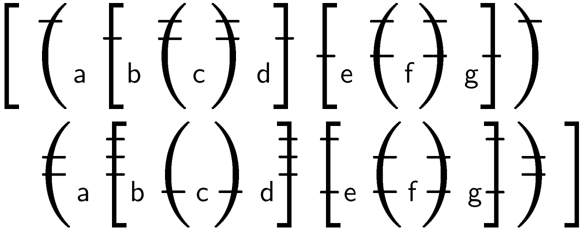
Zebrackets: A Score of Years and Delimiters

Michael Cohen, Blanca Mancilla, and John Plaice

U.Aizu, Mentel, GrammaTech and UNSW Australia

TUG meeting, July 2016

Zebrackets: striated parentheses and brackets



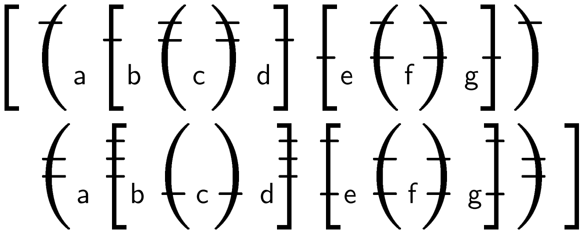
In expressions with many parentheses and brackets, striations can be added programmatically to better distinguish matching pairs in a specified region of a text.

In this example, the delimiters are four times normal size to accentuate the striations.

A bit of history

- *TUG 1993*: Cohen presented the paper “A pseudo-dynamic contextually adaptive font.” His implementation used a combination of Perl, C, and sh.
- 2014: Plaice and Cohen revived this work.
- 2016: Mancilla and Plaice reworked the whole system in Python to make it more flexible and usable.

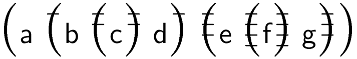
Another look at our example



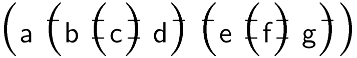
font = cmr	size = 10
magnification = 4	style = foreground
encoding = binary	index = unique
slots = 5	mixcount = true
origin = 0	direction = topdown

Index: unique, depth, or breadth

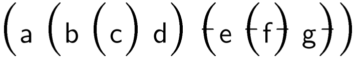
unique:



depth:



breadth:



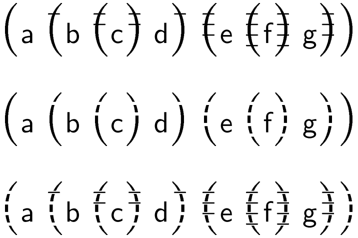
encoding = unary	style = foreground
magnification = 2	mixcount = true
origin = 0	direction = topdown

Style: foreground, background, or hybrid

foreground:

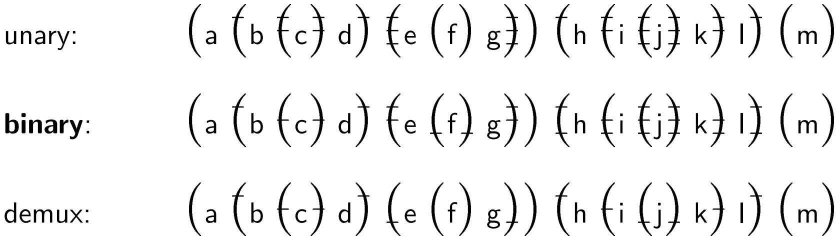
background:

hybrid:



encoding = unary	index = unique
magnification = 2	mixcount = true
origin = 0	direction = topdown

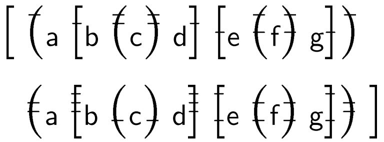
Encoding: unary, binary, or demux


unary: (a (b (c) d) (e (f) g)) (h (i (j) k) l) (m)
binary: (a (b (c) d) (e (f) g)) (h (i (j) k) l) (m)
demux: (a (b (c) d) (e (f) g)) (h (i (j) k) l) (m)

