The `combofont` package

Ulrike Fischer*

May 27, 2017

1 Status: EXPERIMENTAL

This is a EXPERIMENTAL package.

It can disappear without notice e. g. if the `luaotfload` changes so that it no longer work, or if luatex changes, or if `fontspec` includes the code.

It is also possible that syntax and commands change in a incompatible way. So if you use it in a production environment: **You have been warned**.

2 Introduction

In version 2.7. `luaotfload` supports combining characters from multiple fonts into a single virtualized one.

That means that one can build a font that takes e.g. the capital letters from a sans serif font and the lowercase letters from a serif font. Or a font that pulls in missing greek or cyrillic glyphs from another font.

The methods pulls in *only* glyphs. It is not suitable for every imaginable font combination – some drawbacks are described below – and one should use it with care. Nevertheless it is a quite neat extension of the tools to manipulate fonts.

The main problem with the examples in the `luaotfload` manual is that it creates fonts of a fix size. This means that they don't respond to command like `\large` or `\footnotesize`.

After trying around a bit and then asking a question (https://tex.stackexchange.com/questions/371647/call-a-luatex-combo-font-through-nfss) I got from David Carlisle the idea

*fischer@troubleshooting-tex.de

to use a `size`-Funktion which one define with `\DeclareSizeFunction` to inject the needed code to size the combo-font in a nfss-`\DeclareFontShape`-command.

`combofont.sty` is the result.

It is not meant as a production package but as package that helps to exploit the use of combo fonts.

3 Requirements

You need at least an up-to-date TeXLive 2016. TeXLive 2017 with luatex 1.0.4. or a current miktex is better.

4 Using combo fonts

To be able to use a combo font with standard LaTeX font commands you have to do two things (the source code of this documentation is a complete example):

1. Setup and describe the building of the combo font with `\setupcombofont`
2. Write `nfss`-declarations

4.1 Setup the combo font

`\setupcombofont{`⟨*name*⟩`}{`⟨*comma list of basefonts*⟩`}{`⟨*comma list of ranges and code-points*⟩`}`

`{`⟨*name*⟩`}` is the name of the font. It should be some unique ascii-string without spaces. If you intent to define lots of fonts it would be a good idea to think about a sensible naming sheme. In the example here I simply used `combotest-regular` and `combotest-bold`.

`{`⟨*comma list of basefonts*⟩`}` This should be a list of font declarations you want to use to build your combo font. The syntax used is described in the `luaotfload` manual. Example:

```
{
 {file:lmroman10-regular.otf:\combodefaultfeat} at #1pt,
 {file:lmsans10-regular.otf} at \fpeval{#1/10*15}pt,
 {file:cmunrm.otf}           at #1pt
}
```

Important points are:

Order of the fonts The first font is the main font which will receive the glyphs. So think carefully which font is should be and setup its font features correctly. `combofont` defines as a helper command `\combodefaultfeat` which sets `mode=node;script=latn;language=DFLT;+tlig;`.

Size declaration The font description should end with a size declaration line `at #1pt`. When processing the font `#1` will be replaced by the current font size. As you can see in the second font you can do calculations.

`{`⟨*comma list of ranges and code-points*⟩`}` This is a comma list of settings which describe which glyphs are taken from the respective font. Example:

```
{
   {} ,
   0x41-0x5A*0x21*0x3F,
   fallback
}
```

Important points:

1. There should be as many settings as there are fonts.
2. Empty entries should be marked with a pair of braces (normally this should done for the first font).
3. You can add ranges of code points and single code points. Blocks are separated by an asterix `*`.
4. The keyword `fallback` means that this font is used for „missing glyphs“

4.2 Write `nfss`-declarations

After all the fonts you need have been setup, you can write suitable `nfss`-declaration which make it possible to call the font by family and other font commands. Example:

```
\DeclareFontFamily{TU}{combotest}{}
\DeclareFontShape {TU}{combotest}{m}{n} {<->combo*combotest-regular}{}
\DeclareFontShape {TU}{combotest}{bx}{n}{<->combo*combotest-bold}{}
```

The important point is the size-function `combo*` which does all the work.

5 Demonstration

`\fontfamily{combotest}\selectfont` – Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte? `\large`

Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte?

`\tiny`

Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte?

`\bfseries\normalsize`

Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte? `\large`**Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte?**

`\tiny`

Some Text with Capital Words! Eh bien, mon prince. Gênes et Lueques ne sont plus que des apanages, des поместья, de la famille Buonaparte?

6 Remarks and open questions

As mentioned in the introduction a combo font only pulls in glyphs. This has a lot of (not all yet understood or seen) side effects. Here a few things that should be considered when building a combo font:

Kerning Obviously some kerning works (see e.g. the large W before the o in the demonstration). But it is quite unclear which values are used, how bad it can get and if one can correct it.

Font features Only font features of the first font are taken into account. E.g. adding a color setting has an effect only if applied to the first font and then colors all glyphs. `+smcp` (the open type small caps feature) only has an effect if the first font knows it. Mixing scripts and languages is probably not possible (but I didn't try yet).

Speed I didn't try to optimize the loading of the fonts.

Pulling glyphs in other positions One interesting question would if it is possible to switch glyph positions before or after the pull. E.g. is one could move the chars a-z from arial to the math serif positions.

Side effect I naturally directly found a side-effect of such a combo font declaration: https://github.com/lualatex/luaotfload/issues/414. So the question is if there are more.