Some misunderstood or unknown LaTeX 2_ε tricks (XI) and encoding issues (II)

Luca Merciadri

1 Introduction

Three subjects will be detailed:

1. Using LaTeX syntax as an unambiguous way to communicate,
2. Writing a bold `\ell`,
3. Fonts' encodings.

2 Using LaTeX syntax as an unambiguous way to communicate

A recurrent fact with everyday language is that it is subject to various interpretations, which sometimes lead to misconceptions and misunderstandings. This might not be a problem in special situations (some people use these facts in relationships, for example), but when one thinks about communicating on objective subjects, such as courses' material, it becomes detrimental.

When a student attends a course, he might understand, or not, what the lecturer says, for various reasons. He might also think that he has understood, when he has not really understood exposed concepts. Despite the fact that there might be many reasons to a (partial) misunderstanding of some given material, we here take an example: unprecisions due to (oral) language. More specifically, let us consider the case of a science lecturer whose course is based on mathematics.

Science courses require precision in their study, and thereby need to be well understood to be passed. Unfortunately, a lecturer has, as a matter of fact, no time to write all the mathematical expressions on the chart. A traditional way of explaining different things is the oral way: when something will not be written on the black board, it will be said. However, giving mathematical expressions orally can be difficult because of a lack of consistency in the oral expression of mathematical terms.

In some of these situations, it might thus be interesting to adopt a univocal language which can be fully understood provided people know its structure. This language can be LaTeX 'core' mathematical syntax.

For example, I would prefer hearing

Let us consider `\int a b` $f(x)\,\mathrm{d}x$ *[...]*

at the place of

Let us consider the integral from a to b of $f(x)$ *[...]*

There are more obvious cases where the oral expressions are longer when used with an 'oral syntax,' than when explained using a LaTeX syntax. For me, using a LaTeX syntax allows your message to be univocal, and shorter. This is pure benefit for lecturers and students (except that it might be considered as 'too mechanical' and 'charmless').

Another less discutable situation where it might be of interest, is when one needs to communicate by means of raw-text. Let us consider that you are a student and that you send a question (by e-mail) to a lecturer on some specific equations of some given course. You might send him to, say, 'eqs. (3.109) and (3.115)' but if the equations are not in his slides or book, or have changed since his (old) edition, you need to type them. You mainly have two choices:

1. If there are many equations you need to type, you generally make a `.tex` document, compile it, and send the resulting `.pdf` document,
2. If there are only some equations, one generally types them in the e-mail.

If you know the LaTeX syntax, why wouldn't you use it? At the place of writing equations in some pseudo-language (which might be subject to various interpretations, and thus misunderstandings) resembling to LaTeX, you can directly type them using LaTeX syntax.

Even if the receiver is not LaTeX-ly aware, he might use syntax-to-graphical interpreters to understand your somewhat creepy equations.

My advice would be to sacrify equations' reading for equations' graphical representation; that is, if your equations need `\left` and `\right` delimiters, and other more complex constructs, use them even if they decrease the human-readibility of the equations' code, because they will increase the human-readibility of the equation's graphical representation by a LaTeX engine.

There is evidently a compromise to do between unimportant (i.e. not useful in the context of the message) LaTeX syntax and core mathematical syntax, except if you are sure that your receiver interprets them using a compiler or a syntax-to-graphical interpreter.

The same method might be adapted with no difficulty when one needs to type some equations (s)he found after a long exercise. Once again, this helps having a shorter and univocal message. Moreover, you can directly embed your equations in forms, etc., after.

3 Writing a bold `\ell`

There are briefly (at least; [31]) two ways to write a bold ℓ:

1. Using `amsmath` and `amsbsy`, you can define
   ```
   \newcommand*\Bell{\ensuremath{\boldsymbol\ell}}
   ```

so that `\Bell` produces the desired effect,

2. Without any packages, you can use `\boldmath` and `\unboldmath`:

   ```
   \boldmath$\ell$\unboldmath
   ```

   (that you can define in a command `\Bell` too, if desired).

4 Fonts' encodings

4.1 Do not mix the different encodings

In my sixth article [21], I explained various aspects of the *input* encodings in LaTeX, for 'habitual' dialects. Apart from the input encoding, one can also take care of using the 'good' *font* encoding. Some distinctions need to be done before going any further. Briefly,

1. A LaTeX file is encoded in some encoding (for the file),
2. The file's content is interpreted thanks to the `inputenc`oding,
3. This content is translated into an output thanks to a font encoding.

