on top of a more general drawing language, called the *XY-pic kernel*. It supports diagonal arrows of different shapes and in many different, although

not arbitrary, slopes, and it also supports parallel and adjoint (counter-parallel) arrows, curved arrows, and loops. Arrows stretch automatically, and there are ample facilities for defining additional arrow styles. Commutative diagrams are specified as an array of cells, with one object and all departing morphisms in each non-empty cell. Arrow coordinates for the target node are implicit in the name of the arrow for the 16 principal directions, and they can be absolute or relative for all other directions. Different facilities for diagram fine-tuning are provided. It runs on both TEX and LATEX. Documentation is excellent, both a comprehensive guide [6] and a more technical document [7] are provided with the package. The latter also describes the XY-pic kernel.

4.7 Smith

The *Expanded Plain TEX* macro package includes macros for typesetting commutative diagrams, written by Steven Smith, in a file named `arrow.tex`. It supports diagonal arrows only in the usual LATEX slopes, because the macros depend on the LATEX font `line10`, and only a "plain" arrow shape is available, besides pairs of parallel and adjoint (counter-parallel) arrows. Commutative diagrams are specified as an array of cells, with either one object or one or more morphisms in each non-empty cell. There is not any automatic stretching of arrows. Arrow coordinates are implicit in the name of the arrow for the four basic directions, and they are relative addresses for all other directions. Designing a complex diagram using this macro package is as difficult as fine-tuning a simple diagram, even requiring manual computations of horizontal and vertical dimensions to get a desired arrow size and slope. It runs on both TEX and LATEX. Documentation is enough to cover the facilities provided by the macros, seven pages in [8] and a two-page source document named `commdiags.tex`, reproducing eleven textbook commutative diagrams.

4.8 Spivak

LAMS-TEX includes an environment for producing commutative diagrams that supports diagonal arrows of different shapes and in many different, although not arbitrary, slopes. Arrows stretch automatically, and there are ample facilities for defining additional arrow styles. Commutative diagrams are specified as an array of cells, with one object and all departing morphisms in each non-empty cell. Arrow coordinates are relative addresses, and mnemonics can be easily defined for the most common arrow coordinates. Superb facilities for diagram fine-tuning are provided. It only runs on TEX. Documentation is excellent, two chapters in [10] describing every detail from diagram design to coding and fine-tuning.

4.9 Svensson

The most recent addition to the commutative diagrams family is the macro package `kuvio.tex`, developed by Anders Svensson. It supports diagonal arrows of different shapes and in many different, although not arbitrary, slopes (implemented by rotating horizontal arrows through PostScript `\special` commands). Arrows stretch automatically, and there are ample facilities for defining additional arrow styles. Commutative diagrams are specified as an array of cells, with either one object or one or more morphisms in each non-empty cell. Arrow coordinates are implicit in the name of the arrow, and they are complemented with explicit slope and length parameters. Different facilities for diagram fine-tuning are provided. It runs on both TEX and LATEX. The macros are well documented in a 54-page guide and reference manual [12].

4.10 Taylor

A macro package developed by Paul Taylor supports diagonal arrows of different shapes and slopes, and even at arbitrary slopes (implemented by rotating horizontal arrows through PostScript `\special` commands). Arrows stretch automatically, and there are ample facilities for defining additional arrow styles. Commutative diagrams are specified as an array of cells, with either one object or one or more morphisms in each non-empty cell. Arrow coordinates are implicit in the name of the arrow, and they are complemented with explicit slope and length parameters. There are plenty of options for diagram fine-tuning, either global to the whole document or local to a single diagram. It runs on both TEX and LATEX. Documentation is excellent, a quite comprehensive document [13] that is even provided typeset in booklet format.

4.11 Van Zandt

As in the case of the macro packages by Eitan Gurari, PSTricks is a general drawing package built on top of TEX. Instead of extending TEX by defining graphics primitives, however, it is a collection of PostScript-based TEX macros, and it can be seen in fact as a high-level TEX-like interface to the PostScript language. It supports diagonal arrows of different shapes and arbitrary slopes, curved arrows and loops, automatic stretching, and symbolic addressing for both node and arrow coordinates. It runs on both TEX and LATEX. The macros are well documented in [15], although there are only two

pages describing commutative diagrams and only two sample diagrams in the whole document.

5 Discussion

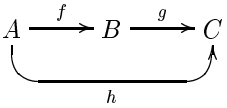
5.1 Syntactic issues

Syntactic issues are so fundamental to user acceptance of a macro package for typesetting commutative diagrams, that a volunteer task within the LaTeX3 project was founded in October 1992 under the name *Research on Syntax for Commutative Diagrams*, with Paul Taylor as co-ordinator and Michael Barr and Kristoffer Rose as members.

After an initiative by Michael Barr, who started a discussion within the categorical community about the best syntax to adopt for commutative diagrams in LaTeX3, a rather heated debate has taken place in the `CATEGORIES` discussion list. There were many contributions between June and August 1993, although the discussion list has been silent in these matters ever since.

5.2 Curved arrows

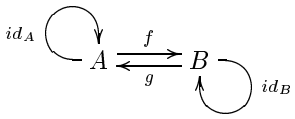
The need for curved arrows arises when "parallel" morphisms have to be distinguished from each other, for instance when it is not known if the morphism $h : A \to C$ is equal to the composition of the morphisms $g : B \to C$ and $f : A \to B$,



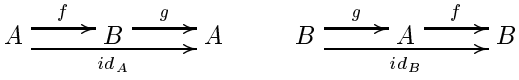
because otherwise the composite morphism would not need to be made explicit.

The need for curved arrows also arises when there are loops in a diagram. For instance, consider the definition of an isomorphism.

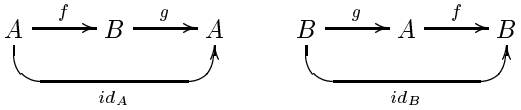
A morphism $f : A \to B$ in a given category is an *isomorphism* if there exists a morphism $g : B \to A$ in that category such that $g \circ f = id_A$ and $f \circ g = id_B$. That is, if the diagram



commutes. One possible trick to eliminate the need for such curved arrows is to "straighten up" the diagram by appropriately duplicating some nodes. For the previous example, a morphism $f : A \to B$ is an isomorphism if the following two diagrams commute:



These diagrams, however, look much better with a curved arrow,



and therefore the need for curved arrows cannot always be eliminated without sacrificing diagram clarity and, perhaps arguably, esthetics. While some authors of category theory textbooks seem to prefer to duplicate nodes, others make a thorough use of curved arrows.

5.3 Design issues

Diagrams are essentially a communication medium, and therefore good design means a design for readability. Although readability issues can be as subjective as esthetic issues, however, some basic principles may help in the design of readable diagrams. The first principle is to follow the natural order of writing, which at least within occidental writing conventions means left to right, top to bottom, and foreground to background. A second principle is to appropriately give depth to three-dimensional diagrams, in such a way that the foreground lies a little below the background. This principle finds no easy justification, because it may seem to contradict the top-to-bottom order of writing by imposing a bottom-to-top order from foreground to background, but it is true of all kinds of pictorial representations.

5.4 User interface

Most of the macro packages provide a simple user interface, consisting of a certain matrix notation. While it adheres to the LaTeX principle of emphasizing structural descriptions, such a specification may become much too obscure for a complex diagram. Some authors have argued against the use of alternative technologies (if you want WYSIWYG, use a pen and paper) but maybe the time has arrived to have a state-of-the-art drawing program with specific facilities for designing commutative diagrams. One possible scenario would be to sketch the arrangement of nodes and arcs on the computer screen using a mouse, and to let the drawing program translate the design into the language of (any of) the macro packages, taking care of all the time-consuming details of computing coordinates, choosing appropriate slopes for the arrows, placing arrow labels, fine-tuning, etc. Further facilities could include, for instance, trying different layouts based both on the structural description of the diagram as a graph and on knowledge of the kind of graphs that commutative diagrams are, and performing specific

operations on descriptions such as, for instance, obtaining the *dual* of a commutative diagram.

5.5 Open issues

Although the conceptual framework used for evaluating the different macro packages resulted from the experience of the author using them and converting diagrams between them, it is precisely because of the evaluation having been carried out by only one person that the resulting data may be somewhat biased. A more general investigation would involve mathematicians and computer scientists writing their own diagrams, as well as (LA)TEX-competent secretaries typing their work, and would produce quantitative measures of learning times for the different macro packages and, once they are fluent in each macro package, measures of the time it takes them to transcribe a diagram drawn on paper.

Further additional investigations include evaluating the degree of help given by each macro package towards improving the quality of the output diagrams, for instance by means of informative messages; quantifying the degree of fine-tuning needed with each macro package in order to produce a complex diagram; evaluating the robustness of the different macro packages when the user makes common errors, such as omitting brackets or mistyping command names; and, last but not least, designing a standard library of common diagrams against which the different macro packages could be evaluated and compared.

6 Acknowledgement

In order to avoid name clashes among the control sequences defined in the different macro packages, all the diagrams have been typeset separately and included in the final document as encapsulated PostScript files. Thanks to Michel Goossens and Sebastian Rahtz for their advice. Ricardo Alberich Martí provided guidance during the design of the experiment to obtain time statistics.

References

[1] Michael Barr. The diagram macros. Electronic document distributed with the package.

[2] Francis Borceux. User's guide for diagram 3. Electronic document distributed with the package.

[3] Eitan M. Gurari. *TEX & LATEX—Drawing and Literate Programming*. McGraw-Hill, New York, 1994.

[4] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 15th printing, 1989.

[5] John Reynolds. User's manual for diagram macros. Electronic document distributed with the package, December 1987.

[6] Kristoffer H. Rose. XY-pic user's guide. Electronic document distributed with the package, October 1994.

[7] Kristoffer H. Rose and Ross Moore. XY-pic reference manual. Electronic document distributed with the package, October 1994.

[8] Steven Smith. Arrow-theoretic diagrams. Electronic document distributed with the package, May 1994. Chapter 5 in Karl Berry and Steven Smith, *Expanded Plain TEX*.

[9] American Mathematical Society. $\mathcal{AMS}$-LATEX version 1.2 user's guide. Electronic document distributed with the package, January 1995.

[10] Michael D. Spivak. *LAMS-TEX—The Synthesis*. The TEXplorators Corporation, Houston, Texas, 1989.

[11] Michael D. Spivak. *The Joy of TEX—A Gourmet Guide to Typesetting with the $\mathcal{AMS}$-TEX Macro Package*. American Mathematical Society, 2nd edition, 1990.

[12] Anders G. S. Svensson. Typesetting diagrams with `kuvio.tex`. Electronic document distributed with the package, January 1995.

[13] Paul Taylor. Commutative diagrams in TEX (version 4). Electronic document distributed with the package, July 1994.

[14] Gabriel Valiente Feruglio. *Knowledge Base Verification using Algebraic Graph Transformations*. PhD thesis, University of the Balearic Islands, December 1994.

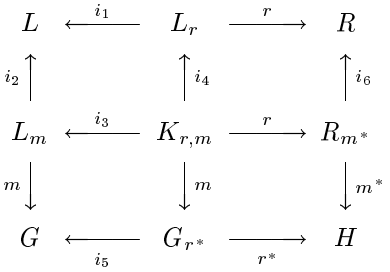
[15] Timothy Van Zandt. PSTricks user's guide. Electronic document distributed with the package, March 1993.

◇ Gabriel Valiente Feruglio
University of the Balearic Islands
Mathematics and Computer Science Dept.
E-07071 Palma de Mallorca (Spain)
`dmigva0@ps.uib.es`

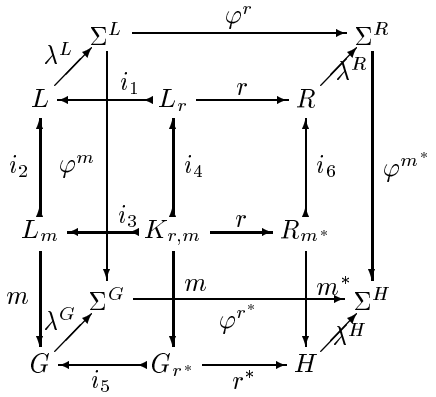
7 Appendix I: Sample diagrams

The following diagrams reproduce a fairly complex commutative diagram, taken from [14], using all the macro packages reviewed in this paper. The diagram consists of a pushout construction of partial closed morphisms of total unary algebras in the foreground, together with a corresponding pushout construction of total morphisms of total signature algebras in the background.

