Typesetting Mathematics with Neateqn

A. G. Rudi

This document briefly introduces Neateqn, an eqn implementation for typesetting mathematical formulas in Neatroff. For further information about Neatroff and Neateqn, see https://dev.rudi.ir/.

The Algorithm

Neatroff follows the rules described in appendix G of Knuth's Texbook, which explains Tex's algorithm for typesetting mathematical formulas. In Tex, sub-formulas of a formula are always typeset in one of eight predefined styles. The formulas inside .EQ/.EN blocks are rendered in displayed styles and inline equations are rendered in text styles. Their main difference is that in text styles formulas are vertically more compact to reduce the amount of extra spacing required to be inserted between lines.

The default value of the parameters of the typesetting algorithm, such as the position of subscripts, can be changed. These parameters are described in appendix G of the Texbook and can be modified in Neatroff with Groff eqn-style "set" commands. See the Groff eqn manual page or the Texbook for a list and explanations.

Defining Custom Brackets

It is possible to adjust the default brackets or to define new ones. Two commands are available for this purpose: one for specifying different bracket sizes (bracketsizes) and one for specifying bracket building glyphs (bracketpieces):

```
bracketsizes sign N glyph1 glyph2 ... glyphN
bracketpieces sign top mid bot cen
```

In these commands, sign is the token placed after the "left" and "right" keywords in equations. In the bracketsizes command, the glyphs should be ordered based on their size with the smallest glyph appearing first. Neateqn tries the specified

glyphs in the same order until it finds the glyph that is large enough for the enclosed formula. If this fails, it tries to build the bracket if its pieces are defined (by default or with bracketpieces). The four arguments of bracketpieces specify the glyphs required for building the bracket. The last argument can be empty if the bracket has no centre (\(lk is the centre of {, for instance).

As an example, the following lines show how the default opening and closing parenthesis can be defined:

```
bracketpieces ( "\(LT" "\(LX" "\(LB" ""
bracketpieces ) "\(RT" "\(RX" "\(RB" ""
```

The following lines do the same for braces:

```
bracketpieces { "\(lt" "\(bv" "\(lb" "\(lk"
bracketpieces } "\(rt" "\(bv" "\(rb" "\(rk"
```

The following line instructs Neateqn to use Tex's open parenthesis glyphs with different sizes (note that in Neatroff \N'gid' is the glyph with device-dependent name gid):

```
bracketsizes ( 5 "(" "\N'parenleftbig'" "\N'parenleftBig'"
     "\N'parenleftbigg'" "\N'parenleftBigg'"
```

Adjusting the Syntax

The logic used in eqn to partition equations may seem odd to new users; for instance in "O(n sup 2)", the expected result may be $O(n^2)$, instead of $O(n^{2)}$. Even experienced eqn users occasionally make these mistakes and some insert spaces around most tokens to prevent possible surprises. Equations like "O (n sup 2)", which prevent most of these problems, are not as readable as the alternative, however. This issue is one of the main advantages of Tex's more concise syntax. In Neateqn it is possible to make equations like the first work.

Neateqn splits (chops) equations at specific characters. Equations are always chopped at spaces and before and after open and close braces. By default, equations are also chopped before and after ^, ~, and " (but this can be changed). The -c option of Neateqn allows specifying the characters around which equations are chopped. For instance, if "~^"()," is passed with -c to Neateqn, "O(n sup 2)" is interpreted as "O (n sup 2)". This may be considered an improvement, but a

more important advantage in the author's opinion is that these characters may be redefined. For instance, one may redefine open and close parenthesis as follows:

```
define ( @{ left "(" @
define ) @right ")" }@
```

Then, it is possible to write "(a over b) sup (c + 5)" to get $\left(\frac{a}{b}\right)^{(c+5)}$. Note that macro arguments are never split away from the macro name, thus one can safely call "log(a, n)", if log is defined as follows.

```
define log @{roman "log" sub {$1}({$2})}@
```

Assigning Character Type

Neateqn determines the spacing between characters in an equation based on their type (see chapter 18 of the Texbook for more information). It is possible to specify or change the type of a character in Neateqn with the "chartype" command. Possible types are "ord" for ordinary atoms, "op" for large operators, "bin" for binary operators, "rel" for relations, "open" for opening brackets, "close" for closing brackets, "punct" for punctuations and "inner" for fractions. As an example, the following line declares backslash as a binary operator:

```
chartype bin \(rs
```

