Vistas for TEX: liberate the typography! (Part I)

Chris Rowley
Semantic Maths for All! Project
Department of Mathematics and Statistics
Faculty of Mathematics, Computing and Technology
Open University, London, UK
c.a.rowley (at) open dot ac dot uk

Abstract

This is a polemic in favour of liberating the core typesetting structures and algorithms around which TEX is built from the monolithic superstructure of the mini-dinosaur of a program called tex and its more or less modernised and approachable derivatives such as xetex and luatex.

Although the high-level aims of the programme of activity advocated here have a lot in common with those of the very exciting and active LuaTEX project, the route I propose seems to me to be very different. The major ambition of the latter project is to embed (something similar to) the whole of the current TEX system within a vastly more complex monolith [*sic*] of an application which will presumably be well-adapted to the formatting needs of oriental languages. To this monolith are now being added many well-oriented intrusions [*sic*] into but a single instance of that ancient bedrock of TEX!

Of course, luatex promises to provide a spectacularly sophisticated and highly hackable system that will eventually enable a great evolutionary radiation of species within the phylum of automated document processing; hence the importance and fascination, for me at least, of the developmental path of the LuaTEX project.

Pursuing the paleontological metaphor well beyond its point of total and painful collapse, my plan can be thought of as providing many tools that can be easily dispersed in such a way that TEX's clever genes can influence (for the good) far more aspects of the evolution of automated typesetting: all this abundance being more speedily and robustly achieved due to not being held back by the decision to build all future systems on a perfectly preserved and complete digestive system from a fossilised ancestral TEXosaur.

I am here also making a plea to the Grand Technical Wizards of TUG to widen support from their development fund's treasure chest to encompass projects that are designed to spread TEX's influence and presence throughout the fertile modern world of document processing via its algorithms alone, without the dead weight of its monolithic, programmatic paradigm and the many somewhat dated aspects of its detailed software design.

Adding topicality and an even longer time-base to the metaphors, please can we have plentiful levels of international funding to support an actual Big Bang to get the elementary particles of TEX spread throughout the typesetting universe, rather than funding only an engineering wonder (for interesting but small-scale experiments to find new TEX-like particles) such as the LHC: *LuaTEX's Hard-problems Cruncher*, a 'shining star in the East' which I fear may spin-off some big black-holes to trap even the most energetic of us mortal, western programmers.

1 Introduction

This could easily have been the shortest genuine paper in this, or any, TEX Users Group proceedings. All it needs to say is: please support Free, Open and Reusable Algorithms from TEX (yes folks, the FORAT Campaign starts here!).

However, I will attempt, in this and subsequent papers, to expand on some examples of what I mean by liberating TEX's formatting algorithms in the form, for example, of C++ libraries, embeddable JavaScript or similar reusable artefacts.

In Section 3 I shall also provide more or less deep discussions, through examples, of the most important and most difficult part of doing this in a practical and useful way: the provision of good external interfaces. But we shall begin in Section 2, with some introductory remarks (not all strictly pertinent) concerning current TEX's use and misuse of its formatting subsystems. The paper concludes with a note on non-TEX math formatters, being an introduction to further study in this area.

The whole is permeated by a sincere plea for greatly increased support, of all types and by the many TUGs and all individuals in the TEX com-

munity, for this outward-looking work so crucial to ensuring a very long life for both the great gifts that arrived within TEX, and also for the inventive and experimental spirit of its pioneers. May this paper be the first of many that promote the creation of Really Useful Software from TEX's Inner Engineering (the RUSTIE project).

2 The structure of TEX

The sum of the 'document manipulation' parts of TEX can be well described as a text processing application whose originality and utility is provided by many specialised formatting engines. Many of these engines are barely recognisable as independent software artefacts due to the programming techniques wisely chosen by Knuth to implement the system [6]. (His methodology can these days be well described as the EO method: Extreme Optimisation, of both space and time!)

Each of these deeply embedded formatting engines will, when it has been disentangled from the sticky mess of connections and optimisations that holds it deep within TEX's embrace, reveal itself as a thing of beauty. Less appealing, each such newly disentangled creature, whilst happy to be breathing free, will also be a very vulnerable beast as it will have no communication interfaces through which to nurture it. Thus its liberators will need to carefully craft protective interfaces in order to ensure the fledgling formatter's viability in the real and exciting, but non-programmatic, world of 21st century documents.

These hidden gems include formatters for words, LR-boxes (with natural and many other width specifications), split-able [*sic*] LR-boxes, leaders of various types, paragraphs, split-able [*sic*] paragraphs, justified lines, formulas, superscripts, fractions, radicals with bar, etc.