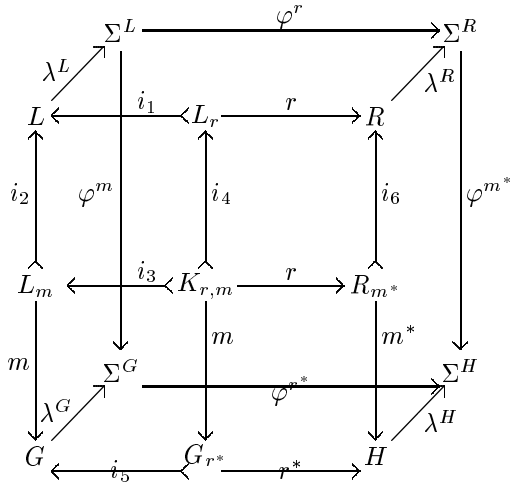
7.1 American Mathematical Society



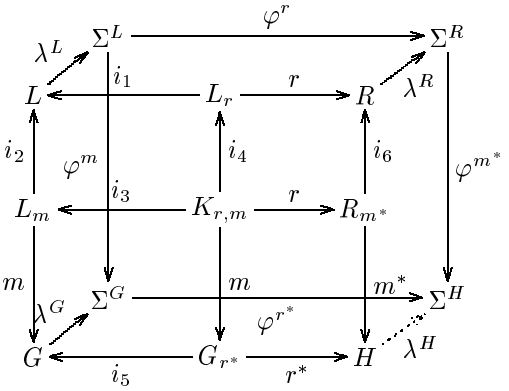
7.2 Michael Barr



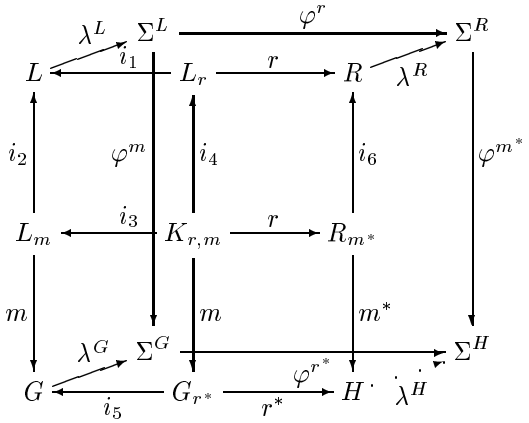
7.3 Francis Borceux



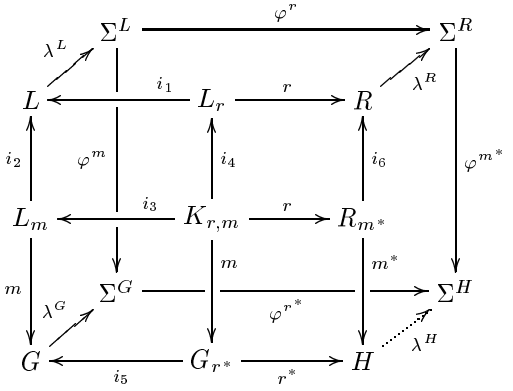
7.4 Eitan Gurari



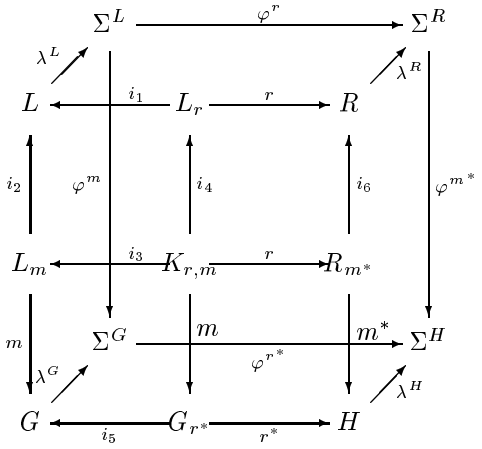
7.5 John Reynolds



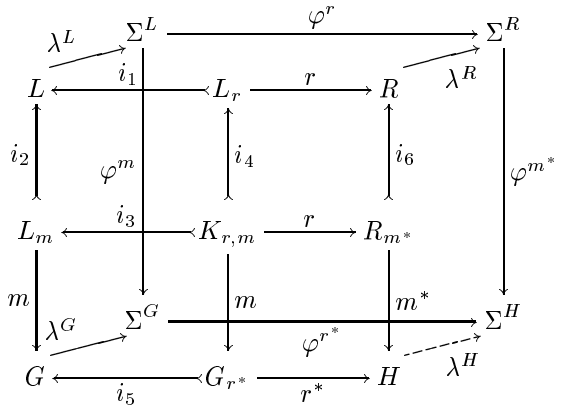
7.6 Kristoffer Rose



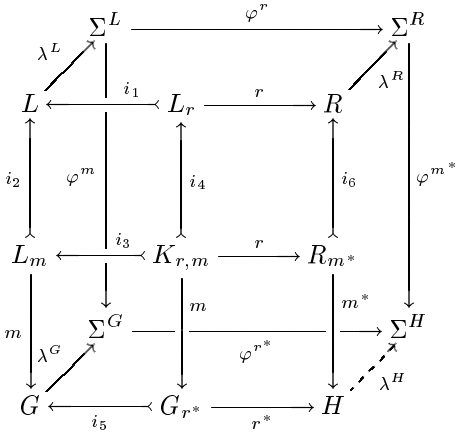
7.7 Steven Smith



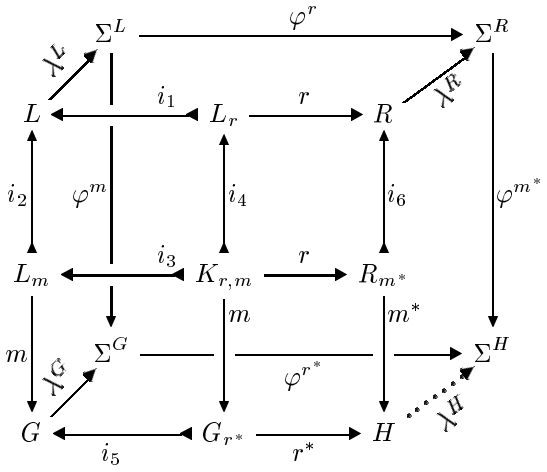
7.8 Michael Spivak



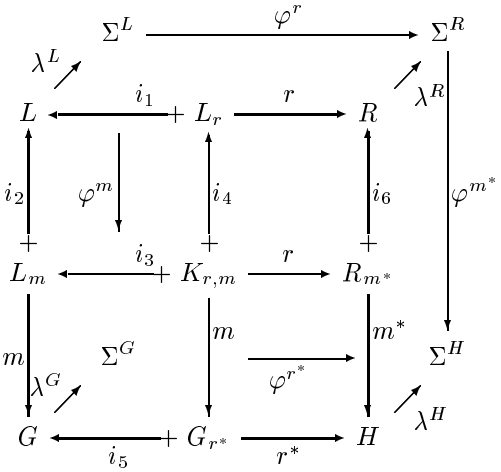
7.9 Anders Svensson



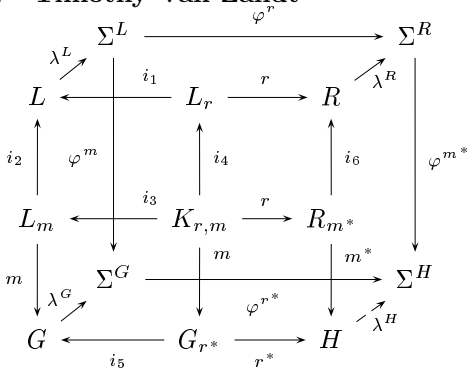
7.10 Paul Taylor



7.11 Paul Taylor emulating F. Borceux



7.12 Timothy Van Zandt



8 Appendix II: Source code for the sample diagrams

```
\newcommand{\up}[1]{\raisebox{1em}{$#1$}}
\newcommand{\down}[1]{\raisebox{-1em}{$#1$}}
\newcommand{\Left}[1]{\makebox[5pt][r]{$#1$}}
\newcommand{\Right}[1]{\makebox[5pt][l]{$#1$}}
```

8.1 American Mathematical Society

```
$$\begin{CD}
L       @<i_1<< L_r     @>r>>   R       \\
@Ai_2AA         @AAi_4A         @AAi_6A \\
L_m     @<i_3<< K_{r,m} @>r>>   R_{m^*} \\
@VmVV           @VVmV           @VVm^*V \\
G       @<<i_5< G_{r^*} @>>r^*> H
\end{CD}$$
```

8.2 Michael Barr

```
$$\bfig
\putsquare<-2‘-2‘-2;500‘500>(0,500)[L‘L_r‘L_m‘K_{r,m};\qquad i_1‘i_2‘i_4‘]
\putsquare<1‘0‘-2‘1;500‘500>(500,500)[\phantom{L_r}‘R‘\phantom{K_{r,m}}‘R_{m^*};r‘‘i_6‘]
\putsquare<0‘1‘1‘-2;500‘500>(0,0)[\phantom{L_m}‘\phantom{K_{r,m}}‘G‘G_{r^*};\qquad i_3‘m‘\up m‘i_5]
\putsquare<0‘0‘1‘1;500‘500>(500,0)%
  [\phantom{K_{r,m}}‘\phantom{R_{m^*}}‘\phantom{G_{r^*}}‘H;r‘‘\up{m^*}‘r^*]
\putsquare<1‘1‘1‘1;1000‘1000>(250,250)%
  [\Sigma^L‘\Sigma^R‘\Sigma^G‘\Sigma^H;\varphi^r‘\varphi^m‘\varphi^{m^*}‘\varphi^{r^*}]
\putmorphism(125,1125)(1,1)%
  [\phantom L‘\phantom{\Sigma^L}‘{\up{\Right{\lambda^L}}}]{0}{1}{l}
\putmorphism(1125,1125)(1,1)%
  [\phantom R‘\phantom{\Sigma^R}‘{\down{\Left{\lambda^R}}}]{0}{1}{r}
\putmorphism(125,125)(1,1)
  [\phantom G‘\phantom{\Sigma^G}‘{\up{\Right{\lambda^G}}}]{0}{1}{l}
\putmorphism(1125,125)(1,1)%
  [\phantom H‘\phantom{\Sigma^H}‘{\down{\Left{\lambda^H}}}]{0}{1}{r}
\efig$$
```

8.3 Francis Borceux

```
\setdefaultscale{40}
\begin{diagram}
? ? \Sigma^L ? ? ? ? \Ear[280] {\varphi^r} ? ? ? ? \Sigma^R ??
? \Near[50] {\lambda^L} ? ? ? ? ? ? ? ? \neaR[50] {\lambda^R} ??
L ? ? \Wmono[130] {\qquad i_1} ? ? L_r ? ? \Ear[130] r ? ? R ?? ??
\Nmono[130] {i_2} ? ? \Sar[280] {\varphi^m} ? ? \nmonO[130] {i_4} ? ? ? ? \nmonO[130] {i_6} ? ?
  \saR[280] {\varphi^{m^*}} ?? ??
L_m ? ? \Wmono[100] {\qquad i_3} ? ? K_{r,m} ? ? \Ear[100] r ? ? R_{m^*} ?? ??
\Sar[130] m ? ? \Sigma^G ? ? \saR[130] {\up{m}} ? ? \eaR[280] {\varphi^{r^*}} ? ?
  \saR[130] {\up{m^*}} ? ? \Sigma^H ??
? \Near[50] {\lambda^G} ? ? ? ? ? ? ? \neaR[50] {\lambda^H} ??
G ? ? \wmonO[130] {i_5} ? ? G_{r^*} ? ? \eaR[130] {r^*} ? ? H ??
\end{diagram}
```

8.4 Eitan Gurari

```
\Draw
\PenSize(0.25pt)
\ArrowSpec(V,5,3,2)
\ArrowHeads(1)
\GridSpace(10,10)
\GridDiagramSpec()(\MyEdge)
\Define\L(4){,+#1..+#2\,L\,#3\,#4}
```

```
\Define\D(4){,+#1..+#2\,D\,#3\,#4}
\Define\MyEdge(5){
   \IF \EqText(#3,D) \THEN
      \EdgeSpec(D)
   \ELSE
      \EdgeSpec(L)
   \FI
   \IF \EqText(#1,#2) \THEN
      \RotateTo(#4)
      \CycleEdge(#1)
      \EdgeLabel(--$#5$--)
   \ELSE
      \Edge(#1,#2)
      \IF \EqText(,#4) \THEN
         \EdgeLabel(--$#5$--)
      \ELSE
         \EdgeLabel[#4](--$#5$--)
      \FI
   \FI}
\GridDiagram(8,8)()()({
& $\Sigma^L$ \L(6,0,+,\mbox{$\varphi^m$}) \L(0,6,,\mbox{$\varphi^r$}) & & & & & & $\Sigma^R$
  \L(6,0,,\mbox{$\varphi^{m^*}$}) //
$L$ \L(-1,1,,\mbox{$\lambda^L$}) & & & $L_r$ \L(0,-3,+,\mbox{$i_1$}) \L(0,3,,\mbox{$r$}) & & &
  $R$ \L(-1,1,+,\mbox{$\lambda^R$}) & //
& & & & & & & //
& & & & & & & //
$L_m$ \L(3,0,+,\mbox{$m$}) \L(-3,0,,\mbox{$i_2$}) & & & $K_{r,m}$ \L(-3,0,+,\mbox{$i_4$})
  \L(3,0,,\mbox{$m$}) \L(0,-3,+,\mbox{$i_3$}) \L(0,3,,\mbox{$r$}) & & & $R_{m^*}$
  \L(3,0,,\mbox{$m^*$}) \L(-3,0,+,\mbox{$i_6$}) & //
& & & & & & & //
& $\Sigma^G$ \L(0,6,+,\mbox{$\varphi^{r^*}$}) & & & & & & $\Sigma^H$ //
$G$ \L(-1,1,,\mbox{$\lambda^G$}) & & & $G_{r^*}$ \L(0,-3,,\mbox{$i_5$}) \L(0,3,+,\mbox{$r^*$})
  & & & $H$ \D(-1,1,+,\mbox{$\lambda^H$}) & //})
\EndDraw
```

8.5 John Reynolds

```
\def\diagramunit{0.6pt}
$$\ctdiagram{
\ctv 0,0:{G}
\ctv 100,0:{G_{r^*}}
\ctv 200,0:{H}
\ctv 0,100:{L_m}
\ctv 100,100:{K_{r,m}}
\ctv 200,100:{R_{m^*}}
\ctv 0,200:{L}
\ctv 100,200:{L_r}
\ctv 200,200:{R}
\ctel 0,100,0,200:{i_2}
\cter 100,100,100,200:{i_4}
\cter 200,100,200,200:{i_6}
\ctel 0,100,0,0:{m}
\cter 100,100,100,0:{m}
\cter 200,100,200,0:{m^*}
\ctetg 100,200,0,200;60:{i_1}
\ctetg 100,100,0,100;60:{i_3}
\cteb 100,0,0,0:{i_5}
\ctet 100,200,200,200:{r}
\ctet 100,100,200,100:{r}
```

```
\cteb 100,0,200,0:{r^*}
\ctv 75,25:{\Sigma^G}
\ctv 275,25:{\Sigma^H}
\ctv 75,225:{\Sigma^L}
\ctv 275,225:{\Sigma^R}
\ctet 0,0,75,25:{\lambda^G}
\ctdot
\cteb 200,0,275,25:{\lambda^H}
\ctsolid
\ctet 0,200,75,225:{\lambda^L}
\cteb 200,200,275,225:{\lambda^R}
\ctelg 75,225,75,25;150:{\varphi^m}
\cterg 275,225,275,25;150:{\varphi^{m^*}}
\ctet 75,225,275,225:{\varphi^r}
\cteb 75,25,275,25:{\varphi^{r^*}}
}$$
```

8.6 Kristoffer Rose

```
\definemorphism{unique}\dotted\tip\notip
\spreaddiagramrows{-1pc}
\spreaddiagramcolumns{-1pc}
\diagram
& \Sigma^L \xto'[1,0]'[3,0]_{\varphi^m}[4,0] \xto[rrrr]^{\varphi^r}
  & & & & \Sigma^R \xto[dddd]^{\varphi^{m^*}} \\
L \urto^{\lambda^L} & & \llto_<<<<{i_1} L_r \rrto^r & & R \urto_{\lambda^R} \\ \\
L_m \uuto^{i_2} \ddto_m & & \llto_<<<<{i_3} \uuto_{i_4} K_{r,m} \ddto^<<<<m \rrto^r
  & & \uuto_{i_6} R_{m^*} \ddto^<<<<{m^*} \\
& \Sigma^G \xto'[0,1]'[0,3]_{\varphi^{r^*}}[0,4] & & & & \Sigma^H \\
G \urto^{\lambda^G} & & \llto^{i_5} G_{r^*} \rrto_{r^*} & & H \urunique_{\lambda^H}
\enddiagram
```

8.7 Steven Smith

```
\harrowlength=45pt
\sarrowlength=.30\harrowlength
$$\gridcommdiag{
& & \Sigma^L & & & & {\harrowlength=100pt\mapright^{\varphi^r}}
  & & & & \Sigma^R \cr
& \arrow(1,1)\lft{\lambda^L} & & & & & & & \arrow(1,1)\rt{\lambda^R} \cr
L & & \mapleft^{\qquad i_1} & & L_r & & \mapright^r & & R \cr \cr
\mapup^{i_2} & & {\varrowlength=100pt\mapdown^{\varphi^m}}
  & & \mapup_{i_4} & & & & \mapup_{i_6} & &
  {\varrowlength=100pt\mapdown_{\varphi^{m^*}}}
  \cr \cr
L_m & & \mapleft^{\qquad i_3} & & K_{r,m} & & \mapright^r & & R_{m^*} \cr \cr
\mapdown^m & & \Sigma^G & & \mapdown_{\up{m}}
  & & {\harrowlength=100pt\mapright_{\varphi^{r^*}}}
  & & \mapdown_{\up{m^*}} & & \Sigma^H \cr
& \arrow(1,1)\lft{\lambda^G} & & & & & & & \arrow(1,1)\rt{\lambda^H} \cr
G & & \mapleft_{i_5} & & G_{r^*} & & \mapright_{r^*} & & H
}$$
```

8.8 Michael Spivak

```
$$\Cgaps{0.5}
\Rgaps{0.5}
\cgaps{1.3;0.7;1;1;1.3}
\rgaps{0.7;1;1;1.3;0.7}
\CD
  & \Sigma^L @() \L{\varphi^m} @(0,-4) @() \L{\varphi^r} @(4,0)
```

```
  & & &
  & \Sigma^R @() \l{\varphi^{m^*}} @(0,-4)
\\
L @() \L{\lambda^L} @(1,1)
  & &
  L_r @() \L{i_1} \Ot @(-2,0) @() \L{r} @(2,0)
  & & R @() \l{\lambda^R} @(1,1)
  &
\\
\\
L_m @() \L{i_2} \Ot @(0,2) @() \L{m} @(0,-2)
  & & K_{r,m} @() \L{i_3} \Ot @(-2,0) @() \L{r} @(2,0) @() \l{i_4} \Ot @(0,2) @() \l{m} @(0,-2)
  & & R_{m^*} @() \l{i_6} \Ot @(0,2) @() \l{m^*} @(0,-2)
  &
\\
  & \Sigma^G @() \l{\varphi^{r^*}} @(4,0)
  & & & & \Sigma^H
\\
  G @() \L{\lambda^G} @(1,1)
  & & G_{r^*} @() \l{i_5} \Ot @(-2,0) @() \l{r^*} @(2,0)
  & & H @() \l{\lambda^H} \a- @(1,1)
& \\
\endCD$$
```