4.2 Brief history

TeX (and LaTeX) traditionally used raster-graphic fonts produced by METAFONT for a specific device resolution. Until the arrival of Postscript, all applications used bitmapped fonts.

METAFONT was different because it used outlines to create the bitmaps and had parameters for optimising them. [7]

`dvips` originally produced PostScript files containing 300 or 600 dpi raster fonts, and so did the PDF files converted from that by e.g. `ps2pdf`. [17]

4.3 'Type' fonts

We first need to make a distinction between Type 3 fonts, that are bitmap (raster) fonts, and Type 1 ones, which are scaleable (vector). As a result, Type 3 fonts are considered resolution-dependent.

4.4 Bitmap vs outline

In principle, given adequate resolution, the screen preview quality of documents set in bitmap fonts, and set in outline fonts, should be comparable, since the outline fonts have to be rasterized dynamically anyway for use on a printer or a display screen. [1] However, things are not that simple.

First, there are some problems because PDF viewers (especially Adobe Acrobat Reader, etc.) sometimes still do a rather bad job when displaying device-dependent Type 3 raster fonts: [17]

1. Texts in raster fonts can be displayed slow on the screen and with no or suboptimal anti-alias filtering. PDF viewers generally also do a poor job of downsampling high-resolution bitmap fonts to low-resolution screen fonts, [1, 17]
2. The 'Type3' raster fonts (previously) inserted by `dvips` lack information about which character each glyph represents, which interferes badly with full-text search and copy & paste. [17]

Second [27],

1. The Type 3 fonts are generated at a specified resolution, which is generally the printer's one. That is, typically 300 or 600 dpi are used,
2. Changing the resolution of bitmap fonts is no easy task and therefore the PDF readers produce terrible results when displaying these fonts on the screen,
3. Printing quality can be quite deceiving if the printer's resolution does not match the one of the bitmap font.

As a result, Type 3 fonts might appear as 'bad.' They are however sometimes unavoidable: there are still many font files of useful or essential specialist symbols that are only available in METAFONT format. [7]

If you've got a recent TeX distribution (later than 2005), `dvips` should already use resolution-independent TeX (Type 1) fonts, by default. [17]

4.5 The OT1 encoding

Using OT1, Don. E. Knuth's original text encoding [5] as an input encoding makes inclusions of Type 1 fonts in the PDF files. It might appear nice at first sight, as we saw that Type 3 fonts were 'bad,' but OT1 has three main problems:

1. When accented characters are required, TeX creates them by combining a normal character with an accent because it is 7-bit and uses fonts that have 128 glyphs (and so do not include the accented characters as individual glyphs). As a result, while the resulting output looks perfect, this approach stops the automatic hyphenation from working inside words containing accented characters, [30, 38]
2. You cannot properly copy-and-paste such words from the output (DVI/PS/PDF). That is, copying/pasting a non-ASCII glyph is impossible without having the glyph split up into its components, [30]
3. Characters like the pipe sign, less than and greater sign give unexpected results in text. [30]

Computer Modern (CM) OT1 fonts are hardly used in raster form, even if they are frequently available in this form. [6]

4.6 Hinting

Preliminary definition: *Hinting* is, briefly, 'information which goes to the rasterizer along with the glyph outlines, and is used to make better decisions about which pixels to turn on.' [19]

4.7 The Cork encoding: T1

4.7.1 Description

As a result of OT1's problems, T1 was defined as a new encoding: the Cork encoding. [5, 16] T1 is used with fonts that use 8-bit encoding. 256 glyphs are possible. [29, 30]

This encoding has been realized in a series of fonts designed with METAFONT, in at least one font series that is available both in Adobe Type 1 format and in OpenType format. [5]

The two encodings (OT1 and T1) are e.g. described at [12, 13].

4.7.2 Together with a font package

The T1 encoding does not necessarily uses Type 1 fonts, because the standard Computer Modern fonts have no *free* 'European' Type 1 equivalent for the T1 font encoding. As a result, some font packages to 'use' conjointly with T1 are generally proposed [18]:

- `lmodern`, Latin Modern fonts: Type 1 converted from METAFONT sources of Computer Modern fonts, [15]
- `cm-super`, Computer Modern Super fonts: Type 1 converted from METAFONT sources of various Computer Modern font families. [37]

The CM-Super family is very large and therefore difficult to maintain. [15] Both [28]

1. are derived from Computer Modern,
2. have problems with hinting.