One aspect of these specialised formatters clearly shows TEX's ancient pedigree, from a time when data flow had kept to a minimum. This is their lack of flexibility, in the following sense. With the exception of using the concept of 'unset glue', TEX's formatters will always combine to produce, from a given input, a unique, fully specified boxful of formatted output (often with a claim of its 'optimal' quality).

As Frank Mittelbach and I have copiously argued over the years, for a sophisticated document formatter a more useful product would be a reasonably sized collection of possible formattings that are all 'good enough'. To this collection could possibly be added some ranking of their absolute quality but even more useful would be a few descriptors or quantisations of how good each is, together with other information that may be of use to other co-operating formatters. Such output could then be used by other 'higher-level' formatters to choose the formatting most suited to their higher purposes.

I shall not pursue such valuable enhancements here as I am only asking for the Moon, not Mars, ... this year; the benefits of this approach to the practical optimisation of formatted documents have been long known and much discussed since Frank Mittelbach and I [18] introduced them.

3 Two (of many) examples

The two of TEX's many embedded formatters about which I have chosen to say a little more here are not necessarily either the most complex or the easiest to specify, but they are both central to TEX's raison d'être. I have somewhat presumptuously chosen to put these into the form of outline draft specifications. I hope that they will start a process that leads smoothly and quickly to full specifications and to the production of partial but usable 'proof-of-concept' library implementations. For the maths formatter some more detailed work has been done (Section 3.3) that will soon form the contents of a funding bid for the project.

3.1 The paragraph formatter

This is just one possible route towards the exposure of a TEX-like paragraph mechanism and it is treated very briefly here. I hope it inspires others to help expand this to a full and carefully explained specification, to be published in this series.

Note that this outline description assumes the existence of methods for formatting 'LR-boxes': these are 'single lines' of text with well-defined spacing of 'words'. In turn that formatter will require a 'word formatter', etc. I plan to provide a fuller explanation of these ideas in a future paper.

The inputs to this formatter would be as follows.

- The material to be formatted (see below).
- Parameter settings. Although it may not be essential, it would be very useful to have the ability to input values for all the parameters that are used by TEX's algorithm for forming and formatting a paragraph. Of course, many of them will have sensible default values that could be fixed or, in context, inherited.

The material to be formatted would consist of the following.

- Pure text (e.g. Unicode strings).
- Some 'formatting information' such as:
  - font selection hints
  - line-breaking hints
  - word-division information

– even direct formatting instructions (!)

all with a precisely defined syntax.

– Pre-formatted material in the form of 'in-line boxes'.

The basic liberated version of this formatter would return two types of material (this is slightly more than current TEX's internal mechanism can be bothered to do, despite the information being available). Applications using it are free to ignore item 2.

1. A formatting of the paragraph: the formatted 'lines' of the paragraph plus other information about their layout in a well-defined format.
2. The number of lines, information about the break-points used (e.g. their location in the input string, whether a word was broken), an evaluation of how 'good' each break-point is and of the quality of the formatting of each line.

Note that here I have not explicitly discussed any of TEX's sophisticated escape mechanisms that implement a small range of specialised extra activities, such as 'vadjust material', marks or 'whatsit nodes'. These are good examples of Knuth's cleverly ad hoc use of small and tightly integrated extensions to a basic algorithm in order to emulate the effect of fully modelling these varied aspects of the document formatting process as separate modules. Such features of TEX thus express a limited collection of ideas from a wider class that is important to automated document processing. Being ad hoc, Knuth's efficient implementations typically use inappropriate models and hence have severe deficiencies. It is therefore probably not sensible to reproduce such escape mechanisms when providing our exposures of The Good Things of TEX™.

A more advanced version of this liberated formatter would allow the replacement of TEX's algorithm for finding break-points but this would entail provision for whole new parametrisations of the process and thus could decrease its immediate usability. A more practical way of providing this formatter may be to split it up into smaller modules that undertake distinct parts of the task, leaving the **master paragraph formatter** with only the two tasks of controlling the whole process and of handling all external interfaces (which would need to be customisable).

3.2 The mathematics formatter

This formatter has turned out to be the central feature in this final form of my diatribe; thus I shall warn you that it gets tediously detailed from here on. Although many of you would like to follow the example of Sebastian Rahtz [3] and banish all mathematics from the TEX world (if not all worlds), we must nevertheless face the reality that **TEX Does Math!!** — both within the TEX world but also, far more and growing rapidly, outside it, including much that will never see the guts of a TEX processor. Hence if, over the next 10 years or so, real world 'TEX for maths' is not to lose all contact with 'TEX the processor', then TEX's maths formatter must get out there and strut its stuff wherever maths is being stuffed into digital form.