8.9 Anders Svensson

```
\scale=.5
\Diagram
& & \Sigma^L & & & & & \rTo^{\varphi^r} & & & & \Sigma^R \\
& \ruTo^{\lambda^L} & & & & & & & & \ruTo_{\lambda^R} & \\
L & & & \dTo_{\varphi^m} \lMono^{i_1}:{.25} \br & & L_r & & & \rTo^r & & R & & \\
& & & & & & & & & & & & \\
\uMono^{i_2} & & & & & \uMono_{i_4} & & & & \uMono_{i_6} & & & \dTo^{\varphi^{m^*}} \\
& & & & & & & & & & & & \\
L_m & & & & \lMono^{i_3}:{.25} \br & & K_{r,m} & & & \rTo^r & & R_{m^*} & & \\
& & & & & & & & & & & & \\
\dTo_m & & & \Sigma^G & & & \rTo_{\varphi^{r^*}} & & \dTo^m:{.25} \br & & & & \dTo^{m^*}:{.25} \br & & & \Sigma^H \\
& \ruTo^{\lambda^G} & & & & & & & & & \ruDashto_{\lambda^H} & & \\
G & & & \lMono_{i_5} & & & G_{r^*} & & & \rTo_{r^*} & & & H & & \\
\endDiagram
```

8.10 Paul Taylor

```
\diagramstyle[heads=littleblack,size=1.5em,PS]
\begin{diagram}
& & \Sigma^L & & & & & \rTo^{\varphi^r} & & & & \Sigma^R \\
& \ruTo^{\lambda^L} & & \vLine & & & & & & & \ruTo_{\lambda^R} & \\
L & & & \HonV & & \lEmbed^{i_1} & & L_r & & & \rTo^r & & & R & & \\
& & & & & & & & & & & & & & & \\
\uEmbed^{i_2} & & & \vLine^{\varphi^m} & & & & \uEmbed_{i_4} & & & & & \uEmbed_{i_6} & & &
  \dTo_{\varphi^{m^*}} \\
& & & & & & & & & & & & & & & \\
L_m & & & \HonV & & \lEmbed^{i_3} & & K_{r,m} & & & \rTo^r & & & R_{m^*} & & \\
& & & \dTo & & & \dTo_m & & & & & \dTo_{m^*} & & & \\
\dTo^m & & & \Sigma^G & & \hLine & & \VonH & & & \hLine_{\varphi^{r^*}} & & & \VonH & & \rTo & & \Sigma^H \\
& \ruTo^{\lambda^G} & & & & & & & & & & & & & \ruDotsto_{\lambda^H} & \\
G & & & \lEmbed_{i_5} & & & & G_{r^*} & & & \rTo_{r^*} & & & H & & \\
\end{diagram}
```

8.11 Paul Taylor emulating Francis Borceux

```
\diagramstyle[size=1.5em]
\begin{diagram}
& & \Sigma^L & & & & \Ear {\varphi^r} & & & & \Sigma^R \\
& \Near {\lambda^L} & & & & & & & & \neaR {\lambda^R} \\
L & & \Wmono {\qquad i_1} & & L_r & & \Ear r & & R \\ \\
\Nmono {i_2} & & \Sar {\varphi^m} & & \nmonO {i_4} & & & & & \nmonO {i_6} & &
  \saR {\varphi^{m^*}} \\ \\
L_m & & \Wmono {\qquad i_3} & & K_{r,m} & & \Ear r & & R_{m^*} \\ \\
\Sar m & & \Sigma^G & & \saR {\up{m}} & & \eaR {\varphi^{r^*}} & & \saR {\up{m^*}} & & \Sigma^H \\
& \Near {\lambda^G} & & & & & & & & \neaR {\lambda^H} \\
G & & \wmonO {i_5} & & G_{r^*} & & \eaR {r^*} & & H \\
\end{diagram}
```

8.12 Timothy Van Zandt

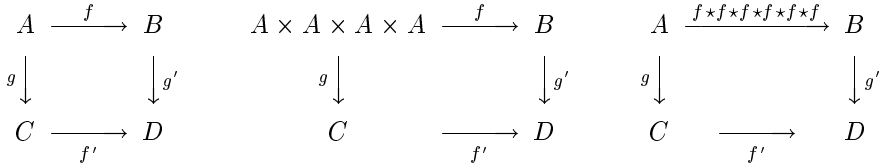
```
$$\setlength{\arraycolsep}{0.1in}
\begin{array}{cccccc}
& \Rnode{SL}{\Sigma^L} & & & & \Rnode{SR}{\Sigma^R} \\ [0.15in]
\Rnode{L}{L} & & \Rnode{Lr}{L_r} & & \Rnode{R}{R} & \\ [0.15in] \\ [0.15in]
\Rnode{Lm}{L_m} & & \Rnode{Krm}{K_{r,m}} & & \Rnode{Rm}{R_{m^*}} & \\ [0.15in]
& \Rnode{SG}{\Sigma^G} & & & & \Rnode{SH}{\Sigma^H} \\ [0.15in]
\Rnode{G}{G} & & \Rnode{Gr}{G_{r^*}} & & \Rnode{H}{H} & \\ [0.15in]
\end{array}
\psset{nodesep=5pt,arrows=->}
\everypsbox{\scriptstyle}
\ncLine{Lr}{R} \Aput{r}
\ncLine{Krm}{Rm} \Aput{r}
\ncLine{Gr}{H} \Bput{r^*}
\ncLine{Lr}{L} \bput{0}(0.3){i_1}
\ncLine{Krm}{Lm} \bput{0}(0.3){i_3}
\ncLine{Gr}{G} \Aput{i_5}
\ncLine{SL}{SR} \Aput{\varphi^r}
\ncLine{SG}{SH} \Bput{\varphi^{r^*}}
\ncLine{SR}{SH} \Aput{\varphi^{m^*}}
\ncLine{SL}{SG} \Bput{\varphi^m}
\ncLine{Lm}{G} \Bput{m}
\ncLine{Krm}{Gr} \aput{0}(0.3){m}
\ncLine{Rm}{H} \aput{0}(0.3){m^*}
\ncLine{Lm}{L} \Aput{i_2}
\ncLine{Krm}{Lr} \Bput{i_4}
\ncLine{Rm}{R} \Bput{i_6}
\ncLine{L}{SL} \Aput[1pt]{\lambda^L}
\ncLine{R}{SR} \Bput[1pt]{\lambda^R}
\ncLine{G}{SG} \Aput[1pt]{\lambda^G}
\ncLine[linestyle=dashed]{H}{SH} \Bput[1pt]{\lambda^H}$$
```

9 Appendix III: Automatic stretching

The following diagrams illustrate the degree of automatic stretching of arrows provided by each of the macro packages. A simple square diagram is typeset with a long label for the top-leftmost node in order to determine if the bottom horizontal arrow stretches to meet its source node, and it is also typeset with a long label for the top horizontal arrow in order to determine if it stretches long enough to fit the label.

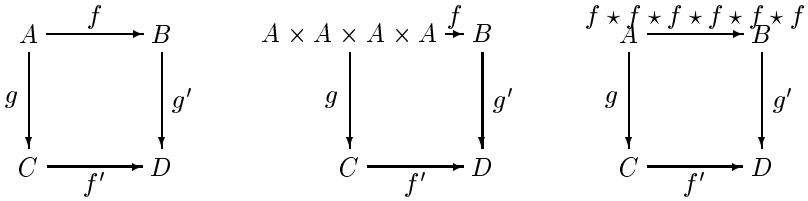
9.1 American Mathematical Society

Arrows do not stretch to meet their source and target nodes, but they stretch to fit their labels, although only the arrow carrying the long label stretches. Manual fine-tuning is needed in order to get the same stretch in all the other arrows lying in the same column of the array.



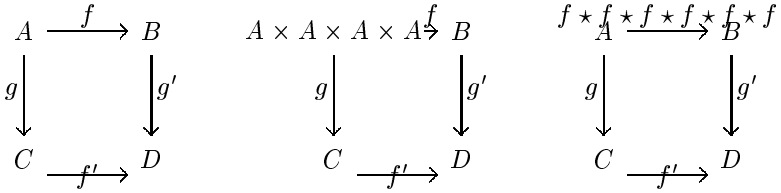
9.2 Michael Barr

Arrows within the shape macros stretch to meet their source and target arrows, but individual arrows obtained with `\putmorphism` do not. In both cases, arrows do not stretch to fit their labels and the required dimensions have to be given explicitly.



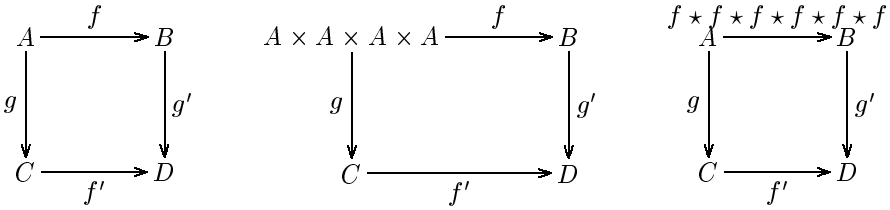
9.3 Francis Borceux

Arrows stretch to meet their source and target nodes, but they do not stretch to fit their labels.



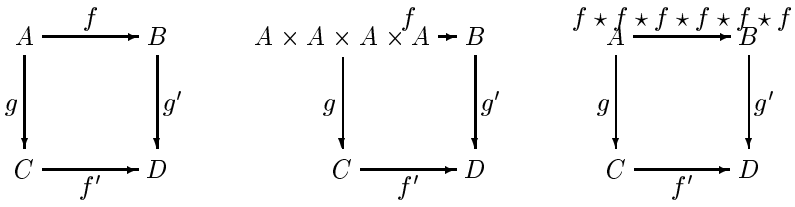
9.4 Eitan Gurari

Arrows stretch to meet their source and target nodes, but they do not stretch to fit their labels.



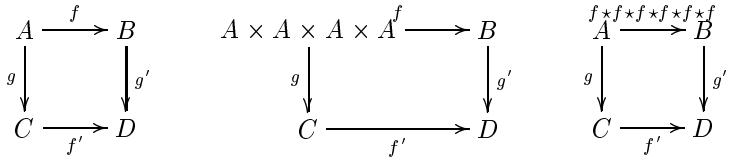
9.5 John Reynolds

Arrows stretch to meet their source and target nodes, although the labels do not get centered on the stretched arrows. They do not stretch to fit their labels.



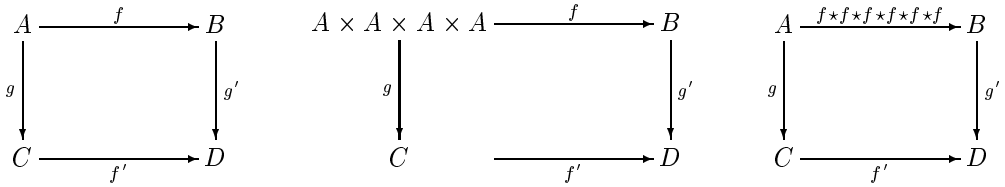
9.6 Kristoffer Rose

Arrows stretch to meet their source and target nodes, although the labels do not get centered on the stretched arrrows. They do not stretch to fit their labels.



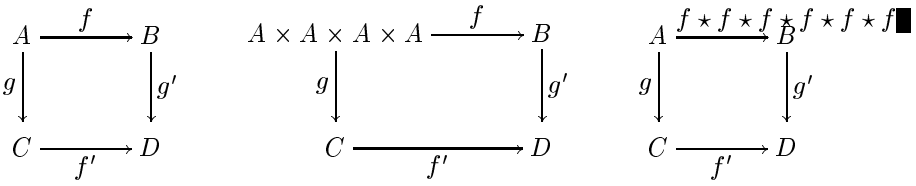
9.7 Steven Smith

Arrows do not stretch to meet their source and target nodes, but they stretch to fit their labels.



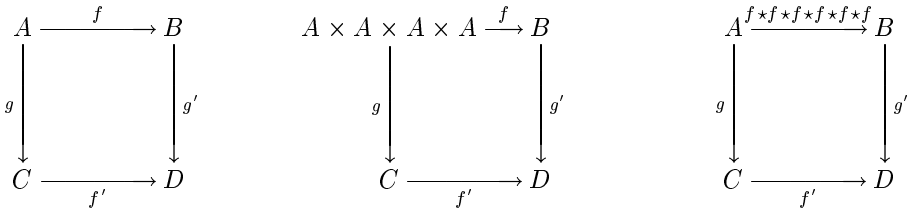
9.8 Michael Spivak

Arrows stretch to meet their source and target nodes, but they do not stretch to fit their labels, even producing overfull \hboxes.



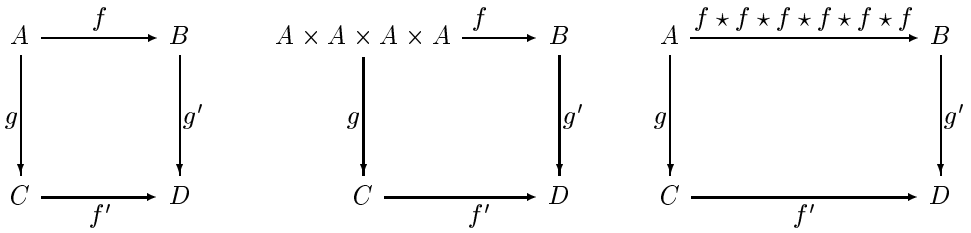
9.9 Anders Svensson

Arrows stretch to meet their source and target nodes, but they do not stretch to fit their labels.



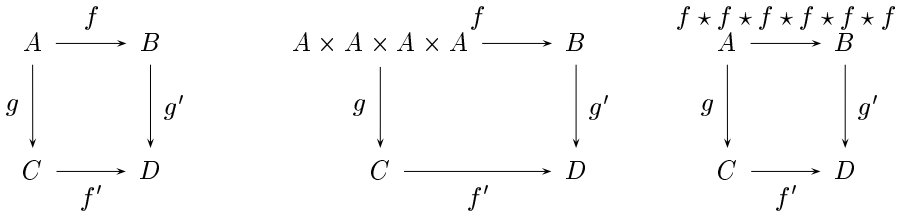
9.10 Paul Taylor

Arrows stretch to meet their source and target nodes, and they also stretch to fit their labels.



9.11 Timothy Van Zandt

Arrows stretch to meet their source and target nodes, although the labels do not get centered on the stretched arrows. They do not stretch to fit their labels, and the required dimensions have to be given explicitly.



10 Appendix IV: Resource requirements

10.1 Package size

The following table lists the size (in kilobytes) of the main macro files that have to be loaded into TEX or LATEX in order to use the respective packages.

package	main files	size
$\mathcal{AMS}$-LATEX	`amscd.sty`	10
Barr	`diagram.tex`	40
Borceux	`Diagram`	270
Gurari	`DraTex.sty` and `AlDraTex.sty`	136
Reynolds	`diagmac.sty`	42
Rose	`xypic.tex` and `xy.tex`	68
Smith	`arrow.tex`	24
Spivak[1]	`amstexl.tex` and `lamstex.tex`	200
Svensson	`kuvio.tex` and `arrsy.tex`	86
Taylor	`diagrams.tex`	86
Van Zandt	`pstricks.tex`, `pst-node.tex` and `pstricks.con`	84

10.2 Time statistics

The following table lists statistics for the time (in seconds) needed to typeset the sample diagrams presented in Appendix I, using TEX and LATEX 2_ε on a Macintosh SE/30, with the different macro packages. The mean time and the confidence interval at a significance level of 95% is given for the total time needed to typeset a diagram and for the marginal time, computed as the difference between the time needed to typeset two copies of the sample diagram using a macro package and the time needed to typeset one copy of the same diagram using the same macro package, where these two random variables are assumed to have a normal distribution and to be independent, and where the mean and the confidence interval have been estimated from a sample of 30 observations.