The second argument of the "chartype" command should be a Troff character name. If the operator is not a character, it can be defined as one. For instance, for "**>>**" and "log" operators, one may define the following two characters (note that the following two lines are Neatroff requests and should be placed outside .EQ/.EN blocks):

```
.char \[eqn.log] "log
.char \[eqn.>>]  ">\h'-.1n'>
```

Then, the type of the operators can be specified as explained above:

```
chartype op  \[eqn.log]
chartype rel \[eqn.>>]
```

Finally, macros like the following may be defined to improve readability:

```
define >>  @\[eqn.>>]@
```

```
define log @\[eqn.log]@
```

Breaking Equations

Neateqn can break equations after top-level operators; This is important especially when there are long inline equations in the text. The "breakcost" command can specify the cost of a line break after different character types: its first argument is the character type, as enumerated in the previous section, and its second argument is the cost of line breaks after the given character type. Costs are specified via Neatroff's \j escape sequence (The document "Neatroff Introduction" explains the meaning of these costs). The default values are:

```
breakcost rel 100
breakcost bin 200
breakcost punct 1000
```

A value of 0 disables breaking equations for the specified character. Note that Neateqn breaks equations after top-level operators only. Thus, equations surrounded by braces will not be broken. The following command instructs Neateqn never to break equations:

```
breakcost any 0
```

Using Tex Mathematical Symbols

In order to use Tex's mathematical symbols in Neatroff, CMEX10 and CMSY10 fonts (or their equivalents, for instance TXEX and TXSY for Txfonts or PXEX and PXSY for Pxfonts) should be mounted and declared as the special font of eqn Roman font (the font declared as grfont in Neateqn).

```
.fp - CMEX CMEX10
.fp - CMSY CMSY10
.fspecial R CMEX CMSY
```

If the italic font lacks Greek characters, CMMI10 (or its equivalents, like RTXMI for Txfonts or RPXMI for Pxfonts) can be mounted and declared as a special font of eqn italic font (the font declared as gfont in Neateqn).

```
.fp - CMMI RPXMI
```

```
.fspecial I CMMI
```

Standard symbols can also be redefined to use Computer Modern glyphs, like those for summation and product:

```
define sum @{vcenter roman "\N'summationdisplay'"}@
define tsum @{vcenter roman "\N'summationtext'"}@
define prod @{vcenter roman "\N'productdisplay'"}@
define tprod @{vcenter roman "\N'producttext'"}@
```

Retrieving and installing Pxfonts and Txfonts

These fonts are optional and Neatroff will run fine without them. This document can make use of these fonts; see the instructions below.
Download both fonts from CPAN:

```
https://ctan.org/pkg/pxfonts (link)
https://ctan.org/pkg/txfonts (link)
```

Unpack both archives in separate directories. In each directory copy the contents of the contained "afm" and "pfb" directories into the "fonts" directory under neatroff_make. Then, edit neateqn.ms (source of this document): Search for "Change this section" and comment/uncomment a few lines as described there. Finally, submit 'make neat' in neatroff_make again, then submit 'make clean all' in this folder ("demo").

Some Samples For Different Fonts

Palatino and Computer Modern mathematical symbols:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

$$[a] + \left(\frac{a}{b}\right) + \left\{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}\right\} + \sqrt{a} + \sqrt{\frac{a}{b}} + \sqrt{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}}$$

Palatino and Pxfonts mathematical symbols:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

$$[a] + \left(\frac{a}{b}\right) + \left\{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}\right\} + \sqrt{a} + \sqrt{\frac{a}{b}} + \sqrt{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}}$$

Times Roman and Txfonts mathematical symbols:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

$$[a] + \left(\frac{a}{b}\right) + \left\{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}\right\} + \sqrt{a} + \sqrt{\frac{a}{b}} + \sqrt{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}}$$

Computer Modern:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

$$[a] + \left(\frac{a}{b}\right) + \left\{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}\right\} + \sqrt{a} + \sqrt{\frac{a}{b}} + \sqrt{\frac{x+\frac{a}{b}}{y+\frac{c}{d}}}$$