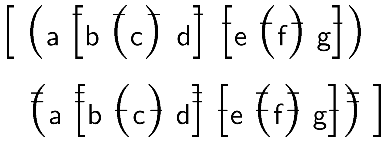
index = unique	style = foreground
magnification = 2	mixcount = true
origin = 0	direction = topdown

Mixcount: true or false

true:



false:

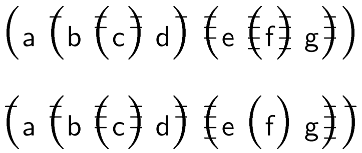


index = unique	style = foreground
magnification = 2	encoding = binary
origin = 0	direction = topdown

Origin: 0 or 1

0:

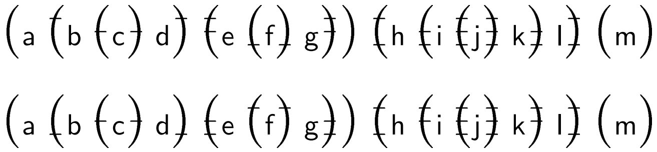
1:



index = unique	style = foreground
magnification = 2	encoding = binary
direction = topdown	mixcount = true

Direction: topdown or bottomup

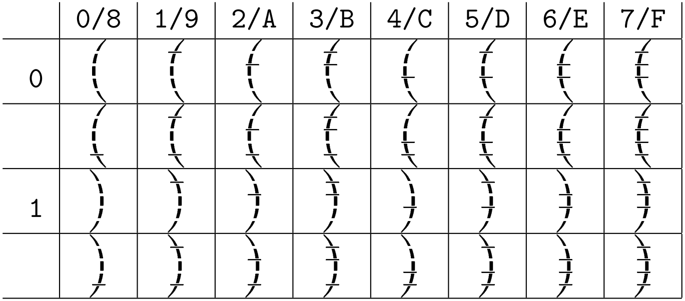
topdown:


(a (b (c) d) (e (f) g)) (h (i (j) k) l) (m)
(a (b (c) d) (e (f) g)) (h (i (j) k) l) (m)

bottomup:

index = unique	style = foreground
magnification = 2	encoding = binary
origin = 0	mixcount = true

Fonts



```
python3 zebrackets/zebraFont.py
        --kind parenthesis --style foreground
        --slots 3 --size 12 --family cmr
        --magnification 2
```

Magnification $= n$ means $\left(2^{\frac{1}{2}}\right)^{n}$.

Architecture of system

- The user writes a LaTeX file with suffix `.zbtex`.
- In this file, the user makes annotations of the form `\zebracketstext{`$\cdots$`}`, or delimited by `\begin{zebrackets}`$\cdots$`\end{zebrackets}` pairs, with appropriate parameters.
- These annotations designate regions of text in which delimiters are to be transformed with striated glyphs.
- This striation is done by explicitly designating which glyph in a font should be used, or automatically.
- For explicit striation, fonts must be created explicitly using the `\zebracketsfonts` command.

Automatic striation

- There is a two-pass algorithm over the text in that region.
- Unless the slots and glyphs are specified explicitly, the first pass computes the number of slots (maximum 7) needed to striate all of the delimiters in that region of text, Should, in theory, more slots than 7 be needed, then 7 are chosen, and the counting for striation purposes wraps through 0 in the appropriate encoding.
- The second pass generates the transformed text, replacing delimiters in the text with specific glyphs in the new fonts, which are generated on-the-fly, as needed.
- All parameters have default values.

Conclusions

- User adds annotations in the source file; these are interpreted by the `zebraParser.py` script to generate a LaTeX file.
- Implementation is slow, because of use of Python, lack of caching of font information, and redundant error checking.
- Proof of concept is successful, future work requires a grand vision, probably not based on TeX.