[1] Although LA$\mathcal{MS}$-TEX offers much more than the macros for commutative diagrams, it has to be loaded as a whole in order to use the macros. Most such macros can be removed from TEX's memory by loading the file `cd.tox` (4 kilobytes), freeing up about 5800 words of memory, and can be later added again by loading the file `cd.tex` (36 kilobytes), but the whole LA$\mathcal{MS}$-TEX has to be loaded before.

package	total time			marginal time		
	mean	95% confidence interval		mean	95% confidence interval	
$\mathcal{AMS}$-LaTeX	18.1367	18.0317	18.2416	1.6600	1.5544	1.7660
Barr	48.8033	48.7731	48.8335	29.9334	29.8800	29.9870
Borceux	127.5630	127.5060	127.6210	28.3170	28.1730	28.4600
Gurari	388.4630	388.4320	388.4950	638.8270	638.5000	639.1490
Reynolds	46.7000	46.6357	46.7643	26.8200	26.7520	26.8880
Rose	242.7400	242.3810	243.0990	210.0730	209.2900	210.8500
Smith	22.9600	22.9031	23.0169	5.2400	5.1817	5.2980
Spivak	37.3000	37.2445	37.3555	11.9833	11.9263	12.0400
Svensson	81.3867	81.2902	81.4831	44.5733	44.4668	44.6799
Taylor	66.8400	66.7553	66.9247	14.3133	14.1420	14.4850
Taylor emul. Borceux	67.3533	67.3243	67.3823	11.4767	11.4360	11.5170
Van Zandt	37.8233	37.7809	37.8657	14.2100	14.1520	14.2680

11 Appendix V: Availability

11.1 Availability

The following table lists the CTAN directories where the different macro packages are stored, together with the authoritative FTP addresses they are mirrored from.

package	CTAN directory	mirrored from
$\mathcal{AMS}$-LaTeX	`macros/latex/packages/amslatex/`	`e-math.ams.org` `/pub/tex/amslatex/`
Barr	`macros/generic/diagrams/barr/`	not mirrored
Borceux	`macros/generic/diagrams/borceux/`	`theory.doc.ic.ac.uk` `/tex/contrib/Borceux/diagram-3/`
Gurari	`macros/generic/dratex/`	`ftp.cis.ohio-state.edu` `/pub/tex/osu/gurari/`
Reynolds	`macros/latex209/contrib/misc/` `diagmac.sty`	not mirrored
Rose	`macros/generic/diagrams/xypic/`	`ftp.diku.dk` `/diku/users/kris/TeX/`
Smith	`macros/eplain/` `arrow.tex`	`ftp.cs.umb.edu` `/pub/tex/eplain/`
Spivak	`macros/lamstex/`	not mirrored
Svensson	`macros/generic/diagrams/kuvio/`	`math.ubc.ca` `/pub/svensson/`
Taylor	`macros/generic/diagrams/taylor/`	`theory.doc.ic.ac.uk` `/tex/contrib/Taylor/tex/`
Van Zandt	`graphics/pstricks/`	`princeton.edu` `/pub/tvz/pstricks/`

The bag of tricks

Victor Eijkhout

Hello all,

One of the things that TEX is commonly said not to be able to do, is letterspacing. I do not want to get involved here in the debate over whether letterspacing is defendable or not. There are places where it's bad, and others where it can safely be used. Karl Berry gave me a particularly neat macro for letterspacing, and I'll leave its application to the reader's discretion[1].

The text to be spaced is passed as an argument:

```
\spreadout{The text} is spread.
```

T h e t e x t is spread.

and expandable material in the text is treated correctly:

```
\def\MoreText{more text}
\spreadout{Here is \MoreText}
than above.
```

H e r e i s m o r e t e x t than above.

The amount of spacing is controlled by a macro with the following default definition:

```
\def\spreadoutfactor{.15}
```

The basic idea behind the macro `\spreadout` is the following. First get rid of all expandable material

```
\def\spreadout#1{%
    \edef\temp{#1}
```

then start processing the result

```
    \dimen0 = \spreadoutfactor em
    \expandafter\dospreadout\temp\endmark
```

where

```
    \def\dospreadout{%
      \afterassignment\findospreadout
      \let\next= }
```

This assigns the first token to `\next`, then calls `\findospreadout`. The latter macro basically amounts to

```
    \next \kern\dimen0
```

except that it has to test for `\endmark`.

Actually, there are a few more gadgets in this macro: the control sequence `\uppercase` is respected by replacing `\uppercase{text}` by `\uppercase{\spreadout{text}}`. Furthermore, a control sequence `\ellipsis` is replaced by three spaced dots.

Here are the actual definitions

```
\def\spreadout#1{%
  \begingroup
    % prevent expansion of \ellipsis
    \def\ellipsis{\noexpand\ellipsis}%
    \xdef\temp{#1}%
  \endgroup
  \dimen0 = \spreadoutfactor em
  \expandafter\dospreadout\temp\endmark
 }
\def\dospreadout{%
  \afterassignment\findospreadout
  \let\next= }
\def\findospreadout{%
  \ifx\next\endmark
    \let\nextaction = \relax
  \else
    \ifx\next\uppercase
      \let\nextaction = \douppercase
    \else
      \ifx\next\ellipsis
        \let\nextaction = \doellipsis
      \else
        \let\nextaction = \dospreadout
        \next
        \kern\dimen0
      \fi
    \fi
  \fi
  \nextaction
 }
\def\douppercase#1{%
  \uppercase{\spreadout{#1}}\dospreadout}
\def\doellipsis{%
  \spreadout{...}\dospreadout}
\def\ellipsis{ellipsis}
\def\endmark{endmark}
```

(The last two definitions are an addition of mine to Karl's macros, since mucking with undefined macros is somewhat dangerous.)

This macro works well, and, although letterspaced words cannot be broken across lines, texts with spaces will be treated as normal paragraphs.

◇ Victor Eijkhout
Department of Mathematics, MS 6363
UCLA
405 Hilgard Avenue
Los Angeles, CA 90024-1555
Internet: eijkhout@math.ucla.edu

[1] Philip Taylor gives macros for letterspacing in *TUGboat* vol. 14, no. 2. Their aim is to letterspace exactly a single line of text in a box.

A TEX Autostereogram Generator

Jacques Richer

Introduction

The Plain TEX code `autostereogram.tex` generates autostereograms. An autostereogram (often called simply stereogram) is a *single* picture that shows objects in 3D but does not require any special device for viewing (other than a normal pair of eyes, of course). Autostereograms have been the subject of a recent craze throughout the world. They now fill entire color albums, such as the following beautiful books:

- *Stereograms*, by Cadence Books, 1994
- *Ultra-3D*, by Montage Publications, 1994
- *Interactive Pictures*, by Benedikt Tashen Verlag, 1994
- *Hidden Dimensions*, by Dan Dyckman, Harmony Books, New York, 1994 (*this is a funny 3D puzzle book*)

They have also been made into postcards and posters. In this article, I show how the basic technique for generating such pictures can be implemented in TEX. I will first explain how `autostereogram.tex` is used and what it does. The TEX coding itself will be described last.

Generalities

The image contents (depth information) is specified to the generator in the form of an ordinary text file, hereafter called the *relief data file*, containing several rows of single digit numbers. These numbers indicate at which height or depth each pixel should be perceived by the viewer. In the original version of this code, no extra file was needed as the image data was generated from within the TEX code itself (by the `\relief` macro), through nested `\if\else\fi` constructs; the external file approach is much more flexible, and easier to use.

The output pixels are identical size `\hbox`es that may contain anything TEX can fit in them: a rule box, a character, or even whole text paragraphs! The whole picture is obtained by tiling the surface with copies of a small number of such `\hbox`es. These basic tiles determine the *texture* of the image, which is totally independent from its 3D contents. Here, I use four tiles consisting in ♡s, in two sizes, the TEX logo, and an empty box, to help produce a lighter texture. Color can be used (here ♡s will be red, if `PSTricks` is available), and is highly recommended.

The image is generated line by line, lines being totally independent from each other. The algorithm consists in repeating horizontally an arbitrary initial pattern of basic tiles, at a repetition rate that is *modulated* by depth information. More precisely, from an initial set of m pixels p_0, $p_{-1}, \ldots, p_{-(m-1)}$ on a line, randomly selected from our basic tile set, one generates n new pixels, where $n \gg m$, using the recurrence relation $p_i = p_{i-\sigma_i}$; σ_i is an integer valued monotonic function of depth d_i at point i. Here, I take that function σ_i to be simply equal to $m - d_i$; clearly, if all depths are equal to 0, for all lines, the final output will consist in identical copies of the first m-pixel wide vertical stripe; hence m is called the *period* of the image. When depths vary, one gets a horizontally distorted version of the zero-depth pattern.

The m starting pixels are not under user control, as far as their depth is concerned, and printing them will lead to more pixels appearing in the picture than the user provided data for; so it may be preferable not to print them. Here they are not printed.

Depth is perceived when the eyes lock their relative orientation so that their aiming points are always separated horizontally by m pixels, or an integer multiple k (k may be negative!) of that distance. Apart from this constraint, *the eyes can move freely and explore the whole picture.* A sort of recursive effect is seen if $|k| > 1$; this is discussed in the paper cited below. For the purpose of this presentation, it is simplest to assume that we are viewing adjacent m-pixel wide stripes. If the two stripes are identical, the 3D impression is that of a flat area floating at some distance away from us; this distance depends on the so-called *vergence angle* between the two lines of sight, that angle depending in turn on the physical width of a *period*.

If the left eye locks onto any stripe — which column it starts in does not matter, as long as it is not too close to the vertical sides — and the right eye locks onto the one immediately to its right, pixels with larger values of d_i are perceived as being closer to the observer than pixels with smaller d_i; if the eyes are crossed ($k < 0$ — left eye locks onto right stripe, and *vice-versa*), larger d_i pixels appear further away. Personally, I find it more difficult to hold the lock with eyes crossed, but that difficulty depends very much on the width of a period and the viewing distance. For the sake of simplicity, I decided to call d_i values depths, even if the word heights would often be more appropriate.

The translation invariance of the algorithm makes it possible to build arbitrarily large pictures.

With autostereograms, 3D perception always occurs as a result of *local distortions* in the repetition cycle; thus an autostereogram covering an entire wall could show 3D in any of its parts. It is easy to see the relationship between autostereograms and stereographic *image pairs* if one thinks in terms of such local disparities between two consecutive stripes; one important difference is that the autostereogram presents us with a *near continuum of stereo pairs* (unfortunately of much lower information content).

For an excellent introduction to the subject of autostereogram generation, by the inventor of the technique, see: Christopher W. Tyler & Maureen B. Clarke, "The Autostereogram", in the Proceedings of the SPIE–The International Society of Optical Engineering, SPIE vol. 1256, "Stereoscopic Displays and Applications", J. O. Merritt & S. S. Fischer, editors, 1990, pages 182–197.

The relief data

To use `autostereogram.tex`, one must supply a relief data file. The first line of that text file should contain the horizontal size of the picture n_{columns}, in pixels, and nothing else; the second line should contain the vertical size n_{lines}, in pixels. The third line must supply the period m. It should be as large as possible, subject to constraints to be discussed below.

The fourth line should contain a small integer g that will tell the generator what depth level is to be interpreted as the *ground plane*. Then should follow n_{lines} lines each containing exactly n_{columns} decimal digits, the d_is (no spaces). The code could be easily generalized to allow an arbitrary number of depth levels. For the following discussion, let's assume that the d_is are in the range $d_{\min}..d_{\max}$.

If $g \leq d_{\min}$, relief will be towards the parallel eye viewer, and if $g \geq d_{\max}$, relief will be away from him/her. Values from $d_{\min}$ to $d_{\max}$ will make it possible to display both receding and advancing objects. As the foregoing sentences imply, g is not limited to the $d_{\min}..d_{\max}$ range, but it is best not to deviate too much from that range. With $g \neq 0$, the algorithm becomes

$$p_i = p_{i-(m-[d_i-g])}$$

and it is necessary to ensure that $m+g+1 \leq i+d_i$ and $d_i \ll m+g$, for any $i > m$, and $d_{\min} \leq d_i \leq d_{\max}$. If $i' = i-(m+1)$ (i' measures the horizontal distance from the right edge of the starting stripe), we must then have $i' \geq g$ for nontrivial pixels: problems can arise if $g > 0$. To prevent such problems the code will extend the initial stripe by exactly g columns, using the ground plane recurrence formula $p_i = p_{i-m}$. The initial stripe (more precisely, one line of it) will then consist (conceptually) of pixels $p_0, p_{-1}, p_{-2} \ldots, p_{m+g-1}$.

```
60
17
10
0
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000002222200000000000000000
000000000000000000000000000000000000002222200000000000000000
000000000000000000000000000000000000002222200000000000000000
000000000000000000000000000000000022222222222220000000000000
000000000000000000000000000000000022222000222220000000000000
000000000000000000022222222000000022222000222220000000000000
000000000000000000022222220000000022222222222220000000000000
000000000000000000022222200000000000000222220000000000000000
000000000000000111122222111000000000002222200000000000000000
000000000000000111122221111000000000002222200000000000000000
000000000000000111122211111000000000000000000000000000000000
000000000000000111122111111000000000000000000000000000000000
000000000000000111121111111000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
```

Figure 1: Input data for a 3-level picture.

The other constraint, $d_i \ll m+g$, is more subtle: $d_i < m+g$ is necessary, but not sufficient, because if the shift $\sigma_i = m-[d_i-g]$ is too small, many of the initial pixels on the current line will never be used, and the output line could easily collapse to a series of identical pixels. So the the input deviations $d_i - g$ should be kept small. There is an inevitable loss of information in the recurrence relation, and it is necessary to have as much information as possible to begin with. This is one reason why a large m is preferable; another reason is that the perceived angular resolution is roughly inversely proportional to m since it is equal to the width of a pixel (which is normally made smaller when m is larger) divided by the viewing distance.

Figure 1 shows a small multilevel example (too small to yield good results). To generate the real example shown in Figure 3, I applied METAFONT to the very nice Irish font `eiad10` to generate a simple 2-level input file. The (UNIX style) commands

```
mf "\mode:=aps; " input eiad10
gftype -i ./eiad10.723gf > eiad.txt
```

produced a text file `eiad.txt` containing drawings of all the characters in the font, where each black pixel is represented by a `*` and each white pixel by a space. From this file I extracted the letter '`b`', which I then edited to change spaces into `0`s and `*` into `1`s, and padded with extra `0`s on all four sides to make it into a rectangular array including a small margin all around. I finally added the four parameter lines, to get the file `eiadb.dat` shown in Figure 2, which `autostereogram.tex` converted to Figure 3.

Note that if the characters are not fat enough, their thinnest parts will not be visible in 3D because of the coarse resolution imposed by the use of character boxes and integer σ_is. It is necessary to call METAFONT with a high resolution `mode` or to require

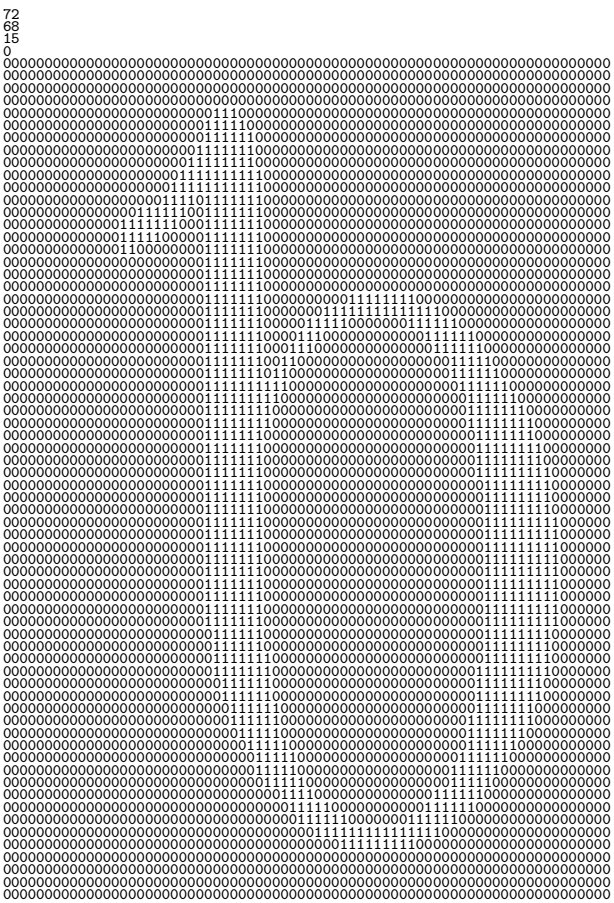


Figure 2: Character 98 from font `eiad10`.

some magnification, (*e.g.*, `mf "\mode:=localfont; mag=magstep(4); " input eiad10`). Our implementation of `gftype` truncates its output at 80 columns; this limits the range of usable magnifications.