There are some key differences between both [28]. For example, `lmodern`

- has a handmade vectorization,
- has revised metrics,
- provides more glyphs, especially diacritical characters,
- 's development goes on,
- is available in the same optical sizes as CM,

Also, `cm-super`

- is a vectorization of CM bitmap fonts but mainly automatically done,
- is available in more optical sizes (this 'feature' of cm-super switched is off by using the `fix-cm` package (highly recommended)),
- is recommended to be used conjointly with the `fix-cm` package to fix a lot of broken design decisions in cm-super (and in addition this makes the final PDF a bit smaller),
- contains Cyrillic fonts and `lmodern` doesn't.

Once installed, these packages do not need to be `\usepackage`d, and directly take part into the game.

Another package is sometimes proposed: `ae`. But despite giving a good-looking PDF,

1. using non-ASCII characters will break Copy & Paste in the PDF output, [28]
2. these fonts do not contain the guillemets, which are necessary for French. As a result, the `aeguill` package has been created to help you. [26]

There is a big difference between `ae` and the two aforementioned packages: `ae` is a *virtual* font package. That is, it allows us to map "bits of DVI file" to single characters in the virtual font; so we can create an "é" character by recreating the DVI commands that would result from the code "`\'e`." However, since this involves two characters being selected from a font, the arrangement is sufficient to fool Acrobat Reader, and that results in the copy & paste problem cited before with `ae`. If you can live with this difficulty, virtual fonts are a useful and straightforward solution to the problem. [24, 36]

4.8 Type 1, Type 3, TrueType and OpenType

Both Type 1 and Type 3 fonts are becoming obsolete if you do not use pure LaTeX: OpenType is more or less the only modern font format that is widely used (there are also AAT and Graphite and possibly others). All input encoding, font encoding and rendering issues vanish once you use a modern engine (LuaTeX, XeTeX) and OpenType fonts. [5] Such engines effectively let you choose an OpenType font. [4, 5, 10]

Type 1 is the default for regular LaTeX. [5] That does not mean that it is impossible to use OpenType fonts in LaTeX: take a look at [14, 22, 23] for example.

OpenType supersedes TrueType, but installing TrueType fonts under LaTeX is possible too; see for example [9, 11, 20, 25, 34]. Using TrueType fonts is also possible under e.g. XeTeX. [32]

4.9 PDFTeX

PDFTeX can use OpenType or TrueType ([2, 3, 25]) fonts, but that requires TFM files (only XeTeX and

LuaTEX can load OpenType fonts directly without the help of TFM files).

XeTEX and LuaTEX can all read Type 1, TrueType and OpenType fonts. PDFTEX built-in deals with Type 1 fonts too. [6]

More generally, pdfTEX is built so that a source must be available for all fonts used in the document, except for the 14 base fonts supplied by Acrobat Reader (Times, Helvetica, Courier, Symbol and Dingbats).

It is possible to use METAFONT-generated fonts in pdfTEX, but it is strongly recommended not to use METAFONT fonts if an equivalent is available in Type 1 or TrueType format, if only because bitmap Type 3 fonts render very poorly in Acrobat Reader.

Given the free availability of Type 1 versions of all the Computer Modern fonts, and the ability to use standard PostScript fonts, most TEX users should be able to experiment with pdfTEX. [33]

4.10 Practical considerations

Nowadays, `dvips` grabs outline fonts automatically, and the PDF output at the end of the routine thus suffers from no 'zooming problem' as you would expect in significantly old distributions relying on `-P` command to `dvips` to grab outline fonts. [5]

With 'standard' languages and recent distributions, there should be no troubles when specifying no font encoding, and Type 1 fonts should be used directly and naturally at compilation. As a result, your documents would not suffer 'zooming problems,' but, as the default encoding is OT1 if you do not specify one, the three main problems given at Subsection 4.5 will arise. To solve this problem, it might be of interest to use a T1 font encoding:

```
\usepackage[T1]{fontenc}
```

We know that the problem of this approach is that it will make extensive use of Type 3 fonts, and thus create zooming problem, because fonts are replaced by 'European Computer Modern' Type 3 fonts.