For the liberated form of this important formatter it is more difficult to specify in suitably general terms the nature of the material to be formatted; the obvious specification is 'Presentation MathML containing Unicode strings' [16] but it is not clear to me at this stage whether this will always be sufficiently rich in information about the mathematical structure of the notation to be typeset. However, a lot of work has been done, and is continuing, on a wide range of uses of mathematical notation in computers, together with their associated description and formatting requirements. Exposing this formatter will greatly help many aspects of this research and development effort, in particular the task of determining what needs to be encoded in the input to ensure high quality maths output.

The output will be material that can be used by an 'LR-box' or 'paragraph' formatter.

The current understanding of the parametrisation needed for this task is also still somewhat empirical. It is known that standard TEX's algorithm is severely under-parametrised for the tasks in which it claims supremacy but, in contrast, many applications will require only a far simpler parametrisation.

It is therefore desirable that the implementation of this algorithm should build in sensible default rules for determining plausible values of all TEX's 'maths parameters' from data as meagre as just the nominal text font size. However, this liberation must at least remove all of current TEX's explicit overloading in the 'standard parameter set' in order to be more generally usable. It may also be sensible to remove some aspects of the parametrisation from the current dependence on the choice of fonts.

The following subsection contains further details of this project; it is an extract from a funding proposal currently being pursued for this task. Throughout, the phrase 'Standard LATEX' refers to a well known and defined (mainly ad hoc at present) subset of LATEX's math mode, but using the full range of Unicode maths characters, for encoding mathematical structures and glyphs. This is not a good phrase for this beast, thus it has also been dubbed 'LoTEX' [17] (as in the common multiple *Lcm*).

3.3 Liberating TEX: maths formatting

A fact and the obvious question: Many current applications, like the above specification, require a maths formatter: why not make it TEX's?

Until quite recently the standard TEX formatter for Computer Modern was the only such application that was both widely available and offered even reasonable typeset quality. However, bits of TEX were, back then, totally incompatible with other applications that used different font resource technologies or series. Thus there are now many applications that, with varying degrees of success, attempt to emulate the quality of TEX without its restrictions on input and output. There are now also some serious rivals to TEX's typeset quality for mathematics but they also share two, at least, of TEX's problems: being too far embedded in widely used and sophisticated document processing systems; and being closely linked to particular font series.

Whilst it may not be feasible to get the highest quality formatting using a given maths font series without careful choice of the set of layout parameters (and their values) specific to that series [7, 8], it is certainly possible to give the world TEX's current algorithm with some liberation of the parametrisation. For example, it is straightforward to remove from its layout parameter set-up the many explicit (and 30-year old) overloadings and relationships that were necessary to Knuth's giving us this wonderful gift, so expertly tuned to a particular font series (Monotype Modern), itself about 100 years old![15]

More importantly, this quality can now be made available in a portable form so that it could be easily linked into any application, such as browsers or office suites, that need to render structured representations of mathematics, particularly MathML 3.0 and 'Standard LATEX' or LoTEX. Post-liberation, this valuable treasure can be further enhanced by making it more configurable so as to allow extensions of its capabilities in the following two areas: the diversity of 1.5-dimensional (or maybe fully 2-dimensional) text-based layouts that it can construct; the range of fonts, glyphs, colour resources and other printing marks that can be used in these constructs (and maybe built-in interactivity).

For many modern applications it will be essential to provide (at least in a derivative version) optimisation for fast parallel rendering of a large collection of (typically small) maths fragments, possibly with different 'quality control requirements'.

A typical such application (from the trendy worlds of Web 2.0 and/or the Semantic Web) of the near-medium future will be fast, highly interactive, agent-supported browsing of large collections of pages that contain a lot of connectivity and mathematical intelligence embedded in semantically rich encodings of mathematical notation [19].

3.3.1 Inputs

- Maths material encoded in a fully specified language that describes, at least, the presentational structure (or, visual semantics) of the material. This currently would typically be either a precisely defined (and large) subset of P-MML (Presentation MathML 3.0 [16]) or a syntactically precise subset of the currently used range of LATEX-related math-mode syntax (LoTEX).[1]

  In order to support a more semantically oriented and user-friendly form of LATEX input, it may be wise to provide a preprocessor that accepts precisely restricted uses of `\newcommand`.

- A modern 'maths font' resource (similar to OpenType with the necessary 'math tables' as supplied with Microsoft's Cambria Math font).

3.3.2 Outputs

This is not so easy to standardise. The high-level specification is that it will consist of 'glyphs plus rules' (plus, possibly, other simple graphical components and colour features) that are absolutely positioned in a local coordinate system relative to a reference or 'base point' that in turn can be used to position and orient the output on a page. It will also contain the necessary pointers to glyph rendering and paint resources, etc.

Since there is no widely accepted standard language for precisely such output, it will be necessary to use a simple fixed internal representation akin to TEX's h/vlist or DVI languages. However, unlike the current TEX paradigm, modules to convert this internal language to a range of commonly needed languages will also be included. Examples of outputs are thus some type of PDF and SVG fragments and other application-specific formats such as RTF or the internal renderable format of a web browser.