The input file represents the whole picture (the same number of pixels will be printed); the relative positions of elements in this "picture" will correspond to their relative positions in the 3D output. There will appear to be a small leftward shift due to the left-right asymmetry in the recurrence relation. Automatically correcting for this asymmetry would require dropping user data on the right and/or adding new ground plane pixels on the left. Another solution is to put in a small rightward shift in the input file to begin with. Here the code makes no centering correction, and leaves that problem to the user.

The code

Most of the code is trivial. First we declare a few counter and dimension variables, most of which have an obvious meaning:

```
% Load PSTricks, if available, for color
%\input pstricks.tex
\newcount\lines\newcount\columns
\newcount\nx\newcount\ny
\newcount\ThisPixelColor
\newcount\ReferenceDepth
\newcount\Period
\newcount\StartingStripeWidth
\newdimen\dx\dx=3pt
\newdimen\dy\dy=\dx
\newread\DepthData\newlinechar=`\^^J
```

Counters `\lines` and `\columns` are read from the data file; `\columns` will be augmented by the program to include the necessary number of invisible pixels, `\StartingStripeWidth`. Dimension `\dx` specifies the pixel size, if we are drawing black and white *square* pixels (a random dot autostereogram); one may set `\dy` to a different value; the code that uses `\dx` and `\dy` is commented out because we will use characters instead, as texturing elements.

`\Period` is the period m of the image, as defined above. It should be as large as possible, for the reasons given in the previous section, but it is limited by the requirement that `\Period` times the pixel width should be of the order of an inch or two (this is about the distance between points your eyes will converge to on the paper; it must be neither too small nor too large, for good comfortable viewing).

A random number generator is used to create the initial stripe on the left hand side of the figure. The rest of the picture will be obtained by copying elements from that stripe. The code for the random number generator, using a simulated shift register, is taken from an article by Hans van der Meer in *TUGboat*, Volume 15 (1994), No. 1, pages 57–58.

```
\catcode`\@=11
\newcount\@SR
\def\@SRconst{2097152}
\def\SRset#1{\global\@SR#1\relax}
\def\@SRadvance{\begingroup
  \ifnum\@SR<\@SRconst\relax \count@=0
  \else \count@=1
  \fi
  \ifodd\@SR \advance\count@ by 1 \fi
  \global\divide\@SR by 2
  \ifodd\count@
      \global\advance\@SR\@SRconst\relax
  \fi
```

```
  \endgroup}
```

It is necessary to initialize the register with some seed value, and to step it a number of times before its output becomes "random". We also define macros `\SRtest` and `\SelectPattern` that use the generator to select randomly between 2 and 4 arguments, respectively.

```
\SRset{1141651}
\nx=20
\loop\ifnum\nx>0
  \@SRadvance\advance\nx by-1\repeat

\def\SRtest#1#2{\@SRadvance
       \ifodd\@SR #1\else #2\fi}
\def\SelectPattern#1#2#3#4{%
  \SRtest{\SRtest{#1}{#2}}%
         {\SRtest{#3}{#4}}}
\catcode`\@=12
```

Different seed values will result in different texturing patterns, so one should experiment with seeds if the output shows distracting accidental defects, like large holes, or lines that seem to be much darker or lighter than the average (this particular kind of defect could also be the result of excessive loss of information, as explained before; use of a larger `\Period` should help cure this).

A test is introduced, `\ifPrinting`, to distinguish between invisible starting pixels, and the printing ones. Continuously testing this flag slows down operations, but it simplifies the code structure.

```
\newif\ifPrinting
```

Next comes a set of macros defining the *basic tiles*: boxes `\abox`, `\bbox`, `\cbox`, `\dbox`, and corresponding macros `\A`, `\B`, `\C`, `\D`. Here, just about anything is allowed, as long as the pixels all have the exact same dimensions. Their aspect ratio can be adjusted, if desired, to compensate for the non-square aspect ratio of characters in the relief data file, when it is being edited. Best results are obtained when pixels are very small, so here I use 3.5pt and 4.5pt characters. The heights and widths of boxes `\abox`...`\dbox` are all coerced to 4.5pt, and depths are coerced to 0pt. Note that this will make the TEX logo overflow into neighboring pixels, but that doesn't matter since the result looks good (at least to me). Box `\bbox` is left empty, to produce a ligther texture.

```
% One way to add a little color:
\ifx\PSTricksLoaded\endinput
  \immediate\write16{Using pstricks}
\else
   \def\black{}
   \def\red{}
\fi

\font\TeXlogo=cmr5 at 3.5pt
\font\Cards=cmsy6 at 4.5pt
\font\Cardlets=cmsy6 at 3.5pt
\newbox\abox\newbox\bbox
\newbox\cbox\newbox\dbox

\setbox\abox=\hbox{%
  \black\kern-1.5pt\TeXlogo\TeX}
\setbox\bbox=\hbox{}
\setbox\cbox=\hbox{\red\Cards\char"7E}
\setbox\dbox=\hbox{\red\Cardlets\char"7E}
\wd\abox=4.5pt
\ht\abox=\wd\abox\dp\abox=0pt
\ht\bbox=\ht\abox
  \wd\bbox=\wd\abox
  \dp\bbox=\dp\abox
\ht\cbox=\ht\abox
  \wd\cbox=\wd\abox
  \dp\cbox=\dp\abox
\ht\dbox=\ht\abox
  \wd\dbox=\wd\abox
  \dp\dbox=\dp\abox
```

The following macros must not insert extra spaces:

```
\def\A{%
  \leftappenditem{\A}\to\CurrentLine%
  \ifPrinting\copy\abox\fi}
\def\B{%
  \leftappenditem{\B}\to\CurrentLine%
  \ifPrinting\copy\bbox\fi}
\def\C{%
  \leftappenditem{\C}\to\CurrentLine%
  \ifPrinting\copy\cbox\fi}
\def\D{%
  \leftappenditem{\D}\to\CurrentLine%
  \ifPrinting\copy\dbox\fi}
```

Boxes containing a black and a white square are also defined: `\noir` and `\blanc`, plus macros `\K` and `\W`; these can be used to generate random dot autostereograms of the sort presented in Christopher Tyler's early papers.

```
\newbox\blanc\newbox\noir
\setbox\blanc=\hbox to \dx{%
   \vrule height\dy
   depth0pt width0pt\hfil}
\setbox\noir=\hbox to \dx{%
   \vrule height\dy depth0pt width\dx}

\def\W{%
  \leftappenditem{\W}\to\CurrentLine%
```

```
  \ifPrinting\copy\blanc\fi}
\def\K{%
  \leftappenditem{\K}\to\CurrentLine%
  \ifPrinting\copy\noir\fi}
```

To implement the $p_i = p_{i-\sigma_i}$ formula, my original plan was to use some of TEX's tables (`lccode`, `uccode`, etc...) as arrays to store these values, but I eventually opted for a well documented more conservative approach: I use the list manipulation techniques presented in Appendix D of the TEXbook. An algorithm based on code tables would certainly show better performance. A list called `\CurrentLine` holds symbolically the contents of the output line that is currently being worked on, in the form of a list of single letter macro tokens. Every time one of the basic tiles has been selected, through a call of the form `\select`σ_i `\of\CurrentLine`, the corresponding macro token (`\A`, `\B`, `\C`, `\D` — or `\K`, `\W`) is inserted with `\leftappenditem` at the head of the `\CurrentLine` list,

```
\toksdef\ta=0 \toksdef\tb=2
\long\def\leftappenditem#1\to#2{%
      \ta={\\#1}%
      \tb=\expandafter{#2}%
      \xdef#2{\the\ta\the\tb}}
```

and the same macro is also immediately executed to copy the box contents at the current position in the printed output line.

The macro `\clearline` resets the `\nx` column counter, and initializes the list to `\outofrange`; `\outofrange` is used as a sentinel and is useful for trapping errors and to accelerate the workings of another important macro, `\GobbleRest`.

```
\def\outofrange{\immediate\write16{%
     ^^JOut-of-range relief:
     left limit reached^^J}}

\def\clearline{%
   \gdef\CurrentLine{\\\outofrange}%
   \global\nx=0}
```

The macro `\select` works basically as explained in the TEXbook (p. 379), except for a little change that dramatically increases its speed (speed is important in the present application).

```
\def\select#1\of#2{%
  \gdef\result{\outofrange}%
  \gdef\\##1{\advance#1-1 %
          \ifnum#1=0 %
            \def\result{##1}%
            \let\\=\GobbleRest%
          \fi}%
   #2\result}
```

```
\def\GobbleRest#1\outofrange{}
```

It does a backward search in the list of p_is for the $(i - \sigma_i)^{\text{th}}$ element. It defines `\\` to be a macro that gobbles the next token, doing nothing else, until the sought after list element has been reached; at that point, it saves the token for later reinsertion in the input stream, and redefines `\\` to be a macro that eats everything else in the stream, in one step, up to the end-of-list marker `\outofrange`. The execution time is no longer proportional to the length of the list, but instead it is roughly proportional to the value of `\Period` (assuming most pixels have $\sigma_i \approx m$)

The top level macro that selects what will appear at the current position is `\MakeThisPixel`. It gets depth values d_i from `\relief`, which was originally written to supply a fixed, hard coded pattern; now it gets the data from the current relief input line `\DepthDataLine`, with the help of `\GetNextInputDigit`.

```
\def\MakeThisPixel{{%
 \ThisPixelColor\expandafter=\relief%
 \shift%
 {\select\ThisPixelColor\of\CurrentLine}%
 }}

\def\relief{\expandafter%
  \GetNextInputDigit%
    \DepthDataLine\endofline}

\def\GetNextInputDigit#1#2\endofline{%
     #1\relax\gdef\DepthDataLine{#2}}
```

The heart of the autostereogram algorithm is the trivial (!) `\shift` macro, whose rôle should be obvious by now: it computes σ_i.

```
\def\shift{%
  \advance\ThisPixelColor
          by-\ReferenceDepth
  \ThisPixelColor=-\ThisPixelColor
  \advance\ThisPixelColor by\Period}

% ******** End of definitions ********
```

After reading the main picture parameters, and doing a simple feasibility check, the code ends with a very simple main loop over lines and columns. The width of the required invisible stripe is calculated, and added to the user specified number of columns.

```
% Now do the work
\openin\DepthData=eiadb.dat
\read\DepthData to\DepthDataLine
         \columns=\DepthDataLine
```

```
\read\DepthData to\DepthDataLine
           \lines=\DepthDataLine
\read\DepthData to\DepthDataLine
          \Period=\DepthDataLine
\read\DepthData to\DepthDataLine
  \ReferenceDepth=\DepthDataLine

\ny=\lines
\nx=\Period\advance\nx by\ReferenceDepth
\ifnum\nx<0
  \immediate\write16{Illegal parameters:
     imply forward recursion.  You should
     probably raise the reference level.
     For this run, it is forced to zero.}
  \ReferenceDepth=0
\fi

\StartingStripeWidth=\Period
\ifnum\ReferenceDepth>0
  \advance\StartingStripeWidth
         by\ReferenceDepth
\fi
\advance\columns by\StartingStripeWidth
\hbadness=10000
\overfullrule=0pt
\offinterlineskip
\parindent=0pt
```

Before the main loop begins, a pair of crosses is written in the center of the page, that are one `\Period` apart*; `\wd\abox` must be replaced by `\dx`, if one is using the black and white squares.

```
\vfill
\nx=0
\line{\hfil\rlap{$+$}%
  \loop\ifnum\nx<\Period%
    \hskip\wd\abox\advance\nx by 1%
  \repeat\rlap{$+$}\hfil}
\vskip5mm
```

[To get the spacing right, it is important not to output any spurious `blank space` in those parts of the code.] These crosses can be used as a practice target, to get used to the particular periodicity of the picture. The viewer is ready to look at the real picture when he/she is able to see 3 crosses in line at the top of the image, in a stable and cozy way.

Looping over columns is done in three parts. First, `\Period` invisible pixels are generated randomly (line 11 or 12, below); `\SelectPattern` selects among four possibilities. To produce a random dot autostereogram, one would replace the `\SelectPattern` line with the `\SRtest{\K}{\W}` line, which is presently commented out, and adjust `\dx` and `\dy` as necessary. Then if necessary, a few more pixels are generated by recurrence (not at random), with $\sigma_i = m$, to complete the invisible stripe (line 14). If printed, these pixels would be perceived as being part of the reference plane. Finally, the visible part of the line is generated with the full recurrence algorithm (line 17).

```
1. \loop\ifnum\ny>0
2.   \clearline
3.   \message{<\number\ny}
4.   \read\DepthData to\DepthDataLine
5.   \nx=0 \Printingfalse
6.   \line{%
7.     \hfil%
8.     \loop\ifnum\nx<\columns%
9.       \ifnum\nx<\StartingStripeWidth
10.        \ifnum\nx<\Period %
11. %       \SRtest{\K}{\W}%
12.         \SelectPattern{\A}{\B}{\C}{\D}%
13.        \else%
14.         {\select\Period\of\CurrentLine}%
15.        \fi%
16.      \else%
17.        \Printingtrue\MakeThisPixel%
18.      \fi%
19.      \advance\nx by 1 %
20.    \repeat%
21.    \hfil}%
22.  \message{>}%
23.  \advance\ny by -1 %
24. \repeat
25. \closein\DepthData
26. \bye
```

The reader with normal vision, who has difficulties perceiving depth in autostereograms, should experiment first with classic stereo pair images, possibly using a 3D viewer; then he/she should move on to simple high resolution black and white random dot autostereograms; these contain fewer distracting features than the character based autostereograms, but are not as pretty.

◇ Jacques Richer
CEntre de Recherche en Calcul
Appliqué (CERCA),
5160, boul. Décarie, bureau 434,
Montréal, PQ, CANADA H3X 2H9
richer@cerca.umontreal.ca

* This part of the code had to be modifed to run within *TUGboat*; this is the unmodified version.

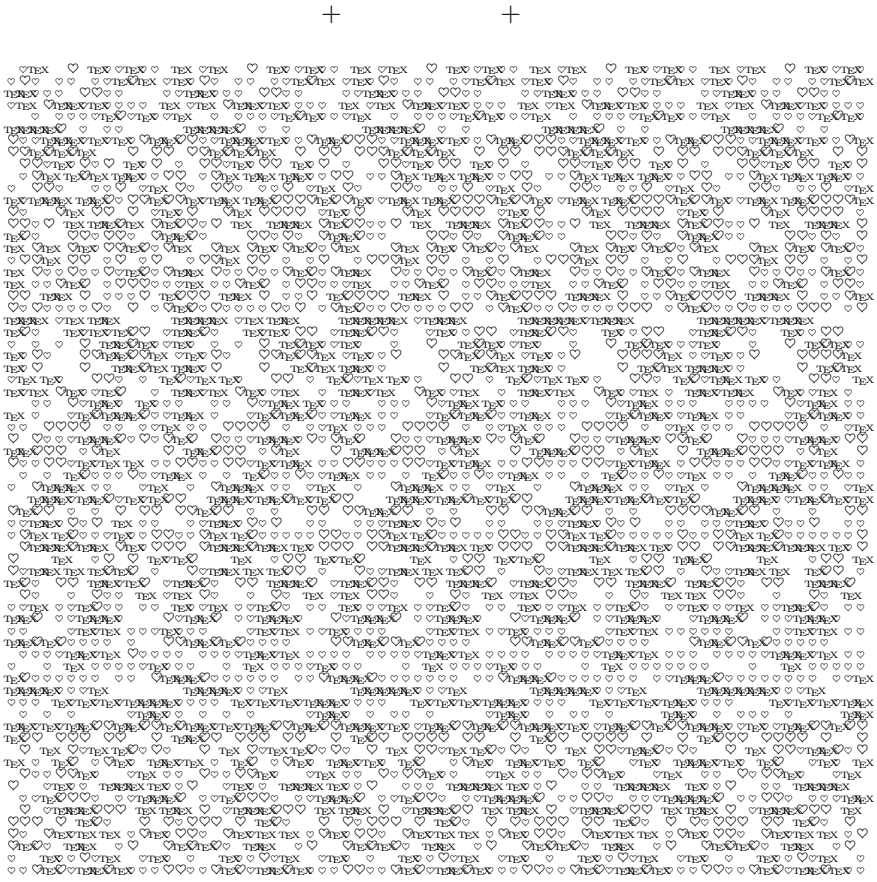


Figure 3: Output generated from Fig. 2.