But if either `lmodern` or `cm-super` is installed on your LATEX distribution, Type 1 fonts will take on. If none of these packages has been installed, you will see zooming problems, and that will be an indicator: you then need to install one, preferably `lmodern` for dealing with Computer Modern-like fonts.

Evidently, you might try another font package, such as `palatino`, which will result in the inclusion of Type 1 fonts, that is, what you were expecting. [16]

The important thing is thus to use a T1 encoding together with Type 1 fonts. Now, you can choose a font of your taste. Have a look at e.g. [35].

Same inclusion remarks generally apply for PDFTEX. (For an example, take [8].)

4.11 Converting `.ps` files that were produced with CM bitmap fonts

If you want to convert to PDF historic PostScript files that were produced with Computer Modern bitmap fonts, try the `pkfix` tool to replace these fonts in the PostScript with their Type 1 equivalents. [17]

4.12 Accents encoding

Many people stick with writing accented characters such as é using `\'e`. It is important to know when this needs to be done, and when it is useless. Consider that your file is encoded using a proper encoding, i.e. 'an encoding that covers the set of characters you wrote in it.'

Now consider the following examples, assuming `lmodern` is installed on your computer:

1. ```
   \documentclass{minimal}
   \usepackage[T1]{fontenc}
   \begin{document}
   é
   \end{document}
   ```
2. ```
   \documentclass{minimal}
   \usepackage[utf8]{inputenc}
   \usepackage[T1]{fontenc}
   \begin{document}
   é
   \end{document}
   ```
3. ```
   \documentclass{minimal}
   \usepackage[utf8]{inputenc}
   \begin{document}
   é
   \end{document}
   ```

Only item 1 will result in a problematic output for evident reasons. There will not be any problem in item 2, and item 3 will make the document use OT1 as a font encoding with Type 1 fonts (which is bad because of the OT1 font encoding, but placed here for demonstration).

However, please note that the `\'` and other schemes are also used to overcome files' encoding clashes through OSes.

4.13 Summary

I personally learnt a lot writing this. As a result, here is a quick summary of what you need to keep in mind after having read this section. These remarks apply for `latex`→`dvips`→`ps2pdf`, and for `pdflatex`.

1. Type 1 fonts are nice, and so are OpenType fonts, because they are 'vector fonts,'
2. OT1 encoding always use Type 1 fonts, but OT1 suffers from three main drawbacks (see 4.5),
   ```

3. T1 encoding should be preferred, but T1 automatically loads Type 3 fonts. As a result, you need to install font packages.

4.14 Internationalization

Note that the above text is only valid for LaTeX-'standard' languages: refer to specialized literature for arabic, cyrillic, slovak, etc., scripts.

◇ Luca Merciadri
University of Liège
Luca.Merciadri (at) student dot ulg dot ac dot be
http://www.student.montefiore.ulg.ac.be/~merciadri/

References

[1] Nelson H.F. Beebe. Outline and bitmap fonts compared , 2006. http://www.math.utah.edu/~beebe/fonts/outline-vs-bitmap-fonts.html.

[2] Keith Briggs. Using truetype fonts with pdfLaTeX. http://keithbriggs.info/ttf-pdflatex.html.

[3] Otfried Cheong. Using Truetype fonts and Unicode in PDFLaTeX. http://tclab.kaist.ac.kr/ipe/pdftex.html.

[4] Existentialtype. Fonts in LaTeX, Part One: XeLaTeX, 2008. http://existentialtype.net/2008/07/12/fonts-in-latex-part-two-pdftex-and-opentype/.

[5] Robin Fairbairns, Frank Mittelbach, and Werner Lemberg. LaTeX font encodings, 2006.

[6] Robin Fairbairns, Philipp Stephani, and Luca Merciadri. Raster fonts questions, 2011. http://groups.google.com/group/comp.text.tex/browse_thread/thread/3ef80548d126759a#.

[7] Peter Flynn. (Personal communication.), 2011.

[8] Archlinux Forums. PDFLaTeX does not use Type 1 fonts. https://bbs.archlinux.org/viewtopic.php?id=26485.

[9] Gordon Grubert. True Type Fonts with LaTeX – Usage for LaTeX under Linux and MiKTeX2.5. http://fachschaft.physik.uni-greifswald.de/~stitch/ttf.html.

[10] Taco Hoekwater. OpenType fonts in LuaTeX. *TUGboat*, 29(1):34–35, 2008.

[11] Klaus Höppner. Truetype-Fonts in LaTeX. http://www.dalug.org/fileadmin/veranstaltungen/Slides/truetype.pdf.

