MFLua 0.8 — Prologue

Luigi Scarso

Abstract

Reflections on the roles of TEX, LuaTEX, METAFONT and MFLua, in the spirit of the theme of the TUG@BachoTEX 2017 conference.

1 Introduction

The opening talk of the TUG@BachoTEX 2017 meeting, given by Hans Hagen, was explicitly focused on the theme of the conference:

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

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

Predictions — what is the future of TEX, what we'd like to achieve and can we influence it.

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

2 Philosophy and history

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 Sometimes the mix returns a blurred image, sometimes a defined one that is already outdated by the course of events, but seldom does a guiding principle emerge from the past to the future; more often, the becoming is sensed as movement that nullifies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

We can assume as indisputable at least two "facts": 1) the spread of global communication and 2) the mathematisation of the society. The first had a big impulse about six hundred years ago, with Gutenberg, but now the time required to exchange a message is roughly several thousand times faster than 100 years ago, and sender and recipient can be almost anywhere — a *novel* situation for the human race. The sensation that the quality of the global communication is not high enough, the naturalness of our act of communicating and the presumption to know that the global communication is not so global after all, hide from us the re-evolutionary step forward we have made in the last 30 years: many common people in the world *can* communicate easily and cheaply. The distinction between synchronous and asynchronous communication, as also between human agent and artificial agent, is irrelevant: we exchange knowledge on a daily basis — and knowledge is a subject of philosophy.

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

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

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

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

3 TEX and LuaTEX

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

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

Different alphabets play against this need, slowing down the process of mathematisation, but, on the other hand, the same process *needs* new symbols when creating new descriptions of concepts (as with category theory) or reality. It was only by means of Knuth that this problem found an optimal solution. Being (for that time) a unique combination of mathematician, computer scientist (winner of a Turing Award) and typographer, the solution was a macro programming language (TEX) to write math as a mathematician would like to, a procedural language (METAFONT), clearly rooted in algebra and geometry, to create new fonts (hence alphabets), while a similar language (METAPOST) was developed a bit later for graphics. Finally a "device independent" (DVI) final format of the document, easily portable to other formats. Another procedural language, WEB (Pascal-based), was used to write the tex family of programs, and this raised another problem: a program is a mathematical proof and as such it must be written. Literate programming was Knuth's answer, of course using TEX.

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

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

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

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

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

As mentioned, apparently these were minor adjustments of Knuth's solution but, under the surface, they reveal a lack of being able to keep up with the environment. Right from the start, the global communication pushed TEX to interact with other languages, going outside the realm of math symbols and "standard" English (i.e. that used in scientific publications). The hyphenation capability of TEX was the answer but the Unicode standard and OpenType demand greater flexibility and, outside TEX, WEB and METAFONT, are simply ignored. Ω and $\mathcal{NTS}$ tried to create a new starting point, but perhaps they used an abstract-to-concrete approach which led to overwhelming complexity.

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

The potential of this orthogonal pair is still unfolding: at the beginning, Lua was used as an input adapter, then its dynamic loading feature was explored, with the SWIGLIB project, through work on how to extend the features of LuaTEX *at runtime*. Suddenly LuaTEX can become a graphics converter program, or a tool for number theory, or a PostScript interpreter, or load at runtime a different text shaper — and the latest release pushes the dynamic loading feature forward, avoiding the compilation of a separate wrapper module. Again, this was an evolution (or "extension") of the `\write18` macro — with the prominent difference that calling an external program is many times slower than calling a function (of course it must be compared with the time required to typeset the document). This opens a new perspective: TEX not only *to write about* math but *to do* math.

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

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

4 MFLua

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

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

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

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

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

It's now possible to have an SVG font, or ttx, and nothing prevents us from having FontForge output, or OTF directly: it's only a matter of having a clear specification. So, suddenly MFLua puts METAFONT back in the game of font design: it took only a few days to modify the ttx backend and make from sourcecode-regular a proof-of-concept variable font.

5 Today's challenges

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

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

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