Stereographic Pictures Using TEX

Reinhard Fößmeier

Abstract

This article shows how to produce stereographic pictures ("magic eye" pictures) using TEX. While it is possible (though slow) to produce such pictures from black and white dots, it is easier to use ordinary glyphs as picture elements.

Resumo

Tiu ĉi artikolo montras kiel produkti kvazaŭ-tridimensiajn bildojn (konatajn sub la nomo "magia okulo") per TEX. Estas eble (kvankam malrapide) konstrui tiajn bildojn el nigraj kaj blankaj punktoj, sed estas pli facile uzi ordinarajn pres-signojn kiel bilderojn.

Introduction

Recently, a new kind of stereographic picture has become rather well known, by names such as "The Magic Eye" or "Stare-E-O". Books containing such pictures (e.g., [1], and other volumes from the same series) for some time were the best-selling non-fiction books in Germany, and no doubt things were similar in a number of other countries.

The new technique differs from the conventional way to present stereographic views: Instead of having two independent views of the same scene, taken from slightly different angles, all the data here are contained in one picture, which must be looked at with a squint, so the visual axes cross behind (or in front of) the picture, and intersect the picture plane at a certain distance.

Since much has been written about it, I won't give here a detailed description of the phenomenon. Suffice it to say that the secret of the 3D effect is that the graphical information in the picture is roughly periodic, the period being the distance between the section points of the visual axes and the picture plane. If the periodicity is perfect, the whole picture is seen as flat. Everywhere the periodicity is disturbed, details stand out in relief, seeming closer

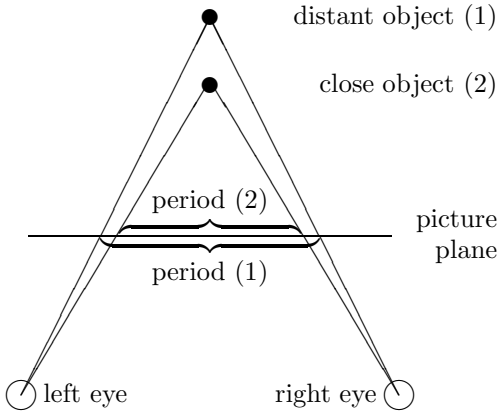


Figure 1: How to look at a magic picture

if the periodicity is less and more distant if it is greater (cf. figure 1).

The `stereo.sty` style

The trick in producing a stereo picture is to start (e.g., at the left) with a certain pattern and to continue it periodically to the right. The period varies according to the distance at which a certain point is to appear: The closer to the eye, the shorter the period. (We assume here that the visual axes cross behind the picture.)

The input form of a picture is a two-dimensional grid of numbers, each number indicating the height of the point it denotes. The number 0 stands for the lowest (most distant) plane. Only numbers between 0 and 9 are allowed. A semicolon delimits a line. The data shown in figure 2 describe a small rectangle that "floats" one unit above the ground plane.

```
000000000000000000000000000000;
000000000000000000000000000000;
000000000000000000000000000000;
000000000000111111000000000000;
000000000000111111000000000000;
000000000000111111000000000000;
000000000000111111000000000000;
000000000000111111000000000000;
000000000000111111000000000000;
000000000000000000000000000000;
000000000000000000000000000000;
000000000000000000000000000000;
```

Figure 2: Input data for a very simple picture

While it would be possible to construct patterns from black and white dots and to draw them, e.g., as `\vrule`s, this process is very slow and consumes a lot of TEX's resources. Unless the pictures are to show very fine details, it is easier to use patterns built from normal TEX glyphs, such as letters and figures. To facilitate the calculation of the period length, we use a mono-spacing font, `\tt`.

Each picture element is produced by a call of the `\Pixel` macro. Its only argument denotes the "altitude" of the point. According to this argument, the pattern (contained in `\Pat`) is shifted around and the correct glyph `\A` for the new element is printed:[1]

```
 1 \newcount\alt
 2 \def\Pixel#1{%
 3    \def\Head##1##2!{##1}%
 4    \def\Tail##1##2!{##2}%
 5    \edef\A{\expandafter\Head\Pat!}%
 6    \edef\Rest{\expandafter\Tail\Pat!}%
 7    \edef\T{\Rest}%
 8    \alt=#1
 9    \loop\ifnum\alt>0
10        \edef\A{\expandafter\Head\T!}%
11        \edef\T{\expandafter\Tail\T!}%
12        \advance\alt -1
13    \repeat
14    \edef\Pat{\Rest\A}%
15    \A%
16 }
```

To produce a line of pixels, this macro must be iterated until a semicolon is reached. This is done by the following macro, `\Line`, which calls itself recursively until it sees a semicolon:

```
17 \def\Line#1{%
18     \if #1;\vskip 0pt \else
19         \Pixel#1%
20         \expandafter\Line
21     \fi
22 }
```

We now could simply give an initial value for `\Pat` and bracket each line of the picture data by `\Line`$\cdots$`;`. The pattern, however, loses details each time it is shifted to a shorter period, and cannot recover the lost information when periods become longer again. We could end up with a pattern consisting of only one sort of glyph. So it is better to restore a reasonable value at the beginning of each line. This is

[1] The linenumbers in the code segments are added for clarity and are *not* part of the macro text.

done in the following macro, `\DLine`; we simply use the same start pattern for all lines.

```
23 \def\DLine#1;{%
24     \edef\Pat{\StartPattern}%
25     \Line#1;%
26 }
```

Now let's have a look at the result of the following calls, shown in figure 3:

```
27 \edef\StartPattern{A-CeL'+MX-/()pd=}
28 \DLine 00000000000000000000000000000000;
```

... (as in figure 3)

```
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/(pd=A-CCeL'+MX-/(pd
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
A-CeL'+MX-/()pd=A-CeL'+MX-/()p
```

Figure 3: Picture from the data in figure 2

The effect of figure 3 is not very convincing as the picture is simply too small: it does not even show two full periods of the pattern. Now we could inflate the data to make the picture larger; on the other hand, TEX can do this for us. We just have to change the following lines in our macro definitions:

```
19          \Pixel#1\Pixel#1\Pixel#1%
25      \Line#1;\Line#1;%
```

to blow up each pixel to a 3×2 matrix. (To do this with loops is left as an exercise to the reader.) The above data (trimmed somewhat to fit into the column) then yield the picture in figure 4.

Now, if you have a trained magic eye, you should be able to see the floating rectangle in the middle of figure 4, and a somewhat more informative picture in figure 5. (If you have no experience with such pictures, ask somebody who has, or simply try staring at it, or forget about the whole thing—it's not really vital after all!) Figure 5 shows that it can be useful to compress the picture vertically, by specifying a negative `\vskip` in line 18.

Why No Dot Patterns?

To be really effective and bring out finer details, magic pictures have to use patterns with a finer resolution than that of glyphs. Basically, this can be done with TEX, e.g. by `\vrule`s and `\kern`s for black and white dots[2]. Details on how to do this can be found in [2]. This process, however, not only is very slow but also consumes a lot of TEX's internal memory. Unless Big TEX is used, pictures are restricted to a rather small size.

Conclusion

Given that TEX's primary domain is typesetting texts, it is no surprise that the easiest way to produce magic pictures with TEX is through the use of character symbols. The pictures produced this way do not come up to the quality of dot graphics but certainly do have a charm of their own.

Possible extensions of the macros presented are:

- use of loops to inflate pixels by variable factors;
- reading picture data from an external file, possibly without the need to insert macro calls into the data;
- starting each line with a copy of the pattern, so details at the left edge of the picture can be seen;
- use of different patterns for each line;
- automatic construction of suitable random patterns.

References

[1] Das magische Auge. Dreidimensionale Illusionsbilder von N. E. Thing Enterprises. München: arsEdition, 1994. Original: Magic Eye, Kansas City: Andrews and McMeel, 1993.

[2] R. Fößmeier: X Bitmaps in TEX. TUGboat 12 (1991), 2, 229–232.

◇ Reinhard Fößmeier
iXOS Software GmbH
Bretonischer Ring 12
DE-85630 Grasbrunn
Germany
Reinhard.Foessmeier@ixos.de

[2] I am indebted to Bernd Raichle, raichle@azu.informatik.uni-stuttgart.de, for this hint.

Figure 4: Picture from the data in figure 2, with inflated pixels

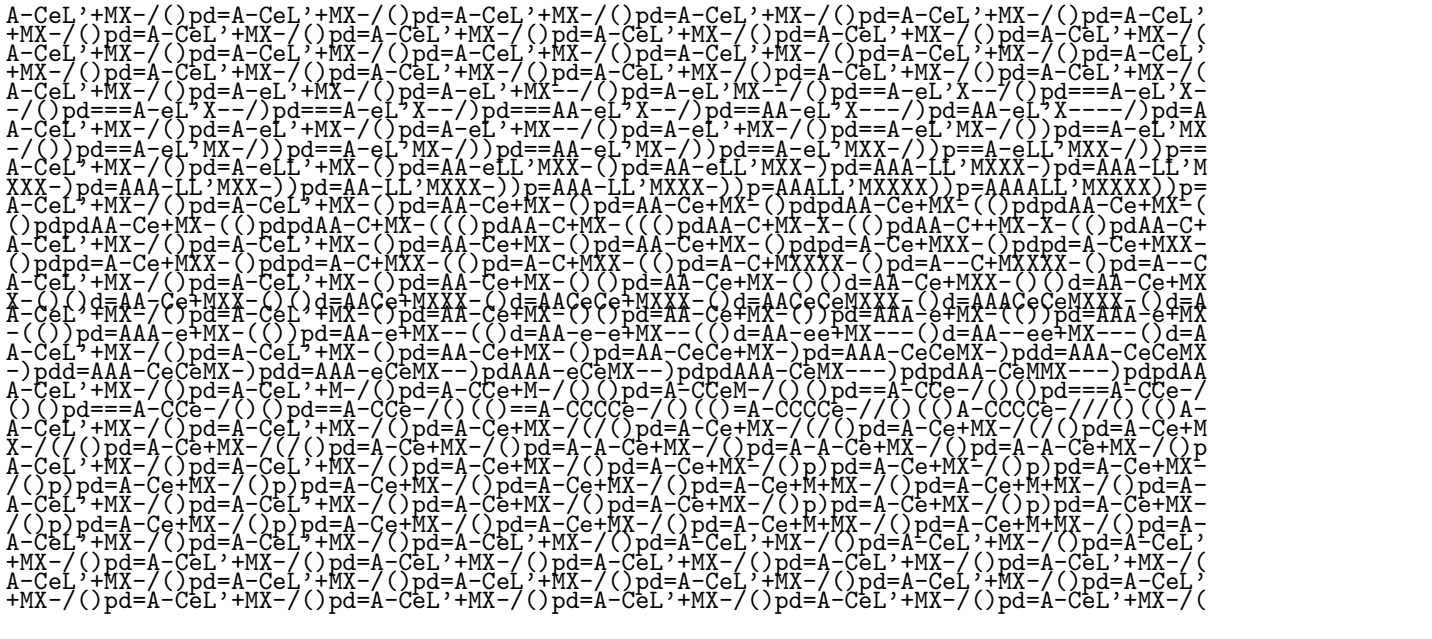


Figure 5: A more technical picture

LaTeX

To reset or not to reset

Johannes Braams

Abstract

This article describes two possible implementations of a `\@removefromreset` macro that can be used to remove a counter from the reset list of another counter.

1 Introduction

When writing a document class it is sometimes necessary to instruct LaTeX that a certain counter has to be reset when another counter gets a new value. This is the case when one wants to number equations within sections. For this purpose LaTeX has the internal command `\@addtoreset`.

Lately people have requested to do the opposite; when they use a document class that has set equation numbering to be within sections they want to be able to number the equations consecutively throughout the document. For this one would need the command `\@removefromreset`, but that command is *not* available in LaTeX.

2 The reset list

When a LaTeX counter is defined using the command `\newcounter` a number of data structures are set up that belong to that counter. Say we allocate a counter foo with the command

```
\newcounter{foo}
```

Then among other things the command `\thefoo` is defined which is used to represent the value of the counter in printed text. One of the other things that are set up is the 'reset list'. This reset list is a list of counters that are to be reset when the counter foo receives a new value with one of the commands `\stepcounter` or `\refstepcounter`. The reset list for the counter foo is stored in the macro `\cl@foo`.

Before we can start to think about the implementation of `\@removefromreset`, we have to know what kind of data structure is used to store a reset list. When we look up the definition of `\@addtoreset` to find out how it works we find the following piece of code:

```
\def\@addtoreset#1#2{%
  \expandafter\@cons
    \csname cl@#2\endcsname {{#1}}}
```

This tells us that `\@addtoreset` is a command that has two arguments, the first of which is the name of a counter to be added to the reset list of the second argument. This is done using the command `\@cons`, so to find out more about the data structure we have to keep digging. Notice that the name of the counter to add to the reset list is passed to `\@cons` inside an extra pair of braces!.

Searching for the definition of `\@cons` reveals:

```
\def\@cons#1#2{%
  \begingroup
    \let\@elt\relax
    \xdef#1{#1\@elt #2}
  \endgroup}
```

This shows us that the reset list is a list of counter names, separated by the command `\@elt`. So the expansion of `\cl@foo` could look like:

```
\cl@foo -> \@elt {bar}\@elt {baz}\@elt {cnt}
```

So, when the command `\stepcounter{foo}` is executed the counters bar, baz and cnt are all reset (get the value 0).

3 Removing an element from the reset list, the idea

Now that we know what the data structure looks like we can start to think about how to remove an element from the list. The essential piece of information we have learned from our search is that *each* counter name in the reset list is preceded by the command `\@elt`.

So the way to the solution to our problem is obvious. We have to give the command `\@elt` a new definition. What should it do? The first thing that comes to mind is that it should compare the name following it with some other name. When those two names are the same we have found the name of the counter to be removed from the list. But how to do that? A solution for this is to build up a new reset list while processing the existing list. If we do that we simply do not include the counter to be removed in the new reset list.

4 Removing an element from the reset list, the implementation

Now that we know the basic idea of how to solve the problem we can start the implementation. I will show two possible implementations.