[12] Micropress Inc. The OT1 font encoding. http://www.micropress-inc.com/fonts/encoding/ot1.htm.

[13] Micropress Inc. The T1 font encoding. http://www.micropress-inc.com/fonts/encoding/t1.htm.

[14] The Odradek Institute. How to install OpenType fonts in LaTeX. http://members.fortunecity.com/odradek5/otf-LaTeX/index.html.

[15] Bogusł aw Jackowski. Latin Modern fonts at eleventh hour. http://www.cstug.cz/aktivity/2005/lm-at11e.pdf.

[16] Christophe Jacquet. Polices et pdflatex, 2011. http://www.jacquet80.eu/blog/post/2006/11/03/60-polices-et-pdflatex.

[17] G. Markus Kuhn. Effective scientific electronic publishing – Use Type 1 vector fonts for generating PDF files with TeX, 2009. http://www.cl.cam.ac.uk/~mgk25/publ-tips/#type1.

[18] LaTeX-Community. Why type 3 fonts with \usepackage[T1]fontenc? http://www.latex-community.org/forum/viewtopic.php?f=5&t=571.

[19] David Lemon. Basic Type 1 Hinting. http://typophile.com/files/hinting.pdf.

[20] Yannick Losser. Utilisation de polices TrueType avec LaTeX. http://pulsar68.org/latex/ttf/.

[21] Luca Merciadri. Some misunderstood or unknown LaTeX 2ε tricks (V) and encoding issues. *TUGboat*. To appear; http://hdl.handle.net/2268/73436.

[22] John D. Owens. Installing OpenType Fonts in LaTeX with the LCDF Typetools. http://www.ece.ucdavis.edu/~jowens/code/otfinst/.

[23] John D. Owens. The Installation and Use of OpenType Fonts in LaTeX. *TUGboat*, 27(2):112–118, 2006.

[24] TeX Frequently Asked Questions. Finding '8-bit' Type 1 fonts. http://www.tex.ac.uk/cgi-bin/texfaq2html?label=type1T1.

[25] Damir Rakityansky. Using TrueType fonts with TeX (LaTeX) and pdfTeX (pdfLaTeX). http://www.radamir.com/tex/ttf-tex.htm.

[26] Denis Roegel. *The aeguill package*, 2003. http://archive.cs.uu.nl/mirror/CTAN/macros/latex/contrib/aeguill/guil-test1.pdf.

[27] Diego Santa Cruz. High quality PDF output from LaTeX and TeX. http://dsanta.users.ch/resources/type1.html.

[28] StackExchange. Latin Modern vs cm-super? http://tex.stackexchange.com/questions/1390/latin-modern-vs-cm-super.

[29] StackExchange. Teletype \textbackslash in alltt environment. http://tex.stackexchange.com/questions/16833/teletype-textbackslash-in-alltt-environment.

[30] StackExchange. Why should I use \usepackage[T1]fontenc ? http://tex.stackexchange.com/questions/664/why-should-i-use-usepackaget1fontenc/677#677.

[31] StackExchange. How to typeset boldface ell?, 2011. http://tex.stackexchange.com/questions/17199/how-to-typeset-boldface-ell.

[32] Stackoverflow. How do I use TrueType fonts with LaTeX. http://stackoverflow.com/questions/2525779/how-do-i-use-truetype-fonts-with-latex.

[33] Hàn Thê Thành, Sebastian Rahtz, and Hans Hagen. *The pdfTeX manual*, 2000. http://tex.loria.fr/moteurs/pdftex-a.pdf.

[34] Sun Tong. LaTeX And TrueType Font. http://xpt.sourceforge.net/techdocs/language/latex/latex33-LaTeXAndTrueTypeFont/.

[35] TUG. The LaTeX Font Catalogue. http://www.tug.dk/FontCatalogue/.

[36] Ulrik Vieth. Surviving the TeX font encoding mess. http://www.scribd.com/doc/2184814/Surviving-the-TeX-font-encoding-mess.

[37] Vladimir Volovich. CM-Super: Automatic creation of efficient Type 1 fonts from METAFONT fonts. *TUGboat*, 24(1):75–78, 2003.

[38] Wikipedia. LaTeX Internationalization – Output encoding, 2011. http://en.wikibooks.org/wiki/LaTeX/Internationalization#Output_encoding.