4 Other mathematics formatters

Time has mitigated against extending this section beyond these very brief comments on some important existing non-TEX maths formatters. So there is another paper or two waiting in the wings.

For a long time there have been non-TEX maths formatters in general use, such as techexplorer [20]. There are now a large number of these; here is a partial list of those that are definitely 'fit-to-purpose',

[1] There is also a far wider need for LoTEX, together with standard translations to/from P-MML.

be that quick, simple and clear rendering of simple maths in browsers, or high quality use of well-designed fonts for more classical paper presentation:

jsmath, Mathplayer for Internet Explorer,
Gecko-Math (used in Firefox et al.),
Microsoft's Rich Edit (in Office 2007),
MathEX (incorporates techexplorer), SWiM.

How much the design of each of these systems uses, or is influenced by, Knuth's algorithm and layout rules ([5], Appendix G) has not I think been much studied; maybe the individual authors of the systems once knew the answers?

Only Microsoft's system [23, 2, 22] (as demonstrated in my talk at San Diego) claims typographical quality that is better than that of TEX; it is also the only one that is not intimately connected with MathML [16] although the two are reasonably friendly. The creators of this RichEdit system state that it uses 'TEX's mathematical typography principles' but they go on to remark that this task was nevertheless 'considerably harder than any of us imagined it would be', taking 15 years of elapsed time to complete (fortunately, throughout this time the then boss of the whole company took a keen personal interest in the whole project). They conclude that 'mathematical typography is very intricate and varied'.

Although the associated product does not seem to be widely used, the GtkMathView project [11] has worked on and documented a lot of interesting ideas and artefacts in this area.

References

[1] Hermann Zapf. About micro-typography and the *hz*-program. *Electronic Publishing* 6(3), pages 283–288, Wiley, 1993. http://cajun.cs.nott.ac.uk/compsci/epo/papers/epoddaui.html

[2] About Rich Edit Controls. http://msdn.microsoft.com/en-us/library/bb787873(VS.85).aspx

[3] Sebastian P. Q. Rahtz. Banish Maths! Daily personal communications, 1990–2005.

[4] D. E. Knuth and M. F. Plass. Breaking paragraphs into lines. *Software – Practice and Experience*, 11(11), pages 1119–1184, 1981.

[5] D. E. Knuth. *TEX: The Program.* Addison-Wesley, 1986, 1993.

[6] D. E. Knuth. *The TEXbook.* Addison-Wesley, 1986, 1993.

[7] Richard Southall. Designing a new typeface with METAFONT, in *TEX for Scientific Documentation*, Springer 1986, pages 161–179. http://www.springerlink.com/content/57432v516731367n/

[8] Peter Karow and Herman Zapf. Digital typography. Private communication, 2003.

[9] John Plaice, Yannis Haralambous and Chris Rowley. An extensible approach to high-quality multilingual typesetting. In *RIDE-MLIM* 2003, IEEE Computer Society Press, 2003.

[10] Extensible Markup Language (XML). http://www.w3c.org/XML

[11] GtkMathView Home Page. http://helm.cs.unibo.it/mml-widget

[12] XETEX. http://scripts.sil.org/xetex

[13] LATEX: A document preparation system. http://www.latex-project.org

[14] LuaTEX. http://www.luatex.org

[15] Monotype Modern: Description. http://www.paratype.com/fstore/default.asp?fcode=871&search=Monotype+Modern

[16] Mathematical Markup Language (MathML) Version 3.0, W3C Working Draft. http://www.w3.org/TR/MathML3

[17] Chris Rowley and Stephen Watt. The need for a 'Standard LATEX'. Private communication, July 2007.

[18] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In *Electronic Publishing*, pages 261–273, Cambridge University Press, 1992.

[19] Christoph Lange and Michael Kohlhase. SWiM: A Semantic Wiki for Mathematical Knowledge Management. Poster at http://kwarc.info/projects/swim/pubs/poster-semwiki06.pdf

[20] Don DeLand. From TEX to XML: The legacy of techexplorer and the future of math on the Web. Abstract in *TUGboat* 28:3, page 369, TUG, *Proceedings of the 2007 Annual Meeting.*

[21] Unicode. http://www.unicode.org

[22] Murray Sargent. Using RichEdit 6.0 for Math. http://blogs.msdn.com/murrays/archive/2007/10/28/using-richedit-6-0-for-math.aspx

[23] Ross Mills and John Hudson, Editors. Mathematical Typesetting: Typesetting Solutions from Microsoft. Glossy brochure distributed at TypeCon 2007, Seattle, August 1–5, 2007.