4.1 First implementation

We are going to define the command `\@removefromreset`. It will have two arguments. The first argument is the name of the counter to remove; the second argument is the name of the counter whose reset list has to be changed.

```
\def\@removefromreset#1#2{%
```

The first thing to do is to start a group and store the name of the counter to remove from the reset list in a command.

```
\begingroup
\def\toremove{#1}%
```

Then we save a copy of the current reset list. We do this because we shouldn't overwrite it while rebuilding a new version.

```
\expandafter\let\expandafter\old@cl
  \csname cl@#2\endcsname
```

In order to rebuild the reset list from scratch, we empty it.

```
\expandafter\let\csname cl@#2\endcsname
                \empty
```

Now we are set up to process the elements of the reset list, except for the proper definition of `\@elt`. Remember that `\@elt` will be defined by the execution of `\@removefromreset` so we have to double the # marks for the argument of `\@elt`.

```
\def\@elt##1{%
```

First we store the argument of `\@elt` in a command in order to be able to use `\ifx` later on for the comparison.

```
  \def\found{##1}%
```

Now we can compare the name of the counter to remove from the list with the name we have just found.

```
  \ifx\toremove\found
```

If they are the same we do nothing, thereby effectively removing it from the list. If they are different we use `\@addtoreset` to build up the new reset list.

```
  \else
    \@addtoreset{##1}{#2}%
  \fi}%
```

Now we have defined `\@elt` so we can simply execute the reset list. This will execute all the occurrences of `\@elt` that are in the list.

```
\old@cl
```

All that is left to do now is to close the group so that TEX forgets about any temporary definitions we made. Notice that the new reset list was built using `\@addtoreset` which uses global definitions inside.

```
\endgroup}
```

4.2 Second implementation

A slightly different approach is taken in the following implementation of `\@removefromreset`.

```
\def\@removefromreset#1#2{%
  \begingroup
```

This time we use a token register to temporarily store the new reset list that is to be built up.

```
  \toksdef\newlist8\newlist{}
```

Again we store the first argument in a control sequence for future use in the `\ifx` test.

```
  \def\toremove{#1}%
```

Again we use `\@elt` to check whether the following list-element is the one we are looking for.

```
  \def\@elt##1{%
```

Store the list element in a control sequence

```
    \def\found{##1}
```

and compare it with the one to remove.

```
    \ifx\found\toremove
    \else
```

If it was not the one we are looking for, add the current list element to the new copy of the list, stored in token register `\newlist`.

```
      \expandafter\newlist\expandafter{%
        \the\newlist\@elt{##1}}
    \fi}
```

Now we can simply execute the reset list which will execute all the occurrences of `\@elt` that are in the list.

```
  \csname cl@#2\endcsname
```

Finally, we have to remember to copy the contents of `\newlist` to the original reset list.

```
  \expandafter\xdef\csname cl@#2\endcsname
                  {\the\newlist}
\endgroup}
```

◇ Johannes Braams
PTT Research
St. Paulusstraat 4
2264 XZ Leidschendam
The Netherlands

Abstracts

Les Cahiers GUTenberg
Contents of Recent Issues

Numéro 18 — septembre 1994

MICHEL GOOSSENS, LaTeX2e, un aperçu [LaTeX2ε: an overview]; pp. 1–34

Author's abstract: "This article gives an overview of the new or extended user commands available with LaTeX2ε, the new LaTeX release, compared to the previous version LaTeX 2.09. After introducing the new preamble commands, the extensions for defining new commands and environments, and handling length and boxes are discussed. The new font selection commands are explained, both for text and math, and it is shown how to easily use different font families. A list of supported class and package files is given and new possibilities for controlling page contents and floats are discussed. Most of this material is described in much greater detail in *The LaTeX Companion* and in the second edition of the LaTeX Reference Manual."

The abstract above is quite complete. The article includes a brief summary of the objectives for LaTeX2ε, neatly capturing what we gain with the new LaTeX:[1]

- create a single LaTeX format, replacing the many (increasingly incompatible) formats available previously
- replace variant LaTeX 'dialects', such as AMS-LaTeX and SliTeX, by add-on packages — `amsmath` and `slides`, in this particular instance — all using the same base format
- add a small number of oft-requested functions
- maintain compatibility at the document level (that is, LaTeX 2.09 source files should not have to be modified in order to run with the new LaTeX)
- retain LaTeX 2.09 conventions in order to make learning the new elements as easy as possible
- the New Font Selection Scheme (NFSS) becomes the standard in LaTeX2ε

This article joins other accessible introductions to the new LaTeX: in the *TUGboat* proceedings issue, "Document Classes and Packages for LaTeX2ε" by Johannes Braams, and "PostScript Fonts in LaTeX2ε" by Alan Jeffrey; and in the Gdansk proceedings, Dag Langmyhr's "How to make your own document style in LaTeX2ε". Braams and Langmyhr can also be found in MAPs #13 (94.2).

DANIEL FLIPO, BERNARD GAULLE, KARINE VANCAUWENBERGHE, Motifs français de césure typographique [French hyphenation patterns]; pp. 35–60

Author's abstract: "The aim of this article is to compare the various current versions of the French hyphenation files and to propose a completely new updated and corrected version. A short introduction is given to French hyphenation as well as to TeX word-splitting mechanisms."

BERNARD GAULLE, Commentaires sur la portabilité des documents (LA)TeX [Comments on (LA)TeX document portability]; pp. 61–86

Author's abstract: "In *Cahier* GUTenberg #15, Daniel Taupin expressed his thoughts and experiences about the portability of TeX documents. This article reviews, point by point, using the same headings (with one or two exceptions) and in the same order each of the elements and discusses their technical validity. This study reveals some rules for increasing the portability of (LA)TeX documents that are both simpler and more elementary than the ones proposed in the article in question."

[A summary of the Taupin article appeared in *TUGboat* 14 #2, page 146.]

Numéro 19 — janvier 1995

Editor's note: Thematic issue entitled "Electronic Document Exchange: from LaTeX to WWW, HTML and Acrobat." The issue contains 158 pages in all, with 9 articles, some of which will be appearing in translation in an upcoming issue of *TUGboat*.

CHRISTIAN ROLLAND, Éditorial: diffusion des documents électroniques [Distribution of electronic documents]; pp. 1–2

With all the talk about the "information highway" and what it will bring, there already is a 'real info highway' in action, with keywords such as CD-ROM, CD-I, the Internet, WWW. The articles in this thematic issue were first presented at the January 19, 1995, one-day meeting organised by GUTenberg and held in Nanterre, France.

LUC OTTAVJ, Systémes d'information sur Internet [Information systems on the Internet]; pp. 3–26

Serving as a solid introduction to the terminology and technology which the remaining articles repeatedly refer to, the article provides a detailed

[1] An elaboration of the main benefits can be found in TTN 2,4:10–11.

overview of how the various elements come together, from communications protocols to Internet structure (both global and in France specifically), from e-mail to telnet, file transfers to browsers. [The paper was originally presented at another conference in October 1994, and published in *Le traitement électronique du document* (1994).]

MICHEL GOOSSENS, Introduction pratique à SGML [Practical Introduction to SGML]; pp. 27–58

Author's abstract: "The international standard SGML (Standard [Generalized] Markup Language) deals with the structural markup of electronic documents. It was adopted by ISO (the International Organisation for Standardisation) in October 1986. SGML soon became very popular, also in Europe, thanks in particular to its enthusiastic acceptance in the publishing world, large multinational companies, and, more recently, by the ubiquity of HTML, the hypertext language of WWW. This article provides an introduction to the basic ideas of SGML and should allow the reader a better understanding of the latest developments in the field of electronic documents in general, and in WWW in particular."

FRANÇOIS DAGORN, World-Wide Web, formulaires électroniques, images réactives, etc. [WWW, electronic forms, clickable images]; pp. 59–66

Author's abstract: "This paper details the mechanisms used to create electronic forms or clickable image maps within the World Wide Web."

CHRISTIAN ROLLAND, Présentation de HTML [Introducing HTML]; pp. 67–84

Author's abstract: "This article presents a markup language, HTML (HyperText Markup Language), which is used to represent hypertext documents in World Wide Web. Tags which indicate the most usual structures are shown; then the hypertext tags and other features are exhibited."

VINCENT QUINT, IRÈNE VATTON, L'édition structurée et le World-Wide Web [Structured editing and the World-Wide Web]; pp. 85–97

Creating documents for the World Wide Web is not an easy task. Many authors create such documents by hand. This means they usually have to wrestle with HTML syntax, even if their text editor provides some tools for the job. An alternative is to use filters which come with various document processing systems, but these don't include all the Web-specific tools. Neither method is therefore completely satisfactory.

This paper presents a solution based on the structured document editor, Grif. The Grif editor has been extended to take into account specific features of the Web and provides a comfortable environment for creating Web documents.

MICHEL GOOSSENS, LaTeX – HTML aller et retour [LaTeX – HTML there and back]; pp. 98–120

Author's abstract: "Both LaTeX and HTML are languages that can express the function of the structural elements of a document, and similarities between these two systems are shown. A detailed study of the `LaTeX2HTML` program, written by Nikos Drakos, is proposed. `LaTeX2HTML` allows a quasi-automatic translation of LaTeX documents into HTML. A brief discussion of the `LaTeX2HTML` and `SGML2TeX` programs that translate HTML into LaTeX concludes the article."

PHILIPPE LOUARN, Documents électroniques: une application [Electronic documents: an application]; pp. 121–126

Author's abstract: "Each year, INRIA produces an activity report. Although this report is typeset in an electronic form, it was never exploited in this way. This paper describes a new process, based on SGML, which allows users to access the report by different ways (WWW, Minitel, ftp, ...). Advantages and disadvantages of this process will be shown and future developments will be presented."

YANNIS HARALAMBOUS, HTML → LaTeX → PDF, ou l'entrée de TeX dans l'ère de l'hypertexte [HTML → LaTeX → PDF, or TeX enters the age of hypertext]; pp. 127–147

Author's abstract: "In this paper we describe the process of creating electronic hyperdocuments via LaTeX and Adobe Acrobat. After a general discussion on the advantages and disadvantages of LaTeX in this field, we give a detailed description of each step and a lot of caveats for the user willing to obtain efficient Acrobat documents.

The reader will find in this paper a discussion of the software tools DVIHPS `repere` and `recticrt` as well as the basic principles of the PDF format."

JACQUES ANDRÉ, MICHEL GOOSSENS, CHRISTIAN ROLLAND, Diffusion des documents électroniques: bibliographie [Distribution of electronic documents: a bibliography]; pp. 148–157

While each article in the thematic issue has its own bibliography, this extensive bibliography includes 99 entries divided into seven headings, including both paper and electronic materials.

(Compiled by Christina Thiele)

Abstracts of the *Proceedings of the Eighth European TEX Conference*, Gdańsk, September 1994

Editor's note: Several of the papers presented at EuroTEX'94 were substantially the same as papers presented at TUG'94; for such papers, a cross-reference is given to the TUG'94 Proceedings.

Principles

JANUSZ S. BIEŃ, Polish texts in multilingual environments (a case study); pp. 3–17

Author's abstract: "There are two possible approaches to the more general use of software originally developed for one specific natural language: to create a version specific to another natural language, or to make a multilingual version preserving all or most of the features of the original, and additionally handling also the other language(s) in question. The second approach is much more difficult than the first, but for an important class of users, especially those from the academic milieu, the first approach is of little use: more and more papers are now being jointly written by multinational teams communicating over the net; being able to use the same version both for national and international papers is an important advantage.

The present paper discusses the most important aspects of the multilingual adaptation of TEX from the point of view of typesetting Polish texts."

BOGUSŁAW JACKOWSKI and MAREK RYĆKO, Labyrinth of METAFONT paths in outline; pp. 18–32

Author's abstract: "The main reason for publishing this text was our need to share with other METAFONT fans our experience in non-standard METAFONT programming. We have shown several examples of universal ("bitmap-free") METAFONT routines and a few possible paths for future development."

KEES VAN DER LAAN, BLUe's Format — The best of both worlds; pp. 32–44

Author's abstract: "An independent format — `blue.fmt` — is proposed to assist authors with creating, formatting, exchanging and maintaining compuscripts during the lifephases of publications. The format builds upon `manmac.tex` and the functionalities provided by `tugboat.sty`. Experience gained by publishers has been picked up too, because of my in-depth study of the activities of AMS with respect to TEX formatting. More recent work of Knuth and co-authors has been borrowed from `gkpmac.tex`.

The design goal was to provide a format which suits me, which is easy to customize — to the world outside, and in general to changing circumstances — and which complies with the adages of software engineering. Another aim of `blue.fmt` is that it can be used throughout the life cycle of publications on modest equipment to format articles, transparencies and you name it. The hoped for [duration] is a lifetime. En passant the design process is accounted for.

New is the handling of a database of references — with cross-referencing — or pictures all in one-pass job."

KLAUS LAGALLY, Bidirectional line breaking with TEX macros; pp. 45–52

Author's abstract: "TEX was originally designed with European languages in mind, and thus, whenever a paragraph contains text portions running in opposite directions, e.g. when combining English and Arabic or Hebrew in the same document, the task of line-breaking becomes rather complicated.

For a clean solution, Knuth and MacKay have proposed a modification to TEX, `TeX-XeT`, which will produce an extended DVI file containing additional directional information to be exploited by a modified DVI driver; and by now there exist several implementations of this idea, including `TeX--XeT` that produces a standard DVI file. The main drawback is just that we have to go outside the TEX standard.

We present a portable technique to handle bidirectional line-breaking by using TEX macros alone, albeit at some sacrifice to quality. This technique has been implemented in version 3.02 of the author's multilingual ArabTEX package."

BERND RAICHLE, Sorting in TEX's Mouth; pp. 53–58

[no abstract available]

PETR SOJKA and PAVEL ŠEVEČEK, Hyphenation in TEX — Quo Vadis?; pp. 59–68

Author's abstract: "Significant progress has been made in the hyphenation ability of TEX since its first version in 1978. However, in practice, we still face problems in many languages such as Czech, German, Swedish, etc. when trying to adopt local typesetting industry standards.

In this paper we discuss problems of hyphenation in multilingual documents in general, we show how we've made Czech and Slovak hyphenation patterns and we describe our results achieved using

the program PATGEN for hyphenation pattern generation. We show that hyphenation of compound words may be partially solved even within the scope of TEX82. We discuss possible enhancements of the process of hyphenation pattern generation and describe features that might be reasonable to think about to be incorporated in Ω or another successor to TEX82."

PHILIP TAYLOR, Defensive programming in TEX: towards a better class of TEX macro; pp. 69–79

Author's abstract: "Defensive programming is a fundamental feature of any professional computer programmer's toolkit; yet the techniques, while widely understood in the worlds both of 'real' and of 'academic' programming, seem to have received less than their fair share of attention from within the world of TEX. It would be unfair to single out any one package as being deficient in this respect, and equally unfair to suggest that all package writers are tarred with the same brush — some, indeed, demonstrate a far better than average awareness of the technique required — but what is perhaps most disturbing is that non-defensive techniques are still not only being used but continue to be advocated. In this paper I seek to redress this imbalance, and to try to illustrate both the pitfalls into which the unwary could stumble, and the safeguards which could be usefully adopted by the wise."

Practice

VLADIMIR BATAGELJ, Combining TEX and PostScript; pp. 83–90

Author's abstract: "PostScript is becoming a de facto standard as a device independent page description language. By embedding PostScript elements in TEX we can extend the use of TEX to new areas of application.

In the first part of this paper we give some general information about PostScript and its features. In the rest of the paper we present some of our own experiences and solutions in combining TEX and PostScript:

- dictionaries, prolog files and how to save a lot of space with PostScript figures produced in *CorelDRAW*, *Mathematica*, ... ;
- writing TEX-PostScript macros, case: drawing graphs (combinatorics) in TEX; PostScript error handling mechanism, an application in function graph drawing macro."

LUTZ BIRKHAHN, Tdb: An X11 TEX Debugger; pp. 91–95

Author's abstract: "Writing TEX macros is an error-prone task, and finding those errors may still require solid (TEX) expert knowledge. Conventional programming languages offer a diversity of debugging tools that are specialized for finding coding errors. But for debugging TEX code, there are, even after 15 years of TEX programming world-wide, only the primitive debugging aids of TEX itself. This paper describes Tdb, a first approach to a TEX debugger, that claims to fill that gap.

Tdb consists of an extension to the TEX formatter, providing an interface to the famous Tool Command Language Tcl and its X11 Window System toolkit Tk. Based on this Tcl interface was built a debugger with a graphical user interface, enabling the user to do things such as setting breakpoints, single-stepping through the code or browsing in the actual macro definitions and variables. The flexible design based on Tcl and the MCV concept (Model, View, Controller) allows it to customize and extend the user interface as well as the debugging functions. Furthermore, Tk's interprocess communication facilities provide a solid basis for integrating Tdb into a complete TEX development environment."

WIETSE DOL and ERIK FRAMBACH, 4TEX: a workbench for MS-DOS PCs; pp. 96–100

Author's abstract: "TEX and all its companions offer an enormous number of possibilities. This is both an advantage and a disadvantage. The advantage is that almost anything is possible; the disadvantage is that you need detailed knowledge of all related programs to fully exploit the possibilities.

The workbench 4TEX is an attempt to integrate all major TEX-related programs in a shell that shields you from the tedious and frustrating job of setting environment variables and program parameters. 4TEX runs under MS-DOS, OS/2, and MS-Windows.

4TEX includes the following tools (amongst others): compilers, previewers, a spell-checker, BIBTEX, MAKEINDEX, TEXCAD, QFIG, graphics convertors such as HP2xx, BM2FONT and GHOSTSCRIPT, text convertors such as WP2LATEX and TROFF2TEX. Note that all programs used by 4TEX are either *freeware* or *shareware*.

Naturally, there is on-line help, and all functions are available through simple menu choices."

MICHEL GOOSSENS and FRANK MITTELBACH, Real life book production — lessons learned from *The LaTeX Companion*; pp. 101–104

Author's abstract: "Some aspects of the production of *The LaTeX Companion* are described."

[See also *TUGboat* **15** no. 3 (1994), 170–173.]

KAREL HORÁK, Fighting with big METAFONT pictures when printing them reversely or [in] landscape; pp. 105–107

[no abstract available]

JÖRG KNAPPEN, Towards a 256-character IPA font; pp. 108–109

[no abstract available]

OLGA LAPKO, MAKEFONT as part of *CyrTUG*-emTeX package; pp. 110–114

[no abstract available]

MARION NEUBAUER, Conversion from WORD/WordPerfect to LaTeX; pp. 115–119

Author's abstract: "Production of a large document with many contributors requires conversion of all submitted manuscripts into the LaTeX format. A large proportion of manuscripts is submitted in the formats of WORD and WordPerfect, two very popular word processing programs. I will discuss different approaches to converting such files to LaTeX format.

First of all the differences between the word processors WORD and WordPerfect versus the document preparation system LaTeX will be explained, and problems encountered during text conversion into LaTeX will be discussed. The conversion can be done either by means of a separate program (external conversion) or using macros, style sheets and a printer driver from within the word processors (internal conversion). Advantages and disadvantages of both methods for different types of text elements such as plain text, lists, tables and mathematical formulas will be discussed. This is followed by an overview of the conversion programs currently available."

ÉRIC PICHERAL, Building and supporting the GUTenberg archive; pp. 120–124

Author's abstract: "Three years ago, the GUTenberg Local TeX Users Group established an archive, accessible via *ftp*. Its main purpose is to supply ready-to-use TeX versions convenient for French-speaking people."

Progress

JOHANNES BRAAMS, Document Classes and Packages for LaTeX2ε; pp. 127–134

Author's abstract: "In the first section of this article I describe what document classes and packages are and how they relate to LaTeX 2.09's style files. Then the process of upgrading existing style files for use with LaTeX2ε is described. Finally I give an overview of the standard packages and document classes that are part of the distribution of LaTeX2ε."

[See also *TUGboat* **15** no. 3 (1994), 255–262.]

YANNIS HARALAMBOUS, Typesetting the Holy Bible in Hebrew, with TeX; pp. 135–152

Author's abstract: "This paper presents *Tiqwah*, a typesetting system for Biblical Hebrew, that uses the combined efforts of TeX, METAFONT and GNU Flex. The author describes its use and its features, discusses issues relevant to the design of fonts and placement of floating diacritics, and gives a list of rare cases and typographical curiosa which can be found in the Bible. The paper concludes with an example of Hebrew Biblical text (the beginning of the book of Genesis) typeset by *Tiqwah*."

[See also *TUGboat* **15** no. 3 (1994), 174–191.]

YANNIS HARALAMBOUS and JOHN PLAICE, Ω, a TeX Extension Including Unicode and Featuring Lex-like Filtering; pp. 153–166

Author's abstract: "Ω consists of a series of extensions to TeX that improve its multilingual capabilities. It allows multiple input and output character sets, and will allow any number of internal codings. Finite state automata can be defined, using a `flex`-like syntax, to pass from one coding to another.

In this paper both a technical introduction and a few applications of the current implementation of Ω are given. The applications concern typesetting problems that cannot be solved by TeX (consequently, by no other typesetting system known to the authors). They cover a wide range, going from calligraphic script fonts (Adobe Poetica), to plain Dutch/Portuguese/Turkish typesetting, to vowelized Arabic, fully diacriticized scholarly Greek, or decently kerned Khmer.

A few problems Ω *cannot* solve are mentioned, as challenges for future Ω versions."

[See also *TUGboat* **15** no. 3 (1994), 344–352.]

DAG F. LANGMYHR, How to make your own document styles in LaTeX2ε; pp. 167–175

[no abstract available]

FRIEDHELM SOWA, Printing colour pictures; pp. 176–181

Author's abstract: "The availability of cheap colour printers has increased the demand for colour support in TEX for text and graphics. This paper shows what components are necessary and available to satisfy this demand. Further, it points out the problems that have to be solved to make the TEX Colour Interface as device independent as possible.

A colour package for printing coloured text was developed by Jim Hafner and Tom Rokicki by defining a set of commands which use the `special` primitive. This was the base for the colour interface in the new LATEX. It represents the first and — up to now — only method to print coloured text.

Printing colour pictures in a TEX document needs a driver program that is able to exploit the capabilities of a colour device. The driver must separate the colours of the picture into the basic colours used by the colour model supported by the output device. This was the purpose [for] develop[ing] the `dvidjc`-drivers for the Hewlett Packard inkjet printers and to upgrade BM2FONT to version 3.0.

The upcoming problems during the development of this dot matrix driver and the integration of colour screens (separated by BM2FONT) showed that text and graphics have to be treated differently. A possible description of a TEX Graphics Interface is proposed."

[See also *TUGboat* **15** no. 3 (1994), 223–227.]

PHILIP TAYLOR, ε-TEX and $\mathcal{NTS}$: a status report; pp. 182–187

Author's abstract: "The $\mathcal{NTS}$ project was created under the aegis of DANTE during a meeting held at Hamburg in 1992; its brief was to investigate the possibility of perpetuating all that is best in TEX whilst being free from the constraints which TEX's author, Prof. Knuth, has placed on its evolution. The group is now investigating both conservative and radical evolutionary paths for TEX-derived systems, these being respectively ε-TEX (extended TEX) and $\mathcal{NTS}$ (a New Typesetting System). The group is also concerned that whilst TEX itself is completely stable and uniform across all platforms, the adjuncts which accompany it vary from implementation to implementation and from site to site, and has therefore proposed that a 'canonical TEX kit' be specified which, once adopted, could safely be assumed to form a part of *every* TEX installation. Work is now well advanced on the ε-TEX project, whilst the group are concurrently involved in identifying the key components of a complete portable TEX system and in investigating sources of funding which will allow the $\mathcal{NTS}$ project to become a reality."

[See also *TUGboat* **15** no. 3 (1994), 353–358.]

JIŘÍ ZLATUŠKA, Surviving in a multilingual world with multiple font encodings; pp. 188–195

Author's abstract: "This paper develops constructions needed for utilizing the encoding scheme concept embedded into LATEX2ε in order to develop a system allowing one to use different font encoding schemes and different languages within one format and to provide mechanisms for fully functional switching between them. The concept of language extends that of TEX based just on proper set of hyphenation patterns. A practical demonstration of this is shown in the example defining hyphen-splitting feature as a modification to the standard line-breaking behaviour of TEX."

MICHEL GOOSSENS and SEBASTIAN RAHTZ, Simple colour design, and colour in LATEX2ε; pp. 196–205

Author's abstract: "This article reviews some basic principles underlying the use of colour. We start by a review of the functional use of colour, explaining how it can help to focus attention, explain relationships, guide the reader/viewer through the presented information so that its contents are easier to absorb and appreciate. Some common rules for optimizing communication using colour elements in documents are discussed. We then explain the colour support in LATEX2ε and give some examples."

[See also *TUGboat* **15** no. 3 (1994), 218–222.]

Editor's note: For more information on EuroTEX'94, held September 26–30 in Gdańsk, Poland, there is a summary in *TTN* 3,4:17–18. Copies of the 200-page Proceedings of the EuroTEX94 conference can be obtained by sending 15 DM (postage included) to Włodek Bzyl, Instytut Matematyki, Uniwersytet Gdański, Wita Stwosza 57, PL 80-952, Poland.

(Compiled by Christina Thiele)

Calendar

1995

Jan 5–8 Linguistic Society of America, 69th Annual Meeting, Fairmont Hotel, New Orleans. For information, contact the LSA office, Washington, DC (202-834-1714, zzlsa@gallua.gallaudet.edu).

Jan 12 DANTE TEX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Oct 6.)

Jan 19 Journée d'information sur la Diffusion des Documents Electroniques de LATEX à HTML, WWW, et Acrobat, Nanterre, France. For information, e-mail tresorerie.gutenberg@ens.fr.

Jan 19 Portable Documents: Acrobat, SGML & TEX, Joint meeting of the UK TEX Users' Group and BCS Electronic Publishing Specialist Group, The Bridewell Theatre, London, UK. For information, contact Malcolm Clark (m.clark@warwick.ac.uk).

TUG Courses, Santa Barbara, California

Jan 30 – Feb 3 Intensive LATEX2ε

Feb 6–10 Beginning/Intermediate TEX

Feb 13–17 Advanced TEX and Macro Writing

Feb 28 – Mar 3 TEX-Tagung DANTE '95, University of Gießen, Germany. For information, contact Günter Partosch ((0641) 702 2170, dante95@hrz.uni-giessen.de).

Apr UK TEX Users' Group, location to be announced. Topic: Maths is what TEX does best of all. For information, e-mail uktug-enquiries@ftp.tex.ac.uk

Apr 29 – May 1 BachoTEX '95, Poland. For information, contact Hanna Kołodzeijska, (gust@camk.edu.pl).

Jun 1–2 GUTenberg '95, "Graphique, TEX et PostScript", La Grande Motte, France. For information, call (33-1) 30-87-06-25, or e-mail treasorerie.gutenberg@ens.fr or aro@lirmm.fr.

Jun 1–2 IWHD '95: International Workshop on Hypermedia Design, Montpellier, France. For information, contact the conference secretariat, Corine Zicler, LIRMM, Montpellier ((33) 6741 8503, zicler@lirmm.fr).

Jul 24–28 **TUG 16th Annual Meeting**: Real World TEX, St. Petersburg Beach, Florida. For information, send e-mail to tug95c@scri.fsu.edu. (For a preliminary announcement, see *TUGboat* 15, no. 2, p. 160.)

Sep 4–8 EuroTEX '95, Papendal, Arnhem, Netherlands. For information, contact eurotex@cs.ruu.nl.

For additional information on the events listed above, contact the TUG office (415-982-8449, fax: 415-982-8559, e-mail: tug@tug.org) unless otherwise noted.

Status as of 31 December 1994

Production Notes

Barbara Beeton

A new approach to *TUGboat* production

Owing to various circumstances beyond the Editor's control, time available for *TUGboat* production has diminished to the point where it is no longer possible for the regular issues of *TUGboat* to remain a one-person operation.

As is quite obvious, this issue is embarrassingly late. But rather than trying to explain why it is late, I would like to describe what has been done to try to avoid such delays in the future.

Mimi Burbank and the system management at SCRI — the Supercomputer Computations Research Institute at Florida State University — have kindly made available copious disk space, login access, and a group identity for a core team of volunteers: Mimi, Robin Fairbairns, Michel Goossens, Sebastian Rahtz, Christina Thiele, and myself. Every member of this team has previous experience in editing or producing *TUGboat*, proceedings issues, or similar TEX publications, so they have been able to "hit the ground running".

In the space allotted, we have set up a full, isolated (LA)TEX system and *TUGboat* work areas. Remaining in a management position, I have populated the tree with the material collected for issues 15(4), 16(1), et seq., identified which ones are encoded using `plain` or LATEX conventions, and encouraged the team members to work first on items that match their interests and expertise. Articles are returned to me as PostScript files to be printed and given a final reading. I have edited the input files directly, where practical, and provided comments by e-mail to the "handler" regarding adjustments in format. The final version is again delivered in PostScript form for printing and inclusion in a growing pile of printer-ready copy. No item has been slighted, with the result that 16(1) is nearly ready to put together, and should be sent to the printer — and thence to members — in no more than a month from 15(4). As I will be out of town for much of this interval, Mimi Burbank has agreed to be the manager for 16(1).

The plan for issue 16(2) is a bit different. For some time, the Publications Committee has been discussing the idea of theme issues — issues devoted to a single topic of narrower or wider scope — under the direction of a guest editor. 16(2) will be the first of such issues, containing articles related to electronic documents, in particular SGML, HTML, hypertext, Acrobat, ..., with Malcolm Clark in charge. Topics for future theme issues will be announced as plans become more firm; one theme issue per year is currently foreseen. Suggestions are welcome for both topics and prospective guest editors.

Input and input processing

Electronic input for articles in this issue was received by e-mail, and was also retrieved from remote sites by anonymous ftp.

In addition to text and various files processable directly by TEX, the input to this issue includes METAFONT source code and many encapsulated PostScript files. More than 200 files were required to generate the final copy; over 100 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; `.dvi` files, device-specific translations, and fonts (`.tfm` files and rasters) are excluded from the total.

Most articles as received were fully tagged for *TUGboat*, using either the usual `plain`-based or LATEX conventions.

By number, 47% of the articles, and 63% of the pages in this issue are in LATEX. (For ease of production, three mostly-text items which were originally prepared using LATEX were converted to `plain`, and one, from `plain` to LATEX.) LATEX2$_\varepsilon$ was the version used, thanks to some major systems work by Robin Fairbairns and Sebastian Rahtz.

Font work was required for the *Indica* article by Haralambous, for MacKay's recycle logo, and for the Chinese fragment in the EuroTEX'94 report.

Articles were processed individually by members of the team according to their own preferred methods, and the final input and output (PostScript) files delivered to the Editor for compilation into an issue. The Editor created the table of contents, the cover and front matter, printed out all the files, checked the copy and conveyed it to the printer.

Output

The bulk of this issue was prepared at SCRI on an IBM RS6000 running AIX, using the *Web2C* implementation of TEX. The remainder was run at the American Mathematical Society from files installed on a VAX 6320 (VMS) and TEX'ed on a server running under UNIX on a Solbourne workstation. Output was printed at AMS at 600 dpi on an HP LaserJet 4M plus; this was used rather than a typesetter for reasons of both cost and speed.

Institutional Members

The Aerospace Corporation,
El Segundo, California

Air Force Institute of Technology,
Wright-Patterson AFB, Ohio

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

Brookhaven National Laboratory,
Upton, New York

Centre Inter-Régional de
Calcul Électronique, CNRS,
Orsay, France

CERN, *Geneva, Switzerland*

College Militaire Royal de Saint
Jean, *St. Jean, Quebec, Canada*

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

Communications
Security Establishment,
Department of National Defence,
Ottawa, Ontario, Canada

CSTUG, *Praha, Czech Republic*

Elsevier Science Publishers B.V.,
Amsterdam, The Netherlands

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Florida State University,
Supercomputer Computations
Research, *Tallahassee, Florida*

Grinnell College,
Noyce Computer Center,
Grinnell, Iowa

Hong Kong University of
Science and Technology,
Department of Computer Science,
Hong Kong

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Iowa State University,
Ames, Iowa

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wisconsin

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Germany

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Basic Research Laboratories,
Tokyo, Japan

Personal TEX, Incorporated,
Mill Valley, California

Princeton University,
Princeton, New Jersey

Smithsonian Astrophysical
Observatory, *Cambridge,*
Massachusetts

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
Heidelberg, Germany

Stanford Linear Accelerator
Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

Texas A & M University,
Department of Computer Science,
College Station, Texas

United States Naval
Postgraduate School,
Monterey, California

United States Naval Observatory,
Washington DC

University of California, Berkeley,
Center for EUV Astrophysics,
Berkeley, California

University of California, Irvine,
Information & Computer Science,
Irvine, California

University of Canterbury,
Christchurch, New Zealand

University College,
Cork, Ireland

University of Delaware,
Newark, Delaware

University of Groningen,
Groningen, The Netherlands

Universität Koblenz–Landau,
Koblenz, Germany

University of Manitoba,
Winnipeg, Manitoba

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

University of Southern California,
Information Sciences Institute,
Marina del Rey, California

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

Università degli Studi di Trieste,
Trieste, Italy

Uppsala University,
Uppsala, Sweden

Vrije Universiteit,
Amsterdam, The Netherlands

Wolters Kluwer,
Dordrecht, The Netherlands

Yale University,
Department of Computer Science,
New Haven, Connecticut